

Université de Montréal

**Understanding, improving, and generalizing generative models**

par

**Alexia Jolicoeur-Martineau**

Département d'informatique et de recherche opérationnelle  
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences  
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)  
en informatique

Août, 2022

© Alexia Jolicoeur-Martineau, 2022.

# Résumé

Les modèles génératifs servent à générer des échantillons d'une loi de probabilité (ex. : du texte, des images, de la musique, des vidéos, des molécules, et beaucoup plus) à partir d'un jeu de données (ex. : une banque d'images, de texte, ou autre). Entraîner des modèles génératifs est une tâche très difficile, mais ces outils ont un très grand potentiel en termes d'applications. Par exemple, dans le futur lointain, on pourrait envisager qu'un modèle puisse générer les épisodes d'une émission de télévision à partir d'un script et de voix générés par d'autres modèles génératifs.

Il existe plusieurs types de modèles génératifs. Pour la génération d'images, l'approche la plus fructueuse est sans aucun doute la méthode de réseaux adverses génératifs (GANs). Les GANs apprennent à générer des images par un jeu compétitif entre deux joueurs, le Discriminateur et le Générateur. Le Discriminateur tente de prédire si une image est vraie ou fausse, tandis que le Générateur tente de générer des images plus réalistes en apprenant à faire croire au discriminateur que ces fausses images générées sont vraies. En complétant ce jeu, les GANs arrivent à générer des images presque photo-réalistes. Il est souvent possible pour des êtres humains de distinguer les fausses images (générés par les GANs) des vraies images (ceux venant du jeu de données), mais la tâche devient plus difficile au fur et à mesure que cette technologie s'améliore. Le plus gros défaut des GANs est que les données générées par les GANs manquent souvent de diversité (ex. : les chats au visage aplati sont rares dans la banque d'images, donc les GANs génèrent juste des races de chats plus fréquentes). Ces méthodes sont aussi souvent très instables. Il y a donc encore beaucoup de chemin à faire avant l'obtention d'images parfaitement photo-réalistes et diverses.

De nouvelles méthodes telles que les modèles de diffusion à la base de score semblent produire de meilleurs résultats que les GANs, donc tout n'est pas gagné pour les GANs. C'est pourquoi cette thèse n'est pas concentrée seulement sur les GANs, mais aussi sur les modèles de diffusion. Notez que cette thèse est exclusivement concentrée sur la génération de données continues (ex. : images, musique, vidéos) plutôt que discrètes (ex. : texte), car cette dernière fait usage de méthodes complètement différentes.

Le premier objectif de cette thèse est d'étudier les modèles génératifs de façon théorique pour mieux les comprendre. Le deuxième objectif de cette thèse est d'inventer de nouvelles astuces (nouvelles fonctions objectives, régularisations, architectures, etc.) permettant d'améliorer les modèles génératifs. Le troisième objectif est de généraliser ces approches au-delà de leur formulation initiale, pour permettre

---

la découverte de nouveaux liens entre différentes approches.

Ma première contribution est de proposer un discriminateur *relativiste* qui estime la probabilité qu'une donnée réelle, soit plus réaliste qu'une donnée fausse (inventée par un modèle générateur). Les GANs relativistes forment une nouvelle classe de fonctions de perte qui apportent beaucoup de stabilité durant l'entraînement. Ma seconde contribution est de prouver que les GANs relativistes forment une mesure de dissimilarité. Ma troisième contribution est de concevoir une variante adverse au appariement de score pour produire des données de meilleure qualité avec les modèles de diffusion. Ma quatrième contribution est d'améliorer la vitesse de génération des modèles de diffusion par la création d'une méthode numérique de résolution pour équations différentielles stochastiques (SDEs).

**Mots clés:** réseaux adverses génératifs, apprentissage profond, modèles génératifs, débruitage par appariement de score avec échantillonnage de Langevin, débruitage de modèles probabilistes de diffusion

# Abstract

Generative models are powerful tools to generate samples (e.g., images, music, text) from an unknown distribution given a finite set of examples. Generative models are hard to train successfully, but they have the potential to revolutionize arts, science, and business. These models can generate samples from various data types (e.g., text, images, audio, videos, 3d). In the future, we can envision generative models being used to create movies or episodes from a TV show given a script (possibly also generated by a generative model).

One of the most successful methods for generating images is Generative Adversarial Networks (GANs). This approach consists of a game between two players, the Discriminator and the Generator. The goal of the Discriminator is to classify an image as real or fake, while the Generator attempts to fool the Discriminator into thinking that the fake images it generates are real. Through this game, GANs are able to generate very high-quality samples, such as photo-realistic images. Humans are still generally able to distinguish real images (from the training dataset) from fake images (generated by GANs), but the gap is lessening as GANs become better over time. The biggest weakness of GANs is that they have trouble generating diverse data representative of the full range of the data distribution. Thus, there is still much progress to be made before GANs reach their full potential.

New methods performing better than GANs are also appearing. One prime example is score-based diffusion models. This thesis focuses on generative models that seemed promising at the time for continuous data generation: GANs and score-based diffusion models.

I seek to improve generative models so that they reach their full potential (Objective 1: Improving) and to understand these approaches better on a theoretical level (Objective 2: Theoretical understanding). I also want to generalize these approaches beyond their original setting (Objective 3: Generalizing), allowing the discovery of new connections between different concepts/fields.

My first contribution is to propose using a *relativistic* discriminator, which estimates the probability that a given real data is more realistic than a randomly sampled fake data. Relativistic GANs form a new class of GAN loss functions that are much more stable with respect to optimization hyperparameters. My second contribution is to take a more rigorous look at relativistic GANs and prove that they are proper statistical divergences. My third contribution is to devise an adversarial variant to denoising score matching, which leads to higher quality data with score-based diffusion models. My fourth contribution is to significantly improve the speed

---

of score-based diffusion models through a carefully devised Stochastic Differential Equation (SDE) solver.

**Keywords:** generative adversarial networks, deep learning, generative models, denoising score matching, denoising diffusion probabilistic models

# Contents

<b>Résumé</b>	<b>ii</b>
<b>Summary</b>	<b>iv</b>
<b>Contents</b>	<b>vi</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Figures</b>	<b>xiv</b>
<b>Acknowledgments</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 What are generative models?	1
1.2 Long-term vision	2
1.3 Focus and goals of this thesis	3
1.4 Overview of the structure	4
1.5 Excluded work	5
<b>2 Background</b>	<b>6</b>
2.1 Neural networks	6
2.1.1 Artificial Neural Networks	6
2.1.2 Convolutional Neural Networks (CNN)	8
2.2 Sampling from a known continuous distribution	12
2.2.1 Classic sampling methods	12
2.2.1.1 Sampling from a uniform distribution	12
2.2.1.2 Inverse Transform Sampling	12
2.2.1.3 Rejection Sampling	13
2.2.2 Markov chain Monte Carlo (MCMC) methods	13
2.2.2.1 Metropolis-Hastings	14
2.2.2.2 Random Walk	14
2.2.2.3 Langevin Monte Carlo (Langevin sampling)	14
2.2.3 Divergence minimization	16
2.3 Sampling from an unknown continuous distribution	18
2.3.1 Score Matching	18

---

2.3.1.1	Traditional score matching . . . . .	18
2.3.1.2	Sliced score matching . . . . .	19
2.3.1.3	Denoising score matching . . . . .	20
2.3.2	Divergence estimation . . . . .	21
2.4	Score-based diffusion models . . . . .	23
2.4.1	Denoising Score Matching with Annealed Sampling (DSM-AS)	24
2.4.1.1	Sampling from the distribution induced by the score function . . . . .	24
2.4.2	Generalizing DSM-AS by reversing a process from data to noise (modern score-based diffusion models) . . . . .	26
2.4.3	Discussion . . . . .	29
2.5	Generative Adversarial Networks (GANs) . . . . .	32
2.5.1	Standard GAN . . . . .	34
2.5.2	Generalizing GANs to other objective functions . . . . .	35
2.5.3	Divergence minimization perspective . . . . .	36
2.5.4	Wasserstein distance and its influence on modern GANs . .	36
2.5.5	Discussion . . . . .	38
<b>3</b>	<b>Prologue to the first article . . . . .</b>	<b>40</b>
3.1	Article Details . . . . .	40
3.2	Context . . . . .	40
3.3	Personal contribution to the paper . . . . .	41
<b>4</b>	<b>The relativistic discriminator: a key element missing from standard GAN</b>	<b>42</b>
4.1	Abstract . . . . .	42
4.2	Introduction . . . . .	43
4.3	Background . . . . .	44
4.3.1	Generative adversarial networks . . . . .	44
4.3.2	Integral probability metrics . . . . .	46
4.4	Missing property of SGAN . . . . .	46
4.4.1	Prior knowledge argument . . . . .	46
4.4.2	Divergence minimization argument . . . . .	47
4.4.3	Gradient argument . . . . .	47
4.5	Method . . . . .	49
4.5.1	Relativistic standard GAN . . . . .	49
4.5.2	Relativistic GANs . . . . .	49
4.5.3	Relativistic average GANs . . . . .	50
4.6	Experiments . . . . .	51
4.6.1	CIFAR-10 . . . . .	52
4.6.2	CAT . . . . .	53

---

4.7	Conclusion and future work . . . . .	55
<b>5</b>	<b>Prologue to the second article . . . . .</b>	<b>56</b>
5.1	Article Details . . . . .	56
5.2	Context . . . . .	56
5.3	Personal contribution to the paper . . . . .	56
<b>6</b>	<b>On relativistic <math>f</math>-divergences . . . . .</b>	<b>57</b>
6.1	Abstract . . . . .	57
6.2	Introduction . . . . .	57
6.3	Background . . . . .	59
6.3.1	Generative Adversarial Networks . . . . .	59
6.3.2	Integral Probability Metrics . . . . .	60
6.3.3	Relativistic GANs . . . . .	61
6.4	Relativistic Divergences . . . . .	61
6.4.1	Main Theorem . . . . .	62
6.4.2	Sketch of the Proof . . . . .	64
6.4.3	Subtypes of Divergences . . . . .	65
6.4.4	Weakness of the Divergence . . . . .	67
6.5	Estimators . . . . .	69
6.5.1	RpGANs . . . . .	69
6.5.2	RaGANs and RalfGANs . . . . .	69
6.6	Experiments . . . . .	71
6.6.1	Bias . . . . .	72
6.6.2	Divergences . . . . .	73
6.7	Conclusion . . . . .	73
<b>7</b>	<b>Prologue to the third article . . . . .</b>	<b>75</b>
7.1	Article Details . . . . .	75
7.2	Context . . . . .	75
7.3	Personal contribution to the paper . . . . .	75
<b>8</b>	<b>Adversarial score matching and improved sampling for image generation</b>	<b>77</b>
8.1	Abstract . . . . .	77
8.2	Introduction . . . . .	77
8.3	Background . . . . .	79
8.3.1	Denoising Score Matching . . . . .	79
8.3.2	Annealed Langevin Sampling . . . . .	80
8.3.3	Expected denoised sample (EDS) . . . . .	81
8.4	Consistent scaling of the noise . . . . .	82
8.4.1	Inconsistencies in ALS . . . . .	82
8.4.2	Algorithm . . . . .	83



---

8.5	Benefits of the EDS on synthetic data and image generation . . . . .	84
8.6	Adversarial formulation . . . . .	86
8.7	Experiments . . . . .	88
8.7.1	Ablation Study . . . . .	88
8.7.2	Non-adversarial and Adversarial score networks have equally high diversity . . . . .	90
8.8	Conclusion . . . . .	91
<b>9</b>	<b>Prologue to the fourth article . . . . .</b>	<b>92</b>
9.1	Article Details . . . . .	92
9.2	Context . . . . .	92
9.3	Personal contribution to the paper . . . . .	92
<b>10</b>	<b>Gotta Go Fast when generating data with score-based models . . . . .</b>	<b>93</b>
10.1	Abstract . . . . .	93
10.2	Introduction . . . . .	93
10.3	Background . . . . .	97
10.3.1	Score-based modeling with SDEs . . . . .	97
10.3.2	Variance Exploding (VE) process . . . . .	98
10.3.3	Variance Preserving (VP) process . . . . .	98
10.3.4	Solving the Reverse Diffusion Process (RDP) . . . . .	99
10.4	Efficient Method for Solving Reverse Diffusion Processes . . . . .	99
10.4.1	Setting up the algorithm . . . . .	99
10.4.1.1	Integration method . . . . .	100
10.4.1.2	Tolerance . . . . .	101
10.4.1.3	Norm of the scaled error . . . . .	102
10.4.1.4	Hyperparameters of the dynamic step size algorithm . . . . .	102
10.4.1.5	Handling the mini-batch . . . . .	103
10.4.2	Algorithm . . . . .	103
10.5	Experiments . . . . .	104
10.5.1	Performance . . . . .	105
10.5.2	Solving an ODE instead of an SDE . . . . .	107
10.5.3	DDIM . . . . .	108
10.6	Limitations . . . . .	108
10.7	Conclusion . . . . .	108
<b>11</b>	<b>Discussion . . . . .</b>	<b>109</b>
	<b>References . . . . .</b>	<b>111</b>
<b>A</b>	<b>The relativistic discriminator: a key element missing from standard GAN . . . . .</b>	<b>126</b>
A.1	Intuitive and memeful visual representation of RaGANs . . . . .	126

---

A.2	Intuition behind RaGANs	126
A.3	Gradients	128
A.3.1	SGAN	128
A.3.2	IPM-based GANs	129
A.4	Simplified form of relativistic saturating and non-saturating GANs	129
A.5	Testing the gradient argument	130
A.6	CIFAR-10 Hard/unstable experiments	132
A.7	Loss functions used in experiments	132
A.7.1	SGAN (non-saturating)	132
A.7.2	RSGAN	133
A.7.3	RaSGAN	133
A.7.4	LSGAN	133
A.7.5	RaLSGAN	134
A.7.6	HingeGAN	134
A.7.7	RaHingeGAN	134
A.7.8	WGAN-GP	135
A.7.9	RSGAN-GP	135
A.7.10	RaSGAN-GP	135
A.8	Algorithms	136
A.9	Architectures	138
A.9.1	Standard CNN	138
A.9.2	DCGAN 64x64	140
A.9.3	DCGAN 128x128	141
A.9.4	DCGAN 256x256	143
A.10	Samples	144
<b>B</b>	<b>On relativistic <math>f</math>-divergences</b>	<b>152</b>
B.1	Proving that the objective functions are divergences	152
B.2	Inequalities between Relativistic Divergences	166
B.3	Bias in RalfGANs, RaGANs, and RcGANs	168
B.3.1	SGAN	169
B.3.2	(Ra) LSGAN	170
B.3.3	(Rc) LSGAN	173
B.3.4	HingeGAN	175
B.4	Architecture	176
<b>C</b>	<b>Adversarial score matching and improved sampling for image generation</b>	<b>178</b>
C.1	Broader Impact	178
C.2	Experiments details	179
C.3	Supplementary experiments	180
C.4	Optimal unconditional score function	181
C.5	Optimal conditional score function	182

---

C.6	ALS Non-geometric proof . . . . .	183
C.7	CAS Geometric proof . . . . .	185
C.8	Update rule . . . . .	186
C.9	Inception Score (IS) . . . . .	186
C.10	Uncurated samples . . . . .	188
<b>D</b>	<b>Gotta Go Fast When Generating Data with Score-Based Models . . . . .</b>	<b>212</b>
D.1	DifferentialEquations.jl . . . . .	212
D.2	Effects of modifying Algorithm 1 . . . . .	212
D.3	Dynamic step size algorithm for solving any type of SDE (rather than just Reverse Diffusion Processes) . . . . .	215
D.4	Implementation Details . . . . .	217
D.5	Inception Score on CIFAR-10 . . . . .	219
D.6	Stability and Bias of the Numerical Scheme . . . . .	219
D.7	Samples . . . . .	223

# List of Tables

4.1	Fréchet Inception Distance (FID) at exactly 100k generator iterations on the CIFAR-10 dataset using stable setups with different GAN loss functions. We used spectral norm in $D$ and batch norm in $G$ . All models were trained using the same <i>a priori</i> selected seed (seed=1). . . . .	53
4.2	Minimum (min), maximum (max), mean, and standard deviation (SD) of the Fréchet Inception Distance (FID) calculated at 20k, 30k . . . , 100k generator iterations on the CAT dataset with different GAN loss functions. The hyper-parameters used were $lr = .0002$ , $\beta = (.50, .999)$ , $n_D = 1$ , and batch norm (BN) in $D$ and $G$ . All models were trained using the same <i>a priori</i> selected seed (seed=1). Note: A missing number imply that the model did not converge and became stuck in the first few iterations. . . . .	54
6.1	Minimum (and standard deviation) of the FID calculated at 10k, 20k, . . . , 100k iterations using different loss functions (see equations 11, 12, 13) and datasets. . . . .	73
8.1	[Non-denoised / Denoised FID] from 10k samples on CIFAR-10 (32x32) with Song and Ermon (2019) score network architecture . . . . .	88
8.2	[Non-denoised / Denoised FID] from 10k samples on LSUN-Churches (64x64) with Song and Ermon (2019) score network architecture . . . . .	89
8.3	[Non-denoised / Denoised FID] from 10k samples on CIFAR-10 (32x32) with Ho et al. (2020) unconditional score network architecture . . . . .	89
8.4	As in Lin et al. (2018), we generated 26k samples and evaluated the mode coverage and KL divergence based on the predicted modes from a pre-trained MNIST classifier. . . . .	90
10.1	Number of score Function Evaluations (NFE) / Fréchet Inception Distance (FID) on CIFAR-10 (32x32) from 50K samples . . . . .	106
10.2	Number of score Function Evaluations (NFE) / Fréchet Inception Distance (FID) on LSUN-Church (256x256) and FFHQ (256x256) from 5K samples . . . . .	107

---

A.1	An illustrative example of the discriminator’s output in standard GAN as traditionally defined ( $P(x_r \text{ is real}) = \text{sigmoid}(C(x_r))$ ) versus the Relativistic average Discriminator (RaD) ( $P(x_r \text{ is real} \overline{C(x_f)}) = \text{sigmoid}(C(x_r) - \overline{C(x_f)})$ ). Breads represent real images, while dogs represent fake images. . . . .	127
A.2	Fréchet Inception Distance (FID) at exactly 100k generator iterations on the CIFAR-10 dataset using unstable setups with different GAN loss functions. Unless otherwise specified, we used $lr = .0002$ , $\beta = (.50, .999)$ , $n_D = 1$ , and batch norm (BN) in $D$ and $G$ . All models were trained using the same <i>a priori</i> selected seed (seed=1). . . . .	133
C.1	Sampling learning rates for non-adversarial ( $\epsilon$ ) and adversarial ( $\epsilon_{adv}$ ) score matching . . . . .	179
D.1	Short experiments with various SDE solvers from <i>DifferentialEquations.jl</i> on the VP model with a small mini-batch. . . . .	212
D.2	Effect of different settings on the [Inception score (IS) / Fréchet Inception Distance (FID) / Number of score Function Evaluations (NFE)] from 10k samples (with mini-batches of 1k samples) with the VP - CIFAR10 model. . . . .	213
D.3	Effect of different settings on the [Inception score (IS) / Fréchet Inception Distance (FID) / Number of score Function Evaluations (NFE)] from 10k samples (with mini-batches of 1k samples) with the VE - CIFAR10 model. . . . .	214
D.4	Inception Score on CIFAR-10 (32x32) from 50K samples . . . . .	219

# List of Figures

2.1	Neural networks used in Generative Adversarial Network . . . . .	7
2.2	$2 \times 2$ convolution over a small black-and-white $3 \times 3$ image . . . . .	9
2.3	Convolutional Neural Networks (CNNs) used in Generative Adversarial Network . . . . .	11
2.4	Samples of humans generated through denoising diffusion models (Ho et al., 2020) . . . . .	24
2.5	Samples of humans, beds, towers, and churches generated using an improved version of denoising score matching (Song and Ermon, 2020) . . . . .	25
2.6	Forward and Negative diffusion processes. We can define a Forward process going from data to noise and reverse this process to go from noise to data. However, to reverse the process, we need to know the score function at every step. . . . .	26
2.7	Comparing the quality and diversity of generated red fishes from ImageNet (Ho et al., 2021; Razavi et al., 2019; Brock et al., 2019) . . . . .	30
2.8	Samples of cats generated by my Relativistic GAN (Jolicoeur-Martineau, 2019) . . . . .	32
2.9	Samples generated by BigGAN (Brock et al., 2019) . . . . .	33
2.10	Samples of humans generated by StyleGAN2 (Karras et al., 2020) . . . . .	33
4.1	Expected discriminator output of the real and fake data for the a) direct minimization of the Jensen–Shannon divergence, b) actual training of the generator to minimize its loss function, and c) ideal training of the generator to minimize its loss function (lines are dotted when they cross beyond the equilibrium to signify that this may or may not be necessary). . . . .	48
6.1	Plots of $f$ with respect to the critic’s difference (CD) using three appropriate choices of $f$ for relativistic divergences. The bottom gray line represents $f(0) = 0$ ; the divergence is zero if all CDs are zero. The above gray line represents the maximum of $f$ ; the divergence is maximized if all CDs leads to that maximum. . . . .	66

---

6.2	Plots of the relative bias (i.e., the biased estimate divided by the unbiased estimate) of relativistic average and centered $f$ -divergences estimators over training time on CIFAR-10 with a mini-batch size of 32. Approximations of the bias were made using 320 independent samples. . . . .	72
8.1	Standard deviation during idealized sampling using a perfect score function $s^*$ . The black curve in (a) corresponds to the true geometric progression, as demonstrated in Proposition 2. . . . .	83
8.2	Langevin sampling on synthetic 2D experiments. Circles are real data points, crosses are generated data points. On both datasets, taking the EDS brings the samples much closer to the real data manifold. . . . .	85
8.3	Partial estimate of FID (lower is better) as a function of the sampling step size on CIFAR-10, with $n_\sigma = 1$ . The interactions between <i>consistent sampling</i> and denoising are shown. . . . .	86
10.1	Comparison between our novel SDE solver at various values of error tolerance and Euler-Maruyama for an equal computational budget. We measure speed through the Number of Function Evaluations (NFE) and the quality of the generated images through the Fréchet Inception Distance (FID; lower is better). See Table 10.1-10.2 for more details. . . . .	95
A.1	Density plots of the mini-batch average of $D(x_r)$ during the first 100 iterations of training on CAT with 256x256 images using only one or two generator updates per discriminator updates. If $D(x_r) = 0$ in all iterations, this would mean that the loss function would be the same as IPM-based GANs. . . . .	131
A.2	64x64 cats with RaLSGAN (FID = 11.97) . . . . .	145
A.3	128x128 cats with RaLSGAN (FID = 15.85) . . . . .	146
A.4	256x256 cats with GAN (5k iterations) . . . . .	147
A.5	256x256 cats with LSGAN (5k iterations) . . . . .	147
A.6	256x256 cats with RaSGAN (FID = 32.11) . . . . .	148
A.7	256x256 cats with RaLSGAN (FID = 35.21) . . . . .	149
A.8	256x256 cats with SpectralSGAN (FID = 54.73) . . . . .	150
A.9	256x256 cats with WGAN-GP (FID > 100) . . . . .	151
C.1	Denoised sample evolving over time for different methods . . . . .	188
C.2	CIFAR-10 Non-adversarial non-consistent $n_\sigma = 1$ . . . . .	189
C.3	CIFAR-10 Adversarial non-consistent $n_\sigma = 1$ . . . . .	189
C.4	CIFAR-10 Non-adversarial non-consistent $n_\sigma = 5$ . . . . .	190
C.5	CIFAR-10 Adversarial non-consistent $n_\sigma = 5$ . . . . .	191

---

C.6	CIFAR-10 Non-adversarial consistent $n_\sigma = 1$	192
C.7	CIFAR-10 Adversarial consistent $n_\sigma = 1$	193
C.8	CIFAR-10 Non-adversarial consistent $n_\sigma = 5$	194
C.9	CIFAR-10 Adversarial consistent $n_\sigma = 5$	195
C.10	LSUN-Churches Non-adversarial non-consistent $n_\sigma = 1$	196
C.11	LSUN-Churches Adversarial non-consistent $n_\sigma = 1$	197
C.12	LSUN-Churches Non-adversarial non-consistent $n_\sigma = 5$	198
C.13	LSUN-Churches Adversarial non-consistent $n_\sigma = 5$	199
C.14	LSUN-Churches Non-adversarial consistent $n_\sigma = 1$	200
C.15	LSUN-Churches Adversarial consistent $n_\sigma = 1$	201
C.16	LSUN-Churches Non-adversarial consistent $n_\sigma = 5$	202
C.17	LSUN-Churches Adversarial consistent $n_\sigma = 5$	203
C.18	Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial non-consistent $n_\sigma = 1$	204
C.19	Ho et al. (2020) network architecture with CIFAR-10 Adversarial non-consistent $n_\sigma = 1$	205
C.20	Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial non-consistent $n_\sigma = 5$	206
C.21	Ho et al. (2020) network architecture with CIFAR-10 Adversarial non-consistent $n_\sigma = 5$	207
C.22	Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial consistent $n_\sigma = 1$	208
C.23	Ho et al. (2020) network architecture with CIFAR-10 Adversarial consistent $n_\sigma = 1$	209
C.24	Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial consistent $n_\sigma = 5$	210
C.25	Ho et al. (2020) network architecture with CIFAR-10 Adversarial consistent $n_\sigma = 5$	211
D.1	VP - CIFAR10	223
D.2	VP-deep - CIFAR10	224
D.3	VE - CIFAR10	225
D.4	VE-deep - CIFAR10	226
D.5	VE - LSUN-Church (256x256)	227
D.6	VE - FFHQ (256x256)	228



# List of acronyms and abbreviations

e.g.	<i>exempli gratia</i> [for instance]
i.e.	<i>ide est</i> [that is]
GANs	Generative Adversarial Networks
MCMC	Markov Chain Monte Carlo
LMC	Langevin Monte Carlo
TV	Television
GPU	Graphics processing unit
SOTA	State Of The Art
ESRGAN	Enhanced Super-Resolution GAN
IPM	Integral Probability Metric
PRNG	PseudoRandom Number Generator
ITS	Inverse Transform Sampling
PDF	Probability Density Function
CDF	Cumulative Distribution Function
IS	Inception Score
FID	Fréchet Inception Distance
DSM	Denoising Score Matching
ALS	Annealed Langevin Sampling
DDP	Denoising Diffusion Processes
ODE	Ordinary Differential Equation
SDE	Stochastic Differential Equation
KL	Kullback-Leibler divergence
JSD	Jensen-Shannon divergence
VE	Variance Exploding
VP	Variance Preserving
FDP	Forward Diffusion Process
PC	Predictor-Corrector
$D$	Discriminator
$G$	Generator
SGAN	Original version of GAN
LSGAN	Least-Squares GAN
HingeGAN	GAN with Hinge loss
RpGAN	Relativistic paired GAN
RaGAN	Relativistic average GAN

---

RpD	Relativistic paired Discriminator
RaD	Relativistic average Discriminator
SpectralGAN	SGAN with spectral normalization
WGAN	Wasserstein GAN
WGAN-GP	Wasserstein GAN with Gradient Penalty
SGD	Stochastic Gradient Descent
SGAD	Stochastic Gradient Ascent/Descent
PACGAN	GAN with discriminator/critic packing
MVUE	Minimum-Variance Unbiased Estimator
CD	Critic's Difference
SyGAN	Symmetric GAN
RcGAN	Relativistic centered GANs
RalfGAN	simplified one-way version of RaGAN
CAS	Consistent Annealed Sampling
EDS	Expected Denoised Sample
MAP	maximum a posteriori probability
CIFAR-10	Dataset of images from 10 categories
CAT	Dataset of cat images
CelebA	Dataset of celebrity images
LSUN-church	Large dataset of churches
MNIST	Dataset of 60k digits
StackedMNIST	Dataset of three digits from MNIST
FFHQ	Dataset of human faces from Flickr
EM	Euler-Maruyama
FDP	Forward Diffusion Process
RDP	Reverse Diffusion Process
DDIM	Denoising Diffusion Implicit Models
NFE	Number of score Function Evaluations

# Notation

Set of real numbers	$\mathbb{R}$
For all $x$	$\forall x$
There exists a $x$	$\exists x$
If and only if	$\iff$
Optimal $\theta$ parameter	$\theta^*$
$x$ is a sample from the distribution $\mathbb{P}$	$x \sim \mathbb{P}$
$x$ converges to $y$	$x \rightarrow y$
$\mathbb{P}$ converges in distribution to $\mathbb{Q}$	$\mathbb{P} \xrightarrow{D} \mathbb{Q}$
Set containing $x_i, x_{i+1}, \dots, x_j$	$\{x_k\}_{k=i}^j$
Summation $x_i + x_{i+1} + \dots + x_j$	$\sum_{k=i}^j x_k$
Integral of $f(x)$ over $\mathcal{X}$	$\int_{\mathcal{X}} f(x) dx$
Expectation of $f(x)$	$\mathbb{E}_{x \sim \mathbb{P}} f(x)$
Neural network parameterized by $\theta$	$f_{\theta}(x)$
Minimum of $f$ from $\theta$	$\min_{\theta} f_{\theta}(x)$
Maximum of $f$ from $\theta$	$\max_{\theta} f_{\theta}(x)$
Parameters $\theta$ that minimizes $f_{\theta}$	$\theta^* = \arg \min_{\theta} f_{\theta}(x)$
Parameters $\theta$ that maximizes $f_{\theta}$	$\theta^* = \arg \max_{\theta} f_{\theta}(x)$
Minimum of $f$ from $C$	$\min_C f(C(x))$
Maximum of $f$ from $C$	$\max_C f(C(x))$
Function $C$ that minimizes $f$	$C^* = \arg \min_C f(C(x))$
Function $C$ that maximizes $f$	$C^* = \arg \max_C f(C(x))$
Supremum	sup
Infimum	inf
PDF of $x$	$p(x)$
PDF of $x$ conditional on $y$	$p(x y)$
PDF induced by $\theta$	$q_{\theta}(x)$
Distribution of (real) data	$\mathbb{P}$
Distribution of (fake) generated data	$\mathbb{Q}$ or $\mathbb{Q}_{\theta}$
Distribution of latent $z$ (in GAN)	$\mathbb{Z}$
Mixture distribution (mix of real and fake)	$\mathbb{M}$
Uniform distribution over $[0, 1]$	$\mathcal{U}(0, 1)$
Univariate Gaussian distribution	$\mathcal{N}(\mu, \sigma^2)$
Multivariate Gaussian distribution	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
Inverse CDF	$F^{-1}(u)$

---

Probability of $x$ at time $t$	$p_t(x) = p(x, t)$
Derivative of $x$ with respect to time	$\dot{x} = \frac{dx}{dt}$
Wiener process	$\mathbf{w}$
Wiener process (backward)	$\bar{\mathbf{w}}$
SDE with drift $f$ and diffusion $g$	$\dot{\mathbf{x}} = f(\mathbf{x}, t) + g(t)\dot{\mathbf{w}}$
SDE with drift $f$ and diffusion $g$	$d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)d\mathbf{w}$
Gradient of $f$ with respect to $x$	$\nabla_x f(x)$
Divergence between two distributions	$\mathcal{D}(\mathbb{P} \parallel \mathbb{Q}_\theta)$
Wasserstein Distance	$\mathcal{D}^W(\mathbb{P} \parallel \mathbb{Q}_\theta)$
Ra Divergence induced by $f$	$\mathcal{D}_f^{Ra}(\mathbb{P} \parallel \mathbb{Q}_\theta)$
Rp Divergence induced by $f$	$\mathcal{D}_f^{Rp}(\mathbb{P} \parallel \mathbb{Q}_\theta)$
Ralf Divergence induced by $f$	$\mathcal{D}_f^{Ralf}(\mathbb{P} \parallel \mathbb{Q}_\theta)$
Rc Divergence induced by $f$	$\mathcal{D}_f^{Rc}(\mathbb{P} \parallel \mathbb{Q}_\theta)$
Wasserstein Divergence	$\mathcal{D}^W(\mathbb{P} \parallel \mathbb{Q}_\theta)$
EDS	$H^*(\tilde{\mathbf{x}}, \sigma) \triangleq \mathbb{E}_{\mathbf{x} \tilde{\mathbf{x}}}[\mathbf{x}]$
vector or matrix transpose	$\mathbf{x}^T$ or $A^T$
trace of a matrix $A$	$\text{tr}(A)$
$\ell_1$ norm, i.e., $\sum_{i=1}^N  x_i $	$ x $
$\ell_p$ norm, i.e., $\sum_{i=1}^N  x_i ^p$	$\ x\ _p$
$\ell_\infty$ norm, i.e., $\max\{ x_1 ,  x_2 , \dots,  x_N \}$	$\ x\ _\infty$
Sigmoid function of $x$	$\text{sigmoid}(x) = \sigma(x)$
Mixed tolerance	$\delta(\mathbf{x}') = \max(\epsilon_{abs}, \epsilon_{rel} \mathbf{x}' )$
Scaled error (with $\ell_q$ norm)	$E_q = \left\  \frac{\mathbf{x}' - \mathbf{x}''}{\delta(\mathbf{x}', \mathbf{x}'_{prev})} \right\ _q$

# Acknowledgments

I want to express my gratitude to my family for their moral support: my wife Emy Gervais, my mother Louise Jolicoeur, my father Roger Martineau, and my cat Bear. My wife has given me invaluable support during difficult times. She taught me many invaluable lessons, such as doing a little bit every day rather than doing a lot sporadically. With her help and support, I was able to finish my Ph.D. without burning out.

I want to give my thanks to all my AI research colleagues who helped me along the way: Alex Lamb, Chris Finlay, Gauthier Gidel, Hugo Bérard, Ian Goodfellow, Ioannis Mitliagkas, Ke Li, Nicolas Loizou, Rémi Piché-Taillerfer, Rémi Tachet des Combes, Reyhane Askari Hemmat, Tal Kachman, Vikram Voleti, Yang Song, and so many others.

I also want to acknowledge the support I received from my colleagues at the Jewish General Hospital. Through my work there and the teachings of Ashley Wazana, I learned how to do research and write papers.

I also want to give my special thanks to Ian Goodfellow, who helped me push forward through his support and guidance.

*It's the little things that make a big difference.*

# 1

# Introduction

---

## 1.1 What are generative models?

The goal of generative modeling is to draw new samples from a probability distribution; we call this *sampling from a distribution*. Since the advent of deep learning, generative models have generated all sorts of complex high-dimensional data such as images, text, music, videos, and animations. As a consequence, generative models have the potential to revolutionize arts, science, and business. However, training generative models is hard and comes with many hurdles.

Generative models have existed for a long time. However, the ability to generate samples from complex high-dimensional data (e.g., images, text, music, etc.) was only developed in the last decade. This success can be attributed to the advent of deep learning (LeCun et al., 2015) and modern generative models such as: VAE (Kingma and Welling, 2013), GANs (Goodfellow et al., 2020), Flow-based models (Dinh et al., 2014, 2017), auto-regressive models (Germain et al., 2015), Implicit Maximum Likelihood (Li and Malik, 2018), Denoising Score Matching with Annealed Sampling (DSL-AS) (Song and Ermon, 2019), Diffusion Reverse Processes (Sohl-Dickstein et al., 2015), and many more.

Although generative models are more powerful than ever before, they remain limited in their capabilities. Firstly, generating high-quality samples is very difficult, and most approaches lead to blurry images (Goodfellow et al., 2020; Larsen et al., 2016) or produce mild artifacts (Brock et al., 2019; Karras et al., 2020). Some models can generate realistic images (Brock et al., 2019; Karras et al., 2019), audio (Oord et al., 2016), or text (Brown et al., 2020). However, these approaches still generate plenty of unrealistic samples, clearly not part of the true data distribution. Secondly, methods that generate higher quality samples tend to be less diverse than what we see in the true data distribution. Less common modes of the data are often ignored (e.g., generating only blue and red cats, when black cats

---

exist in the dataset); we call this phenomenon *mode collapse*, and it is pervasive in GANs (Salimans et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017).

In the simplest scenario, one may know the distribution that they want to sample from a priori. In this case, they can use classic and semi-modern sampling methods to generate new samples. Some of these techniques are: inverse transform sampling (Casella and Berger, 2002), rejection sampling (Casella and Berger, 2002), and Markov Chain Monte Carlo (MCMC) methods (Robert and Casella, 2011) such as Metropolis-Hastings (Metropolis et al., 1953), Gibbs (Geman and Geman, 1984), and Langevin Monte Carlo (Grenander and Miller, 1994). An alternative to sampling methods is to generate data in approximation by minimizing some distance between real data and fake (generated) data; we call such a distance a divergence. We discuss these fundamental techniques in Chapter 2.

Note that these techniques assume that we know the distribution of the data and can extract related functions (e.g., density function, gradient log-density function, cumulative distribution function, inverse cumulative distribution function, divergence). In practice, however, we generally do not know the data distribution. All we have is a dataset of samples from that distribution (e.g., a set of images, songs, or texts); this problem set corresponds to modern generative models.

Since we do not know the distribution of the data, we must estimate the essential functions required for sampling using our dataset. For example, we can still use LMC by estimating the density function or the gradient log-density function (score function). To generate data by minimizing a divergence without knowing the probability distributions, we can learn a distance between real and fake data from only the samples. We show this in Chapter 3.

---

## 1.2 Long-term vision

Given the difficulty of the task at hand, modern approaches still have many limitations regarding the quality and diversity of generated samples; this is where my work comes in. I seek to be on the frontier of generative models to help them reach their full potential.

My long-term objective is to construct a generative model that can generate movies or TV episodes from a script (possibly also generated by a generative model);

---

this is something we can envision happening in a few decades. I seek to improve generative models in order to pave the path towards this ultimate goal. I also want to provide better tools for artists to automatize certain aspects of their work or directly produce art. I am highly gratified that my approach called Relativistic GAN (Jolicoeur-Martineau, 2019) is now used (through ESRGAN (Wang et al., 2018)) to improve the graphics of old video games (by increasing the resolution of the textures), which makes them much more enjoyable to play (Vincent, 2019; Patrikspacek, 2019; Papadopoulos, 2019). I take great pride in knowing that the community uses my research.

---

### 1.3 Focus and goals of this thesis

Considering how quickly technologies change, my thesis focus on only the most promising emerging approaches; doing so ensures that I stay ahead of the curve and that the tools I create remain relevant for a longer time.

So far, one of the most successful approaches has been Generative Adversarial Networks (GANs) (Goodfellow et al., 2020). This approach consists of training a neural network (the generator) to fool a classifier (the discriminator) into thinking that its fake generated images are actually real. GANs can generate high-quality samples, such as photo-realistic images.

However, in recent years a new contender to GANs has arrived: score-based diffusion models (Song and Ermon, 2019, 2020; Ho et al., 2020); this new approach seems to perform better than GANs in terms of both quality and diversity. Score-based diffusion models consist in adding noise to your data samples through a Forward Diffusion Process (FDP) and then learning to reverse that process to go from noise to data.

Since GANs and score-based diffusion models seem to be some of the most promising approaches for continuous data generation (discrete data such as text need different tools), this research project focuses on these two approaches (until a new, better approach comes along).

My goals are to 1) improve generative models so that they reach their full potential (Objective 1: Improving), 2) understand these approaches better on a theoretical level (Objective 2: Theoretical understanding), and 3) generalize these



---

approaches beyond their original setting (Objective 3: Generalizing), allowing the discovery of new connections between different concepts/fields.

My first contribution is to propose using a *relativistic* discriminator, which estimates the probability that a given real data is more realistic than randomly sampled fake data. This new class of GAN loss functions is a generalization of Integral Probability Metrics GANs (Objective 3) with increased training stability (Objective 1). My second contribution is to take a more rigorous look at relativistic GANs and prove that they are proper statistical divergences (Objective 2). My third contribution is to devise an adversarial variant to score-based diffusion models (Objective 3), which leads to higher quality data (Objective 1). My fourth contribution is to significantly improve the speed of score-based diffusion models through a carefully devised Stochastic Differential Equation (SDE) solver (Objective 1).

---

## 1.4 Overview of the structure

In chapter 1, I explain what generative models are and why they matter. I describe my long-term goal of using generative models to produce art and entertainment (such as tv shows or movies). I mention that my thesis is focused on two very promising approaches for continuous data generation: Generative Adversarial Networks (GANs) and score-based diffusion models. I describe my thesis's three research objectives: 1) improve, 2) understand, and 3) generalize generative models. I state my contributions.

Chapter 2 is the background section.

In Section 2.2, I provide a non-exhaustive overview of the methods used for generating samples from a known continuous distribution. I introduce most of the classic sampling methods and a few modern techniques. I focus mainly on Langevin Monte Carlo (LMC) and divergence minimization, given their importance to score-based diffusion models and GANs.

In Section 2.3, I explain how to adapt the sampling methods presented in the previous chapter to function when we have no information on the distribution we want to sample from, except for having a finite dataset of samples from that distribution. First, I show how to estimate the score function (the gradient log-density function) needed to sample from the data distribution using LMC. Second,

---

I show how to estimate the divergence in divergence minimization with a critic (classifier) network to be able to use this method without knowing the probability density function. These two approaches form the basis of score-based diffusion models and GANs.

In Sections 2.4 and 2.5, I respectively discuss the two generative models I study in my thesis: score-based diffusion models and GANs.

The following Chapters consist of my four articles.

Finally, Chapter 11 concludes my thesis, discusses the impact of my research and what may come from the future.

---

## 1.5 Excluded work

To keep my thesis succinct and coherent, I removed some of my other contributions. See a list of my other work below:

1. First/Sole Author: GANs beyond divergence minimization (Objective 2) (Jolicœur-Martineau, 2018)
2. Co-author (I produced the experiments): Stochastic Hamiltonian Gradient Methods for Smooth Games (Objective 2) (Loizou et al., 2020)
3. First author: Connections between Support Vector Machines, Wasserstein distance and gradient-penalty GANs (Objective 2 and 3) (Jolicœur-Martineau and Mitliagkas, 2019)

# 2 Background

---

## 2.1 Neural networks

This section explains the basics of convolutional neural. I do not provide all the details but instead, show a somewhat high-level explanation of their functioning. For a more detailed explanation of convolutional neural networks and their optimization, please see [Goodfellow et al. \(2016\)](#).

### 2.1.1 Artificial Neural Networks

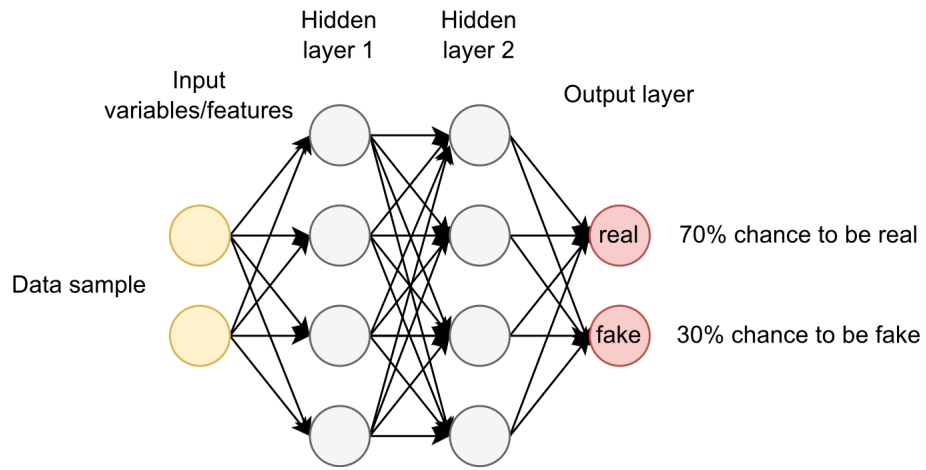
Artificial neural networks ([McCulloch and Pitts, 1943](#)) form the basis of most modern AI techniques and generative models. Neural networks are models that progressively transform an input data into an output through multiple layers of transformations.

A neural network has learnable parameters and the goal is to update the parameters over time so that the output matches the desired output (e.g., a cat image is transformed so that the output is as close as possible to a vector (cat=1, dog=0, car=0)).

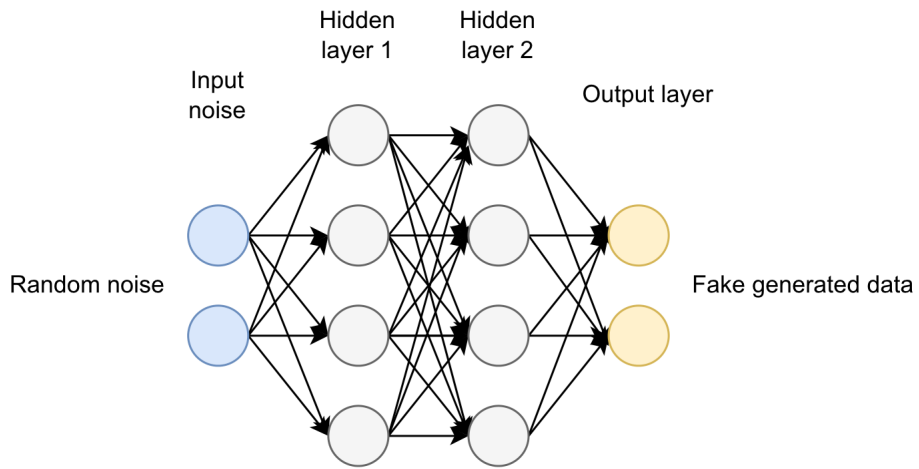
The fact that neural networks progressively transform the data over many layers is very powerful; the universal approximation theorem states that neural networks with at least one hidden layer and a large enough number of neurons can approximate any continuous function ([Cybenko, 1989](#)). In practice, we balance depth (number of layers) and width (number of neurons) for efficient processing.

There are four basic components in neural networks: 1) linear layers, 2) activation functions, and 3) normalization layers.

Linear layers are simple affine transformations of the data  $y = \beta x + c$  that serves to transform the data. When every node of the previous layer connects to the next layer, we call those linear layers *fully connected*. With simple tabular data (e.g., age, weight, height, hair color, salary, etc. of a person), we generally use fully



(a) Discriminator



(b) Generator

**Figure 2.1** – Neural networks used in Generative Adversarial Network

---

connected layers.

Activation functions are non-linear transformations of the data (e.g., ReLU  $y = \max(0, x)$ ). Adding non-linearities provides neural networks with the ability to represent any transformation from input to output, leading to the universal approximation theorem.

Normalization layers standardize the data, which serve to prevent variance explosion through the networks (Ioffe and Szegedy, 2015) and are especially useful for improving training stability. They are optional.

In Figure 2.3, we show a high-level illustration of the artificial neural networks used in Generative Adversarial Networks (GANs). The Discriminators take a data sample and transform it into class labels. Meanwhile, the Generator takes random noise and transforms it into fake generated data. More details on GANs will be shown in Section 2.5. Assume that every arrow represents a multiplication (linear transformation) and that every node contains an activation function and, optionally, a normalization layer.

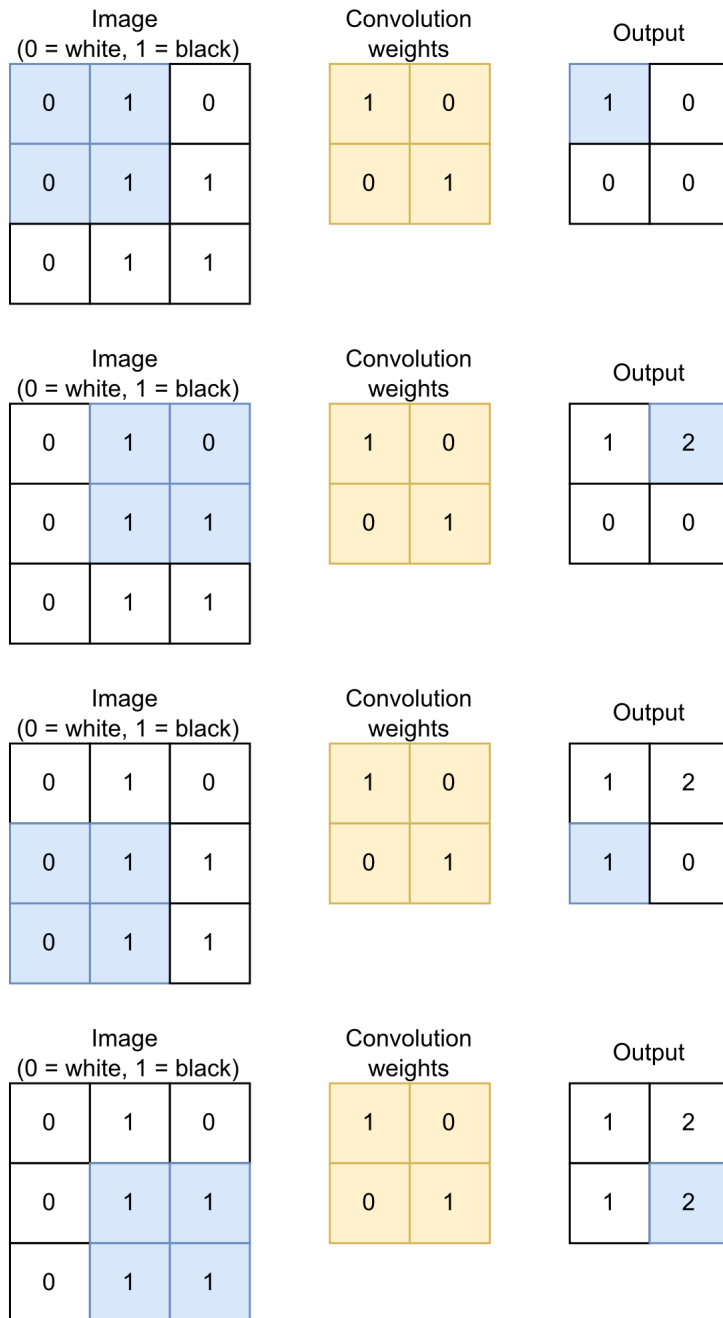
Although powerful, adapting neural networks to handle image data is non-trivial and is explained in the subsection below.

### 2.1.2 Convolutional Neural Networks (CNN)

Typically, we represent images as tensor objects of size  $C \times H \times W$  where  $C$  is the number of color channels (generally, we use three channels for RGB (Red, Blue, and Green) colors),  $H$  is the height, and  $W$  is the width. Convolutional Neural Networks (CNN) (Fukushima and Miyake, 1982) are a specific type of neural network which is particularly effective to process such tensor image data.

A CNN is a neural network where the linear transformation layers are replaced with 2D convolution filters. A 2D convolution is a weighted sum (similar to linear layers) over a small sliding window of  $k \times k$  weights. It is best understood through visualization; please see Figure 2.2.

There are multiple reasons why we use CNNs over fully-connected neural networks. Firstly, CNNs are closely inspired by the brain visual system Lindsay (2021). Secondly, fully-connected linear layers lead to an excessive number of parameters; for example, if our model has  $c$  nodes/channels in the next layer, and we have a  $3 \times 64 \times 64$  image, the number of parameters is  $3 * 64 * 64 * c$ . Meanwhile, a typical

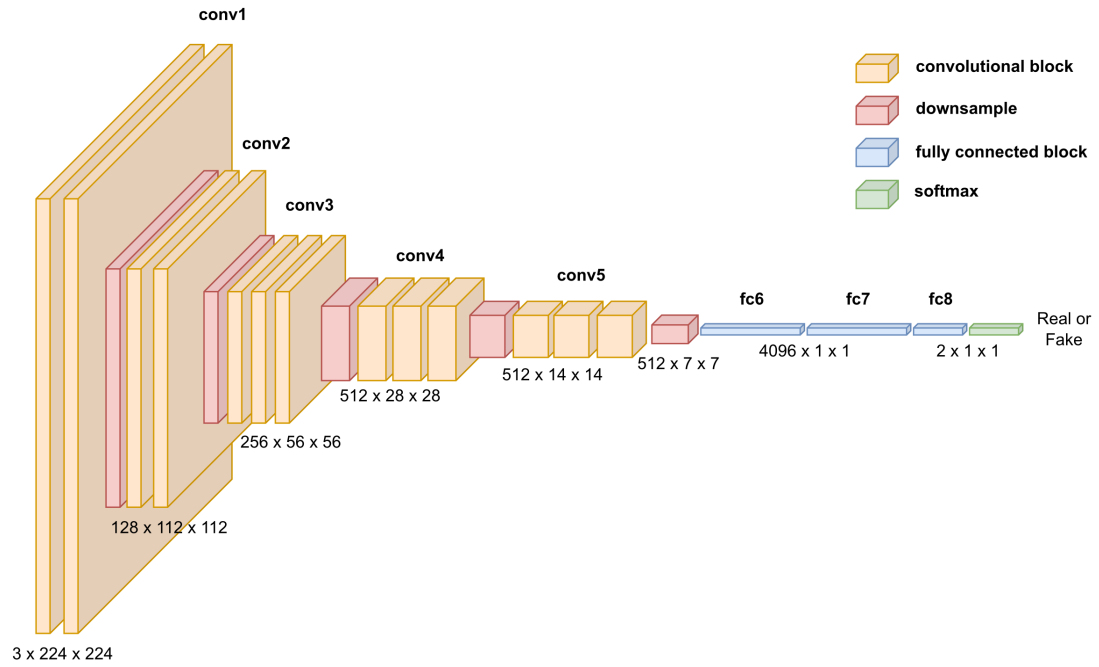


**Figure 2.2** –  $2 \times 2$  convolution over a small black-and-white  $3 \times 3$  image

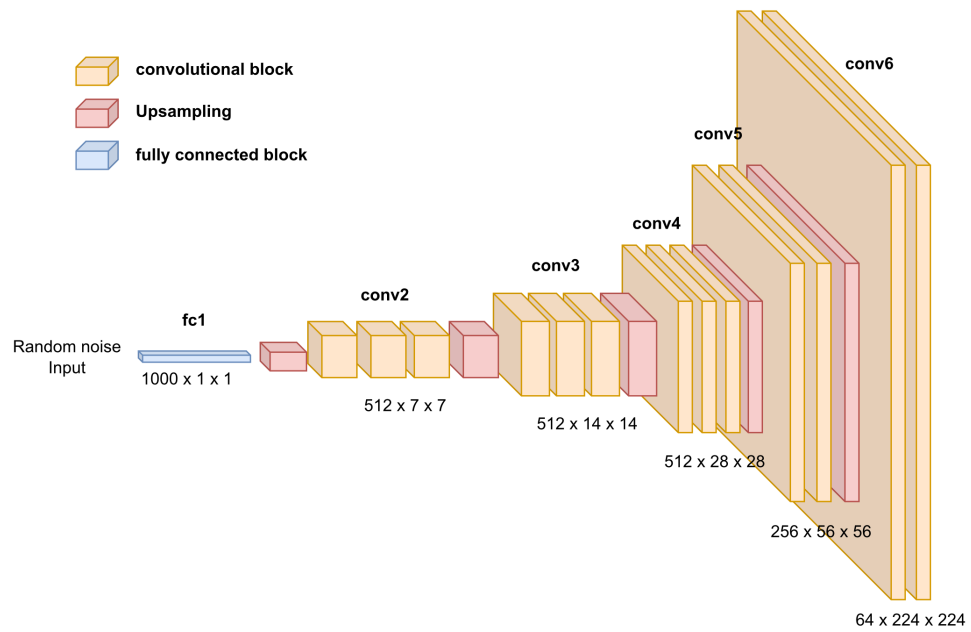
---

$3 \times 3$  convolution only requires  $3 * 3 * 3 * c$  parameters. Thirdly, convolutions have translation invariance, meaning that a cat found in the top-left of the image is still seen as a cat when found in the image's bottom-right.

A visualization of CNNs in the use case of GANs can be found in Figure 2.1. You can see that we process the image so that the resolution (height and width) decreases over time to transform the high-dimensional image into a low-dimensional vector. In practice, we find that CNNs process the images so that early layers find edges, middle layers find simple shapes (e.g., ears, mouths), and end layers find complex shapes (e.g., face, body) (Goodfellow et al., 2016).



(a) Discriminator



(b) Generator

**Figure 2.3** – Convolutional Neural Networks (CNNs) used in Generative Adversarial Network



---

## 2.2 Sampling from a known continuous distribution

Sampling from finite discrete distribution is trivial, but sampling from a continuous distribution is much harder. To sample from a continuous distribution, we generally need access to certain functions that pertain to the distribution (i.e., the density function, inverse cumulative function, or the score function). In this chapter I assume that we have access to all these functions. I will show how to generate samples using classic and MCMC methods. I will also briefly discuss the concept of divergence minimization as a way to generate samples in an approximate matter.

### 2.2.1 Classic sampling methods

#### 2.2.1.1 Sampling from a uniform distribution

The most basic distribution is the uniform distribution. To use any sampling method, we generally need to sample from a uniform distribution first. We can do so through a pseudorandom number generator (PRNG) (James, 1990), such as the highly popular Mersenne Twister (Matsumoto and Nishimura, 1998). A PRNG follows a recursive equation to generate a sequence of integer numbers (between 0 and  $K$ ). One can then sample from a standard uniform variable by taking these integers and dividing them by  $K$ .

#### 2.2.1.2 Inverse Transform Sampling

The standard way to sample from a continuous distribution  $\mathbb{P}$ , as taught in graduate mathematical statistic courses, is the Inverse Transform Sampling (ITS) (Casella and Berger, 2002). It works in the following way:

1. Sample  $u \sim \mathcal{U}(0, 1)$ ,
2.  $x = F^{-1}(u)$  is now a sample from  $\mathbb{P}$ ,

where  $F^{-1}$  is the inverse Cumulative Distribution Function (CDF).

Although very simple, this approach requires knowing the inverse CDF, which is extremely non-trivial to compute. ITS is thus only used for simple distributions for which we have extracted the inverse CDF.

---

### 2.2.1.3 Rejection Sampling

To be able to sample without knowing the CDF, a classic approach is rejection sampling (Casella and Berger, 2002). Let  $p$  be the density of  $\mathbb{P}$  and  $q$  be the density of a (proposal) distribution  $\mathbb{Q}$  that is easy to sample from. Choose a small constant  $M$  such that  $p(x) \leq Mq(x) \forall x$ ; then, rejection sampling works in the following way:

1. Sample  $u \sim \mathcal{U}(0, 1)$ ,
2. Sample  $x$  from a proposal distribution  $q$ ,
3. Accept  $x$  as a sample from  $\mathbb{P}$  if  $u < \frac{p(x)}{Mq(x)}$

This method generates samples from a simpler distribution and only a small proportion of the samples as being part of the desired distribution.  $M$  corresponds to the expected number of iterations needed to accept a sample. The higher the number of dimensions, the larger  $M$  needs to be. This method does not scale well to high-dimension and can require an absurd amount of iterations for a single accept.

## 2.2.2 Markov chain Monte Carlo (MCMC) methods

The methods presented above either required knowing the inverse CDF or are not scalable. This brings us to Markov chain Monte Carlo (MCMC) methods (Robert and Casella, 2011), which form a set of highly powerful tools for sample generations. Some of the most popular MCMC methods are Metropolis-Hastings (Metropolis et al., 1953), Gibbs (Geman and Geman, 1984), Langevin Monte Carlo (Grenander and Miller, 1994), and Hamiltonian (Duane et al., 1987) sampling. These approaches start from an initialization sample (generally pure Gaussian noise) and generate a new sample based on a random modification of the previous sample (i.e., following a Markov chain). In theory, after enough iterations, the samples should converge to real samples of the distribution.

To converge in distribution from a Markov Chain, we need a Markov process, which asymptotically reaches the unique stationary distribution with density  $p(x)$ . To ensure that this is the case, we need two conditions: detailed balance and ergodicity (Robert and Casella, 2013). Ergodicity requires that the Markov process must be aperiodic (i.e., one must not return to the sample state after a fixed amount of steps), and the expected number of steps needed to return to a previous state must be finite. Detailed balance is about ensuring that the process is reversible

---

and that  $p(x_i)q(x_j|x_i) = p(x_j)q(x_i|x_j)$ , where  $q(x_i|x_j)$  is the probability that the state  $x_i$  transitions to  $x_j$ . In practice, we generally have ergodicity but not detailed balance.

### 2.2.2.1 Metropolis-Hastings

To enforce detailed balanced, we can use Metropolis-Hastings (Metropolis et al., 1953). This approach consists of accepting or rejecting the next proposed sample. We accept the proposed sample ( $x'_{k+1} = x_{k+1}$ ) with probability:

$$\min\left(1, \frac{p(x_{k+1})q(x_k|x_{k+1})}{p(x_k)q(x_{k+1}|x_k)}\right); \quad (2.1)$$

otherwise, we reject the proposed sample ( $x'_{k+1} = x_k$ ).

The accept/reject step ensures that we have detailed balance, and thus the chain  $\{x'_k\}_{k=1}^K$  produced by the process has  $p(x)$  as the density of its unique stationary distribution.

### 2.2.2.2 Random Walk

A classic choice of proposed sample for the Markov Chain is  $x_{k+1} \sim \mathcal{N}(x_k, \sigma^2)$ , where  $\sigma$  is a constant; this corresponds to a random Gaussian walk. The transition probability becomes  $q(x'|x) = \mathcal{N}(x' - x, \sigma^2)$ . Although the Random Walk proposal works, it could lead to many rejections (especially in high-dimensions) since we move aimlessly in any direction.

### 2.2.2.3 Langevin Monte Carlo (Langevin sampling)

Instead of the random walk proposal distribution, we would like a proposal distribution that brings us closer to the distribution and produces samples that are more likely to be accepted. To achieve this goal, Langevin Monte Carlo (LMC) proposes a new sample by following the gradient-log-density (i.e., a vector field which points toward samples of increasing probabilities); this corresponds to following the Langevin dynamics.

The (over-damped) Langevin dynamics is defined as the following Stochastic

---

Differential Equation (SDE):

$$\dot{x} = \nabla_x \log p(x) + \sqrt{2}\dot{w},$$

where  $w$  is the standard Brownian motion. To simulate from Langevin dynamics, we need to discretize the SDE; we can do by assuming a fixed step size  $\lambda$  and using the Euler–Maruyama method (Kloeden and Platen, 2013):

$$x_{k+1} \leftarrow x_k + \lambda \nabla_x \log p(x_k) + \sqrt{2\lambda}\epsilon, \quad (2.2)$$

where  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ . The transition probability becomes  $q(x'|x) = \mathcal{N}(x' - x - \lambda \nabla_x \log p(x), 2\lambda \mathbf{I})$ .

We see that this amount to Gradient Ascent of the log-density with some added Gaussian noise. We know that the mode (most probable value of the distribution) is reached at the maximum of  $p(x)$ , which is when  $\nabla_x \log p(x_k) = 0$ . Thus, following Equation 2.2, but setting  $\epsilon = 0$  would bring the sample toward the mode. This means that we can intuitively think of Langevin dynamics as going toward the mode of the distribution while adding some noise to ensure that we explore areas around the mode. This ensures a much greater acceptance ratio.

Due to the Metropolis-Hastings accept/reject step, Langevin Monte Carlo requires that one knows both the density  $p(x)$  and the score function (i.e., the gradient log-density  $\nabla_x \log p(x)$ ). Below, I show that, by decreasing the step size, we can effectively remove the need for the accept/reject step (and thus, for the density function).

Langevin dynamics with decreasing step sizes consists in the following iterative process:

$$x_{k+1} \leftarrow x_k + \lambda_k \nabla_x \log p(x_k) + \sqrt{2\lambda_k}\epsilon, \quad (2.3)$$

where  $0 \leq \lambda_{k+1} \leq \lambda_k$ ,  $\sum_{k=1}^{\infty} \lambda_k = 0$ ,  $\sum_{k=1}^{\infty} \lambda_k^2 = \infty$ .

As can be seen, we now decrease the step size over time toward zero. It has been shown that samples at the end of this process converge in distribution to  $p(x)$  (Welling and Teh, 2011). Notably, the probability of acceptance in the Metropolis-Hastings step goes to 1 as the step size goes to 0; this means that we are almost always guaranteed to accept our next proposal sample when the step size becomes small. Due to this, in practice, the algorithm works well without any Metropolis-

---

Hastings accept/reject step (Welling and Teh, 2011).

Thus, we can use Langevin dynamics with decreasing step sizes without having access to  $p(x)$ ; all we need is to use this approach is the score function. In the next chapter, I show how we can estimate the score function using a score network. This will allow us to sample from an unknown distribution using only samples from a dataset.

### 2.2.3 Divergence minimization

Although not a sampling method in the traditional sense, one can train a Generator model to produce samples from the distribution by minimizing a divergence. We explain how this works below.

Assume we have a neural network  $G_\theta$  (the Generator), which takes as input a set of random variables  $z$  (generally standard uniform or standard Gaussian noise) and produces an output of the same shape/dimension as the data from  $\mathbb{P}$ . The input noise serves to produce diversity so that there is an infinite amount of possible generated outputs. The goal of the Generator is to produce samples of the data distribution. We can achieve this goal by training  $G_\theta$  to minimize the distance between the distribution of its fake generated samples ( $\mathbb{Q}_\theta$ ) and the distribution of real data ( $\mathbb{P}$ ). We call distances between distributions *statistical divergences*. We can formalize this idea as:

$$\min_{\theta} \mathcal{D}(\mathbb{P} \parallel \mathbb{Q}_\theta) \quad (2.4)$$

At the optimum, the divergence is zero, and samples from the Generator are effectively samples from the true data distribution.

To be able to minimize a divergence, we need have a divergence  $\mathcal{D}$ , but most divergences are highly computationally demanding or require knowing the probability density functions of real data  $p(x)$  and fake (generated) data  $q_\theta(x)$ . As an example, let's focus on a popular class of divergences called  $f$ -divergences (Rényi et al., 1961):

$$\mathcal{D}_f(\mathbb{P} \parallel \mathbb{Q}_\theta) = \int_{\mathcal{X}} q_\theta(x) f\left(\frac{p(x)}{q_\theta(x)}\right) dx, \quad (2.5)$$

where  $f$  is a convex function and  $f(1) = 0$ .

If we do not have  $p(x)$  and  $q_\theta(x)$ , we cannot solve this equation. However, there are ways around this. As we will show in the next chapter, one can learn

---

a divergence only using real and fake samples (without knowing  $p(x)$  or  $q_\theta(x)$ ). Through a learned divergence, we will be able to generate samples from an unknown distribution using only samples from a dataset.

---

## 2.3 Sampling from an unknown continuous distribution

In the previous section, I introduced how to use Langevin Monte Carlo (LMC) to sample from a distribution using only the score function (gradient log-density). I also showed how to learn a Generator to sample from a distribution by minimizing a divergence (distance between probability distributions), which requires knowing the density function. However, in practice, we generally do not possess the score function nor the density function. This chapter shows how to estimate the score function and a divergence using samples from a dataset to allow the use of LMC and divergence minimization when the distribution is unknown.

Note that there are many other ways of generating samples from an unknown continuous distribution which do not follow from the classic sampling methods introduced in the previous chapter; we mention them briefly in Chapter 1. For a more complete literature of the subject, I recommend the Standard course on Deep Generative Models (CS236), whose notes and slides are freely available online<sup>1</sup>.

### 2.3.1 Score Matching

#### 2.3.1.1 Traditional score matching

The most straightforward way to estimate the score function of a distribution  $\mathbb{P}$  through a score network  $s_\theta(x)$  is to solve the following problem:

$$\min_{\theta} \frac{1}{2} \mathbb{E}_{x \sim \mathbb{P}} [\|s_\theta(x) - \nabla_x \log p(x)\|_2^2] \quad (2.6)$$

This consists of approximating the score function at every sample from a dataset. However, we do not know  $\nabla_x \log p(x)$ , which makes this loss function completely unusable. However, this equation can be shown to be equivalent to the following minimization problem (Vincent, 2011):

$$\min_{\theta} \mathbb{E}_{x \sim \mathbb{P}} \left[ \text{tr}(\nabla_x s_\theta(x)) + \frac{1}{2} \|s_\theta(x)\|_2^2 \right] \quad (2.7)$$

---

1. <https://deepgenerativemodels.github.io>

---

Astonishingly, this equation does not depend on the actual score function; it can be used to train the score network using only samples from the dataset. However, due to the  $\text{tr}(\nabla_x s(x))$  term, this approach does not scale well to high-dimensional data. We show another solution below.

### 2.3.1.2 Sliced score matching

A more scalable alternative to the traditional score matching approach is sliced score matching (Song et al., 2020):

$$\min_{\theta} \mathbb{E}_{v \sim \mathcal{N}(0, I)} \mathbb{E}_{x \sim \mathbb{P}} \left[ v^T \nabla_x s_{\theta}(x) v + \frac{1}{2} \|s_{\theta}(x)\|_2^2 \right] \quad (2.8)$$

Although solving 2.7 or 2.8 allows us to estimate the score function, both of these approaches are slow. If the data has a large dimension  $K$  (e.g., for a small 3x32x32 image,  $K = 3072$ ), the score network has the same dimension, and the gradient of the score network has dimension  $K^2$  (e.g., for a small 3x32x32 image,  $K^2 = 9437184$  dimensions). Thus, the fact that the sliced score matching approach still relies on the score network’s gradient and does matrix multiplication is a problem.

Besides, both previous approaches have one critical problem: any region such that  $p(x) \approx 0$ , will not contain samples from the dataset and will lead to regions where the score network is undefined. In the case of a Dirac distribution (which corresponds to a dataset of  $K$  samples where each sample is assigned a probability  $1/k$ ), we see that we only learn the score function at data samples and nowhere else. If our goal is to use the score network as a gradient field, which points where one should go, this will not do as we may reach such a low-density area (not containing any training data sample) after any Langevin step. It puts us at a high risk of getting stuck or going in random directions. The big problem being that with complex high-dimensional data such as images, most of the space is filled with low-density/empty areas. There is, thankfully, a solution to this problem, as shown next.



---

### 2.3.1.3 Denoising score matching

One way to solve the previous issue is to use Denoising Score Matching (DSM) (Vincent, 2011):

$$\min_s \frac{1}{2} \mathbb{E}_{p(\tilde{x}, x, \sigma)} [\lambda(\sigma) \|s(\tilde{x}, \sigma) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x)\|_2^2] \quad (2.9)$$

where  $\lambda(\sigma)$  is a weighting function,  $p(\tilde{x}, x, \sigma) = q_\sigma(\tilde{x}|x)p(x)p(\sigma)$ ,  $p(x)$  is the density of the data distribution,  $p(\sigma)$  is the density of the level of noise, and  $q_\sigma(\tilde{x}|x)$  is the perturbation kernel. This approach consists in adding some noise to the data samples so that we learn the score function over varying levels of noise.

For an optimal score network  $s^*(x, \sigma)$ , it can be shown (Vincent, 2011) that

$$\lim_{\sigma \rightarrow 0} s(x, \sigma) = \nabla_x \log p(x).$$

This means that we obtain the score function when the noise is zero.

In the approach by Song and Ermon (2019), they use  $q_\sigma(\tilde{x}|x) = \mathcal{N}(0, \sigma^2 \mathbf{I})$ ,  $\lambda(\sigma) = \sigma^2$ , and  $p(\sigma)$  is the uniform distribution over a set  $\{\sigma_i\}_{i=1}^L$  corresponding to different levels of noise; in practice, this set is defined as a geometric progression between  $\sigma_1$  and  $\sigma_L$  (with  $L$  chosen according to some computational budget):

$$\{\sigma_i\}_{i=1}^L = \left\{ \gamma^i \sigma_1 \mid i \in \{0, \dots, L-1\}, \gamma \triangleq \frac{\sigma_2}{\sigma_1} = \dots = \left( \frac{\sigma_L}{\sigma_1} \right)^{\frac{1}{L-1}} < 1 \right\}. \quad (2.10)$$

With such Gaussian noise, the DSM loss function can then be reduced to:

$$\min_s \frac{1}{2} \mathbb{E}_{p(\tilde{x}, x, \sigma)} \left[ \left\| \sigma s(\tilde{x}, \sigma) + \frac{\tilde{x} - x}{\sigma} \right\|_2^2 \right] \quad (2.11)$$

The optimal score network for such a loss estimates the score function at various levels of Gaussian noise corruption. At high noise levels, the score network landscape is very smooth with support on the whole space. Meanwhile, at low noise levels, the score network is sharp and concentrated close to real data. Thus, rather than using  $s(x, 0)$ , we could, in theory, use  $s(x, \sigma)$  with a  $\sigma$  small enough to be close to the truth, but large enough so that most of the support is covered (and we do not reach an undefined region after a Langevin step). Rather than use a single  $\sigma$ ,

---

Song and Ermon (2019) propose starting with  $s(x, \sigma)$  where  $\sigma$  is large and slowly decreasing  $\sigma$  until it is close to zero; this effectively corresponds to using Langevin dynamics with decreasing step sizes and a noisy score function. We discuss this idea in more detail in Chapter 2.4 as it forms the basis of score-based diffusion models.

### 2.3.2 Divergence estimation

As mentioned in the previous chapter, most divergences, such as  $f$ -divergences, require knowing the density function of real and fake (generated) data. For  $f$ -divergences, a solution is to make use of their dual form (Nguyen et al., 2010) which states that:

$$\mathcal{D}_f(\mathbb{P} \parallel \mathbb{Q}) \geq \max_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} [C(x)] - \mathbb{E}_{y \sim \mathbb{Q}} [f^*(C(x))], \quad (2.12)$$

where  $f^*(t) = \sup_{u \in \text{domain}_f} \{ut - f(u)\}$  is the Fenchel conjugate (Hiriart-Urruty and Lemaréchal, 2012) of  $f$ .

It can be shown that the inequality ( $\geq$ ) of Equation 2.12 becomes an equality ( $=$ ) under certain mild conditions (Nowozin et al., 2016). Thus, we can estimate  $f$ -divergences by solving a maximization problem.

$f$ -divergences form a very broad set of divergences. For example,  $f$ -divergences encompass Kullback-Leibler (KL) (Kullback, 1997) and Jensen-Shannon (JSD) (Lin, 1991) divergences, two very popular choices of  $f$ -divergences. However, other popular divergences, such as the Wasserstein distance (Vaserstein, 1969), are not  $f$ -divergences; they are instead Integral Probability Metrics (IPMs). IPMs form the following set of divergences:

$$\text{IPM}_{\mathcal{F}}(\mathbb{P} \parallel \mathbb{Q}) = \sup_{\substack{C: \mathcal{X} \rightarrow \mathbb{R} \\ C \in \mathcal{F}}} \mathbb{E}_{x \sim \mathbb{P}} [C(x)] - \mathbb{E}_{y \sim \mathbb{Q}} [C(x)], \quad (2.13)$$

where  $\mathcal{F}$  is a class of functions (e.g., 1-Lipschitz functions, convex functions, bounded functions, functions with a bounded gradient norm). The class of function serves to ensure that the supremum exists, and it is not infinity (which would be the case if we optimized this objective over any function). Very popular divergences such as the Wasserstein distance (Vaserstein, 1969), Total variation distance, and

---

Maximum mean discrepancy (Gretton et al., 2007) are IPMs. Most divergences used in the literature are  $f$ -divergences or IPMs.

We see that IPMs correspond to a maximization problem, just as the dual form of  $f$ -divergences. In practice, we parametrize the critic as a neural network  $C_\theta$  and we maximize its objective function over the parameters  $\theta$ . Thus, the problem of minimizing an  $f$ -divergence or IPM can be represented as:

$$\min_{\theta} \mathcal{D}(\mathbb{P} \parallel \mathbb{Q}_\theta) = \min_{\theta} \max_{\phi} \mathcal{D}_\phi(\mathbb{P} \parallel \mathbb{Q}_\theta), \quad (2.14)$$

where  $\mathcal{D}_\phi$  is an objective function that depends on a critic and leads to a divergence when we choose the critic that maximize it. The critic's objective function serves to estimate a divergence and the goal of the generator is to minimize that divergence. This formulation has the benefit of not depending on the density function; we can thus estimate a divergence using only real samples from a dataset and fake samples.

In Chapter 2.5, we will show that divergence minimization, as represented in equation 2.13, forms the basis of GANs. In addition, we will explain how to solve the min-max problem given that it is highly non-trivial to solve.

---

## 2.4 Score-based diffusion models

Yang Song developed Denoising Score Matching with Annealed Sampling (DSM-AS) (Song and Ermon, 2019) during his Ph.D. at Stanford. This approach is one of the initial iterations of what we now know as score-based diffusion models. The idea behind it is to estimate a noisy version of the gradient log-density using a score (neural) network (i.e., denoising score matching) and then generate samples from the distribution through Annealed Langevin sampling (Grenander and Miller, 1994) by using the score network.

The combination of denoising score matching (DSM) with Langevin sampling makes this approach shine as the results obtained were outstanding at the time it was published. From my perspective and one of many others, this approach appeared to be entirely novel and groundbreaking. In reality, this approach was one drop in a large ocean of earlier work on diffusion processes and denoising auto-encoders (Bengio et al., 2013; Sohl-Dickstein et al., 2015; Alain et al., 2016; Goyal et al., 2017). Nevertheless, what really distinguished DSM-AS from earlier work was how exceptionally well it performed (competitive with SOTA methods) and scaled to high-dimensional data.

Given the impressive start of DSM-AS, I started studying DSM-AS and exploring ways to improve it. As time went by, this approach started to show amazing results in image generation (Song and Ermon, 2020; Ho et al., 2020; Song et al., 2021; Ho et al., 2021; Vahdat et al., 2021). As an example, the paper by Ho et al. (2020) came out a year after the original paper by Song and Ermon (2019) and obtained results on CIFAR-10 (Krizhevsky and Hinton, 2009) better than any SOTA generative model in terms of image quality/diversity (as determined by the Fréchet Inception Distance (Heusel et al., 2017)). I show examples of images generated by this variant and the improved version of DSM-AS (Song and Ermon, 2020) in 2.4 and 2.5 respectively.

In this section, I first introduce the original DSM-AS approach by Song and Ermon (2019), which uses Langevin Monte Carlo (LMC) while using a score network conditional on the amount of Gaussian noise and decreasing the level of noise. Then, I introduce a more general framework that forms the basis of modern score-based diffusion models. This approach works in the following way: define a forward stochastic process which slowly corrupt real data into Gaussian noise (from  $t = 0$

---

to  $t = T$ ), use Denoising Score Matching (DSM) to estimate the score function at any step  $t$ , and reverse the process (from  $t = T$  to  $t = 0$ ) (i.e., transforming Gaussian noise into real data). Finally, I discuss the current limitations/problems associated with this class of techniques and briefly discuss current and possible future solutions.

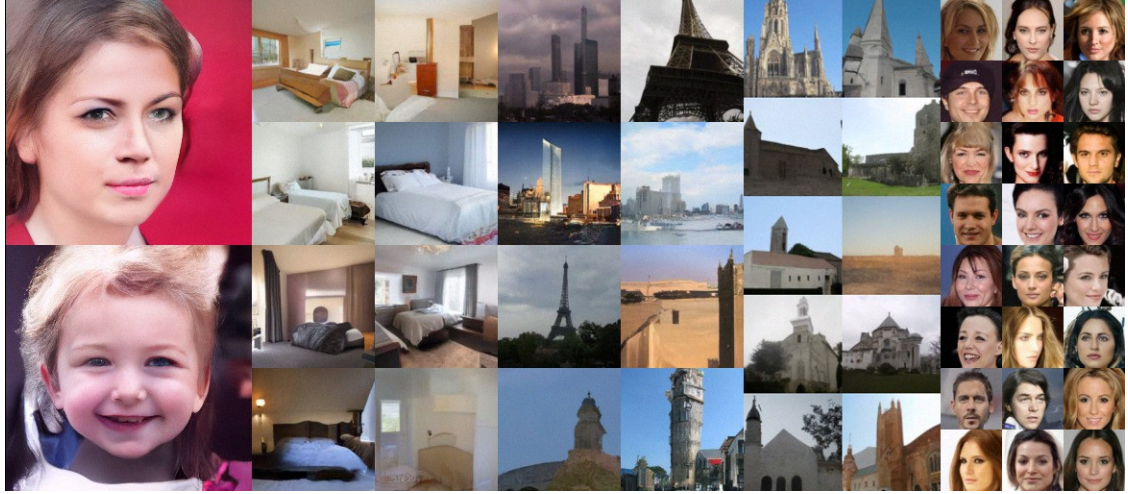
### 2.4.1 Denoising Score Matching with Annealed Sampling (DSM-AS)



**Figure 2.4** – Samples of humans generated through denoising diffusion models (Ho et al., 2020)

#### 2.4.1.1 Sampling from the distribution induced by the score function

As shown in section 2.3.1.3, we can estimate a noisy version of the score function using a conditional score network  $s(x, \sigma)$  which depends on the level of Gaussian noise. Although  $s(x, \sigma)$  converges to the true score function when  $\sigma$  is small, at that level of noise, we still have the issue of low-density areas of the space not being well-defined. Since we cannot explore the full support of the distribution using a small  $\sigma$ , Song and Ermon (2019) suggested starting the LMC with a large  $\sigma$ , and then slowly decrease  $\sigma$  over time. They call this approach Annealed Langevin Sampling (ALS) and it bears strong similarity to Langevin with decreasing step



**Figure 2.5** – Samples of humans, beds, towers, and churches generated using an improved version of denoising score matching (Song and Ermon, 2020)

sizes (Equation 2.3) except that the score network changes over time (instead of being a fixed score function). ALS is defined formally in Algorithm 1.

---

**Algorithm 1** Annealed Langevin Sampling

---

**Require:**  $s_\theta, \{\sigma_i\}_{i=1}^L, \epsilon, n_\sigma$ .

- 1: Initialize  $\mathbf{x}$
- 2: **for**  $i \leftarrow 1$  to  $L$  **do**
- 3:      $\alpha_i \leftarrow \epsilon \sigma_i^2 / \sigma_L^2$
- 4:     **for**  $t \leftarrow 1$  to  $n_\sigma$  **do**
- 5:         Draw  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
- 6:          $\mathbf{x} \leftarrow \mathbf{x} + \alpha_i s_\theta(\mathbf{x}, \sigma_i) + \sqrt{2\alpha_i} \mathbf{z}$

**return**  $\mathbf{x}$

---

The inner loop of ALS consists of Langevin sampling with a fixed step size. Thus, if the number of Langevin steps  $L$  is large enough, we should converge in distribution to the distribution implied by  $s(x, \sigma)$ . The outer loop of ALS simultaneously reduces the step size, and the level of noise  $\sigma$  in the score network. Thus, we start by sampling from the distribution implied by the noisy score function and decreases  $\sigma$  until we are approximately sampling from the actual score function. This allows the process to explore better the full support of the distribution (mode coverage) while converging to the distribution. With only one small  $\sigma$ , the process

would not be able to cover the distribution.

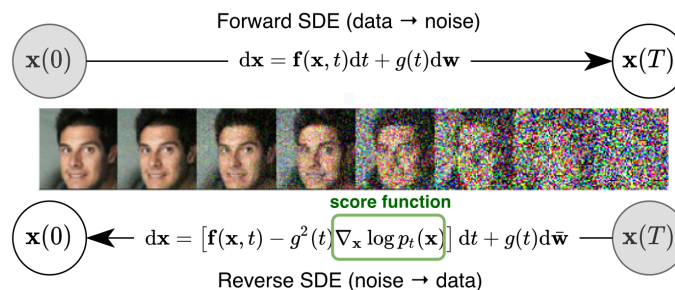
Song and Ermon (2020) found that mode coverage was poor unless starting from a very large  $\sigma$ . They recommend using the maximum  $L_2$  distance between two real data samples as the starting  $\sigma_0$ ; doing so ensures that one can quickly move away from one mode of the distribution to another and thus sample from the full range of the distribution. In addition, they recommend having the set of noise levels  $\{\sigma_i\}_{i=1}^L$  follow a geometric progression (i.e.,  $\sigma_{i+1} = \gamma\sigma_i$ , where  $0 < \gamma < 1$ ), choosing  $L$  as large as possible, and choosing the smallest noise level to not be too small; they suggests  $\sigma_L = .01$ .

This summarizes the basic idea of DSM-AS, but a few improvements and generalizations of this approach have been made since then. I discuss them below.

## 2.4.2 Generalizing DSM-AS by reversing a process from data to noise (modern score-based diffusion models)

Since the paper on DSM-AS, a different but very similar approach based on Denoising Diffusion Processes (DDPs) has emerged (Ho et al., 2020). DDPs are motivated very differently, and yet, they also use Denoising Score Matching and a sampling process very similar to LMC. In this section, we explain the general framework that leads to DSM-AS and DDPs as special cases (Song et al., 2021).

As we will show, the framework consists of defining a Forward Diffusion Process (FDP) from real data to Gaussian noise. Then, one can generate samples from the data distribution by going through the Reverse process from Gaussian noise to real data. See Figure 2.6 for a visual representation of the Forward and Reverse processes.



**Figure 2.6** – Forward and Negative diffusion processes. We can define a Forward process going from data to noise and reverse this process to go from noise to data. However, to reverse the process, we need to know the score function at every step.

---

The main difference between DSM-AS and DDPs is the choice of perturbation kernel  $q(\tilde{x}|x)$ , which underlie an assumed forward degenerative process (i.e., a process transforming real data samples into Gaussian noise). Let  $t \in [0, \dots, T]$  and denote the sample  $x$  at time  $t$  as  $x_t$ . We start from real data at  $x_0$  and choose a perturbation  $q(x_t|x_0)$  so that  $x_0$  only has a negligible influence on  $x_T$ . Thus, at the end of the forward process,  $x_T$  effectively does not depend on  $x_0$  and can be sampled independently (generally as Gaussian noise).

In DSM-AS, we let the perturbation kernel be  $q(x_t|x_0) = \mathcal{N}(x_0, \sigma_t^2 \mathbf{I})$ . This corresponds to the process resulting from the following discrete Markov Chain  $x_{t+1}|x_t$ :

$$x_{t+1} \leftarrow x_t + \sqrt{\sigma_{t+1} - \sigma_t} z_t,$$

where  $z_t \sim \mathcal{N}(0, \mathbf{I})$ . In the continuous-time limit, replacing  $\{\sigma_t\}_{t=1}^T$  by a function  $\sigma(t)$  over  $t \in [0, 1]$ , we obtain the following FDP (Song et al., 2021):

$$\dot{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} \dot{w}. \quad (2.15)$$

We call this the Variance Exploding (VE) Forward process. Importantly, if  $\sigma_T$  is large, we have approximately that  $x_T \sim \mathcal{N}(0, \sigma_T^2 \mathbf{I})$  since the impact of  $x_0$  is minuscule compared to the added noise. Thus,  $x_T$  can be sampled independently from  $x_0$ .

In DDP, the authors use the other way around; they first define the discrete Markov Chain  $x_{t+1}|x_t$  as:

$$x_{t+1} \leftarrow \sqrt{1 - \beta_t} x_t + \sqrt{\beta_t} z_t,$$

where  $\{\beta_t\}_{t=1}^T$  is a set of parameters between 0 and 1. In the continuous-time limit, replacing  $\{\beta_t\}_{t=1}^T$  by a function  $\beta(t)$  over  $t \in [0, 1]$ , we obtain the following FDP: (Song et al., 2021):

$$\dot{x} = -\frac{1}{2} \beta(t) x + \sqrt{\beta(t)} \dot{w}. \quad (2.16)$$

We call this the Variance Preserving (VP) Forward process. The perturbation kernel can be found to be  $q(x_t|x_0) = \sqrt{\prod_{t=1}^T [1 - \beta_t]} x_0 + \sqrt{\prod_{t=1}^T [\beta_t]} z_t$ . Importantly, we see that  $\prod_{t=1}^T [1 - \beta_t] \rightarrow 0$  as  $T \rightarrow \infty$ , which means that  $x_T \sim \mathcal{N}(0, \mathbf{I})$ . Thus,  $x_T$  can be sampled independently from  $x_0$ .



---

As we will show, we create the Forward process so that we may be able to reverse this process in order to generate samples starting from Gaussian noise: ( $\mathcal{N}(0, \sigma_T \mathbf{I})$  with VE and  $\mathcal{N}(0, \mathbf{I})$  with VP).

Since there is a Forward process from real data to Gaussian noise, there could also be a Reverse process from Gaussian noise to real data. In fact, [Anderson \(1982\)](#) showed that every Forward diffusion process of the following form:

$$\dot{x} = f(x, t) + g(t)\dot{w} \quad (2.17)$$

can be reversed using the Reverse Diffusion Process (RDP):

$$\dot{x} = f(x, t) - g(t)^2 \nabla_x \log p_t(x) + g(t)\dot{w}_r, \quad (2.18)$$

where  $\tilde{w}_r$  is the standard Brownian motion in reverse time, and  $p_t$  is the time-conditional distribution of the samples at time  $t$ . Both VE and VP Forward processes, Equations 2.15 and 2.16 respectively, have the the form of Equation 2.17; thus, they can be reversed using Equation 2.18.

Given that we do not possess the time-conditional score functions, we can estimate them using denoising score matching in the following way:

$$\min_{\theta} \frac{1}{2} \mathbb{E}_{p(x_t, x_0, t)} [\lambda(t) \|s_{\theta}(x_t, t) - \nabla_{x_t} \log q(x_t|x_0)\|_2^2], \quad (2.19)$$

where  $p(x_t, x_0, t) = q(x_t|x_0)p(x_0)p(t)$ , where  $p(t) \sim \mathcal{U}(0, T)$ .

As can be seen, this is simply a different way of stating 2.9. It can be shown that for an optimal score network, we have  $s^*(x, t) = \nabla_x \log p_t(x)$  for all  $t$ .

Thus, after training a time-conditional score network, we can sample from the data distribution by starting from Gaussian noise and solving the RDP 2.18 using any SDE solver.

[Song et al. \(2021\)](#) discretized the RDP of VE and VP in the same way as their respective Forward Process. They found that many iterations were needed to converge well and, thus, they proposed to augment their algorithm with Langevin Monte Carlo (LMC). They use the reverse step in an outer loop and use LMC in an inner loop to act as a correction-step. They call this approach PC sampling, and it is represented in Algorithm 2 for the VE process and in Algorithm 3 for the VP process.

---

Algorithm 2 PC sampling (VE SDE)	Algorithm 3 PC sampling (VP SDE)
1: $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$	1: $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: <b>for</b> $i = N - 1$ <b>to</b> $0$ <b>do</b>	2: <b>for</b> $i = N - 1$ <b>to</b> $0$ <b>do</b>
3: $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2) \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, \sigma_{i+1})$	3: $\mathbf{x}'_i \leftarrow (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i + 1)$
4: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$	4: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2} \mathbf{z}$	5: $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}} \mathbf{z}$
6: <b>for</b> $j = 1$ <b>to</b> $M$ <b>do</b>	6: <b>for</b> $j = 1$ <b>to</b> $M$ <b>do</b>
7: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$	7: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
8: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i} \mathbf{z}$	8: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i} \mathbf{z}$
9: <b>return</b> $\mathbf{x}_0$	9: <b>return</b> $\mathbf{x}_0$

We see that the PC sampling resembles the Annealing sampling (Algorithm 1) except that they add the reverse step in the outer loop. In fact, if we take small enough steps so that  $\sigma_{i+1} \approx \sigma_i$ , the PC sampling for the VE process (Algorithm 2) is exactly equivalent to the Annealing Sampling.

Importantly, it can be shown that the RDP 2.18 has an equivalent ODE formulation. There exist an ODE such that the marginal distribution  $p_t(x)$  at any time  $t$  is the same as the one produced by the reverse SDE (Song et al., 2021). This ODE can be shown to be:

$$\dot{x} = f(x, t) - \frac{1}{2} g(t)^2 \nabla_x \log p_t(x). \quad (2.20)$$

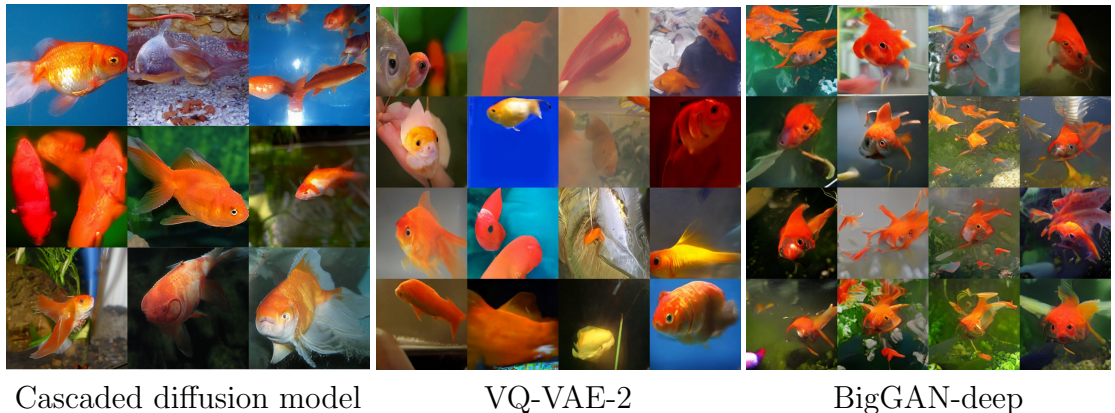
Thus, instead of solving an SDE, one can solve an ODE, which is generally much easier.

The framework described above provides us with an alternate perspective in which we are trying to reverse a destructive Forward process. This perspective is more theoretically grounded than annealing  $\sigma$  over time within Langevin sampling. It also can take advantage of ODE solvers to balance quality/speed. However, Algorithm 2 and Algorithm 3 require more score network evaluations than Algorithm 1, and the correction-step is not well theoretically justified. Also, it is not yet clear whether using Algorithm 2, Algorithm 3, or solving the RDP 2.18 with an ODE solver performs significantly better than the original Algorithm 1.

### 2.4.3 Discussion

As can be seen from the samples generated by some of the first score-based diffusion models from 2019/2020 (Figures 2.4 and 2.5), the images were high-quality,

but they still had significant visual corruptions. Given the quality limitations at the time, I explored possible improvements to quality using adversarial score matching (Jolicoeur-Martineau et al., 2021). Since then, progress has been steady, and some score-based diffusion models now produce images that look even more photo-realistic than those generated by GANs (see Figure 2.7).



**Figure 2.7** – Comparing the quality and diversity of generated red fishes from ImageNet (Ho et al., 2021; Razavi et al., 2019; Brock et al., 2019)

Now that we have a general framework for score-based diffusion models, we need to uncover what variation works best in terms of quality and speed. For example, Song et al. (2021) proposed using a different FDP to align the objective closer to maximum likelihood estimation and makes it easier to extract the log-likelihood. Meanwhile, Nachmani et al. (2021) showed that using Gamma noise instead of Gaussian noise leads to higher quality images.

Currently, the biggest issue with score-based diffusion models is that generating samples is very slow and requires many iterations. More work is needed to improve speed so that high-resolution images can be produced in a reasonable amount of time. Ideally, we would like to find a way to obtain near real-time sample generation in order to compete with GANs, which can generate samples in real-time. At the time of writing, the most promising fast sampling technique seems to be Progressive Distillation (Salimans and Ho, 2022) which manages to generate data with no significant loss in quality through only 4 to 16 iterations.

Regarding sampling, it is also unclear why one would use LMC over other more modern MCMC sampling methods. Either we can do better with these other sampling methods, or something fundamentally important about LMC needs to be uncovered. An exciting line of direction is high-order denoising score matching

---

which allows using second-order Langevin dynamics to potentially sample in much fewer iterations (Meng et al., 2021).

---

## 2.5 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) (Goodfellow et al., 2020) is considered to be one of the best approach to generative modelling. This approach came to be as a result of a drunken argument at the bar *Les 3 Brasseurs* (Giles, 2018). Ian Goodfellow, a Ph.D. student at UdeM at the time, wondered if he could generate realistic images by making two neural networks, the generator  $G$  and the critic  $C$ , compete with one another. He defined the goal of  $C$  to be determining how likely a sample is to be real or fake (i.e.,  $C$  is a classifier) and the goal of  $G$  to be "fooling"  $C$  into thinking that the fake data generated by  $G$  are actually real.

His approach turned out to work exceptionally well, as GANs have become one, if not the most, successful approach to generative modeling. GANs are now used everywhere. One of its biggest successes is generating high-quality images. I show examples of images produced by GANs in Figures 2.8, 2.9, and 2.10. As can be observed, samples generated by GANs are very realistic, which is by the design of the adversarial game, since it trains the generator to make samples realistic from the perspective of the critic.



**Figure 2.8** – Samples of cats generated by my Relativistic GAN (Jolicoeur-Martineau, 2019)

I start this chapter with a review of the original GAN. I show how to extend this game to other objective functions. I show that this game can be re-interpreted as divergence minimization (a concept we have introduced in sections 2.2.3 and 2.3.2). I then discuss the Wasserstein distance and how this specific divergence influences how we train GANs. I finish with a discussion on the central issues of GANs and how we can partially fix these issues.



**Figure 2.9** – Samples generated by BigGAN (Brock et al., 2019)



**Figure 2.10** – Samples of humans generated by StyleGAN2 (Karras et al., 2020)

---

### 2.5.1 Standard GAN

In the original version of GAN (Saturating-GAN) (Goodfellow et al., 2020), we try to solve the following game:

$$\min_{G:Z \rightarrow \mathcal{X}} \max_{C:\mathcal{X} \rightarrow \mathbb{R}} -\mathbb{E}_{x \sim \mathbb{P}} [\log(\sigma(C(x)))] - \mathbb{E}_{z \sim \mathbb{Z}} [\log(1 - \sigma(C(G(z))))], \quad (2.21)$$

where  $\sigma(x)$  is the sigmoid function,  $\mathbb{P}$  is the distribution of real data with support  $\mathcal{X}$ ,  $\mathbb{Z}$  is a latent distribution (generally a multivariate normal distribution),  $C(x)$  is the critic evaluated at  $x$ ,  $G(z)$  is the generator evaluated at  $z$ , and  $G(z) \sim \mathbb{Q}$ , where  $\mathbb{Q}$  is the distribution of fake data.

This equation corresponds to a cross-entropy objective function, where the *discriminator*  $D(x) = \sigma(C(x))$  is a classifier which outputs the probability that the given sample  $x$  is real (from  $\mathbb{P}$ ) rather than fake (from  $\mathbb{Q}$ ). The first term of the equation makes  $C(x) \rightarrow \infty$  (or equivalently  $D(x) \rightarrow 1$ ), while the second term makes  $C(G(z)) \rightarrow -\infty$  (or equivalently  $D(G(z)) \rightarrow 0$ ). Meanwhile,  $G$  tries to counter by making  $C(G(z)) \rightarrow \infty$  (or equivalently  $D(G(z)) \rightarrow 1$ ); in other words,  $D$  determines the probability a given sample is real and  $G$  attempt to make fake images appear real from the point of view of the critic.

Since we cannot optimize functions directly, we parametrize  $G$  and  $C$  using neural networks with trainable parameters  $\theta$  and  $\phi$  respectively. We train  $G$  and  $C$  iteratively, so we denote them  $G_{\theta_k}$  and  $C_{\phi_k}$ , where  $k \in [1, \dots, K]$ .

To train  $C$  and  $G$  to play this min-max game, we generally use Stochastic Gradient Ascent/Descent (SGAD) (Robbins and Monro, 1951). Two choices arise: we can train both networks successively (alternating SGAD) or simultaneously (simultaneous SGAD).

Let  $\mathcal{O}_C(\phi, \theta)$  and  $\mathcal{O}_G(\phi, \theta)$  be respectively the objective function of  $C$  (to be maximized) and of  $G$  (to be minimized). In the case of standard GAN,  $\mathcal{O}_C(\phi, \theta) = \mathcal{O}_G(\phi, \theta) = -\mathbb{E}_{x \sim \mathbb{P}} [\log(\sigma(C_{\phi}(x)))] - \mathbb{E}_{z \sim \mathbb{Z}} [\log(1 - \sigma(C_{\phi}(G_{\theta}(z))))]$ .

Alternating SGAD consists in the following update rules:

$$\phi_{k+1} \leftarrow \phi_k + \nabla_{\phi_k} [\mathcal{O}_C(\phi_k, \theta_k)], \quad (2.22)$$

$$\theta_{k+1} \leftarrow \theta_k - \nabla_{\theta_k} [\mathcal{O}_G(\phi_{k+1}, \theta_k)]. \quad (2.23)$$

---

Simultaneous SGAD consists in the following update rules:

$$\phi_{k+1} \leftarrow \phi_k + \nabla_{\phi_k} [\mathcal{O}_C(\phi_k, \theta_k)], \quad (2.24)$$

$$\theta_{k+1} \leftarrow \theta_k - \nabla_{\theta_k} [\mathcal{O}_G(\phi_k, \theta_k)]. \quad (2.25)$$

In practice, alternating SGAD tends to work better in GANs (Goodfellow et al., 2020) even though both simultaneous and alternating SGAD can converge under certain conditions (Mescheder et al., 2018). Thus, practitioners generally use alternating SGAD.

Importantly, in practice, Saturating-GAN tends to have vanishing gradients; thus, Goodfellow et al. (2020) suggested using a different formulation with the same optimum with less risk of vanishing gradients (but as I will discuss, still tends to happen). This alternate formulation (non-saturating-GAN) can be formulated as to:

$$\max_{C: \mathcal{X} \rightarrow \mathbb{R}} -\mathbb{E}_{x \sim \mathbb{P}} [\log(\sigma(C(x)))] - \mathbb{E}_{z \sim \mathbb{Z}} [\log(1 - \sigma(C(G(z))))], \quad (2.26)$$

$$\max_{G: \mathbb{Z} \rightarrow \mathcal{X}} -\mathbb{E}_{z \sim \mathbb{Z}} [\log(\sigma(C(G(z))))], \quad (2.27)$$

We see that generator still makes  $C(G(z)) \rightarrow \infty$  (or equivalently  $D(G(z)) \rightarrow 1$ ), thus the goal is fundamentally the same. In practice, we use the non-saturating version.

## 2.5.2 Generalizing GANs to other objective functions

Since the goal is to make  $C$  output a large positive number for real data and a small negative number for fake data, we can generalize saturating and non-saturating GANs to any classifier loss function. GANs can thus be defined very generally as:

$$\max_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} [f_1(C(x))] + \mathbb{E}_{y \sim \mathbb{Q}} [f_2(C(y))], \quad (2.28)$$

$$\min_{G: \mathbb{Z} \rightarrow \mathcal{X}} \mathbb{E}_{z \sim \mathbb{Z}} [g(C(G(z)))], \quad (2.29)$$

for some choice of  $f_1, f_2, g: \mathbb{R} \rightarrow \mathbb{R}$ .

Irrespective of the objective function, GANs with a saturating loss have  $g = -f_2$ , while GANs with a non-saturating loss have  $g = f_1$ . Most GAN variants use the non-saturating loss or are such that the non-saturating and saturating losses are the same. Unless otherwise specified, I assume the non-saturating loss function.



---

There are many variants for the objective functions, but three are very popular and successful: Non-saturating GAN, Least-Squares GAN (LSGAN) (Mao et al., 2017), and Hinge GAN (Lim and Ye, 2017). LSGAN corresponds to  $f_1(z) = -(1 - z)^2$ ,  $f_2(z) = -(1 + z)^2$ , and  $g(z) = f_1(z)$ . HingeGAN (Lim and Ye, 2017) corresponds to  $f_1(z) = -\max(0, 1 - z)$ ,  $f_2(z) = -\max(0, 1 + z)$ , and  $g(z) = -z$ . SGAD can be applied irrespective of the loss function used.

### 2.5.3 Divergence minimization perspective

I previously showed how GANs correspond to an adversarial game between two players. However, there is a different perspective that can be used to define GANs. From sections 2.2.3 and 2.3.2, we have seen that estimating  $f$ -divergences and IPMs (which encompass most divergences used in the literature) can be done by training a critic  $C$  to classify real from fake data. We had formalized this idea as:

$$\min_{\theta} \mathcal{D}(\mathbb{P} \parallel \mathbb{Q}_{\theta}) = \min_{\theta} \max_{\phi} \mathcal{D}_{\phi}(\mathbb{P} \parallel \mathbb{Q}_{\theta}), \quad (2.30)$$

where  $\mathcal{D}_{\phi}$  is an objective function that depends on a critic.

As it turns out most GANs can be interpreted from the perspective of divergence minimization. The critic’s objective function in GAN, LSGAN, and HingeGAN corresponds respectively to the Jensen-Shannon divergence (JSD) (Lin, 1991), Pearson  $\chi^2$  divergence (Lin, 1991), and Reverse KL divergence (Kullback, 1997).

### 2.5.4 Wasserstein distance and its influence on modern GANs

For most divergences, the mapping  $\theta \rightarrow \mathcal{D}(\mathbb{P} \parallel \mathbb{Q}_{\theta})$  is not continuous even when the mapping  $\theta \rightarrow G_{\theta}$  is continuous. Arjovsky et al. (2017) showed this to be the case with Kullback-Leibler (KL) (Kullback, 1997) and the Jensen-Shannon divergence (JSD) (Lin, 1991), two very popular choices of  $f$ -divergences. The latter, JSD, is the divergence used in the original GAN. The fact that these divergences are not continuous with respect to the parameters means that that optimization is difficult, and instability is expected.

On the other, Arjovsky et al. (2017) showed that the mapping  $\theta \rightarrow W(\mathbb{P}, \mathbb{Q}_{\theta})$  is continuous, when  $W$  is the Wasserstein distance (Vaserstein, 1969). Furthermore,  $W(\mathbb{P}, \mathbb{Q}_{\theta})$  is differentiable everywhere. Thus, they propose using this divergence.

---

Given a metric  $d(x_1, x_2)$ , the Wasserstein’s distance is defined as:

$$W(\mathbb{P}, \mathbb{Q}) := \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{M \times M} d(x_1, x_2) d\pi(x_1, x_2),$$

where  $\Pi(\mathbb{P}, \mathbb{Q})$  is the set of all distributions with marginals  $\mathbb{P}$  and  $\mathbb{Q}$  and we call  $\pi$  a coupling. Unless otherwise specified, we assume the metric  $d(x_1, x_2) = \|x_1 - x_2\|_2$ .

Estimating the Wasserstein distance is far from trivial, but it can be shown through Kantorovich-Rubinstein duality (Villani, 2008) that its dual form is a particular case of Integral Probability Metric (IPM):

$$\text{IPM}_{\mathcal{F}}(\mathbb{P} \parallel \mathbb{Q}_{\theta}) = \sup_{\substack{C: \mathcal{X} \rightarrow \mathbb{R} \\ C \in \mathcal{F}}} \mathbb{E}_{x \sim \mathbb{P}} [C(x)] - \mathbb{E}_{y \sim \mathbb{Q}} [C(x)], \quad (2.31)$$

for some specific choice of functions  $\mathcal{F}$ . The Wasserstein distance corresponds to the IPM where  $\mathcal{F}$  is the set of 1-Lipschitz functions (i.e.,  $\frac{f(x)-f(y)}{d(x,y)} \leq 1$  for all  $x, y \in \mathcal{X}$ ). Thus, if we can solve the maximization problem while ensuring that the critic is 1-Lipschitz, we can estimate the Wasserstein distance.

Given that the Wasserstein distance is continuous and differentiable everywhere, it can make for a good choice of divergence. In practice, it has been found to work well and make training much more stable (Arjovsky et al., 2017). Many ways to estimate the Wasserstein distance for GAN purposes have been devised, but I will focus on the two most popular approaches: 1) spectral normalization (Miyato et al., 2018), and 2) gradient penalty (Gulrajani et al., 2017).

The first approach, Spectral normalization, constrains the critic to be 1-Lipschitz by normalizing the weight matrix’s spectral norm at each layer of the neural network to be Lipschitz-1 (Miyato et al., 2018). This approach is easy to apply and does not lead to significant computing overhead.

The second approach, gradient penalty, requires a bit more background to understand. Gulrajani et al. (2017) showed that the optimal critic  $C^*(x)$  of the Wasserstein distance (from equation 2.31) is such that for any  $\hat{x} = \alpha x_1 + (1 - \alpha)x_2$ ,  $0 \leq \alpha \leq 1$ , where  $(x_1, x_2)$  comes from the optimal coupling  $\pi^*$ , we have that  $\|\nabla C^*(\hat{x})\|_2 = 1$  almost everywhere. Since we do not know the optimal coupling, we cannot sample from it. As a simple solution, Gulrajani et al. (2017) proposed to enforce the gradient-norm one constraint using a soft penalty on every convex combination of real and fake samples; this corresponds to penalizing  $\mathbb{E}_{\hat{x}}[\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1]^2$

---

to be small, where  $\tilde{x} = \alpha x_1 + (1 - \alpha)x_2$ ,  $\alpha \sim U(0, 1)$ ,  $x_1 \sim \mathbb{P}$ , and  $x_2 \sim \mathbb{Q}$ . This approach is called Wasserstein GAN with gradient-penalty (WGAN-GP).

Importantly, WGAN-GP does not necessarily estimate the Wasserstein distance since we are not sampling from  $\pi^*$  (Petzka et al., 2018). However, it can be shown (Jolicoeur-Martineau and Mitliagkas, 2019) that if  $d(x_1, x_2) = \|x_1 - x_2\|_2$  and  $f$  is a differentiable function, we have that:

$$\|\nabla_x f(x)\|_2 \leq 1 \iff f \text{ is 1-Lipschitz.} \quad (2.32)$$

This means that one should not penalize the gradient to be one, but to be less-or-equal than one. This can be enforced by penalizing  $\mathbb{E}_{\tilde{x}}[\max(0, \|\nabla_{\tilde{x}} C(\tilde{x})\|_2 - 1)]$  to be small (Petzka et al., 2018; Jolicoeur-Martineau and Mitliagkas, 2019).

Gradient norm penalties are highly popular in GANs as they have been shown theoretically and empirically to help the stability, and convergence of most GANs (Wasserstein or Non-Wasserstein) (Fedus et al., 2018; Kodali et al., 2017; Mescheder et al., 2018; Karras et al., 2019). I have also shown that gradient penalties (and thereby 1-Lipschitz constraints) arise in Non-Wasserstein GANs from the use of a maximum-margin classifier as GAN critic (Jolicoeur-Martineau and Mitliagkas, 2019).

### 2.5.5 Discussion

GANs can generate highly realistic samples; however, they have two main flaws: instability and mode collapse.

First, training of GANs can be highly unstable. This issue has mostly been improved through the use of gradient penalties (Gulrajani et al., 2017), Spectral normalization (Miyato et al., 2018), more stable loss functions (Gulrajani et al., 2017; Jolicoeur-Martineau, 2019; Mao et al., 2017; Lim and Ye, 2017), and better model architectures (Radford et al., 2015; Brock et al., 2019; Miyato et al., 2018; Zhang et al., 2019; Karras et al., 2018, 2019, 2020).

Second, GANs can generate high-quality samples, but the diversity in samples can be low. This is because GANs can solve the min-max problem simply by generating a single perfectly realistic image. Thus, without any additional constraint, GANs tend to mode collapse. There are, however, solutions to this problem. Both Spectral normalization and gradient penalty seem to improve diversity and reduce

---

mode collapse (Miyato et al., 2018; Gulrajani et al., 2017). A theoretically grounded solution is PACGAN (Lin et al., 2018), which consists of showing the critic more than one sample at a time by concatenating multiple input samples into one. With PACGAN, for samples to be considered realistic by the critic, multiple *realistic* and *different* samples must be provided at a time (unless the true distribution has extremely low diversity or is discrete). This ensures good diversity in theory and practice.

Although the solutions above reduce the problems of instability and mode collapse, they are imperfect. For example, Brock et al. (2019) found that gradient penalty prevented their model from mode collapsing, but at the expense of producing much lower quality images. I had similar observations from using PACGAN. Meanwhile, without gradient penalty, even if using Spectral normalization, Brock et al. (2019) found that GANs inevitably mode collapse at some point in training. Thus, more effort still needs to be made to improve GANs in order to have both high-quality and high-diversity.

# 3

## Prologue to the first article

---

### 3.1 Article Details

Jolicoeur-Martineau, A. (2019). [The relativistic discriminator: a key element missing from standard GAN](#). *International Conference on Learning Representations (ICLR)*.

---

### 3.2 Context

At the time, training GANs was very difficult and generally led to severe instabilities; this was especially true when trying to generate high-dimensional data such as high-resolution images. In trying to understand GANs better, I ended up discovering a new class of GAN loss functions which appeared to have advantages based on a few theoretical reasons (see the three arguments in the abstract). I then found that these *relativistic* loss functions were significantly more stable than their non-relativistic counterparts. With relativistic loss functions, I successfully generated high-quality samples in 256x256 for a very small data-set of  $N = 2011$ , a very challenging task at the time.

I invented this approach on my own, using my single GTX 1060 GPU to train the models. While models were training, I was unable to use my computer. I could only leave train a model during the night or for longer when away from home. My friends and family must have thought I was crazy, but I knew I had something really promising. Now, Relativistic GANs have received over 500 citations and are used by many. It seems to be particularly good in super-resolution (Wang et al., 2018), text generation (Nie et al., 2019), and for relatively small datasets with unruly/unstable settings.

---

I refer to myself as "we" instead of "I" in the paper because papers need to be anonymized during peer review, and speaking in plural makes more sense for this purpose.

---

### **3.3 Personal contribution to the paper**

I wrote the whole paper.

# The relativistic discriminator: a key element missing from standard GAN

---

## 4.1 Abstract

In standard generative adversarial network (SGAN), the discriminator  $D$  estimates the probability that the input data is real. The generator  $G$  is trained to increase the probability that fake data is real. We argue that it should also simultaneously decrease the probability that real data is real because 1) this would account for *a priori* knowledge that half of the data in the mini-batch is fake, 2) this would be observed with divergence minimization, and 3) SGAN would be more similar to integral probability metric (IPM) GANs.

We show that this property can be induced by using a “relativistic discriminator” which estimate the probability that the given real data is more realistic than a randomly sampled fake data. We also present a variant in which the discriminator estimate the probability that the given real data is more realistic than fake data, on average. We generalize both approaches to non-standard GAN loss functions and we refer to them respectively as Relativistic GANs (RGANs) and Relativistic average GANs (RaGANs). We show that IPM-based GANs are a subset of RGANs which use the identity function.

Empirically, we observe that 1) RGANs and RaGANs are significantly more stable and generate higher quality data samples than their non-relativistic counterparts, 2) Standard RaGAN with gradient penalty generate data of better quality than WGAN-GP while only requiring a single discriminator update per generator update (reducing the time taken for reaching the state-of-the-art by 400%), and 3) RaGANs are able to generate plausible high resolutions images (256x256) from a very small sample (N=2011), while GAN and LSGAN cannot; these images are of significantly better quality than the ones generated by WGAN-GP and SGAN with spectral normalization.

The code is freely available on <https://github.com/AlexiaJM/RelativisticGAN>.

---

## 4.2 Introduction

Generative adversarial networks (GANs) (Goodfellow et al., 2020) form a broad class of generative models in which a game is played between two competing neural networks, the discriminator  $D$  and the generator  $G$ .  $D$  is trained to discriminate real from fake data, while  $G$  is trained to generate fake data that  $D$  will mistakenly recognize as real. In the original GAN by Goodfellow et al. (2020), which we refer to as Standard GAN (SGAN),  $D$  is a classifier, thus it is predicting the probability that the input data is real. When  $D$  is optimal, the loss function of SGAN is approximately equal to the Jensen–Shannon divergence (JSD) (Goodfellow et al., 2020).

SGAN has two variants for the generator loss functions: saturating and non-saturating. In practice, the former has been found to be very unstable, while the latter has been found to be more stable (Goodfellow et al., 2020). Under certain conditions, Arjovsky and Bottou (2017) proved that, if real and fake data are perfectly classified, the saturating loss has zero gradient and the non-saturating loss has non-zero, but volatile gradient. In practice, this means that the discriminator in SGAN often cannot be trained to optimality or with a too high learning rate; otherwise, gradients may vanish and, if so, training will stop. This problem is generally more noticeable in high-dimensional setting (e.g., high resolution images and discriminator architectures with high expressive power) given that there are enough degrees of freedom available to perfectly classify the training set.

To improve on SGAN, many GAN variants have been suggested using different loss functions and discriminators that are not classifiers (e.g., LSGAN (Mao et al., 2017), WGAN (Arjovsky et al., 2017)). Although these approaches have partially succeeded in improving stability and data quality, the large-scale study by Lucic et al. (2018) suggests that these approaches do not consistently improve on SGAN. Additionally, some of the most successful approaches, such as WGAN-GP (Gulrajani et al., 2017), are much more computationally demanding than SGAN.

Many of the recent successful GANs variants have been based on Integral probability metrics (IPMs) (Müller, 1997) (e.g., WGAN, WGAN-GP, Fisher GAN (Mroueh and Sercu, 2017), Sobolev GAN (Mroueh et al., 2018)). In IPM-based GANs, the discriminator is real-valued and constrained to a specific class of function which regularize the discriminator. See Mroueh et al. (2018) for a review of



---

the different IPMs.

These IPM constraints have been shown to be beneficial even in non-IPM based GANs. Spectral normalization (Miyato et al., 2018) improves the stability of various GANs and it consists in making the discriminator Lipschitz-1, which is the constraint of WGAN. Similarly, the gradient penalty of WGAN-GP also provides improve the stability of SGAN (Fedus et al., 2018). Although this shows that certain IPM constraints improve the stability of GANs, it does not explain why IPM-based GANs generally provide increased stability over other metrics/divergences in GANs (e.g., JSD for SGAN,  $f$ -divergences for  $f$ -GANs (Nowozin et al., 2016)).

Note that although powerful, IPM-based GANs tend to more computationally demanding than other GANs. Certain IPM-based GANs use a gradient penalty (e.g. WGAN-GP, Sobolev GAN) which is very computationally costly and most IPM-based GANs need more than one discriminator update per generator update (WGAN-GP requires at least 5 (Gulrajani et al., 2017)). Assuming equal training time for  $D$  and  $G$ , every additional discriminator update increase training time by a significant 50%.

In this paper, we argue that non-IPM-based GANs are missing a key ingredient, a relativistic discriminator, which IPM-based GANs already possess. We show that a relativistic discriminator is necessary to make GANs analogous to divergence minimization and produce sensible predictions based on the *a priori* knowledge that half of the samples in the mini-batch are fake. We provide empirical evidence showing that GANs with a relativistic discriminator are more stable and produce data of higher quality.

---

## 4.3 Background

### 4.3.1 Generative adversarial networks

GANs can be defined very generally in terms of the discriminator in the following way:

$$L_D = \mathbb{E}_{x_r \sim \mathbb{P}} \left[ \tilde{f}_1(D(x_r)) \right] + \mathbb{E}_{z \sim \mathbb{P}_z} \left[ \tilde{f}_2(D(G(z))) \right], \quad (4.1)$$

---

and

$$L_G = \mathbb{E}_{x_r \sim \mathbb{P}} [\tilde{g}_1(D(x_r))] + \mathbb{E}_{z \sim \mathbb{P}_z} [\tilde{g}_2(D(G(z)))], \quad (4.2)$$

where  $\tilde{f}_1, \tilde{f}_2, \tilde{g}_1, \tilde{g}_2$  are scalar-to-scalar functions,  $\mathbb{P}$  is the distribution of real data,  $\mathbb{P}_z$  is generally a multivariate normal distribution centered at 0 with variance 1,  $D(x)$  is the discriminator evaluated at  $x$ ,  $G(z)$  is the generator evaluated at  $z$  ( $\mathbb{Q}$  is the distribution of fake data, thus of  $G(z)$ ). Note that, through the paper, we refer to real data as  $x_r$  and fake data as  $x_f$ . Without loss of generality, we assume that both  $L_D$  and  $L_G$  are loss functions to be minimized.

Most GANs can be separated into two classes: non-saturating and saturating loss functions. GANs with the saturating loss are such that  $\tilde{g}_1 = -\tilde{f}_1$  and  $\tilde{g}_2 = -\tilde{f}_2$ , while GANs with the non-saturating loss are such that  $\tilde{g}_1 = \tilde{f}_2$  and  $\tilde{g}_2 = \tilde{f}_1$ . Saturating GANs are intuitive as they can be interpreted as alternating between maximizing and minimizing the same loss function. After training  $D$  to optimality, the loss function is generally an approximation of a divergence (e.g., Jensen–Shannon divergence (JSD) for SGAN (Goodfellow et al., 2020),  $f$ -divergences for F-GANs (Nowozin et al., 2016), and Wassertein distance for WGAN (Arjovsky et al., 2017)). Thus, training  $G$  to minimize  $L_G$  can be roughly interpreted as minimizing the approximated divergence (although this is not technically true; see Jolicoeur-Martineau (2018)). On the other hand, non-saturating GANs can be thought as optimizing the same loss function, but swapping real data with fake data (and vice-versa). In this article, unless otherwise specified, we assume a non-saturating loss for all GANs.

SGAN assumes a cross-entropy loss, i.e.,  $\tilde{f}_1(D(x)) = -\log(D(x))$  and  $\tilde{f}_2(D(x)) = -\log(1-D(x))$ , where  $D(x) = \text{sigmoid}(C(x))$ , and  $C(x)$  is the non-transformed discriminator output (which we call the *critic* as per Arjovsky et al. (2017)). In most GANs,  $C(x)$  can be interpreted as *how realistic the input data is*; a negative number means that the input data looks fake (e.g., in SGAN,  $D(x) = \text{sigmoid}(-5) = 0$ ), while a positive number means that the input data looks real (e.g., in SGAN,  $D(x) = \text{sigmoid}(5) = 1$ ).

---

### 4.3.2 Integral probability metrics

IPMs are statistical divergences represented mathematically as:

$$IPM_F(\mathbb{P}||\mathbb{Q}) = \sup_{C \in \mathcal{F}} \mathbb{E}_{x \sim \mathbb{P}}[C(x)] - \mathbb{E}_{x \sim \mathbb{Q}}[C(x)],$$

where  $\mathcal{F}$  is a class of real-valued functions.

IPM-based GANs can be defined using equation 1 and 2 where  $\tilde{f}_1(D(x)) = \tilde{g}_2(D(x)) = -D(x)$  and  $\tilde{f}_2(D(x)) = \tilde{g}_1(D(x)) = D(x)$ , where  $D(x) = C(x)$  (i.e., no transformation is applied). It can be observed that both discriminator and generator loss functions are unbounded and would diverge to  $-\infty$  if optimized directly. However, IPMs assume that the discriminator is of a certain class of function that does not grow too quickly which prevent the loss functions from diverging. Each IPM applies a different constraint to the discriminator (e.g., WGAN assumes a Lipschitz  $D$ , WGAN-GP assumes that  $D$  has a gradient norm equal to 1 around real and fake data).

---

## 4.4 Missing property of SGAN

We argue that the key missing property of SGAN is that the probability of real data being real ( $D(x_r)$ ) should decrease as the probability of fake data being real ( $D(x_f)$ ) increase. We provide three arguments suggesting that SGAN should have this property.

### 4.4.1 Prior knowledge argument

Assuming a rational human was shown half real data and half fake data. If they perceived all samples shown as equally real ( $C(x_f) \approx C(x_r)$  for most  $x_r$  and  $x_f$ ), they would assume that each sample has probability .50 of being real. However, this is not the case for the discriminator in SGAN. If all samples looked real ( $C(x_f) \approx C(x_r) \geq 3$ ),  $D$  would assume incorrectly that they are indeed all real ( $D(x) \approx 1$  for all  $x$ ). Of course, once trained with the labels,  $D$  would decrease  $D(x_f)$  and thus would obtain more reasonable estimates. However, if  $D(x_r)$  decreased as  $D(x_f)$  increased, we would have had that  $D(x) \approx .50$  for all  $x$  before even retraining

---

$D$ . A rational human would not require retraining. IPM-based GANs implicitly account for the fact that some of the samples must be fake because they compare how realistic real data is compared to fake data. This is the behavior that we would want.

#### 4.4.2 Divergence minimization argument

In SGAN, when optimized, we have that the discriminator loss function is equal to the Jensen–Shannon divergence (JSD) (Goodfellow et al., 2020). The JSD is minimized ( $JSD(\mathbb{P}||\mathbb{Q}) = 0$ ) when  $D(x_r) = D(x_f) = \frac{1}{2}$  for all  $x_r \in \mathbb{P}$  and  $x_f \in \mathbb{Q}$  and maximized ( $JSD(\mathbb{P}||\mathbb{Q}) = \log(2)$ ) when  $D(x_r) = 1$ ,  $D(x_f) = 0$  for all  $x_r \in \mathbb{P}$  and  $x_f \in \mathbb{Q}$ . Thus, if we were directly minimizing the divergence from maximum to minimum, we would expect  $D(x_r)$  to smoothly decrease from 1 to .50 for most  $x_r$  and  $D(x_f)$  to smoothly increase from 0 to .50 for most  $x_f$  (Figure 1a). However, when minimizing the saturating loss in SGAN, we are only increasing  $D(x_f)$ , we are not decreasing  $D(x_r)$  (Figure 1b). Furthermore, we are bringing  $D(x_f)$  closer to 1 rather than .50.

This means that SGAN dynamics are very different from the minimization of the JSD. To bring SGAN closer to divergence minimization, training the generator should not only increase  $D(x_f)$  but also decrease  $D(x_r)$  (Figure 1c). Note that although specific to the JSD, similar dynamics are true for other divergences; when the divergence is maximal,  $D(x_r)$  and  $D(x_f)$  are very far from one another, but they converge to the same value as the divergence approach zero. Thus, this argument applies to other divergences.

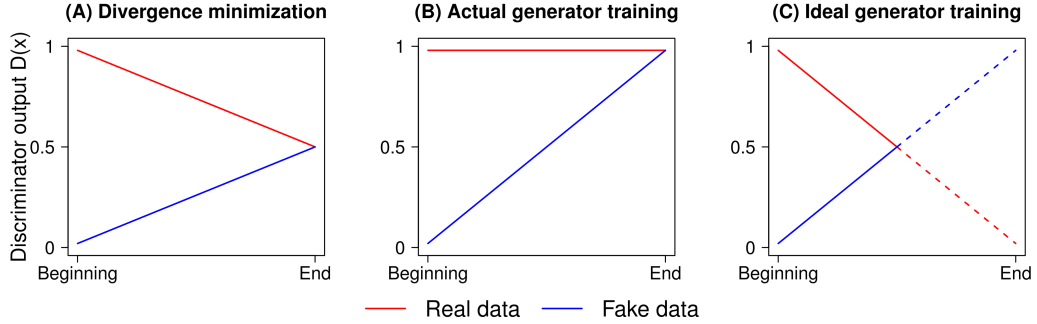
#### 4.4.3 Gradient argument

We compare the gradients of standard GAN and IPM-based GANs for further insight. It can be shown that the gradients of the discriminator and generator in non-saturating SGAN are respectively:

$$\nabla_w L_D^{GAN} = -\mathbb{E}_{x_r \sim \mathbb{P}} [(1 - D(x_r)) \nabla_w C(x_r)] + \mathbb{E}_{x_f \sim \mathbb{Q}_\theta} [D(x_f) \nabla_w C(x_f)], \quad (4.3)$$

$$\nabla_\theta L_G^{GAN} = -\mathbb{E}_{z \sim \mathbb{P}_z} [(1 - D(G(z))) \nabla_x C(G(z)) J_\theta G(z)], \quad (4.4)$$

where  $J$  is the Jacobian.



**Figure 4.1** – Expected discriminator output of the real and fake data for the a) direct minimization of the Jensen–Shannon divergence, b) actual training of the generator to minimize its loss function, and c) ideal training of the generator to minimize its loss function (lines are dotted when they cross beyond the equilibrium to signify that this may or may not be necessary).

It can be shown that the gradients of the discriminator and generator in IPM-based GANs are respectively:

$$\nabla_w L_D^{IPM} = -\mathbb{E}_{x_r \sim \mathbb{P}}[\nabla_w C(x_r)] + \mathbb{E}_{x_f \sim \mathbb{Q}_\theta}[\nabla_w C(x_f)], \quad (4.5)$$

$$\nabla_\theta L_G^{IPM} = -\mathbb{E}_{z \sim \mathbb{P}_z}[\nabla_x C(G(z)) J_\theta G(z)], \quad (4.6)$$

where  $C(x) \in \mathcal{F}$  (the class of functions assigned by the IPM).

From these equations, it can be observed that SGAN leads to the same dynamics as IPM-based GANs when we have that:

1.  $D(x_r) = 0, D(x_f) = 1$  in the discriminator step of SGAN
2.  $D(x_f) = 0$  in the generator step of SGAN.
3.  $C(x) \in \mathcal{F}$

Assuming that the discriminator and generator are trained to optimality in each step (which we sometimes do for  $D$ , but never for  $G$ ) and that it is possible to perfectly distinguish real from the fake data (strong assumption, but generally true early in training); we would have that  $D(x_r) = 1, D(x_f) = 0$  in the generator step and that  $D(x_r) = 1, D(x_f) = 1$  in the discriminator step for most  $x_r$  and  $x_f$  (Figure 1b). Thus, the only missing assumption would be that  $D(x_r) = 0$  in the discriminator step.

Although the above scenario is not realistic (because we never train  $G$  to optimality), if all the assumptions were respected and the generator could indirectly

---

influence  $D(x_r)$ , we would have that  $D(x_r) = 0$ ,  $D(x_f) = 1$ . Thus, SGAN would have the same gradients as IPM-based GANs. We conjecture that making SGAN more similar to IPM-based GANs could potentially improve its stability (our results will show that it does in fact improve stability).

---

## 4.5 Method

### 4.5.1 Relativistic standard GAN

In standard GAN, the discriminator can be defined, in term of the non-transformed layer  $C(x)$ , as  $D(x) = \text{sigmoid}(C(x))$ . A simple way to make discriminator relativistic (i.e., having the output of  $D$  depends on both real and fake data) is to sample from real/fake data pairs  $\tilde{x} = (x_r, x_f)$  and define it as  $D(\tilde{x}) = \text{sigmoid}(C(x_r) - C(x_f))$ .

We can interpret this modification in the following way: *the discriminator estimates the probability that the given real data is more realistic than a randomly sampled fake data*. Similarly, we can define  $D_{rev}(\tilde{x}) = \text{sigmoid}(C(x_f) - C(x_r))$  as the probability that the given fake data is more realistic than a randomly sampled real data. An interesting property of this discriminator is that we do not need to include  $D_{rev}$  in the loss function through  $\log(1 - D_{rev}(\tilde{x}))$  because we have that  $1 - D_{rev}(\tilde{x}) = 1 - \text{sigmoid}(C(x_f) - C(x_r)) = \text{sigmoid}(C(x_r) - C(x_f)) = D(\tilde{x})$ ; thus,  $\log(D(\tilde{x})) = \log(1 - D_{rev}(\tilde{x}))$ .

The discriminator and generator (non-saturating) loss functions of the Relativistic Standard GAN (RSGAN) can be written as:

$$L_D^{RSGAN} = -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_r) - C(x_f)))]. \quad (4.7)$$

$$L_G^{RSGAN} = -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_f) - C(x_r)))]. \quad (4.8)$$

### 4.5.2 Relativistic GANs

More generally, we consider any discriminator defined as  $a(C(x_r) - C(x_f))$ , where  $a$  is the activation function, to be relativistic. This means that almost any

---

GAN can have a relativistic discriminator. This forms a new class of models which we call Relativistic GANs (RGANs).

Most GANs can be parametrized very generally in terms of the critic:

$$L_D^{GAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [f_1(C(x_r))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [f_2(C(x_f))] \quad (4.9)$$

and

$$L_G^{GAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [g_1(C(x_r))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [g_2(C(x_f))], \quad (4.10)$$

where  $f_1$ ,  $f_2$ ,  $g_1$ ,  $g_2$  are scalar-to-scalar functions. If we use a relativistic discriminator, these GANs now have the following form:

$$L_D^{RGAN} = \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_2(C(x_f) - C(x_r))] \quad (4.11)$$

and

$$L_G^{RGAN} = \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [g_1(C(x_r) - C(x_f))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [g_2(C(x_f) - C(x_r))]. \quad (4.12)$$

If one use the identity function (i.e.,  $f_1(y) = g_2(y) = -y$ ,  $f_2(y) = g_1(y) = y$ ), this results in a degenerate case since there is no supremum/maximum. However, if one adds a constraint so that  $C(x_r) - C(x_f)$  is bounded, then there is a supremum and one arrives at IPM-based GANs. Thus, although different, IPM-based GANs share a very similar loss function focused on the difference in critics.

Importantly,  $g_1$  is normally ignored in GANs because its gradient is zero since the generator does not influence it. However, in RGANs,  $g_1$  is influenced by fake data, thus by the generator.

### 4.5.3 Relativistic average GANs

The discriminator has a very different interpretation in SGAN compared to RSGAN. In SGAN,  $D(x)$  estimates the probability that  $x$  is real, while in RGANs,  $D(x_r, x_f)$  estimates the probability that  $x_r$  is more realistic than  $x_f$ . As a middle ground, we developed an alternative to the Relativistic Discriminator, which retains approximately the same interpretation as the discriminator in SGAN while still being relativistic.

We propose the Relativistic average Discriminator (RaD) which compares the

---

critic of the input data to the average critic of samples of the opposite type. The discriminator loss function for this approach can be formulated as:

$$L_D^{RaSGAN} = -\mathbb{E}_{x_r \sim \mathbb{P}} [\log (\bar{D}(x_r))] - \mathbb{E}_{x_f \sim \mathbb{Q}} [\log (1 - \bar{D}(x_f))], \quad (4.13)$$

where

$$\bar{D}(x) = \begin{cases} \text{sigmoid}(C(x) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f)) & \text{if } x \text{ is real} \\ \text{sigmoid}(C(x) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r)) & \text{if } x \text{ is fake.} \end{cases} \quad (4.14)$$

RaD has a more similar interpretation to the standard discriminator than the relativistic discriminator. With RaD, *the discriminator estimates the probability that the given real data is more realistic than fake data, on average.*

As before, we can generalize this approach to work with any GAN loss function using the following formulation:

$$L_D^{RaGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [f_1 (C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [f_2 (C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))]. \quad (4.15)$$

$$L_G^{RaGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [g_1 (C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [g_2 (C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))]. \quad (4.16)$$

We call this general approach Relativistic average GAN (RaGAN).

---

## 4.6 Experiments

Experiments were conducted on the CIFAR-10 dataset (Krizhevsky and Hinton, 2009) and the CAT dataset (Zhang et al., 2008). Code was written in Pytorch (Paszke et al., 2017) and models were trained using the Adam optimizer (Kingma and Ba, 2014) for 100K generator iterations with seed 1 (which shows that we did not fish for the best seed, instead, we selected the seed *a priori*). We report the Fréchet Inception Distance (FID) (Heusel et al., 2017), a measure that is generally better correlated with data quality than the Inception Distance (Salimans et al., 2016) (Borji, 2019); lower FID means that the generated images are of better quality.



---

For the models architectures, we used the standard CNN described by Miyato et al. (2018) on CIFAR-10 and a relatively standard DCGAN architecture (Radford et al., 2015) on CAT (see Appendix). We also provide the source code required to replicate all analyses presented in this paper.

### 4.6.1 CIFAR-10

In these analyses, we compared standard GAN (SGAN), least-squares GAN (LSGAN), Wassertein GAN improved (WGAN-GP), Hinge-loss GAN (HingeGAN) (Miyato et al., 2018), Relativistic SGAN (RSGAN), Relativistic average SGAN (RaSGAN), Relativistic average LSGAN (RaLSGAN), and Relativistic average HingeGAN (RaHingeGAN) using the standard CNN architecture on stable setups (See Appendix for details on the loss functions used). Additionally, we tested RSGAN and RaSGAN with the same gradient-penalty as WGAN-GP (named RSGAN-GP and RaSGAN-GP respectively).

We used the following two known stable setups: (DCGAN setup)  $lr = .0002$ ,  $n_D = 1$ ,  $\beta_1 = .50$  and  $\beta_2 = .999$  (Radford et al., 2015), and (WGAN-GP setup)  $lr = .0001$ ,  $n_D = 5$ ,  $\beta_1 = .50$  and  $\beta_2 = .9$  (Gulrajani et al., 2017), where  $lr$  is the learning rate,  $n_D$  is the number of discriminator updates per generator update, and  $\beta_1, \beta_2$  are the ADAM momentum parameters. For optimal stability, we used batch norm (Ioffe and Szegedy, 2015) in  $G$  and spectral norm (Miyato et al., 2018) in  $D$ .

Results are presented in Table 1. We observe that RSGAN and RaSGAN generally performed better than SGAN. Similarly, RaHingeGAN performed better than HingeGAN. RaLSGAN performed on par with LSGAN, albeit slightly worse. WGAN-GP performed poorly in the DCGAN setup, but very well in the WGAN-GP setup. RasGAN-GP performed poorly; however, RSGAN-GP performed better than all other loss functions using only one discriminator update per generator update. Importantly, the resulting FID of 25.60 is on par with the lowest FID obtained for this architecture using spectral normalization, as reported by Miyato et al. (2018) (25.5). Overall, these results show that using a relativistic discriminator generally improve data generation quality and that RSGAN works very well in conjunction with gradient penalty to obtain state-of-the-art results.

**Table 4.1** – Fréchet Inception Distance (FID) at exactly 100k generator iterations on the CIFAR-10 dataset using stable setups with different GAN loss functions. We used spectral norm in  $D$  and batch norm in  $G$ . All models were trained using the same *a priori* selected seed (seed=1).

Loss	$lr = .0002$	$lr = .0001$
	$\beta = (.50, .999)$	$\beta = (.50, .9)$
	$n_D = 1$	$n_D = 5$
SGAN	40.64	41.32
RSGAN	36.61	55.29
RaSGAN	31.98	37.92
LSGAN	29.53	187.01
RaLSGAN	30.92	219.39
HingeGAN	49.53	80.85
RaHingeGAN	39.12	37.72
WGAN-GP	83.89	<b>27.81</b>
RSGAN-GP	<b>25.60</b>	28.13
RaSGAN-GP	331.86	

## 4.6.2 CAT

CAT is a dataset containing around 10k pictures of cats with annotations. We cropped the pictures to the faces of the cats using those annotations. After removing outliers (hidden faces, blurriness, etc.), the CAT dataset contained 9304 images  $\geq 64 \times 64$ , 6645 images  $\geq 128 \times 128$ , and 2011 images  $\geq 256 \times 256$ . The CAT dataset is particularly challenging due to its small sample size and high-resolution images; this makes it perfect for testing the stability of different GAN loss functions.

We trained different GAN loss functions on  $64 \times 64$ ,  $128 \times 128$ ,  $256 \times 256$  images. For  $256 \times 256$  images, we compared RaGANs to known stable approaches: SpectralSGAN (SGAN with spectral normalization in  $D$ ) and WGAN-GP. Although some approaches were able to train on  $256 \times 256$  images, they did so with significant mode collapse. To alleviate this problem, for  $256 \times 256$  images, we packed the discriminator (Lin et al., 2018) (i.e.,  $D$  took a concatenated pair of images instead of a single image). We looked at the minimum, maximum, mean and standard deviation (SD) of the FID at 20k, 30k, ..., 100k generator iterations; results are presented in Table 2.

Overall, we observe lower minimum FID, maximum FID, mean and standard

**Table 4.2** – Minimum (min), maximum (max), mean, and standard deviation (SD) of the Fréchet Inception Distance (FID) calculated at 20k, 30k . . . , 100k generator iterations on the CAT dataset with different GAN loss functions. The hyper-parameters used were  $lr = .0002$ ,  $\beta = (.50, .999)$ ,  $n_D = 1$ , and batch norm (BN) in  $D$  and  $G$ . All models were trained using the same *a priori* selected seed (seed=1). Note: A missing number imply that the model did not converge and became stuck in the first few iterations.

Loss	Min	Max	Mean	SD
64x64 images (N=9304)				
SGAN	16.56	310.56	52.54	96.81
RSGAN	19.03	42.05	32.16	7.01
RaSGAN	15.38	33.11	20.53	5.68
LSGAN	20.27	224.97	73.62	61.02
RaLSGAN	<b>11.97</b>	<b>19.29</b>	<b>15.61</b>	2.55
HingeGAN	17.60	50.94	32.23	14.44
RaHingeGAN	14.62	27.31	20.29	3.96
RSGAN-GP	16.41	22.34	18.20	1.82
RaSGAN-GP	17.32	22	19.58	<b>1.81</b>
128x128 images (N=6645)				
SGAN	-	-	-	-
RaSGAN	21.05	<b>39.65</b>	28.53	<b>6.52</b>
LSGAN	19.03	51.36	30.28	10.16
RaLSGAN	<b>15.85</b>	40.26	<b>22.36</b>	7.53
256x256 images (N=2011)				
SGAN <sup>1</sup>	-	-	-	-
RaSGAN	<b>32.11</b>	102.76	<b>56.64</b>	21.03
SpectralSGAN	54.08	<b>90.43</b>	64.92	<b>12.00</b>
LSGAN <sup>1</sup>	-	-	-	-
RaLSGAN	35.21	299.52	70.44	86.01
WGAN-GP	155.46	437.48	341.91	101.11

deviation (sd) for RGANs and RaGANs than their non-relativistic counterparts (SGAN, LSGAN, RaLSGAN).

In 64x64 resolution, both SGAN and LSGAN generated images with low FID, but they did so in a very unstable matter. For example, SGAN went from a FID of 17.50 at 30k iterations, to 310.56 at 40k iterations, and back to 27.72 at 50k iterations. Similarly, LSGAN went from a FID of 20.27 at 20k iterations, to 224.97

---

at 30k iterations, and back to 51.98 at 40k iterations. On the other hand, RaGANs were much more stable (lower max and SD) while also resulting in lower minimum FID. Using gradient-penalty did not improve data quality; however, it reduced the SD lower than without gradient penalty, thus increasing stability further.

SGAN was unable to converge on 128x128 or bigger images and LSGAN was unable to converge on 256x256 images. Meanwhile, RaGANs were able to generate plausible images with low FID in all resolutions. Although SpectralSGAN and WGAN-GP were able to generate 256x256 images of cats, the samples they generated were of poor quality (high FID). Thus, in this very difficult setting, relativism provided a greater improvement in quality than gradient penalty or spectral normalization.

---

## 4.7 Conclusion and future work

In this paper, we proposed the relativistic discriminator as a way to fix and improve on standard GAN. We further generalized this approach to any GAN loss and introduced a generally more stable variant called RaD. Our results suggest that relativism significantly improve data quality and stability of GANs at no computational cost. Furthermore, using a relativistic discriminator with other tools of the trade (spectral norm, gradient penalty, etc.) may lead to better state-of-the-art.

Future research is needed to fully understand the mathematical implications of adding relativism to GANs. Furthermore, our experiments were limited to certain loss functions using only one seed, due to computational constraints. More experiments are required to determine which relativistic GAN loss function is best over a wide-range of datasets and hyper-parameters. We greatly encourage researchers and machine learning enthusiasts with greater computing power to experiment further with our approach.

# 5

# Prologue to the second article

---

## 5.1 Article Details

Jolicoeur-Martineau, A. (2020). *On Relativistic  $f$ -Divergences*. *International Conference on Machine Learning (ICML)*.

---

## 5.2 Context

After writing the paper on Relativistic GANs, I wanted to understand them further from a strong theoretical point-of-view. In this paper, I took a more rigorous look at Relativistic GANs. I showed that  $f$ -divergences with a relativistic discriminator lead to relativistic  $f$ -divergences, and I proved that they were proper statistical divergences. I studied the weak topology of the divergence and showed how to construct unbiased estimators. Unbiased estimators performed worse empirically than biased estimators. In the end, this work left me with more questions than answers!

I refer to myself as "we" instead of "I" in the paper because papers need to be anonymized during peer review, and speaking in plural makes more sense for this purpose.

---

## 5.3 Personal contribution to the paper

I wrote the whole paper.

# 6

# On relativistic $f$ -divergences

---

## 6.1 Abstract

We take a more rigorous look at Relativistic Generative Adversarial Networks (RGANs) and prove that the objective function of the discriminator is a statistical divergence for any concave function  $f$  with minimal properties ( $f(0) = 0$ ,  $f'(0) \neq 0$ ,  $\sup_x f(x) > 0$ ). We devise additional variants of relativistic  $f$ -divergences. We show that the Wasserstein distance is weaker than  $f$ -divergences which are weaker than relativistic  $f$ -divergences. Given the good performance of RGANs, this suggests that Wasserstein GAN does not perform well primarily because of the weak metric, but rather because of regularization and the use of a relativistic discriminator. We introduce the minimum-variance unbiased estimator (MVUE) for Relativistic GANs and show that it does not perform better. We show that the estimator of Relativistic average GANs (RaGANs) is asymptotically unbiased and that the finite-sample bias is small; removing this bias does not improve performance.

---

## 6.2 Introduction

Generative adversarial networks (GANs) (Goodfellow et al., 2020) are a very popular approach to approximately generate data from a complex probability distribution using only samples of data (without any information on the true data distribution). Most notably, it has been very successful at generating photo-realistic images (Karras et al., 2018, 2019). It consists in a game between two neural networks, the generator  $G$  and the discriminator  $D$ . The goal of  $D$  is to classify real from fake (generated) data. The goal of  $G$  is to generate fake data that appears to be real, thus "fooling"  $D$  into thinking that fake data is actually real.

---

There are many GAN variants and most of them consist of changing the loss function of  $D$ . To name a few: Standard GAN (SGAN) (Goodfellow et al., 2020), Least-Squares GAN (LSGAN) (Mao et al., 2017), Hinge-loss GAN (HingeGAN) (Miyato et al., 2018), Wasserstein GAN (WGAN) (Arjovsky et al., 2017).

For most GAN variants, training  $D$  is equivalent to estimating a divergence: SGAN estimates the Jensen–Shannon divergence (JSD), LSGAN estimates the Pearson  $\chi^2$  divergence, HingeGAN estimates the Reverse-KL divergence, and WGAN estimates the Wasserstein distance. Even more generally,  $f$ -GANs (Nowozin et al., 2016) estimate any  $f$ -divergence (which includes most of the popular divergences), while IPM-based GANs (Mroueh and Sercu, 2017) estimate any Integral probability metric (IPM) (Müller, 1997). Thus, intuitively, GANs can be thought of as estimating a diverge and then minimizing it (this is not technically correct; see Jolicoeur-Martineau (2018)).

Recently, Jolicoeur-Martineau (2019) showed that IPM-based GANs possess a unique type of discriminator which they call a Relativistic Discriminator (RD). They explained that one can construct  $f$ -GANs while using a RD and that doing so improves the stability of the training and quality of generated data. They called this approach Relativistic GANs (RGANs). They proposed two variants: Relativistic paired GANs (RpGANs)<sup>1</sup> and Relativistic Average GANs (RaGANs).

Jolicoeur-Martineau (2019) provided mathematical and intuitive arguments as to why using a Relativistic Discriminator (RD) may be helpful. However, they did not prove that the loss functions are mathematically sensible. Furthermore, the estimators that they used are not the minimum-variance unbiased estimators (MVUE).

The contributions of this paper are the following:

1. We prove that the objective functions of the discriminator in RGANs are divergences (relativistic  $f$ -divergences).
2. We devise additional variants of Relativistic  $f$ -divergences.
3. We show that the Wasserstein Distance is weaker than  $f$ -divergences which are weaker than relativistic  $f$ -divergences.
4. We present the minimum-variance unbiased estimator (MVUE) of RpGANs and show that using it hinders the performance of the generator.

---

1. We added the word "paired" to better distinguish the variant with paired real/fake data (originally called RGANs) and the general approach called Relativistic GANs (RGANs).

- 
5. We show that RaGANs are only asymptotically unbiased, but that the finite-sample bias is small. Removing this bias does not improve the performance of the generator.

---

## 6.3 Background

For the rest of the paper, we will refer to the "critic"  $C(x)$  instead of the discriminator  $D(x)$ . The critic is the discriminator before applying the activation function ( $D(x) = a(C(x))$ ), where  $a$  is an activation function and  $C(x) \in \mathbb{R}$ . Intuitively, the critic can be thought of as describing how realistic  $x$  is. In the case of SGAN and HingeGAN, a large  $C(x)$  means that  $x$  is realistic, while a small  $C(x)$  means that  $x$  is not realistic. We use this notation because Relativistic GANs are defined in terms of the critic rather than the discriminator.

### 6.3.1 Generative Adversarial Networks

GANs can be defined very generally in the following way:

$$\sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} [f_1(C(x))] + \mathbb{E}_{y \sim \mathbb{Q}} [f_2(C(y))], \quad (6.1)$$

$$\sup_{G: \mathbb{Z} \rightarrow \mathcal{X}} \mathbb{E}_{x \sim \mathbb{P}} [g_1(C(x))] + \mathbb{E}_{z \sim \mathbb{Z}} [g_2(C(G(z)))], \quad (6.2)$$

where  $f_1, f_2, g_1, g_2 : \mathbb{R} \rightarrow \mathbb{R}$ ,  $\mathbb{P}$  is the distribution of real data with support  $\mathcal{X}$ ,  $\mathbb{Z}$  is the latent distribution (generally a multivariate normal distribution),  $C(x)$  is the critic evaluated at  $x$ ,  $G(z)$  is the generator evaluated at  $z$ , and  $G(z) \sim \mathbb{Q}$ , where  $\mathbb{Q}$  is the distribution of fake data. See [Brock et al. \(2019\)](#) for details on how different choices of  $\mathbb{Z}$  performs. The critic and the generator are generally trained with stochastic gradient descent (SGD) in alternating steps.

Most GANs can be separated in two classes: non-saturating and saturating loss functions. GANs with the saturating loss are such that  $g_1 = -f_1$  and  $g_2 = -f_2$ , while GANs with the non-saturating loss are such that  $g_1 = f_2$  and  $g_2 = f_1$ . In this paper, we will assume that the non-saturating loss is used as it generally works best in practice ([Goodfellow et al., 2020](#)) ([Nowozin et al., 2016](#)). Note that  $g_1$  generally



---

has no impact on training since its gradient with respect to  $G$  is zero; we can thus ignore it.

Although not always the case, the most popular GAN loss functions (SGAN, LS-GAN with labels -1/1, HingeGAN, WGAN) are symmetric (i.e.,  $f_2(x) = f_1(-x)$ ). For simplicity, in this paper, we restrict ourselves to symmetric loss functions.

Non-saturating Symmetric GANs (SyGANs) can be represented more simply as:

$$\sup_{C:\mathcal{X}\rightarrow\mathbb{R}} \mathbb{E}_{x\sim\mathbb{P}} [f(C(x))] + \mathbb{E}_{y\sim\mathbb{Q}} [f(-C(y))], \quad (6.3)$$

$$\sup_{G:Z\rightarrow\mathcal{X}} \mathbb{E}_{z\sim\mathbb{Z}} [f(C(G(z)))], \quad (6.4)$$

for some function  $f : \mathbb{R} \rightarrow \mathbb{R}$ . For easier optimization, we generally want  $f$  to be concave with respect to the critic. This is the case in symmetric  $f$ -GANs.

In this paper, we restrict our relativistic divergences to symmetric cases with concave  $f$ . Although this may be somewhat constraining, not making these assumptions would be very problematic for GANs. By not assuming concavity, we could have an objective function that diverges to infinity (and thus an infinite divergence). This is particularly problematic for GANs because early in training, we expect  $\mathbb{P}$  and  $\mathbb{Q}$  to be perfectly separated (because of fully disjoint supports). This would cause the objective function to explode towards infinity and thereby causing severe instabilities. The Kullback–Leibler (KL) divergence is a good example of such a problematic divergence for GANs. If a single sample from the support of  $\mathbb{Q}$  is not part of the support of  $\mathbb{P}$ , the divergence will be  $\infty$ . Also, note that the dual form of the KL divergence cannot be represented as a SyGAN with equation (3) since  $f_1(x) = x$  and  $f_2(x) = -e^{x-1}$  are not symmetric (Nowozin et al., 2016).

### 6.3.2 Integral Probability Metrics

Rather than using a concave function  $f$  to ensure a maximum on the objective function, IPM-based GANs instead force the critic to respect some constraint so that it does not grow too quickly. IPM-based GANs are defined in the following way:

$$\sup_{\substack{C:\mathcal{X}\rightarrow\mathbb{R} \\ C\in\mathcal{F}}} \mathbb{E}_{x\sim\mathbb{P}} [C(x)] - \mathbb{E}_{y\sim\mathbb{Q}} [C(y)], \quad (6.5)$$

$$\sup_{G:Z \rightarrow \mathcal{X}} \mathbb{E}_{z \sim \mathbb{Z}} [C(G(z))], \quad (6.6)$$

where  $\mathcal{F}$  is a class of functions such that the IPM is not infinite. See Mroueh et al. (2018) for an extensive review of the choices of  $\mathcal{F}$ .

### 6.3.3 Relativistic GANs

Rather than training the critic on real and fake data separately, Relativistic GANs tries to maximize the critic’s difference (CD). In Relativistic paired GANs (RpGANs), the CD is defined as  $C(x) - C(y)$ , while in Relativistic average GANs (RaGANs), the CD is defined as  $C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y)$  (or vice-versa). The CD can be understood as how much more realistic real data is from fake data. The optimal size of the CD is determined by the choice of  $f$ . With a least-square loss, the CD must be exactly equal to 1. On the other hand, with a log-sigmoid loss, the CD is grown to around 2 or 3 (after-which the gradient of  $f$  vanishes to zero). This will be explained in more details in the next section. Again, we focus only on choices of  $f$  that have symmetry (as done with SyGANs).

Relativistic paired GANs (RpGANs) are defined in the following way:

$$\sup_{C:\mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))], \quad (6.7)$$

$$\sup_{G:Z \rightarrow \mathcal{X}} \mathbb{E}_{\substack{x \sim \mathbb{P} \\ z \sim \mathbb{Z}}} [f(C(G(z)) - C(x))]. \quad (6.8)$$

Relativistic average GANs (RaGANs) are defined in the following way:

$$\sup_{C:\mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C(x) - C(y) \right) \right], \quad (6.9)$$

$$\sup_{G:Z \rightarrow \mathcal{X}} \mathbb{E}_{z \sim \mathbb{Z}} \left[ f \left( C(G(z)) - \mathbb{E}_{x \sim \mathbb{P}} C(x) \right) \right] + \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( \mathbb{E}_{z \sim \mathbb{P}_z} C(G(z)) - C(x) \right) \right]. \quad (6.10)$$

---

## 6.4 Relativistic Divergences

We define statistical divergences in the following way:

---

**Definition 6.4.1.** Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions and  $S$  be the set of all probability distributions with common support. A function  $D : (S, S) \rightarrow \mathbb{R}_{>0}$  is a divergence if it respects the following two conditions:

$$D(\mathbb{P}, \mathbb{Q}) \geq 0$$

$$D(\mathbb{P}, \mathbb{Q}) = 0 \iff \mathbb{P} = \mathbb{Q}.$$

In other words, divergences are distances between probability distributions. The distribution of real data ( $\mathbb{P}$ ) is fixed and our goal is to modify the distribution of fake data ( $\mathbb{Q}$ ) so that the divergence decreases over time through the training process.

It is important to show that we use a divergence; this ensures that it is not possible to obtain a critic which cannot distinguish real from fake sample ( $D(\mathbb{P}, \mathbb{Q}) = 0$ ) when the two distributions (real and fake) are not the same ( $\mathbb{P} \neq \mathbb{Q}$ ). If we did not have a divergence, it could be possible to reach a situation where the generator cannot learn (since the critic returns the same value for real and fake samples) while the generator still isn't generating samples from the real distribution.

### 6.4.1 Main Theorem

As discussed in the introduction, in most GANs, the objective function of the critic at optimum is a divergence. We show that the objective function of the critic in RpGANs, RaGANs, and other variants also estimate a divergence. The theorem is as follows:

**Theorem 1.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Let  $\mathbb{M} = \frac{1}{2}\mathbb{P} + \frac{1}{2}\mathbb{Q}$ . Then, we

---

have that

$$\begin{aligned}
D_f^{Rp}(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))] \\
D_f^{Ra}(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right] + \\
&\quad \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C(x) - C(y) \right) \right] \\
D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} 2 \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right] \\
D_f^{Rc}(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{m \sim \mathbb{M}} C(m) \right) \right] + \\
&\quad \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{m \sim \mathbb{M}} C(m) - C(y) \right) \right]
\end{aligned}$$

are divergences.

We ask that the supremum of  $f(x)$  is reached at some positive  $x$  (or at  $\infty$ ). This is purely to ensure that a larger CD can be interpreted as leading to a larger divergence (rather than the opposite). This does not reduce the generality of Theorem 3.1. If  $f(x)$  is maximized at  $x < 0$ , we have that  $g(x) = f(-x)$  is maximized at  $x > 0$  and one can simply use  $g$  instead of  $f$ .

We require that  $f$  is differentiable at zero and its derivative to be non-zero. This assumption may not be necessary, but it is needed for one of our main lemma which we use to prove that these objective functions are divergences.

Note that  $D_f^{Rp}(\mathbb{P}, \mathbb{Q})$  corresponds to RpGANs,  $D_f^{Ra}(\mathbb{P}, \mathbb{Q})$  corresponds to RaGANs,  $D_f^{Ralf}(\mathbb{P}, \mathbb{Q})$  corresponds to a simplified one-way version of RaGANs (RalfGANs), and  $D_f^{Rc}(\mathbb{P}, \mathbb{Q})$  corresponds to a new type of RGAN called Relativistic centered GANs (RcGANs). RalfGANs are not particularly interesting as they simply represent a simpler version of RaGANs. On the other hand, RcGANs are interesting as they center the critic scores using the mean of the whole mini-batch (rather than the mean of only real or only fake mini-batch samples). This divergence also has similarities to the Jensen–Shannon divergence (JSD) since the JSD is the sum of the KL-divergence between  $\mathbb{P}$  and  $\mathbb{M}$  to the KL-divergence between  $\mathbb{Q}$  and  $\mathbb{M}$ .

A logical extension to RcGANs would be to standardize the critic scores; however, this would not lead to a divergence given that we could not control the size of

---

the elements inside  $f$ . To make it a divergence, we would need a learn-able scaling weight (as in batch norm (Ioffe and Szegedy, 2015)), but this would counter the effect of the standardization. Thus, standardizing and scaling would just correspond to an equivalent re-parametrization of  $D_f^{Rc}$ .

A sketch of the proof can be found below; the full proof is found in Appendix A.

### 6.4.2 Sketch of the Proof

Although the four divergences need separate proofs, a similar framework is used in each of them. Each proof consists of three steps. For clarity of notation, let  $D_f(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} F(\mathbb{P}, \mathbb{Q}, C, f)$  be the divergence, where  $F$  is any of the objective functions in Theorem 3.1.

First, we show that  $D_f(\mathbb{P}, \mathbb{Q}) \geq 0$ . This is easily proven by taking the simplest possible choice of critic, which does not depend on the probability distributions, i.e.,  $C^w(x) = k$  for all  $x$ . This critic always leads to  $f(0)$  and thus to a objective function equal to 0. This means that

$$D_f(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} F(\mathbb{P}, \mathbb{Q}, C, f) \geq F(\mathbb{P}, \mathbb{Q}, C^w, f) = 0.$$

Second, we show that  $\mathbb{P} = \mathbb{Q} \implies D_f(\mathbb{P}, \mathbb{Q}) = 0$ . This step generally relies on Jensen's inequality (for concave functions) which we use to show that  $D_f(\mathbb{P}, \mathbb{P}) \leq 0$ . Given that  $D_f(\mathbb{P}, \mathbb{P}) \geq 0$  and  $D_f(\mathbb{P}, \mathbb{P}) \leq 0$ , we have that  $D_f(\mathbb{P}, \mathbb{P}) = 0$ .

Third, we show that  $D_f(\mathbb{P}, \mathbb{Q}) = 0 \implies \mathbb{P} = \mathbb{Q}$ . This step is by far the most difficult to prove. Instead of showing it directly, we instead prove it by contraposition, i.e., we show that  $\mathbb{P} \neq \mathbb{Q} \implies D_f(\mathbb{P}, \mathbb{Q}) > 0$ . To prove this, we use the fact that if  $\mathbb{P} \neq \mathbb{Q}$ , there must be values of the probability density functions,  $p(x)$  and  $q(x)$  respectively, such that  $p(x) > q(x)$  (and vice versa). Let  $T = \arg \sup_S \mathbb{P}(S) - \mathbb{Q}(S)$ , we know that this set is not empty. Note that when  $\mathbb{P}$  and  $\mathbb{Q}$  have probability density functions  $p(x)$  and  $q(x)$  respectively, we have that  $T = \{x | p(x) > q(x)\}$ . To make the proof as simple as possible, we use the following sub-optimal critic:

$$C'(x) = \begin{cases} \nabla & \text{if } x \in T \\ 0 & \text{else,} \end{cases}$$

---

where  $\nabla \neq 0$ . This critic function is very simple, but, as we will show, there exists a  $\nabla > 0$  such that this leads to an objective function greater than 0 which means that the divergence is also greater than 0.

With this critic in mind, our goal is to transform the problem into the following:

$$\begin{aligned}
D_f(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} F(\mathbb{P}, \mathbb{Q}, C, f) \geq F(\mathbb{P}, \mathbb{Q}, C', f) \\
&\geq L(\nabla) \\
&> 0,
\end{aligned}$$

where  $L(\nabla) = af(\nabla) + bf(-\nabla)$ , for some  $a > 0$  and  $b > 0$  s.t.  $a > b$ . We have been able to show this with all divergences.

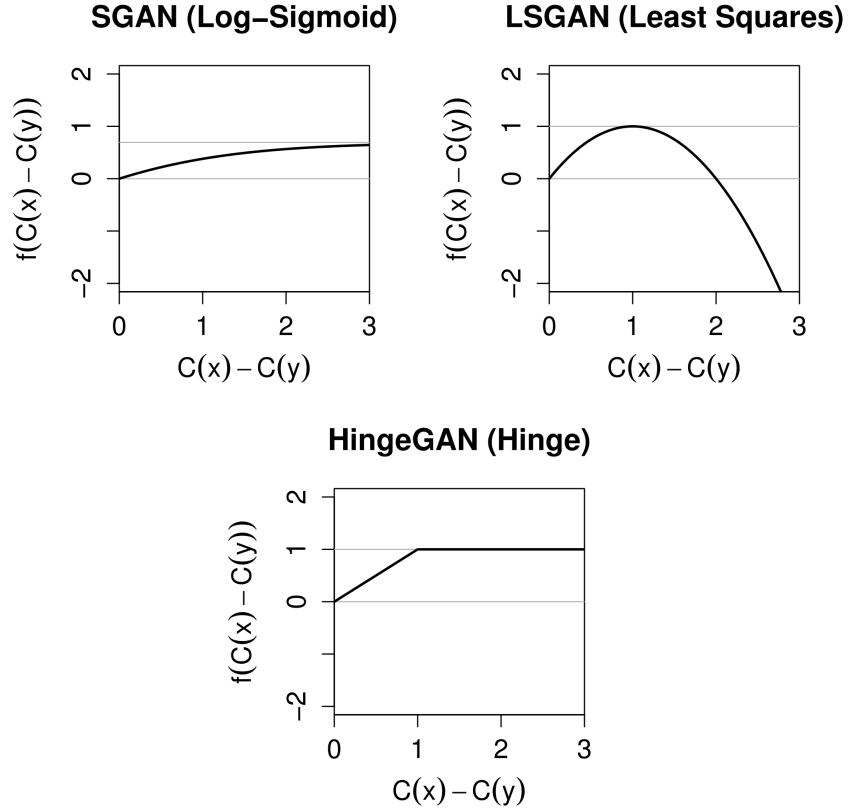
We want to find a  $\nabla > 0$  large enough so that the positive term ( $f(\nabla)$ ) is big, but small enough so that the negative term ( $f(-\nabla)$ ) is not too big. The main caveat is that, by concavity,  $f(\nabla) \leq |f(-\nabla)|$ . This means that the negative term is always bigger in absolute value than the positive term. This is problematic, since  $a$  could be very close to  $b$  and we want  $af(\nabla) > bf(-\nabla)$  to get  $L(\nabla) > 0$  which proves that we have a divergence. The solution is to choose  $\nabla$  to be very small. By continuity of the concave function, if we make  $\nabla$  small enough (very close to 0), we can reach a point where ( $f(\nabla) \approx -f(-\nabla)$ ). In which case, if  $a = b + \epsilon$ , we have that

$$\begin{aligned}
L(\nabla) &= af(\nabla) + bf(-\nabla) \approx af(\nabla) - bf(\nabla) \\
&= bf(\nabla) + \epsilon f(\nabla) - bf(\nabla) \\
&= \epsilon f(\nabla) \\
&> 0.
\end{aligned}$$

In the actual proof, we show that there always exists a  $\delta > 0$  small enough such that any  $\nabla \in (0, \delta)$  leads to  $L(\nabla) > 0$ . This concludes the sketch of the proof.

### 6.4.3 Subtypes of Divergences

Figure 1 shows three examples of concave  $f$  with the necessary properties to be used in relativistic divergences; they are the concave functions used in SGAN, LS-GAN (with labels 1/-1), and HingeGAN. Their respective mathematical functions



**Figure 6.1** – Plots of  $f$  with respect to the critic’s difference (CD) using three appropriate choices of  $f$  for relativistic divergences. The bottom gray line represents  $f(0) = 0$ ; the divergence is zero if all CDs are zero. The above gray line represents the maximum of  $f$ ; the divergence is maximized if all CDs leads to that maximum.

are

$$f_S(z) = \log(\text{sigmoid}(z)) + \log(2), \quad (6.11)$$

$$f_{LS}(z) = -(z - 1)^2 + 1, \quad (6.12)$$

$$f_{Hinge}(z) = -\max(0, 1 - z) + 1. \quad (6.13)$$

Interestingly, we see that they form three different types of functions. Firstly, we have functions that grow exponentially less as  $x$  increases and thus reach their supremum at  $\infty$ . Secondly, we have functions that grow to a maximum and then forever decrease (thus penalizing large CDs). Thirdly, we have functions that grow to a maximum and then never change. SGAN is of the first type, LSGAN is of the second, and HingeGAN is of the third type.

---

This shows that for all three types, we have that the CD is only encouraged to grow until a certain point. With the first type, we never truly force the CD to stop growing, but the gradients vanish to zero. Thus, SGD effectively prevents the CDs from growing above a certain level (sigmoid saturates at around 2 or 3).

It is useful to keep in mind that Figure 1 also represents the concave functions used for SyGANs, in which case  $f$  applies to real and fake data separately ( $f(x)$  and  $f(-y)$ ).

#### 6.4.4 Weakness of the Divergence

The paper by [Arjovsky et al. \(2017\)](#) on using the Wasserstein distance (and other IPMs) for GANs has been extremely influential. In this paper, the authors suggest that the Wasserstein distance is more appropriate than  $f$ -divergences for training a critic since it induces the weakest topology possible. Rather than giving a formal definition in terms of topologies, we use a simpler definition (as also done by [Arjovsky et al. \(2017\)](#)):

**Definition 6.4.2.** Let  $\mathbb{P}$  be a probability distribution with support  $\mathcal{X}$ ,  $(\mathbb{P}_n)_{n \in \mathbb{N}}$  be a sequence of distributions converging to  $\mathbb{P}$ , and  $D_1$  and  $D_2$  be statistical divergences (per definition 3.1).

We say that  $D_1$  is weaker than  $D_2$  if we have that:

$$D_2(\mathbb{P}_n, \mathbb{P}) \rightarrow 0 \implies D_1(\mathbb{P}_n, \mathbb{P}) \rightarrow 0 \quad \forall (\mathbb{P}_n)_{n \in \mathbb{N}},$$

but the converse is not true.

We say that  $D_1$  is a weakest distance if we have that:

$$D_1(\mathbb{P}_n, \mathbb{P}) \rightarrow 0 \iff \mathbb{P}_n \xrightarrow{D} \mathbb{P} \quad \forall (\mathbb{P}_n)_{n \in \mathbb{N}},$$

where  $\xrightarrow{D}$  represents convergence in distribution.

Thus, intuitively, a weaker divergence can be thought of as converging more easily. [Arjovsky et al. \(2017\)](#) showed that the Wasserstein distance is a weakest divergence and that it is weaker than common  $f$ -divergences (as used in  $f$ -GANs and standard GANs). They also showed that the Wasserstein distance is continuous with respect to its parameters and they attributed this property to the weakness of the divergence.



---

Considering this argument, one would expect that RaGANs would be weaker than RpGANs which would be weaker than Symmetric GANs since this is generally the order of their relative performance and stability (however, note that this is not always true and GANs can perform better than RaGANs). Instead, we found the opposite relationship:

**Theorem 2.** Let  $\mathbb{P}$  be a probability distribution with support  $S$ ,  $(\mathbb{P}_n)_{n \in \mathbb{N}}$  be a sequence of distributions converging to  $\mathbb{P}$ ,  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Then, we have that

$$\begin{aligned} D^W(\mathbb{P}, \mathbb{Q}) &\text{ is weakest,} \\ D^W(\mathbb{P}, \mathbb{Q}) &\text{ is weaker than } D_f^{Sy}(\mathbb{P}, \mathbb{Q}), \\ D_f^{Sy}(\mathbb{P}, \mathbb{Q}) &\text{ is weaker than } D_f^{Rp}(\mathbb{P}, \mathbb{Q}), \\ D_f^{Rp}(\mathbb{P}, \mathbb{Q}) &\text{ is weaker than } D_f^{Ra}(\mathbb{P}, \mathbb{Q}), \end{aligned}$$

where  $D^W$  is the Wasserstein distance and  $D^{Sy}$  is the distance in Symmetric GANs (see equation 3).

The proof is in Appendix B.

Given the good performance of RaGANs, this suggests that the argument made by Arjovsky et al. (2017) is insufficient. It only focuses on a perfect sequence of converging distributions, but the generator training does not guarantee a converging sequence of fake data distributions. It ignores the complex dynamics and intricacies of the generator training, which are still not well understood. Furthermore, it assumes an optimal critic which is effectively unobtainable. In practice, obtaining a semi-optimal critic requires training the critic for multiple iterations before training the generator; this significantly increase the computational time.

Furthermore, it has been found that WGAN does not provide a good approximation of the Wasserstein distance and that better approximations of the Wasserstein distance lead to worse GANs (Mallasto et al., 2019). This provides further argument towards the idea that the weakness of the divergence is not a good indicator of a good divergence for GANs. As previously suggested (Jolicoeur-Martineau, 2019), we hypothesize that what make WGAN good for GANs are likely 1) the constraint of the critic (a Lipschitz critic) and 2) the use of a relativistic discriminator, rather

than the weakness of the divergence.

---

## 6.5 Estimators

### 6.5.1 RpGANs

To estimate RpGANs, Jolicoeur-Martineau (2018) used the following estimator<sup>2</sup>:

$$\widehat{D}_f^{\text{Rp}}(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \frac{2}{k} \sum_{i=1}^k [f(C(x_i) - C(y_i))],$$

where  $x_1, \dots, x_k$  and  $y_1, \dots, y_k$  are samples from  $\mathbb{P}$  and  $\mathbb{Q}$  respectively.

Although this is an unbiased estimator of  $D_f^{\text{Rp}}(\mathbb{P}, \mathbb{Q})$ , it is not the estimator with the minimal variance for a given mini-batch. Using the two-sample version (Lehmann, 1951) of the U-statistic theorem (Hoeffding, 1992) and given that the loss function is symmetric with respect to its arguments, one can show the following:

**Corollary 1.** Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Let  $x_1, \dots, x_k$  and  $y_1, \dots, y_k$  be i.i.d. samples from  $\mathbb{P}$  and  $\mathbb{Q}$  respectively. Then, we have that

$$\widehat{D}_f^{\text{Rp}^*}(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \frac{2}{k^2} \sum_{i=1}^k \sum_{j=1}^k [f(C(x_i) - C(y_j))]$$

is the minimum-variance unbiased estimator (MVUE) of  $D_f^{\text{Rp}}(\mathbb{P}, \mathbb{Q})$ .

Although it is the MVUE, this estimator requires  $O(k^2)$  operations instead of  $O(k)$ . In the experiments, we will show that using this estimator does not lead to good performance. Given the quadratic scaling and lack of performance gain, it may not be worth using.

### 6.5.2 RaGANs and RalfGANs

The divergences of RaGANs and RalfGANs assume that one knows the true expectation of the critic of real and fake data. However, in practice, we can only es-

---

2. Note that they actually used  $\frac{1}{k}$  instead of  $\frac{2}{k}$  because of how they defined the divergence.

estimate the expectation. Although never explicitly mentioned, Jolicoeur-Martineau (2019) simply replaced all expectations by the mini-batch mean:

$$\mathbb{E}[C(x)] \approx \frac{1}{k} \sum_{i=1}^k C(x_i),$$

where  $k$  is the size of the mini-batch.

Given the non-linear function applied after calculating the CD, the divergences of RaGANs are biased with finite batch size  $k$ . This means that RaGANs are only asymptotically unbiased. How large  $k$  must be for the bias to become negligible is unclear.

We attempted to find a close form for the bias with  $f_S$ ,  $f_{LS}$ , and  $f_{Hinge}$  (equations 11, 12, 13 and Figure 1), but we were only able to find a closed form with  $f_{LS}$ . The bias with  $f_{LS}$  has a simple form and can be removed, as shown below:

**Corollary 2.** Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Then, we have that

$$\begin{aligned} & \sup_{C:\mathcal{X}\rightarrow\mathbb{R}} \frac{1}{k} \left( \hat{\sigma}_{C(x)} + \hat{\sigma}_{C(y)} - \sum_{i=1}^k \left[ (C(x_i) - \hat{\mu}_{C(y)} - 1)^2 \right] \right. \\ & \quad \left. - \sum_{j=1}^k \left[ (\hat{\mu}_{C(x)} - C(y_j) - 1)^2 \right] \right) + 2, \\ & \sup_{C:\mathcal{X}\rightarrow\mathbb{R}} \frac{2}{k} \left( \hat{\sigma}_{C(y)} - \sum_{i=1}^k \left[ (C(x_i) - \hat{\mu}_{C(y)} - 1)^2 \right] \right) + 1, \\ & \inf_{C:\mathcal{X}\rightarrow\mathbb{R}} \frac{1}{k} \left( \frac{1}{2} \hat{\sigma}_{C(x)} + \frac{1}{2} \hat{\sigma}_{C(y)} + \sum_{i=1}^k \left[ (C(x_i) - \hat{\mu}_C - 1)^2 \right] \right. \\ & \quad \left. + \sum_{j=1}^k \left[ (\hat{\mu}_C - C(y_j) - 1)^2 \right] \right) - 2 \end{aligned}$$

are unbiased estimator of  $D_{f_{LS}}^{Ra}(\mathbb{P}, \mathbb{Q})$ ,  $D_{f_{LS}}^{Ral}(\mathbb{P}, \mathbb{Q})$ , and  $D_{f_{LS}}^{Rc}(\mathbb{P}, \mathbb{Q})$  respectively. Furthermore,

$$\hat{\mu}_{C(x)} = \frac{1}{k} \sum_{i=1}^k C(x_i),$$

---


$$\hat{\mu}_{C(y)} = \frac{1}{k} \sum_{i=1}^k C(y_i),$$

$$\hat{\mu}_C = \frac{1}{k} \sum_{i=1}^k \left( \frac{C(x_i) + C(y_i)}{2} \right),$$

$$\hat{\sigma}_{C(x)} = \frac{1}{(k-1)} \sum_{i=1}^k (C(x_i) - \hat{\mu}_{C(x)})^2,$$

$$\hat{\sigma}_{C(y)} = \frac{1}{(k-1)} \sum_{i=1}^k (C(y_i) - \hat{\mu}_{C(y)})^2.$$

See Appendix C for the proof. This means that we can estimate the loss functions in RaLSGAN, RalFLSGAN, and RcLSGAN without bias. In the experiments, we will show that the bias is negligible with the usual choices of  $f$  (equations 11, 12, 13) and batch size (32 or higher).

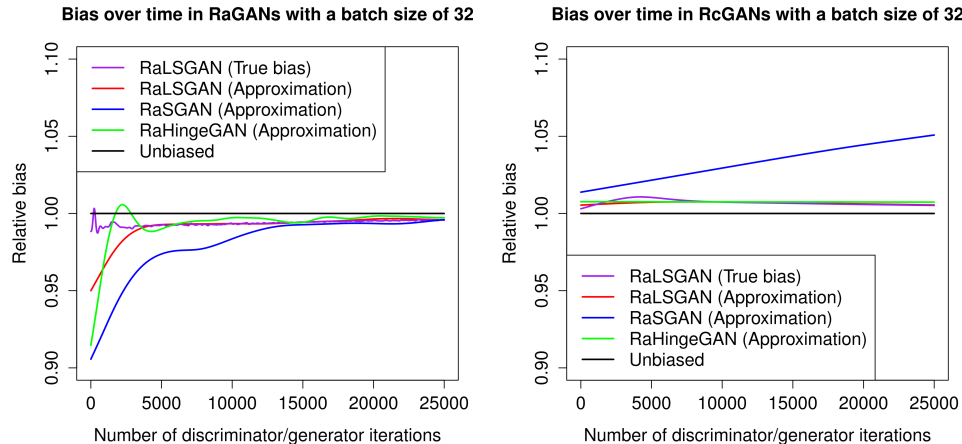
---

## 6.6 Experiments

All experiments were done with the spectral GAN architecture for 32x32 images (Miyato et al., 2018) in Pytorch (Paszke et al., 2017). We used the standard hyperparameters: learning rate (lr) = .0002, batch size (k) = 32, and the ADAM optimizer (Kingma and Ba, 2014) with parameters  $(\alpha_1, \alpha_2) = (.50, .999)$ . We trained the models for 100k iterations with one critic update per generator update. For the datasets, we used CIFAR-10 (50k training images from 10 categories) (Krizhevsky and Hinton, 2009), CelebA (200k of face images from celebrities) (Liu et al., 2015) and CAT (10k images of cats) (Zhang et al., 2008). All models were trained using the same seed (seed=1) with a single GPU. To evaluate the quality of generated outputs, we used the Fréchet Inception Distance (FID) (Heusel et al., 2017). For a review of the different evaluation metrics for GANs, please see Borji (2019). CAT was preprocessed by cropping all images to the faces of the cats, removing outliers (faces hidden by background), and removing images smaller than 32x32. CelebA images were center cropped to 160x160 before being resized to 32x32. See code for details; the code to reproduce the experiments is available on <https://github.com/AlexiaJM/relativistic-f-divergences>.

### 6.6.1 Bias

We approximated the bias of RaGANs and RcGANs by estimating the real/fake critic mean from 320 samples rather than the 32 mini-batch samples. For  $f_{LS}$ , we were able to calculate the true value of the bias (in expectation, see Corollary 4.2). Results on CIFAR-10 are shown in Figure 2.



**Figure 6.2** – Plots of the relative bias (i.e., the biased estimate divided by the unbiased estimate) of relativistic average and centered  $f$ -divergences estimators over training time on CIFAR-10 with a mini-batch size of 32. Approximations of the bias were made using 320 independent samples.

For RaGANs, the approximation of the relative bias with  $f_{LS}$  was correct from 4k iterations and onwards. For all choices of  $f$ , we observed the same pattern of low approximated relative bias which stabilized after a certain number of iterations. We suspect that this may be due to the important instabilities of the first iterations when the discriminator is not optimal. At 15k iterations, all biases were stabilized. We calculated the average of the bias with different  $f$  starting at 15k iterations: .995 for the true relative bias with  $f_{LS}$ , .996 for the approximated relative bias with  $f_{LS}$ , .994 for the approximated relative bias with  $f_S$ , and .997 for the approximated relative bias with  $f_{Hinge}$ .

For RcGANs, the approximation of the bias with  $f_{LS}$  was correct from the very beginning of training. All biases were relatively stable over time with the exception of  $f_S$  which increased linearly over time (up to around 1.05). We calculated the average of the bias with different  $f$ : 1.007 for the true relative bias with  $f_{LS}$ , 1.007 for the approximated relative bias with  $f_{LS}$ , 1.03 for the approximated relative bias with  $f_S$ , and 1.007 for the approximated relative bias with  $f_{Hinge}$ .

Overall, this shows that the bias in the estimators of RaGANs and RcGANs tends to be small. Furthermore, with the exception of  $f_S$ , the bias is relatively stable over time. Thus, accounting for the bias, may not be necessary.

## 6.6.2 Divergences

To test the new relativistic divergences proposed (and verify whether removing the bias in RaGANs is useful), we ran experiments on CIFAR-10 using  $f_{LS}$ , on LSUN bedrooms using  $f_{Hinge}$ , and on CAT using  $f_{Hinge}$  (these choices of  $f$  were arbitrary). Results are shown in Table 1.

**Table 6.1** – Minimum (and standard deviation) of the FID calculated at 10k, 20k, ... , 100k iterations using different loss functions (see equations 11, 12, 13) and datasets.

Loss	CIFAR-10	CelebA	CAT
	$f_{LS}$	$f_{Hinge}$	$f_S$
GAN	31.1 (8)	15.3 (52)	15.2 (11)
RpGAN	31.5 (8)	16.7 (4)	12.9 (2)
RpGAN <sub>MVUE</sub>	30.2 (12)	21.9 (3)	18.2 (3)
RaGAN	<b>29.2 (7)</b>	<b>15.9 (5)</b>	<b>12.3 (1)</b>
RaGAN <sub>unbiased</sub>	30.3 (13)	-	-
RcGAN	31.7 (8)	18.1 ( <b>3</b> )	16.5 (7)
RcGAN <sub>unbiased</sub>	32.3 (9)	-	-

Using the MVUE for RpGAN resulted in the generator having a worse performance on CIFAR-10 with  $f_{LS}$  ( $\beta = .37$ ,  $p = .72$ ), CelebA with  $f_{Hinge}$  ( $\beta = 2.08$ ,  $p = .07$ ), and CAT with  $f_S$  ( $\beta = 4.02$ ,  $p = .003$ ). Similarly, using the unbiased estimator made the generator perform slightly worse for RaLSGAN ( $\beta = 2.37$ ,  $p = .04$ ) and RcLSGAN ( $\beta = 1.33$ ,  $p = .05$ ). These results are surprising as they suggest that using noisy or slightly biased estimators may be beneficial.

---

## 6.7 Conclusion

Most importantly, we proved that the objective function of the critic in RGANs is a divergence. In addition, we showed that  $f$ -divergences are weaker than relativistic  $f$ -divergences. Thus, the weakness of the topology induced by a divergence

---

alone cannot explain why WGAN performs well. Finally, we took a closer look at the estimators of RGANs and found that 1) the estimator of RpGANs used by [Jolicœur-Martineau \(2019\)](#) is not the minimum-variance unbiased estimator (MVUE) and 2) the estimators of RaGANs and RalfGANs are slightly biased with finite batch-sizes. Surprisingly, we found that neither using the MVUE with RpGANs or using an unbiased estimator with RaGANs and RalfGANs improved the performance. On the contrary, using better estimators always slightly decreased the quality of generated samples. This suggests that using noisy estimates of the divergences may be beneficial as a regularization mechanism. This could be explained by vanishing gradients when the discriminator becomes closer to optimality ([Arjovsky and Bottou, 2017](#)).

It still remains a mystery as to why RaGANs are better than RpGANs and the direct mechanism that leads to RGANs performing in a much more stable matter. Future work should attempt to better understand the effect of the critic’s difference on training. Our experiments were limited to the generation of small images; thus, we encourage further experiments with the MVUE and the unbiased estimator of RaLSGAN.

# 7

## Prologue to the third article

---

### 7.1 Article Details

Jolicoeur-Martineau, A.\* , Piché-Taillefer, R.\* , Mitliagkas, I., Tachet des Combes, R. (2021). [Adversarial score matching and improved sampling for image generation](#). *International Conference on Learning Representations (ICLR)*.

\* Equal contribution

---

### 7.2 Context

At the time, score-based diffusion models had just come up, and they did not produce as high-quality samples as GANs. I came up with the idea of encouraging realistic images from the point-of-view of a discriminator, as in GANs, to help ensure that the images generated by the score network are high quality.

Generating data with score-based diffusion models was also very unstable, and it was unclear how to properly select the noise in the sampling stage (this was later determined theoretically in [Song et al. \(2021\)](#)). This led Rémi and I to investigate the matter, which then became the Consistent Annealed Sampling proposed in the paper.

---

### 7.3 Personal contribution to the paper

I proposed the adversarial score-matching approach and initiated the project. Rémi Piché-Taillefer proposed the Consistent Annealed Sampling. Me and Rémi Piché-Taillefer worked equally on the writing and experiments. Ioannis Mitliagkas



---

supervised the project and helped with the writing. Rémi Tachet des Combes helped with the writing and the theoretical proofs/derivations.

# Adversarial score matching and improved sampling for image generation

---

## 8.1 Abstract

Denoising Score Matching with Annealed Langevin Sampling (DSM-ALS) has recently found success in generative modeling. The approach works by first training a neural network to estimate the score of a distribution, and then using Langevin dynamics to sample from the data distribution assumed by the score network. Despite the convincing visual quality of samples, this method appears to perform worse than Generative Adversarial Networks (GANs) under the Fréchet Inception Distance, a standard metric for generative models. We show that this apparent gap vanishes when *denoising* the final Langevin samples using the score network. In addition, we propose two improvements to DSM-ALS: 1) Consistent Annealed Sampling as a more stable alternative to Annealed Langevin Sampling, and 2) a hybrid training formulation, composed of both Denoising Score Matching and adversarial objectives. By combining these two techniques and exploring different network architectures, we elevate score matching methods and obtain results competitive with state-of-the-art image generation on CIFAR-10.

---

## 8.2 Introduction

Song and Ermon (2019) recently proposed a novel method of generating samples from a target distribution through a combination of Denoising Score Matching (DSM) (Hyvärinen, 2005; Vincent, 2011; Raphan and Simoncelli, 2011) and Annealed Langevin Sampling (ALS) (Welling and Teh, 2011; Roberts et al., 1996). Since convergence to the distribution is guaranteed by the ALS, their approach (DSM-ALS) produces high-quality samples and guarantees high diversity. Though, this comes at the cost of requiring an iterative process during sampling, contrary

---

to other generative methods. These generative methods can notably be used to diverse tasks like colorization, image restoration and image inpainting (Song and Ermon, 2019; Kadkhodaie and Simoncelli, 2020).

Song and Ermon (2020) further improved their approach by increasing the stability of score matching training and proposing theoretically sound choices of hyperparameters. They also scaled their approach to higher-resolution images and showed that DSM-ALS is competitive with other generative models. Song and Ermon (2020) observed that the images produced by their improved model were more visually appealing than the ones from their original work; however, the reported Fréchet Inception Distance (FID) (Heusel et al., 2017) did not correlate with this improvement.

Although DSM-ALS is gaining traction, Generative adversarial networks (GANs) (Goodfellow et al., 2020) remain the leading approach to generative modeling. GANs are a very popular class of generative models; they have been successfully applied to image generation (Brock et al., 2019; Karras et al., 2018, 2019, 2020) and have subsequently spawned a wealth of variants (Radford et al., 2015; Miyato et al., 2018; Jolicoeur-Martineau, 2019; Zhang et al., 2019). The idea behind this method is to train a Discriminator ( $D$ ) to correctly distinguish real samples from fake samples generated by a second agent, known as the Generator ( $G$ ). GANs excel at generating high-quality samples as the discriminator captures features that make an image plausible, while the generator learns to emulate them.

Still, GANs often have trouble producing data from all possible modes, which limits the diversity of the generated samples. A wide variety of tricks have been developed to address this issue in GANs (Kodali et al., 2017; Gulrajani et al., 2017; Arjovsky et al., 2017; Miyato et al., 2018; Jolicoeur-Martineau and Mitliagkas, 2019), though it remains an issue to this day. DSM-ALS, on the other hand, does not suffer from that problem since ALS allows for sampling from the full distribution captured by the score network. Nevertheless, the perceptual quality of DSM-ALS higher-resolution images has so far been inferior to that of GAN-generated images. Generative modeling has since seen some incredible work from Ho et al. (2020), who achieved exceptionally low (better) FID on image generation tasks. Their approach showcased a diffusion-based method (Sohl-Dickstein et al., 2015; Goyal et al., 2017) that shares close ties with DSM-ALS, and additionally proposed a convincing network architecture derived from Salimans et al. (2017).

---

In this paper, after introducing the necessary technical background in the next section, we build upon the work of Song and Ermon (2020) and propose improvements based on theoretical analyses both at training and sampling time. Our contributions are as follows:

- We propose Consistent Annealed Sampling (CAS) as a more stable alternative to ALS, correcting inconsistencies relating to the scaling of the added noise;
- We show how to recover the *expected denoised sample* (EDS) and demonstrate its unequivocal benefits *w.r.t* the FID. Notably, we show how to resolve the mismatch observed in DSM-ALS between the visual quality of generated images and its high (worse) FID;
- We propose to further exploit the EDS through a hybrid objective function, combining GAN and Denoising Score Matching objectives, thereby encouraging the EDS of the score network to be as realistic as possible.

In addition, we show that the network architecture used by Ho et al. (2020) significantly improves sample quality over the RefineNet (Lin et al., 2017) architecture used by Song and Ermon (2020). In an ablation study performed on CIFAR-10 and LSUN-church, we demonstrate how these contributions bring DSM-ALS in range of the state-of-the-art for image generation tasks *w.r.t.* the FID.

---

## 8.3 Background

### 8.3.1 Denoising Score Matching

Denoising Score Matching (DSM) (Hyvärinen, 2005) consists of training a score network to approximate the gradient of the log density of a certain distribution ( $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ ), referred to as the score function. This is achieved by training the network to approximate a noisy surrogate of  $p$  at multiple levels of Gaussian noise corruption (Vincent, 2011). The score network  $s$ , parametrized by  $\theta$  and condi-

---

tioned on the noise level  $\sigma$ , is tasked to minimize the following loss:

$$\frac{1}{2} \mathbb{E}_{p(\tilde{\mathbf{x}}, \mathbf{x}, \sigma)} \left[ \left\| \sigma s_\theta(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma} \right\|_2^2 \right], \quad (8.1)$$

where  $p(\tilde{\mathbf{x}}, \mathbf{x}, \sigma) = q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p(\mathbf{x})p(\sigma)$ . We define further  $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 \mathbf{I})$  the corrupted data distribution,  $p(\mathbf{x})$  the training data distribution, and  $p(\sigma)$  the uniform distribution over a set  $\{\sigma_i\}$  corresponding to different levels of noise. In practice, this set is defined as a geometric progression between  $\sigma_1$  and  $\sigma_L$  (with  $L$  chosen according to some computational budget):

$$\{\sigma_i\}_{i=1}^L = \left\{ \gamma^i \sigma_1 \mid i \in \{0, \dots, L-1\}, \gamma \triangleq \frac{\sigma_2}{\sigma_1} = \dots = \left( \frac{\sigma_L}{\sigma_1} \right)^{\frac{1}{L-1}} < 1 \right\}. \quad (8.2)$$

Rather than having to learn a different score function for every  $\sigma_i$ , one can train an unconditional score network by defining  $s_\theta(\tilde{\mathbf{x}}, \sigma_i) = s_\theta(\tilde{\mathbf{x}})/\sigma_i$ , and then minimizing Eq. 8.1. While unconditional networks are less heavy computationally, it remains an open question whether conditioning helps performance. Li et al. (2019) and Song and Ermon (2020) found that the unconditional network produced better samples, while Ho et al. (2020) obtained better results than both of them using a conditional network. Additionally, the denoising autoencoder described in Lim et al. (2020) gives evidence supporting the benefits of conditioning when the noise becomes small (also see App. C.5 and C.4 for a theoretical discussion of the difference). While our experiments are conducted with unconditional networks, we believe our techniques can be straightforwardly applied to conditional networks; we leave that extension for future work.

### 8.3.2 Annealed Langevin Sampling

Given a score function, one can use Langevin dynamics (or Langevin sampling) (Welling and Teh, 2011) to sample from the corresponding probability distribution. In practice, the score function is generally unknown and estimated through a score network trained to minimize Eq. 8.1. Song and Ermon (2019) showed that Langevin sampling has trouble exploring the full support of the distribution when the modes are too far apart and proposed Annealed Langevin Sampling (ALS) as a solution. ALS starts sampling with a large noise level and progressively anneals

it down to a value close to 0, ensuring both proper mode coverage and convergence to the data distribution. Its precise description is shown in Algorithm 4.

Algorithm 4 ALS	Algorithm 5 CAS
<b>Require:</b> $s_\theta, \{\sigma_i\}_{i=1}^L, \epsilon, n_\sigma$	<b>Require:</b> $s_\theta, \{\sigma_i\}_{i=1}^L, \gamma, \epsilon$
1: Initialize $\mathbf{x}$	1: Initialize $\mathbf{x}$
2: <b>for</b> $i \leftarrow 1$ to $L$ <b>do</b>	2: $\beta \leftarrow \sqrt{1 - (1 - \epsilon/\sigma_L^2)^2 / \gamma^2}$
3: $\alpha_i \leftarrow \epsilon \sigma_i^2 / \sigma_L^2$	3: <b>for</b> $i \leftarrow 1$ to $L$ <b>do</b>
4: <b>for</b> $n_\sigma$ <b>steps do</b>	4: $\alpha_i \leftarrow \epsilon \sigma_i^2 / \sigma_L^2$
5: Draw $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$	5: Draw $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
6: $\mathbf{x} \leftarrow \mathbf{x} + \alpha_i s_\theta(\mathbf{x}, \sigma_i) + \sqrt{2\alpha_i} \mathbf{z}$	6: $\mathbf{x} \leftarrow \mathbf{x} + \alpha_i s_\theta(\mathbf{x}, \sigma_i) + \beta \sigma_{i+1} \mathbf{z}$
<b>return</b> $\mathbf{x}$	<b>return</b> $\mathbf{x}$

### 8.3.3 Expected denoised sample (EDS)

A little known fact from Bayesian literature is that one can recover a denoised sample from the score function using the Empirical Bayes mean (Robbins, 1955; Miyasawa, 1961; Raphan and Simoncelli, 2011):

$$s^*(\tilde{\mathbf{x}}, \sigma) = \frac{H^*(\tilde{\mathbf{x}}, \sigma) - \tilde{\mathbf{x}}}{\sigma^2}, \quad (8.3)$$

where  $H^*(\tilde{\mathbf{x}}, \sigma) \triangleq \mathbb{E}_{\mathbf{x} \sim q_\sigma(\mathbf{x}|\tilde{\mathbf{x}})}[\mathbf{x}]$  is the expected denoised sample given a noisy sample (or Empirical Bayes mean), conditioned on the noise level. A different way of reaching the same result is through the closed-form of the optimal score function, as presented in Appendix C.5. The corresponding result for unconditional score function is presented in Appendix C.4 for completeness.

The EDS corresponds to the expected real image given a corrupted image; it can be thought of as what the score network believes to be the true image concealed within the noisy input. It has also been suggested that denoising the samples (i.e., taking the EDS) at the end of the Langevin sampling improves their quality (Saremi and Hyvarinen, 2019; Li et al., 2019; Kadkhodaie and Simoncelli, 2020). In Section 8.5, we provide further evidence that denoising the final Langevin sample brings it closer to the assumed data manifold. In particular, we show that the Fréchet Inception Distance (FID) consistently decreases (improves) after denoising. Finally, in Section 8.6, we build a hybrid training objective using the properties of the EDS

---

discussed above.

There are interesting links to be made between ALS and the RED algorithm (Romano et al., 2017; Reehorst and Schniter, 2018). The RED algorithm attempts to find the maximum a posteriori probability (MAP) denoised sample (i.e., the most plausible real data) given a noisy sample. It does so by solving an optimization problem to obtain a sample close to the noisy sample for which the EDS is a fixed point (denoising the sample does not change it because it is a real sample). Thus, just like ALS, the RED algorithm generates plausible real data given a score network. However, this algorithm does not ensure that we sample from the distribution and obtain full mode coverage. Thus, ALS’s key benefit is ensuring that we sample from the full support of the distribution.

---

## 8.4 Consistent scaling of the noise

In this section, we present inconsistencies in ALS relating to the noise scaling and introduce Consistent Annealed Sampling (CAS) as an alternative.

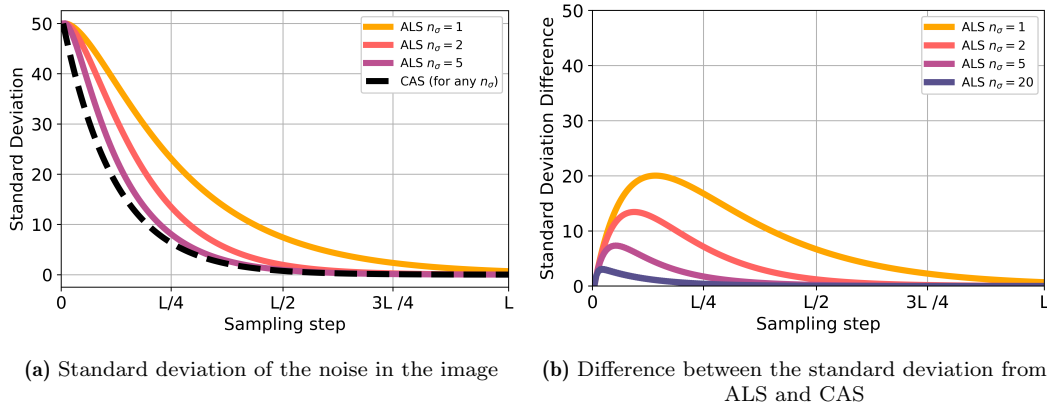
### 8.4.1 Inconsistencies in ALS

One can think of the ALS algorithm as a sequential series of Langevin Dynamics (inner loop in Algorithm 4) for decreasing levels of noise (outer loop). If allowed an infinite number of steps  $n_\sigma$ , the sampling process will properly produce samples from the data distribution.

In ALS, the score network is conditioned on geometrically decreasing noise ( $\sigma_i$ ). In the unconditional case, this corresponds to dividing the score network by the noise level (i.e.,  $s_\theta(\tilde{\mathbf{x}}, \sigma_i) = s_\theta(\tilde{\mathbf{x}})/\sigma_i$ ). Thus, in both conditional and unconditional cases, we make the assumption that the noise of the sample at step  $i$  will be of variance  $\sigma_i^2$ , an assumption upon which the quality of the estimation of the score depends. While choosing a geometric progression of noise levels seems like a reasonable (though arbitrary) schedule to follow, we show that ALS does not ensure such schedule.

Assume we have the true score function  $s^*$  and begin sampling using a real image with some added zero-centered Gaussian noise of standard deviation  $\sigma_0 = 50$ .

In Figure 8.1a, we illustrate how the intensity of the noise in the sample evolves through ALS and CAS, our proposed sampling, for a given sampling step size  $\epsilon$  and a geometric schedule in this idealized scenario. We note that, although a large  $n_\sigma$  approaches the real geometric curve, it will only reach it at the limit ( $n_\sigma \rightarrow \infty$  and  $\epsilon \rightarrow 0$ ). Most importantly, Figure 8.1b highlights how even when the annealing process does converge, the progression of the noise is never truly geometric; we prove this formally in Proposition 1.



**Figure 8.1** – Standard deviation during idealized sampling using a perfect score function  $s^*$ . The black curve in (a) corresponds to the true geometric progression, as demonstrated in Proposition 2.

**Proposition 1.** *Let  $s^*$  be the optimal score function from Eq. 8.3. Following the sampling described in Algorithm 4, the variance of the noise component in the sample  $\mathbf{x}$  will remain greater than  $\sigma_t^2$  at every step  $t$ .*

The proof is presented in Appendix C.6. In particular, for  $n_\sigma < \infty$ , sampling has not fully converged and the remaining noise is carried over to the next iteration of Langevin Sampling. It also follows that for any  $s_\theta$  different from the optimal  $s^*$ , the actual noise at every iteration is expected to be even higher than for the best possible score function  $s^*$ .

## 8.4.2 Algorithm

We propose Consistent Annealed Sampling (CAS) as a sampling method that ensures the noise level will follow a prescribed schedule for any sampling step size  $\epsilon$  and number of steps  $L$ . Algorithm 5 illustrates the process for a geometric schedule.



---

Note that for a different schedule,  $\beta$  will instead depend on the step  $t$ , as in the general case,  $\gamma_t$  is defined as  $\sigma_{t+1}/\sigma_t$ .

**Proposition 2.** *Let  $s^*$  be the optimal score function from Eq. 8.3. Following the sampling described in Algorithm 5, the variance of the noise component in the sample  $\mathbf{x}$  will consistently be equal to  $\sigma_t^2$  at every step  $t$ .*

The proof is presented in Appendix C.7. Importantly, Proposition 2 holds no matter how many steps  $L$  we take to decrease the noise geometrically. For ALS,  $n_\sigma$  corresponds to the number of steps per level of noise. It plays a similar role in CAS: we simply dilate the geometric series of noise levels used during training by a factor of  $n_\sigma$ , such that  $L_{\text{sampling}} = (L_{\text{training}} - 1)n_\sigma + 1$ . Note that the proposition only holds when the initial sample is a corrupted image (*i.e.*,  $\mathbf{x}_0 = \mathcal{I} + \sigma_0 \mathbf{z}_0$ ). However, by defining  $\sigma_0$  as the maximum Euclidean distance between all pairs of training data points (Song and Ermon, 2020), the noise becomes in practice much greater than the true image; sampling with pure noise initialization (*i.e.*,  $\mathbf{x}_0 = \sigma_0 \mathbf{z}_0$ ) becomes indistinguishable from sampling with data initialization.

---

## 8.5 Benefits of the EDS on synthetic data and image generation

As previously mentioned, it has been suggested that one can obtain better samples (closer to the assumed data manifold) by taking the EDS of the last Langevin sample. We provide further evidence of this with synthetic data and standard image datasets.

It can first be observed that the sampling steps correspond to an interpolation between the previous point and the EDS, followed by the addition of noise.

**Proposition 3.** *Given a noise-conditional score function, the update rules from Algorithm 4 and Algorithm 5 are respectively equivalent to the following update rules:*

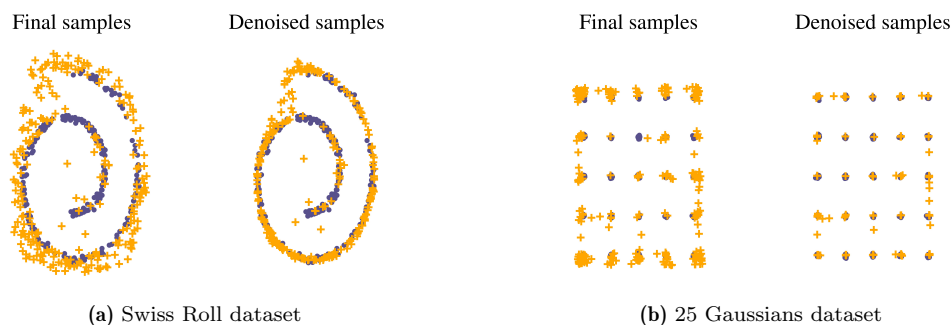
$$\begin{aligned} \mathbf{x} &\leftarrow (1 - \eta)\mathbf{x} + \eta H(\mathbf{x}, \sigma_i) + \sqrt{2\eta}\sigma_i \mathbf{z} && \text{for } \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \text{ and } \eta = \frac{\epsilon}{\sigma_L^2} \\ \mathbf{x} &\leftarrow (1 - \eta)\mathbf{x} + \eta H(\mathbf{x}, \sigma_i) + \beta \sigma_{i+1} \mathbf{z} \end{aligned}$$

The demonstration is in Appendix C.8. This result is equally true for an unconditional score network, with the distinction that  $\eta$  would no longer be independent of  $\sigma_i$  but rather linearly proportional to it.

Intuitively, this implies that the sampling steps slowly move the current sample towards a moving target (the EDS). If the sampling behaves appropriately, we expect the final sample  $\mathbf{x}$  to be very close to the EDS, *i.e.*,  $\mathbf{x} \approx H(\mathbf{x}, \sigma_L)$ . However, if the sampling step size is inappropriate, or if the EDS does not stabilize to a fixed point near the end of the sampling, these two quantities may be arbitrarily far from one another. As we will show, the FIDs from Song and Ermon (2020) suffer from such distance.

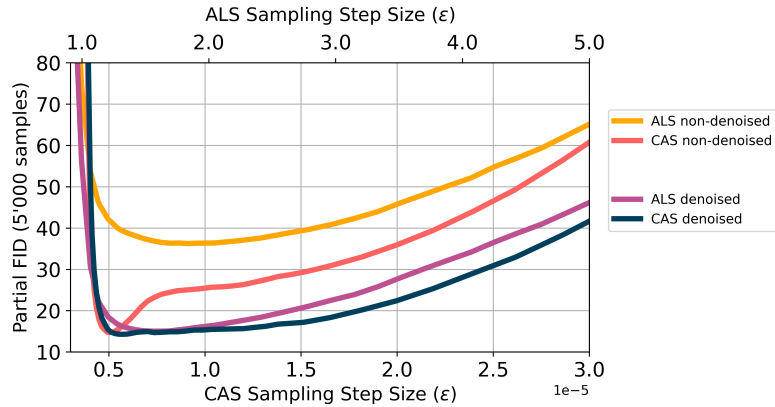
From Proposition 3, we see that CAS shares some similarities with the algorithm by Kadkhodaie and Simoncelli (2020). While the weight we give to the denoiser ( $\eta$ ) decreases geometrically (by its linearity in  $\sigma$ ), their schedule appears to be much steeper. They also estimate the residual noise in their samples by the  $\ell_2$  norm instead of determining it through a schedule, as CAS strives to do. As a note, we had found weak evidence during development that estimating the residual noise worsened the FID.

The equivalence showed in Proposition 3 suggests instead to take the expected denoised sample at the end of the Langevin sampling as the final sample; this would be equivalent to the update rule  $\mathbf{x} \leftarrow H(\mathbf{x}, \sigma_L)$  at the last step. Synthetic 2D examples shown in Figure 8.2 demonstrate the immediate benefits of this technique.



**Figure 8.2** – Langevin sampling on synthetic 2D experiments. Circles are real data points, crosses are generated data points. On both datasets, taking the EDS brings the samples much closer to the real data manifold.

We train a score network on CIFAR-10 (Krizhevsky and Hinton, 2009) and report the FID from both ALS and CAS as a function of the sampling step size and of denoising in Figure 8.3. The first observation to be made is just how critical



**Figure 8.3** – Partial estimate of FID (lower is better) as a function of the sampling step size on CIFAR-10, with  $n_\sigma = 1$ . The interactions between *consistent sampling* and denoising are shown.

denoising is to the FID score for ALS, even as its effect cannot be perceived by the human eye. For CAS, we note that the score remains small for a much wider range of sampling step sizes when denoising. Alternatively, the sampling step size must be very carefully tuned to obtain results close to the optimal.

Figure 8.3 also shows that, with CAS, the FID of the final sample is approximately equal to the FID of the denoised samples for small sampling step sizes. Furthermore, we see a smaller gap in FID between denoised and non-denoised for larger sampling step sizes than ALS. This suggests that consistent sampling is resulting in the final sample being closer to the assumed data manifold (i.e.,  $\mathbf{x} \approx H_\theta(\mathbf{x}, \sigma_L)$ ).

Interestingly, when Song and Ermon (2020) improved their score matching method, they could not explain why the FID of their new model did not improve even though the generated images looked better visually. To resolve that matter, they proposed the use of a new metric (Zhou et al., 2019) that did not have this issue. As shown in Figure 8.3, denoising resolves this mismatch.

---

## 8.6 Adversarial formulation

The score network is trained to recover an uncorrupted image from a noisy input minimizing the  $\ell_2$  distance between the two. However, it is well known from the image restoration literature that  $\ell_2$  does not correlate well with human perception

---

of image quality (Zhang et al., 2012; Zhao et al., 2016). One way to take advantage of the EDS would be to encourage the score network to produce an EDS that is more realistic from the perspective of a discriminator. Intuitively, this would incentivize the score network to produce more discernible features at inference time.

We propose to do so by training the score network to simultaneously minimize the score-matching loss function and maximize the probability of denoised samples being perceived as real by a discriminator. We use alternating gradient descent to sequentially train a discriminator for a determined number of steps at every score function update.

In our experiments, we selected the Least Squares GAN (LSGAN) (Mao et al., 2017) formulation as it performed best (see Appendix C.2 for details). For an unconditional score network, the objective functions are as follows:

$$\max_{\phi} \mathbb{E}_{p(\mathbf{x})} [(D_{\phi}(\mathbf{x}) - 1)^2] + \mathbb{E}_{p(\tilde{\mathbf{x}}, \mathbf{x}, \sigma)} [(D_{\phi}(H_{\theta}(\tilde{\mathbf{x}}, \sigma) + 1)^2] \quad (8.4)$$

$$\min_{\theta} \mathbb{E}_{p(\tilde{\mathbf{x}}, \mathbf{x}, \sigma)} \left[ (D_{\phi}(H_{\theta}(\tilde{\mathbf{x}}, \sigma)) - 1)^2 + \frac{\lambda}{2} \left\| \sigma s_{\theta}(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma} \right\|_2^2 \right], \quad (8.5)$$

where  $H_{\theta}(\tilde{\mathbf{x}}, \sigma) = s_{\theta}(\tilde{\mathbf{x}}, \sigma)\sigma^2 + \tilde{\mathbf{x}}$  is the EDS derived from the score network. Eq. 8.4 is the objective function of the LSGAN discriminator, while Eq. 8.5 is the adversarial objective function of the score network derived from Eq. 8.1 and from the LSGAN objective function.

We note the similarities between these objective functions and those of an LSGAN adversarial autoencoder (Makhzani et al., 2015; Tolstikhin et al., 2018; Tran et al., 2018), with the distinction of using a denoising autoencoder  $H$  as opposed to a standard autoencoder. We can highlight this difference by reformulating Eq. 8.5 as:

$$\min_{\theta} \mathbb{E}_{p(\tilde{\mathbf{x}}, \mathbf{x}, \sigma)} \left[ (D_{\phi}(H_{\theta}(\tilde{\mathbf{x}}, \sigma)) - 1)^2 + \frac{\lambda}{2\sigma^2} \|H_{\theta}(\tilde{\mathbf{x}}, \sigma) - \mathbf{x}\|_2^2 \right], \quad (8.6)$$

As GANs favor quality over diversity, there is a concern that this hybrid objective function might decrease the diversity of samples produced by the ALS. In Section 8.7.1, we first study image generation improvements brought by this method

and then address the diversity concerns with experiments on the 3-StackedMNIST (Metz et al., 2017) dataset in Section 8.7.2.

---

## 8.7 Experiments

### 8.7.1 Ablation Study

We ran experiments on CIFAR-10 (Krizhevsky and Hinton, 2009) and LSUN-churches (Yu et al., 2015) with the score network architecture used by Song and Ermon (2020). We also ran similar experiments with an unconditional version of the network architecture by Ho et al. (2020), given that their approach is similar to Song and Ermon (2019) and they obtain very small FIDs. For the hybrid adversarial score matching approach, we used an unconditional BigGAN discriminator (Brock et al., 2019). We compared three factors in an ablation study: adversarial training, Consistent Annealed Sampling and denoising.

Details on how the experiments were conducted are found in Appendix C.2. Unsuccessful experiments with large images are also discussed in Appendix C.3. See also Appendix C.9 for a discussion pertaining to the use of the Inception Score (Heusel et al., 2017), a popular metric for generative models.

Results for CIFAR-10 and LSUN-churches with Song and Ermon (2019) score network architecture are respectively shown in Table 8.1 and 8.2. Results for CIFAR-10 with Ho et al. (2020) score network architecture are shown in Table 8.3.

**Table 8.1** – [Non-denoised / Denoised FID] from 10k samples on CIFAR-10 (32x32) with Song and Ermon (2019) score network architecture

Sampling	Non-adversarial	Adversarial
non-consistent ( $n_\sigma = 1$ )	36.3 / 13.3	30.0 / 11.8
non-consistent ( $n_\sigma = 5$ )	33.7 / 10.9	26.4 / <b>9.5</b>
consistent ( $n_\sigma = 1$ )	14.7 / 12.3	11.9 / 10.8
consistent ( $n_\sigma = 5$ )	12.7 / 11.2	9.9 / 9.7

We always observe an improvement in FID from denoising and by increasing  $n_\sigma$  from 1 to 5. We observe an improvement from using the adversarial approach with

**Table 8.2** – [Non-denoised / Denoised FID] from 10k samples on LSUN-Churches (64x64) with Song and Ermon (2019) score network architecture

Sampling	Non-adversarial	Adversarial
non-consistent ( $n_\sigma = 1$ )	43.2 / 40.3	40.9 / 36.7
non-consistent ( $n_\sigma = 5$ )	42.0 / 39.2	40.0 / 35.8
consistent ( $n_\sigma = 1$ )	41.5 / 40.7	38.2 / 36.7
consistent ( $n_\sigma = 5$ )	39.5 / 39.1	36.3 / <b>35.4</b>

**Table 8.3** – [Non-denoised / Denoised FID] from 10k samples on CIFAR-10 (32x32) with Ho et al. (2020) unconditional score network architecture

Sampling	Non-adversarial	Adversarial
non-consistent ( $n_\sigma = 1$ )	25.3 / 7.5	21.6 / 7.5
non-consistent ( $n_\sigma = 5$ )	20.0 / <b>5.6</b>	17.7 / 6.1
consistent ( $n_\sigma = 1$ )	7.8 / 7.1	7.7 / 7.1
consistent ( $n_\sigma = 5$ )	6.2 / 6.1	6.1 / 6.5

Song and Ermon (2019) network architecture, but not on denoised samples with the Ho et al. (2020) network architecture. We hypothesize that this is a limitation of the architecture of the discriminator since, as far as we know, no variant of BigGAN achieves an FID smaller than 6. Nevertheless, it remains advantageous for more simple architectures, as shown in Table 8.1 and 8.2. We observe that consistent sampling outperforms non-consistent sampling on the CIFAR-10 task at  $n_\sigma = 1$ , the quickest way to sample.

We calculated the FID of the non-consistent denoised models from 50k samples in order to compare our method with the recent work from Ho et al. (2020). We obtained a score of 3.65 for the non-adversarial method and 4.02 for the adversarial method on the CIFAR-10 task when sharing their architecture; these scores are close to their reported 3.17. Although not explicit in their approach, Ho et al. (2020) denoised their final sample. This suggests that taking the EDS and using an architecture akin to theirs were the two main reasons for outperforming Song and Ermon (2020). Of note, our method only trains the score network for 300k iterations, while Ho et al. (2020) trained their networks for more than 1 million iterations to achieve similar results.

---

## 8.7.2 Non-adversarial and Adversarial score networks have equally high diversity

To assess the diversity of generated samples, we evaluate our models on the 3-Stacked MNIST generation task (Metz et al., 2017) (128k images of 28x28), consisting of numbers from the MNIST dataset (LeCun et al., 1998) superimposed on 3 different channels. We trained non-adversarial and adversarial score networks in the same way as the other models. The results are shown in Table 8.4.

We see that each of the 1000 modes is covered, though the KL divergence is still inferior to PACGAN (Lin et al., 2018), meaning that the mode proportions are not perfectly uniform. Blindness to mode proportions is thought to be a fundamental limitation of score-based methods (Wenliang, 2020). Nevertheless, these results confirm a full mode coverage on a task where most GANs struggle and, most importantly, that using a hybrid objective does not hurt the diversity of the generated samples.

3-Stacked MNIST		
	Modes (Max 1000)	KL
DCGAN (Radford et al., 2015)	99.0	3.40
ALI (Dumoulin et al., 2017)	16.0	5.40
Unrolled GAN (Metz et al., 2017)	48.7	4.32
VEEGAN (Srivastava et al., 2017)	150.0	2.95
PacDCGAN2 (Lin et al., 2018)	1000.0	0.06
WGAN-GP (Gulrajani et al., 2017)	959.0	0.73
PresGAN (Dieng et al., 2019)	999.6	0.115
MEG (Kumar et al., 2019)	1000.0	0.03
Non-adversarial DSM (ours)	1000.0	1.36
Adversarial DSM (ours)	1000.0	1.49

**Table 8.4** – As in Lin et al. (2018), we generated 26k samples and evaluated the mode coverage and KL divergence based on the predicted modes from a pre-trained MNIST classifier.

---

## 8.8 Conclusion

We proposed Consistent Annealed Sampling as an alternative to Annealed Langevin Sampling, which ensures the expected geometric progression of the noise and brings the final samples closer to the data manifold. We showed how to extract the expected denoised sample and how to use it to further improve the final Langevin samples. We proposed a hybrid approach between GAN and score matching. With experiments on synthetic and standard image datasets; we showed that these approaches generally improved the quality/diversity of the generated samples.

We found equal diversity (coverage of all 1000 modes) for the adversarial and non-adversarial variant of the difficult StackedMNIST problem. Since we also observed better performance (from lower FIDs) in our other adversarial models trained on images, we conclude that making score matching adversarial increases the quality of the samples without decreasing diversity. These findings imply that score matching performs better than most GANs and on-par with state-of-the-art GANs. Furthermore, our results suggest that hybrid methods, combining multiple generative techniques together, are a very promising direction to pursue.

As future work, these models should be scaled to larger batch sizes on high-resolution images, since GANs have been shown to produce outstanding high-resolution images at very large batch sizes (2048 or more). We also plan to further study the theoretical properties of CAS by considering its corresponding stochastic differential equation.



# 9

# Prologue to the fourth article

---

## 9.1 Article Details

Jolicoeur-Martineau, A., Li, K., Piché-Taillefer, R., Kachman, T., Mitliagkas, I. (2022). [Gotta Go Fast When Generating Data with Score-Based Models](#). *International Conference on Learning Representations (ICLR)*.

---

## 9.2 Context

Although score-based models provide high-quality samples and generate from the full support of the distribution (contrary to most other generative models), they are extremely slow. Generating a mini-batch of data can take minutes to hours. Most existing faster methods are specific to the VP process or work poorly with VE while requiring heavy hyper-parameter tuning. I wanted to find a method that works well for VE while requiring as little tuning as possible. I concluded that we could get rid of the hyperparameter tuning and slow speed by devising a Stochastic Differential Equation (SDE) solver that works well on score-based models.

---

## 9.3 Personal contribution to the paper

I created the new Algorithm, initiated the project, wrote most of the paper, I wrote most of the code and ran the experiments. Rémi Piché-Taillefer helped with the code and figures. Other members helped in guiding the high-level discussion (the project started with a completely different direction). Ioannis Mitliagkas supervised the project.

# Gotta Go Fast when generating data with score-based models

---

## 10.1 Abstract

Score-based (denoising diffusion) generative models have recently gained a lot of success in generating realistic and diverse data. These approaches define a forward diffusion process for transforming data to noise and generate data by reversing it (thereby going from noise to data). Unfortunately, current score-based models generate data very slowly due to the sheer number of score network evaluations required by numerical SDE solvers.

In this work, we aim to accelerate this process by devising a more efficient SDE solver. Existing approaches rely on the Euler-Maruyama (EM) solver, which uses a fixed step size. We found that naively replacing it with other SDE solvers fares poorly - they either result in low-quality samples or become slower than EM. To get around this issue, we carefully devise an SDE solver with adaptive step sizes tailored to score-based generative models piece by piece. Our solver requires only two score function evaluations per step, rarely rejects samples, and leads to high-quality samples. Our approach generates data 2 to 10 times faster than EM while achieving better or equal sample quality. For high-resolution images, our method leads to significantly higher quality samples than all other methods tested. Our SDE solver has the benefit of requiring no step size tuning.

Code is available on [https://github.com/AlexiaJM/score\\_sde\\_fast\\_sampling](https://github.com/AlexiaJM/score_sde_fast_sampling).

---

## 10.2 Introduction

Score-based generative models (Song and Ermon, 2019, 2020; Ho et al., 2020; Jolicœur-Martineau et al., 2021; Song et al., 2021; Rémi Piché-Taillefer, 2021) have been very successful at generating data from various modalities, such as images (Ho

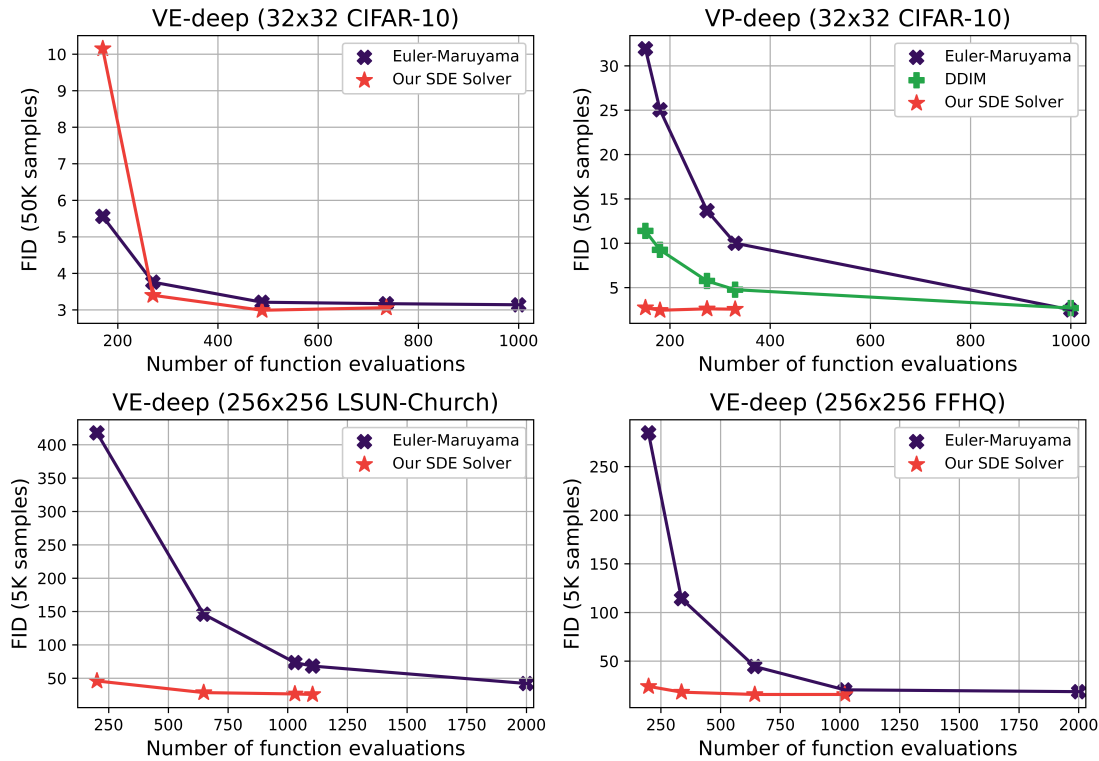
---

et al., 2020; Song et al., 2021), audio (Chen et al., 2020; Kong et al., 2021; Mittal et al., 2021; Kameoka et al., 2020), and graphs (Niu et al., 2020). They have further been used effectively for super-resolution (Salimans et al., 2016; Kadkhodaie and Simoncelli, 2020), inpainting (Kadkhodaie and Simoncelli, 2020), source separation (Jayaram and Thickstun, 2020), and image-to-image translation (Sasaki et al., 2021). In most of these applications, score-based models achieved superior performances in terms of quality and diversity than the historically dominant Generative Adversarial Networks (GANs) (Goodfellow et al., 2020).

Score-based models can be understood in two main classes: those based on a Variance Exploding (VE) diffusion process (Song and Ermon, 2019) and those based on a Variance Preserving (VP) one (Ho et al., 2020). Both diffusion processes progressively transform real data into Gaussian noise;  $\mathcal{N}(\mathbf{0}, \sigma_{max}^2 \mathbf{I})$  for VE where  $\sigma_{max}^2$  is very large, and  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  for VP.

The diffusion process (VE, VP, etc.) is then reversed in order to generate real data from Gaussian noise. Reversing the process requires the score function, which is estimated with a neural network (known as a score network). Although very powerful, score-based models generate data through an undesirably long iterative process; meanwhile, other state-of-the-art methods such as GANs generate data from a single forward pass of a neural network. Increasing the speed of the generative process is thus an active area of research.

Chen et al. (2020) and San-Roman et al. (2021) proposed faster step size schedules for VP diffusions that still yield relatively good quality/diversity metrics. Although fast, these schedules are arbitrary, require careful tuning, and the optimal schedules will vary from one model to another.



**Figure 10.1** – Comparison between our novel SDE solver at various values of error tolerance and Euler-Maruyama for an equal computational budget. We measure speed through the Number of Function Evaluations (NFE) and the quality of the generated images through the Fréchet Inception Distance (FID; lower is better). See Table 10.1-10.2 for more details.

Block et al. (2020) proposed generating data progressively from low to high-resolution images and show that the scheme improves speed. Similarly, Nichol and Dhariwal (2021) proposed generating low-resolution images and then upscale them since generating low-resolution images is quicker. They further suggested to accelerate VP-based models by learning dimension-specific noise rather than assuming equal noise everywhere. Note that these methods do not affect the data generation algorithm and would thus be complementary to our methods.

Song et al. (2021) and Song et al. (2020) proposed removing the noise from the data generation algorithm and solve an Ordinary Differential Equation (ODE) rather than a Stochastic Differential Equation (SDE); they report being able to converge much faster when there is no noise. Although it improves the generation speed, Song et al. (2021) report obtaining lower-quality images when using the ODE formulation for the VE process (Song et al., 2021). We will later show that

---

our SDE solver generally leads to better results than ODE solvers at similar speeds.

Thus, existing methods for acceleration often require considerable step size/schedule tuning (this is also true for the baseline approach) and do not always work for both VE and VP processes. To improve speed and remove the need for step size/schedule tuning, we propose to solve the reverse diffusion process using SDE solvers with adaptive step sizes.

It turns out that off-the-shelf SDE solvers are ill-suited for generative modeling and exhibit either (1) divergence, (2) slower data generation than the baseline, or (3) significantly worse quality than the baseline (see Appendix D.1). This can be attributed to distinct features of the SDEs that arise in score-based generative models that set them apart from the SDEs traditionally considered in the numerical SDE solver literature, namely: (1) the codomain of the unknown function is extremely high-dimensional, especially in the case of image generation; (2) evaluating the score function is computationally expensive, requiring a forward pass of a large mini-batch through a large neural network; (3) the required precision of the solution is smaller than usual because we are satisfied as long as the error is not perceptible (e.g., one RGB increment on an image).

We devise our own SDE solver with these features in mind, resulting in an algorithm that can get around the problems encountered by off-the-shelf solvers. To address high dimensionality, we use the  $\ell_2$  norm rather than the  $\ell_\infty$  norm to measure the error across different dimensions to prevent a single pixel from slowing down the solver. To address the cost of score function evaluations while still obtaining high precision, we (1) take the minimum number of score function evaluations needed for adaptive step sizes (two evaluations), and (2) use extrapolation to get high precision at no extra cost. To take advantage of the reduced requirement for precision, we set the absolute tolerance for the error according to the range of RGB values.

Our main contribution is a new SDE solver tailored to score-based generative models with the following benefits:

- Our solver is much faster than the baseline methods, i.e. reverse-diffusion method with Langevin dynamics and Euler-Maruyama (EM);
- It yields higher quality/diversity samples than EM when using the same computing budget;
- It does not require any step size or schedule tuning;

---

— It can be used to quickly solve any type of diffusion process (e.g., VE, VP)

---

## 10.3 Background

### 10.3.1 Score-based modeling with SDEs

Let  $\mathbf{x}(0) \in \mathbb{R}^d$  be a sample from the data distribution  $p_{\text{data}}$ . The sample is gradually corrupted over time through a Forward Diffusion Process (FDP), a common type of Stochastic Differential Equation (SDE):

$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \quad (10.1)$$

where  $f(\mathbf{x}, t) : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$  is the drift,  $g(t) : \mathbb{R} \rightarrow \mathbb{R}$  is the diffusion coefficient and  $\mathbf{w}(t)$  is the Wiener process indexed by  $t \in [0, 1]$ . Data points and their probability distribution evolve along the trajectories  $\{\mathbf{x}(t)\}_{t=0}^1$  and  $\{p_t(\mathbf{x})\}_{t=0}^1$  respectively, with  $p_0 \equiv p_{\text{data}}$ . The functions  $f$  and  $g$  are chosen such that  $\mathbf{x}(1)$  be approximately Gaussian and independent from  $\mathbf{x}(0)$ . Inference is achieved by reversing this diffusion, drawing  $\mathbf{x}(1)$  from its Gaussian distribution and solving the Reverse Diffusion Process (RDP) equal to:

$$d\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t)d\bar{\mathbf{w}}, \quad (10.2)$$

where  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$  is referred to as the score of the distribution at time  $t$  (Hyvärinen, 2005) and  $\bar{\mathbf{w}}(t)$  is the Wiener process in which time flows backward (Anderson, 1982).

One can observe from Equation 10.2 that the RDP requires knowledge of the score (or  $p_t$ ), which we do not have access to. Fortunately, it can be estimated by a neural network (referred to as the score network) by optimizing the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}(t) \sim p(\mathbf{x}(t)|\mathbf{x}(0)), \mathbf{x}(0) \sim p_{\text{data}}} \left[ \frac{\lambda(t)}{2} \left\| s_{\theta}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t)|\mathbf{x}(0)) \right\|_2^2 \right], \quad (10.3)$$

---

where  $\lambda(t) : \mathbb{R} \rightarrow \mathbb{R}$  is a weighting function generally chosen to be inversely proportional to:

$$\mathbb{E} \left[ \left\| \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t)|\mathbf{x}(0)) \right\|_2^2 \right].$$

One can demonstrate that the minimizer of that objective  $\theta^*$  will be such that  $s_{\theta^*}(\mathbf{x}, t) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$  (Vincent, 2011), allowing us to approximate the reverse diffusion process. As can be seen, evaluating the objective requires the ability to generate samples from the FDP at arbitrary times  $t$ . Thankfully, as long as the drift is affine (i.e.,  $f(\mathbf{x}, t) = \mathbf{A}\mathbf{x} + \mathbf{B}$ ), the transition kernel  $p(\mathbf{x}(t)|\mathbf{x}(0))$  will always be normally distributed (Särkkä and Solin, 2019), which means that we can solve the forward diffusion in a single step. Furthermore, the score of the Gaussian transition kernel is trivial to compute, making the loss an inexpensive training objective.

There are two primary choices for the FDP in the literature, which we discuss below.

### 10.3.2 Variance Exploding (VE) process

The Variance Exploding (VE) process consists in the following FDP:

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} d\mathbf{w}.$$

Its associated transition kernel is:

$$\mathbf{x}(t)|\mathbf{x}(0) \sim \mathcal{N}(\mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I}) \approx \mathcal{N}(\mathbf{x}(0), \sigma^2(t)\mathbf{I}).$$

In practice, we let  $\sigma(t) = \sigma_{min} \left( \frac{\sigma_{max}}{\sigma_{min}} \right)^t$ , where  $\sigma_{min} = 0.01$  and  $\sigma_{max} \approx \max_i \sum_{j=1}^N \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|$  is the maximum Euclidean distance between two samples from the dataset  $\{\mathbf{x}^{(i)}\}_{i=1}^N$  (Song and Ermon, 2020). Using the maximum Euclidean distance ensures that  $\mathbf{x}(1)$  does not depend on  $\mathbf{x}(0)$ ; thus,  $\mathbf{x}(1)$  is approximately distributed as  $\mathcal{N}(\mathbf{0}, \sigma^2(1)\mathbf{I})$ .

### 10.3.3 Variance Preserving (VP) process

The Variance Preserving (VP) process consists in the following FDP:

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}dt + \sqrt{\beta(t)}d\mathbf{w}.$$

---

Its associated transition kernel is:

$$\mathbf{x}(t)|\mathbf{x}(0) \sim \mathcal{N}(\mathbf{x}(0) e^{-\frac{1}{2} \int_0^t \beta(s) ds}, (1 - e^{-\int_0^t \beta(s) ds}) \mathbf{I}).$$

In practice, we let  $\beta(t) = \beta_{min} + t(\beta_{max} - \beta_{min})$ , where  $\beta_{min} = 0.1$  and  $\beta_{max} = 20$ . Thus,  $\mathbf{x}(1)$  is approximately distributed as  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  and does not depend on  $\mathbf{x}(0)$ .

### 10.3.4 Solving the Reverse Diffusion Process (RDP)

There are many ways to solve the RDP; the most basic one being Euler-Maruyama (Kloeden and Platen, 1992), the SDE analog to Euler’s method for solving ODEs. Song et al. (2021) also proposed *Reverse-Diffusion*, which consists in ancestral sampling (Ho et al., 2020) with the same discretization used in the FDP. With the Reverse-Diffusion, (Song et al., 2021) obtained poor results unless applying an additional Langevin dynamics step after each Reversion-Diffusion step. They named this approach Predictor-Corrector (PC) sampling, with the predictor corresponding to Reverse-Diffusion and the corrector to Langevin dynamics. Although using a corrector step leads to better samples, PC sampling is only heuristically motivated and the theoretical underpinnings are not yet understood. Nevertheless, (Song et al., 2021) report their best results (in terms of lowest Fréchet Inception Distance (Heusel et al., 2017)) using the Reverse-Diffusion with Langevin dynamics for VE models. For VP models, they obtain their best results using Euler-Maruyama.

---

## 10.4 Efficient Method for Solving Reverse Diffusion Processes

### 10.4.1 Setting up the algorithm

We start with a general algorithm for solving an SDE (similar to most ODE/SDE solvers). We choose the various options/hyper-parameters based on a mixture of



---

theory and experiments; an ablation study of the different hyper-parameters can also be found in Appendix D.2.

#### 10.4.1.1 Integration method

Solving the RDP to generate data can take an undesirably long time. One would assume that solving SDEs with high-order methods would improve speed over Euler-Maruyama, just like high-order ODE solvers improve speed over Euler’s method when solving ODEs. However, this is not always the case: while higher-order solvers may achieve lower discretization errors, they require more function evaluations, and the improved precision might not be worth the increased computation cost (Lehn et al., 2002; Lamba, 2003).

Our preliminary attempts at using SDE solvers with the *DifferentialEquations.jl* Julia package (Rackauckas and Nie, 2017b) confirmed that higher-order methods were significantly slower (6 to 8 times slower; see Appendix D.1). Lamba’s algorithm (Lamba, 2003), a low-order adaptive method, yielded the fastest results, thus motivating us to restrict our search to the space of low-order methods. Still, the resulting images were of lower quality.

Using a fixed step-size while solving an ODE/SDE requires some tuning and one should be able to advance faster (from  $t = 1$  to  $t = 0$ ) in regions of low noise. To gain more speed, one can dynamically adjust the step size over time; this is a common approach used in most fast ODE/SDE solvers. Such technique generally use two integration methods: a lower-order ( $\mathbf{x}'$ ) method and a higher-order ( $\mathbf{x}''$ ) method. The local error  $E(\mathbf{x}', \mathbf{x}'') = \mathbf{x}' - \mathbf{x}''$  is used to determine how stable the lower-order method is at the current step size; the closer to zero, the more appropriate the step size is. From this information, we can dynamically adjust the step size and decide whether or not to accept the proposed sample of the lower-order method. Alternatively, one can select  $\mathbf{x}''$  as the proposal, which we will refer to as *extrapolating*.

Rather than using the Improved Euler ODE solver (Süli and Mayers, 2003) as in Lamba (2003) or a high-order stochastic Runge-Kutta method (Rößler, 2010) as in Rackauckas and Nie (2017a) (which did not work well in our preliminary attempts with the Julia package) we instead rely on the more recent Improved Euler SDE solver (Roberts, 2012) as our higher order method. This method is very similar to the classical Improved Euler ODE solver, but it is made to work with SDEs instead

---

of ODEs. Importantly, this method only requires two score function evaluations and re-uses the same score function evaluation used for EM, meaning that it is only twice as expensive as EM. Similarly to Lamba’s algorithm, this method, albeit quick, leads to images of poor quality. However, by using extrapolation (taking  $\mathbf{x}''$  instead of  $\mathbf{x}'$  as our proposal), we were able to match and improve over the baseline approach (EM). Thus, using the stochastic Improved Euler was the key to taking bigger steps without sacrificing precision. Note that Lamba’s algorithm cannot use extrapolation due to its use of a non-stochastic ODE solver (Improved Euler).

An algorithm has strong-order  $p$  when the local error from  $t$  to  $t+h$  is  $\mathcal{O}(h^{p+1})$ . Euler-Maruyama has strong-order 0.5 while Improved Euler has strong-order 1 (Roberts, 2012). The highest strong-order found in the *DifferentialEquations.jl* Julia package (Rackauckas and Nie, 2017b) are order 1.5. Thus, our method obtains a balance between methods that are 1) low precision, but fast and 2) high precision, but slow.

#### 10.4.1.2 Tolerance

In ODE/SDE solvers, the local error is divided by a *tolerance* term. Traditionally, the mixed tolerance  $\delta(\mathbf{x}') : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is calculated as the maximum between the absolute and relative tolerance:

$$\delta(\mathbf{x}') = \max(\epsilon_{abs}, \epsilon_{rel}|\mathbf{x}'|), \quad (10.4)$$

where the absolute value  $|\cdot|$  is applied element-wise.

The *DifferentialEquations.jl* Julia package instead calculates the mixed tolerance through the maximum of the current and previous sample:

$$\delta(\mathbf{x}', \mathbf{x}'_{prev}) = \max(\epsilon_{abs}, \epsilon_{rel} \max(|\mathbf{x}'|, |\mathbf{x}'_{prev}|)). \quad (10.5)$$

Having no trivial prior for which approach to use, we tried both and found the latter approach (Equation 10.5) to converge much faster for VE models (see Appendix D.2).

Given our focus on image generation, we can set  $\epsilon_{abs}$  a priori. During training and at the end of the data generation, images are represented as floating-point tensors with range  $[y_{min}, y_{max}]$ . When evaluated, they must be transformed into

---

8-bit color images; this means that images are scaled to the range  $[0, 255]$  and converted to the nearest integer (to represent one of the 256 values per color channel). Given the 8-bit color encoding, an absolute tolerance  $\epsilon_{abs} = \frac{y_{max} - y_{min}}{256}$  corresponds to tolerating local errors of at most one color (e.g.,  $x'_{ij}$  with Red=5 and  $x''_{ij}$  with Red=6 is accepted, but  $x'_{ij}$  with Red=5 and  $x''_{ij}$  with Red=7 is not) channel-wise. One-color differences are not perceptible and should not influence the metrics used for evaluating the generated images. For VP models, which have range  $[-1, 1]$ , this corresponds to  $\epsilon_{abs} = 0.0078$  while for VE models, which have range  $[0, 1]$ , this corresponds to  $\epsilon_{abs} = 0.0039$ .

To control speed/quality, we vary  $\epsilon_{rel}$ , where large values lead to more speed but less precision, while small values lead to the converse.

#### 10.4.1.3 Norm of the scaled error

The scaled error (the error scaled by the mixed tolerance) is calculated as

$$E_q = \left\| \frac{\mathbf{x}' - \mathbf{x}''}{\boldsymbol{\delta}(\mathbf{x}', \mathbf{x}'_{prev})} \right\|_q.$$

Many algorithms use  $q = \infty$  (Lamba, 2003; Rackauckas and Nie, 2017a), where  $\|\mathbf{x}\|_\infty = \max(\mathbf{x}_1, \dots, \mathbf{x}_k)$  over all  $k$  elements of  $\mathbf{x}$ . Although this can work with low-dimensional SDEs, this is highly problematic for high-dimensional SDEs such as those in image-space. The reason is that a single channel of a single pixel (out of 65536 pixels for a  $256 \times 256$  color image) with a large local error will cause the step size to be reduced for all pixels and possibly lead to a step size rejection. Indeed, our experiments confirmed that using  $q = \infty$  slows down generation considerably (see Appendix D.2). To that effect, we instead use a scaled  $\ell_2$  norm:

$$\|\mathbf{x}\|_2 = \sqrt{\frac{1}{n} \sum_{i=1}^k \left( \frac{\mathbf{x}' - \mathbf{x}''}{\boldsymbol{\delta}(\mathbf{x}', \mathbf{x}'_{prev})} \right)_k^2}.$$

#### 10.4.1.4 Hyperparameters of the dynamic step size algorithm

Upon calculating the scaled error, we accept the proposal  $\mathbf{x}''$  if  $E_q \leq 1$  and increment the time by  $h$ . Whether or not it is accepted, we update the next step

size  $h$  in the usual way:

$$h \leftarrow \min(h_{\max}, \theta h E_q^{-r}), \quad (10.6)$$

where  $h_{\max}$  is the maximum step size,  $\theta$  is the safety parameter which determines how strongly we adapt the step size (0 being very safe; 1 being fast, but high rejections rate), and  $r$  is an exponent-scaling term.

Although ODE theory tells us that we should let  $r = \frac{1}{p+1}$  with  $p$  being the order of the lower-order integration method, there is no such theory for SDEs (Rackauckas and Nie, 2017a). Thus, as Rackauckas and Nie (2017a) suggest, we resorted to empirically testing values and found that any  $r \in [0.5, 1]$  works well on both VE and VP processes, but that  $r \in [0.8, 0.9]$  is slightly faster (see Appendix D.2). We arbitrarily chose  $r = 0.9$  as the default setting.

Finally, we defaulted to setting  $\theta = 0.9$  for the safety parameter as is common in the literature, and choose  $h_{\max}$  to be equal to the largest step size possible, namely the remaining time  $t$ .

#### 10.4.1.5 Handling the mini-batch

Using the same step size for every sample of a mini-batch means that every images would be slowed down by the other images. Since every image’s RDP is independent from one another, we apply a different step size to each data sample; some images may converge faster than others, but we wait for all images to have converged.

### 10.4.2 Algorithm

In Section 10.4.1, We defined every aspect of the algorithm needed to numerically solve Equation 10.2 for image generation. The algorithm thus consists in using adaptive step sizes through Equation 10.6 with the hyperparameters defined in the previous subsection ( $q = \infty$ ,  $\theta = 0.9$ ,  $r = 0.9$ ,  $\epsilon_{abs} = \frac{y_{max} - y_{min}}{256}$ ) with Euler-Maruyama as the low-order method and Improved Euler as the high-order method. The resulting algorithm is described in Algorithm 6. This algorithm is straightforward to parallelize across the batch dimension.

Note that this algorithm is only for solving an RDP; a more general version for solving an arbitrary forward-time diffusion process can be found in Appendix

D.3. Additionally, we present a demonstration in Appendix D.6 showing that the extrapolation step conserves the stability and convergence of the EM step.

---

**Algorithm 6** Dynamic step size extrapolation for solving Reverse Diffusion Processes

---

**Require:**  $s_\theta, \epsilon_{rel}, \epsilon_{abs}, h_{init} = 0.01, r = 0.9, \theta = 0.9$      $\triangleright$  for images:  $\epsilon_{abs} = \frac{y_{max} - y_{min}}{256}$

$t \leftarrow 1$

$h \leftarrow h_{init}$

Initialize  $\mathbf{x}$

$\mathbf{x}'_{prev} \leftarrow \mathbf{x}$

**while**  $t > 0$  **do**

  Draw  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{x}' \leftarrow \mathbf{x} - hf(\mathbf{x}, t) + hg(t)^2 s_\theta(\mathbf{x}, t) + \sqrt{h}g(t)\mathbf{z}$      $\triangleright$  Euler-Maruyama

$\tilde{\mathbf{x}} \leftarrow \mathbf{x} - hf(\mathbf{x}', t-h) + hg(t-h)^2 s_\theta(\mathbf{x}', t-h) + \sqrt{h}g(t-h)\mathbf{z}$

$\mathbf{x}'' \leftarrow \frac{1}{2}(\mathbf{x}' + \tilde{\mathbf{x}})$      $\triangleright$  Improved Euler (SDE version)

$\boldsymbol{\delta} \leftarrow \max(\epsilon_{abs}, \epsilon_{rel} \max(|\mathbf{x}'|, |\mathbf{x}'_{prev}|))$      $\triangleright$  Element-wise operations

$E_2 \leftarrow \frac{1}{\sqrt{n}} \|(\mathbf{x}' - \mathbf{x}'') / \boldsymbol{\delta}\|_2$

**if**  $E_2 \leq 1$  **then**     $\triangleright$  Accept

$\mathbf{x} \leftarrow \mathbf{x}''$      $\triangleright$  Extrapolation

$t \leftarrow t - h$

$\mathbf{x}'_{prev} \leftarrow \mathbf{x}'$

$h \leftarrow \min(t, \theta h E_2^{-r})$      $\triangleright$  Dynamic step size update

**return**  $\mathbf{x}$

---

## 10.5 Experiments

To test Algorithm 6 on RDPs, we apply it to various pre-trained models from Song et al. (2021). To start, we generate low-resolution images (32x32) by testing the VP, VE, VP-deep, and VE-deep models on CIFAR-10 (Krizhevsky and Hinton, 2009). Then, we generate higher-resolutions images (256x256) by testing the VE models on LSUN-Church (Yu et al., 2015), and Flickr-Faces-HQ (FFHQ) (Karras et al., 2019). See implementation details in Appendix D.4. We used four or less V100 GPUs to run the experiments.

To measure the performance of the image generation, we calculate the Fréchet Inception Distance (FID) (Heusel et al., 2017) and the Inception Score (IS) (Salimans et al., 2016), where low FID and high IS correspond to higher quality/diver-

---

sity. We compare our method to the three base solvers used in Song et al. (2021): Reverse-Diffusion with Langevin dynamics, Euler-Maruyama (EM), and Probability Flow, where the latter solves an ODE instead of an SDE using Runge-Kutta 45 (Dormand and Prince, 1980). We also compare against the fast solver by (Song et al., 2020) called denoising diffusion implicit models (DDIM), which is only defined for VP models. We define the *baseline* approach as the solver used by Song et al. (2021) which leads to the lowest FID (EM for VP models and Reverse-Diffusion with Langevin for VE models). For our algorithm, the only free hyperparameter is the relative tolerance which we set to  $\epsilon_{rel} \in \{0.01, 0.02, 0.05, 0.1, 0.5\}$ .

The FID and the Number of score Function Evaluations (NFE) are described in Table 10.1 for low-resolution images and Table 10.2 for high-resolution images. The Inception Score (IS) is described for CIFAR-10 in Appendix D.5.

### 10.5.1 Performance

Compared to EM, we observe that our method is immediately advantageous in terms of quality/diversity for high-resolution images, along with 2 to  $3\times$  speedups ( $\epsilon_{rel} = 0.02$ ). While this advantage becomes less obvious in terms of the FID on CIFAR-10, our method still offers  $> 5\times$  computational speedups at no apparent disadvantage ( $\epsilon_{rel} \in \{0.02, 0.05\}$ ).

Reverse-Diffusion with Langevin achieves the lowest FID for VE models on CIFAR-10, though at the cost of a  $4\times$  computational overhead over our method. Furthermore, their advantage vanishes for VP models and when generating high-resolution images.

We further compare our SDE solver to EM given the same computational budget and observe that our method is always immensely preferable in high-resolutions and for VP models. For VE models on CIFAR-10, we observe that our algorithm leads to a better FID as long as the NFE is sufficiently large (270). Note that since our algorithm takes two score function evaluations per step, EM has the advantage of doing twice as many steps given the same NFE, which appears to be a factor more important than the order of the method at low budget in low-resolution VE. Nevertheless, comparing for equal number of iterative step, the results still point to our method being always preferable. For high-resolution images, we see that EM cannot converge on moderate to small NFEs due to the high-dimensionality,

**Table 10.1** – Number of score Function Evaluations (NFE) / Fréchet Inception Distance (FID) on CIFAR-10 (32x32) from 50K samples

Method	VP	VP-deep	VE	VE-deep
Reverse-Diffusion & Langevin	1999 / 3.41	1999 / 3.28	1999 / <b>2.40</b>	1999 / <b>2.21</b>
Euler-Maruyama	1000 / <b>2.55</b>	1000 / <b>2.49</b>	1000 / 2.98	1000 / 3.14
DDIM	1000 / 2.86	1000 / 2.69	–	–
Ours ( $\epsilon_{rel} = 0.01$ )	329 / 2.70	330 / 2.56	738 / 2.91	736 / 3.06
Euler-Maruyama (same NFE)	329 / 10.28	330 / 10.00	738 / 2.99	736 / 3.17
DDIM (same NFE)	329 / 4.81	330 / 4.76	–	–
Ours ( $\epsilon_{rel} = 0.02$ )	274 / 2.74	274 / 2.60	490 / <b>2.87</b>	488 / <b>2.99</b>
Euler-Maruyama (same NFE)	274 / 14.18	274 / 13.67	490 / 3.05	488 / 3.21
DDIM (same NFE)	274 / 5.75	274 / 5.74	–	–
Ours ( $\epsilon_{rel} = 0.05$ )	179 / <b>2.59</b>	180 / <b>2.44</b>	271 / 3.23	270 / 3.40
Euler-Maruyama (same NFE)	179 / 25.49	180 / 25.05	271 / 3.48	270 / 3.76
DDIM (same NFE)	179 / 9.20	180 / 9.25	–	–
Ours ( $\epsilon_{rel} = 0.10$ )	147 / 2.95	151 / 2.73	170 / 8.85	170 / 10.15
Euler-Maruyama (same NFE)	147 / 31.38	151 / 31.93	170 / 5.12	170 / 5.56
DDIM (same NFE)	147 / 11.53	151 / 11.38	–	–
Ours ( $\epsilon_{rel} = 0.50$ )	49 / 72.29	48 / 82.42	52 / 266.75	50 / 307.32
Euler-Maruyama (same NFE)	49 / 92.99	48 / 95.77	52 / 169.32	50 / 271.27
DDIM (same NFE)	49 / 37.24	48 / 38.71	–	–
Probability Flow (ODE)	142 / 3.11	145 / 2.86	183 / 7.64	181 / 5.53

**Table 10.2** – Number of score Function Evaluations (NFE) / Fréchet Inception Distance (FID) on LSUN-Church (256x256) and FFHQ (256x256) from 5K samples

Method	VE (Church)	VE (FFHQ)
Reverse-Diffusion & Langevin	3999 / 29.14	3999 / 16.42
Euler-Maruyama	2000 / 42.11	2000 / 18.57
Ours ( $\epsilon_{rel} = 0.01$ )	1104 / <b>25.67</b>	1020 / <b>15.68</b>
Euler-Maruyama (same NFE)	1104 / 68.24	1020 / 20.45
Ours ( $\epsilon_{rel} = 0.02$ )	1030 / <b>26.46</b>	643 / <b>15.67</b>
Euler-Maruyama (same NFE)	1030 / 73.47	643 / 44.42
Ours ( $\epsilon_{rel} = 0.05$ )	648 / 28.47	336 / 18.07
Euler-Maruyama (same NFE)	648 / 145.96	336 / 114.23
Ours ( $\epsilon_{rel} = 0.10$ )	201 / 45.92	198 / 24.02
Euler-Maruyama (same NFE)	201 / 417.77	198 / 284.61
Probability Flow (ODE)	434 / 214.47	369 / 135.50

making of our SDE solver the way to go.

Generally, we observe that the VE process cannot be solved as fast as the VP process; this is due to the enormous Gaussian noise in the VE process causing larger local errors. This reflects the issue mentioned in Section 10.4.1.1 regarding high-order SDE solvers not always being beneficial in terms of speed for SDEs with heavy Gaussian noise. In practice, for VE, the algorithm uses a small step size in the beginning to ensure high accuracy and eventually increases the step size as the noise becomes less considerable.

## 10.5.2 Solving an ODE instead of an SDE

We see that our SDE solver generally does better than Probability Flow, especially in high-resolution, where we obtain greatly lower FIDs with a similar budget. Our algorithm attains the same NFE as Probability Flow when  $\epsilon_{rel} = 0.10$  for low-resolution images and when  $0.05 < \epsilon_{rel} < 0.10$  for high-resolution images. For the same budget, Probability Flow has higher FID than our approach on all but low-resolution VE models. However, in that case, our algorithm achieves a much lower FID when  $\epsilon_{rel} \leq 0.05$ , albeit slower. In high-resolution, Probability Flow leads to very poor FIDs, suggesting no convergence.



---

### 10.5.3 DDIM

Contrary to Song et al. (2020), the FID of DDIM worsens significantly when the NFE decreases. This could be due to differences between Song et al. (2021) continuous-time score-matching and the DDIM training procedure and architecture. Nevertheless, the FID increase engendered by a reduced budget is much less dramatic than for EM. As of note, DDIM succeeds in maintaining a lower FID than our solver at extremely small NFEs ( $< 50$ ), albeit with extremely poor FID.

---

## 10.6 Limitations

Although we tested our approach on a wide range of settings, we nevertheless only tested on continuous-time image generation models. We did so because solving the SDE requires continuous-time and the only such pre-trained models at time of publishing are the one by Song et al. (2021).

Although our approach removes step size and schedule tuning, we still need to choose a value of the relative tolerance, which indirectly affects the number of steps taken; one could theoretically tune this hyper-parameter to optimize a certain metric, going against the point of removing tuning. Still, letting  $\epsilon_{rel} = 0.01$  for precise results and  $\epsilon_{rel} = 0.05$  for fast results are reasonable choices, as all evidence points to the FID being stable w.r.t.  $\epsilon_{rel}$ .

---

## 10.7 Conclusion

We built an SDE solver that allows for generating images of comparable (or better) quality to Euler-Maruyama at a much faster speed. Our approach makes image generation with score-based models more accessible by shrinking the required computational budgets by a factor of 2 to  $5\times$ , and presenting a sensible way of compromising quality for additional speed.

# 11 Discussion

In Chapter 1, I discussed the extremely promising applications of generative models, especially regarding art content generation. I envision a future where artists can collaborate with generative models to produce complex media such as paintings, tv shows, video games, and comic books. To pave the way toward this goal, I have focused on some of the most promising generative models: GANs and score-based diffusion models. In Chapter 2, I discussed both approaches and their pros and cons.

As discussed in Section 2.5, GANs can provide high-quality, but they suffer from training instabilities, and they often do not sample from the entire distribution (mode collapse). In Chapters 4 and 6, I provided a well-grounded method called Relativistic GAN which significantly improves the stability of GANs and has also been shown to prevent mode collapse (Sun et al., 2020).

As discussed in Section 2.4, score-based diffusion models initially suffered from significant instabilities, and their remaining issue is how slow they are in generating data. In Chapter 8, I provided an adversarial score-based model which improves sample quality, and I discussed an improved sampling technique for improving stability. In Chapter 10, I devised a new SDE solver to significantly improve the speed of the generation with score-based diffusion models.

Thus, for both GANs and score-based diffusion models, I improved the various issues associated with each technique. In particular, Relativistic GAN has already had a massive impact, having been cited already more than 500 times; they are now being used for super-resolution (Wang et al., 2018) text generation (Nie et al., 2019), and relatively small datasets with unruly/unstable settings. As score-based diffusion gains traction, my work to improve quality, stability, and speed will be used to help achieve this.

There still remains a long way for fully realistic complex media generation such as video generation. I suspect score-based diffusion methods will be important for such systems, but I do not think it will be enough. More work will need to be

---

done to fix the following problems: 1) sequences of frames for which the motion does not make physical sense, 2) lack of long-term memory, 3) colossal memory requirements, and 4) lack of integration of audio and text into the model. I will continue working on this problem until we finally reach the point of generating full-fledged videos with AI. Hopefully, this time will come sooner than later!

# Bibliography

- Alain, G., Y. Bengio, L. Yao, J. Yosinski, E. Thibodeau-Laufer, S. Zhang, and P. Vincent (2016). GSNs: generative stochastic networks. *Information and Inference: A Journal of the IMA* 5(2), 210–249.
- Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications* 12(3), 313–326.
- Arjovsky, M. and L. Bottou (2017). Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*.
- Arjovsky, M., S. Chintala, and L. Bottou (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223.
- Arnold, L. (1974). Stochastic differential equations. *New York*.
- Artemiev, S. S. and T. A. Averina (2011). *Numerical analysis of systems of ordinary and stochastic differential equations*. Walter de Gruyter.
- Barratt, S. and R. Sharma (2018). A Note on the Inception Score. *arXiv preprint arXiv:1801.01973*.
- Bengio, Y., L. Yao, G. Alain, and P. Vincent (2013). Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907.
- Block, A., Y. Mroueh, A. Rakhlin, and J. Ross (2020). Fast mixing of multi-scale langevin dynamics under the manifold hypothesis. *arXiv preprint arXiv:2006.11166*.
- Borji, A. (2019). Pros and Cons of GAN Evaluation Measures. *Computer Vision and Image Understanding* 179, 41–65.

- 
- Brock, A., J. Donahue, and K. Simonyan (2019). Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.
- Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems 33*, 1877–1901.
- Casella, G. and R. L. Berger (2002). *Statistical inference*, Volume 2. Duxbury Pacific Grove, CA.
- Chen, N., Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan (2020). WaveGrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems 2*(4), 303–314.
- Dieng, A. B., F. J. Ruiz, D. M. Blei, and M. K. Titsias (2019). Prescribed generative adversarial networks. *arXiv preprint arXiv:1910.04302*.
- Dinh, L., D. Krueger, and Y. Bengio (2014). NICE: Non-linear Independent Components Estimation. *arXiv preprint arXiv:1410.8516*.
- Dinh, L., J. Sohl-Dickstein, and S. Bengio (2017). Density estimation using real NVP. In *International Conference on Learning Representations*.
- Dormand, J. R. and P. J. Prince (1980). A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics 6*(1), 19–26.
- Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth (1987). Hybrid monte carlo. *Physics letters B 195*(2), 216–222.
- Dumoulin, V., I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville (2017). Adversarially Learned Inference. In *International Conference on Learning Representations*.
- Efron, B. (2011). Tweedie’s formula and selection bias. *Journal of the American Statistical Association 106*(496), 1602–1614.

- 
- Fedus, W., M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow (2018). Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step. In *International Conference on Learning Representations*.
- Fukushima, K. and S. Miyake (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pp. 267–285. Springer.
- Gardiner, C. (2009). *Stochastic methods*, Volume 4. Springer Berlin.
- Geirhos, R., P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel (2019). ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*.
- Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence* (6), 721–741.
- Germain, M., K. Gregor, I. Murray, and H. Larochelle (2015). Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889.
- Gidel, G., R. A. Hemmat, M. Pezeshki, R. Le Priol, G. Huang, S. Lacoste-Julien, and I. Mitliagkas (2019). Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1802–1811.
- Giles, M. (2018, Feb). The GANfather: The man who’s given machines the gift of imagination. <https://www.technologyreview.com/2018/02/21/145289/the-ganfather-the-man-whos-given-machines-the-gift-of-imagination/>.
- Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep learning*. MIT press.
- Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2020). Generative adversarial networks. *Communications of the ACM* 63(11), 139–144.

- 
- Goyal, A. G. A. P., N. R. Ke, S. Ganguli, and Y. Bengio (2017). Variational walk-back: Learning a transition operator as a stochastic recurrent net. In *Advances in Neural Information Processing Systems*, pp. 4392–4402.
- Grenander, U. and M. I. Miller (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)* 56(4), 549–581.
- Gretton, A., K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola (2007). A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems*, pp. 513–520.
- Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville (2017). Improved training of wasserstein GANs. *Advances in neural information processing systems* 30.
- Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in neural information processing systems* 30.
- Hiriart-Urruty, J.-B. and C. Lemaréchal (2012). *Fundamentals of convex analysis*. Springer Science & Business Media.
- Ho, J., A. Jain, and P. Abbeel (2020). Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems* 33, 6840–6851.
- Ho, J., C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans (2021). Cascaded diffusion models for high fidelity image generation. *arXiv preprint arXiv:2106.15282*.
- Hoeffding, W. (1992). A class of statistics with asymptotically normal distribution. In *Breakthroughs in Statistics*, pp. 308–334. Springer.
- Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research* 6(Apr), 695–709.
- Ioffe, S. and C. Szegedy (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International conference on machine learning*, pp. 448–456. PMLR.

- 
- James, F. (1990). A review of pseudorandom number generators. *Computer physics communications* 60(3), 329–344.
- Jayaram, V. and J. Thickstun (2020). Source separation with deep generative priors. In *International Conference on Machine Learning*, pp. 4724–4735. PMLR.
- Jolicoeur-Martineau, A. (2018). GANs beyond divergence minimization. *arXiv preprint arXiv:1809.02145*.
- Jolicoeur-Martineau, A. (2019). The relativistic discriminator: a key element missing from standard GAN. In *International Conference on Learning Representations*.
- Jolicoeur-Martineau, A. (2020). On relativistic  $f$ -divergences. In *International Conference on Machine Learning*, pp. 4931–4939. PMLR.
- Jolicoeur-Martineau, A. and I. Mitliagkas (2019). Connections between Support Vector Machines, Wasserstein distance and gradient-penalty GANs. *arXiv preprint arXiv:1910.06922*.
- Jolicoeur-Martineau, A., R. Piché-Taillefer, I. Mitliagkas, and R. T. des Combes (2021). Adversarial score matching and improved sampling for image generation. In *International Conference on Learning Representations*.
- Kadkhodaie, Z. and E. P. Simoncelli (2020). Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser. *arXiv preprint arXiv:2007.13640*.
- Kameoka, H., T. Kaneko, K. Tanaka, and N. Hojo (2020). VoiceGrad: Non-Parallel Any-to-Many Voice Conversion with Annealed Langevin Dynamics. *arXiv preprint arXiv:2010.02977*.
- Karras, T., T. Aila, S. Laine, and J. Lehtinen (2018). Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*.
- Karras, T., S. Laine, and T. Aila (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410.



- 
- Karras, T., S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila (2020). Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119.
- Kingma, D. P. and J. Ba (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.
- Kingma, D. P. and M. Welling (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.
- Kloeden, P. E. and E. Platen (1992). Stochastic differential equations. In *Numerical Solution of Stochastic Differential Equations*, pp. 103–160. Springer.
- Kloeden, P. E. and E. Platen (2013). *Numerical solution of stochastic differential equations*, Volume 23. Springer Science & Business Media.
- Kodali, N., J. Abernethy, J. Hays, and Z. Kira (2017). On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*.
- Kong, Z., W. Ping, J. Huang, K. Zhao, and B. Catanzaro (2021). DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *International Conference on Learning Representations*.
- Krizhevsky, A. and G. Hinton (2009). Learning Multiple Layers of Features from Tiny Images. *Master’s thesis, Department of Computer Science, University of Toronto*.
- Kullback, S. (1997). *Information theory and statistics*. Courier Corporation.
- Kumar, R., S. Ozair, A. Goyal, A. Courville, and Y. Bengio (2019). Maximum Entropy Generators for Energy-Based Models. *arXiv preprint arXiv:1901.08508*.
- Lamba, H. (2003). An adaptive timestepping algorithm for stochastic differential equations. *Journal of computational and applied mathematics* 161(2), 417–430.
- Larsen, A. B. L., S. K. Sønderby, H. Larochelle, and O. Winther (2016). Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pp. 1558–1566. PMLR.

- 
- LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature* 521(7553), 436–444.
- LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11), 2278–2324.
- Lehmann, E. L. (1951). Consistency and unbiasedness of certain nonparametric tests. *The Annals of Mathematical Statistics*, 165–179.
- Lehn, J., A. Rößler, and O. Schein (2002). Adaptive schemes for the numerical solution of SDEs—a comparison. *Journal of computational and applied mathematics* 138(2), 297–308.
- Li, K. and J. Malik (2018). Implicit Maximum Likelihood Estimation. *arXiv preprint arXiv:1809.09087*.
- Li, Z., Y. Chen, and F. T. Sommer (2019). Learning Energy-Based Models in High-Dimensional Spaces with Multi-scale Denoising Score Matching. *arXiv*, arXiv–1910.
- Lim, J. H., A. Courville, C. Pal, and C.-W. Huang (2020). AR-DAE: Towards Unbiased Neural Entropy Gradient Estimation. In *International Conference on Machine Learning*, pp. 6061–6071. PMLR.
- Lim, J. H. and J. C. Ye (2017). Geometric GAN. *arXiv preprint arXiv:1705.02894*.
- Lin, G., A. Milan, C. Shen, and I. Reid (2017). RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1925–1934.
- Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37(1), 145–151.
- Lin, Z., A. Khetan, G. Fanti, and S. Oh (2018). PacGAN: The Power of Two Samples in Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, pp. 1498–1507.
- Lindsay, G. W. (2021). Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *Journal of cognitive neuroscience* 33(10), 2017–2031.

- 
- Liu, Z., P. Luo, X. Wang, and X. Tang (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.
- Loizou, N., H. Berard, A. Jolicoeur-Martineau, P. Vincent, S. Lacoste-Julien, and I. Mitliagkas (2020). Stochastic Hamiltonian Gradient Methods for Smooth Games. In *International Conference on Machine Learning*, pp. 6370–6381. PMLR.
- Lucic, M., K. Kurach, M. Michalski, S. Gelly, and O. Bousquet (2018). Are GANs Created Equal? A Large-Scale Study. *Advances in neural information processing systems* 31.
- Makhzani, A., J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey (2015). Adversarial Autoencoders. *arXiv preprint arXiv:1511.05644*.
- Mallasto, A., G. Montúfar, and A. Gerolin (2019). How Well Do WGANs Estimate the Wasserstein Metric? *arXiv preprint arXiv:1910.03875*.
- Mao, X., Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley (2017). Least Squares Generative Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2813–2821. IEEE.
- Matsumoto, M. and T. Nishimura (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8(1), 3–30.
- McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5(4), 115–133.
- Meng, C., Y. Song, W. Li, and S. Ermon (2021). Estimating High Order Gradients of the Data Distribution by Denoising. *Advances in Neural Information Processing Systems* 34.
- Mescheder, L., A. Geiger, and S. Nowozin (2018). Which training methods for GANs do actually converge? In *International conference on machine learning*, pp. 3481–3490. PMLR.
- Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of State Calculations by Fast Computing Machines. *The journal of chemical physics* 21(6), 1087–1092.

- 
- Metz, L., B. Poole, D. Pfau, and J. Sohl-Dickstein (2017). Unrolled Generative Adversarial Networks. In *International Conference on Learning Representations*.
- Mittal, G., J. Engel, C. Hawthorne, and I. Simon (2021). Symbolic Music Generation with Diffusion Models. *arXiv preprint arXiv:2103.16091*.
- Miyasawa, K. (1961). An empirical Bayes estimator of the mean of a normal population. *Bull. Inst. Internat. Statist* 38(181-188), 1–2.
- Miyato, T., T. Kataoka, M. Koyama, and Y. Yoshida (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.
- Mroueh, Y., C.-L. Li, T. Sercu, A. Raj, and Y. Cheng (2018). Sobolev GAN. In *International Conference on Learning Representations*.
- Mroueh, Y. and T. Sercu (2017). Fisher GAN. *Advances in neural information processing systems* 30.
- Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability* 29(2), 429–443.
- Nachmani, E., R. S. Roman, and L. Wolf (2021). Denoising Diffusion Gamma Models. *arXiv preprint arXiv:2110.05948*.
- Nguyen, X., M. J. Wainwright, and M. I. Jordan (2010). Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory* 56(11), 5847–5861.
- Nichol, A. and P. Dhariwal (2021). Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*.
- Nie, W., N. Narodytska, and A. Patel (2019). RelGAN: Relational Generative Adversarial Networks for Text Generation. In *International Conference on Learning Representations*.
- Niu, C., Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon (2020). Permutation Invariant Graph Generation via Score-Based Generative Modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 4474–4484. PMLR.

- 
- Nowozin, S., B. Cseke, and R. Tomioka (2016).  $f$ -GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *Advances in neural information processing systems* 29.
- Oord, A. v. d., S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu (2016). WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499*.
- Papadopoulos, J. (2019, Jan). Max Payne gets an amazing HD Texture Pack using ESRGAN that is available for download. <https://www.dsogaming.com/news/max-payne-gets-an-amazing-hd-texture-pack-using-esrgan-and-is-available-for-download/>.
- Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer (2017). Automatic differentiation in PyTorch.
- Patrikspacek (2019, Jan). Enhanced Super-Resolution Generative Adversarial Networks. <https://esrgan.blogspot.com/2019/01/blog-post.html>.
- Petzka, H., A. Fischer, and D. Lukovnikov (2018). On the regularization of Wasserstein GANs. In *International Conference on Learning Representations*.
- Rackauckas, C. and Q. Nie (2017a). Adaptive methods for stochastic differential equations via natural embeddings and rejection sampling with memory. *Discrete and continuous dynamical systems. Series B* 22(7), 2731.
- Rackauckas, C. and Q. Nie (2017b). Differentialequations.jl—a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of open research software* 5(1).
- Radford, A., L. Metz, and S. Chintala (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Raphan, M. and E. P. Simoncelli (2011). Least squares estimation without priors or supervision. *Neural computation* 23(2), 374–420.
- Razavi, A., A. van den Oord, and O. Vinyals (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pp. 14866–14876.

- 
- Reehorst, E. T. and P. Schniter (2018). Regularization by denoising: Clarifications and new interpretations. *IEEE transactions on computational imaging* 5(1), 52–67.
- Rémi Piché-Taillefer (2021). Hamiltonian Monte Carlo and Consistent Sampling for Score-Based Generative Modeling. *Master’s thesis, Department of Computer Science, Université de Montréal.*
- Rényi, A. et al. (1961). On Measures of Entropy and Information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California.
- Robbins, H. (1955). *An empirical Bayes approach to statistics*. Office of Scientific Research, US Air Force.
- Robbins, H. and S. Monro (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407.
- Robert, C. and G. Casella (2011). A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data. *Statistical Science*, 102–115.
- Robert, C. and G. Casella (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.
- Roberts, A. (2012). Modify the Improved Euler scheme to integrate stochastic differential equations. *arXiv preprint arXiv:1210.0933*.
- Roberts, G. O., R. L. Tweedie, et al. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli* 2(4), 341–363.
- Romano, Y., M. Elad, and P. Milanfar (2017). The little engine that could: Regularization by denoising (RED). *SIAM Journal on Imaging Sciences* 10(4), 1804–1844.
- Rößler, A. (2010). Runge–Kutta methods for the strong approximation of solutions of stochastic differential equations. *SIAM Journal on Numerical Analysis* 48(3), 922–952.

- 
- Saito, Y. and T. Mitsui (1996). Stability analysis of numerical schemes for stochastic differential equations. *SIAM Journal on Numerical Analysis* 33(6), 2254–2267.
- Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, pp. 2234–2242.
- Salimans, T. and J. Ho (2022). Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*.
- Salimans, T., A. Karpathy, X. Chen, and D. P. Kingma (2017). PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *arXiv preprint arXiv:1701.05517*.
- San-Roman, R., E. Nachmani, and L. Wolf (2021). Noise Estimation for Generative Diffusion Models. *arXiv preprint arXiv:2104.02600*.
- Saremi, S. and A. Hyvarinen (2019). Neural Empirical Bayes. *Journal of Machine Learning Research* 20, 1–23.
- Särkkä, S. and A. Solin (2019). *Applied stochastic differential equations*, Volume 10. Cambridge University Press.
- Sasaki, H., C. G. Willcocks, and T. P. Breckon (2021). UNIT-DDPM: UNpaired Image Translation with Denoising Diffusion Probabilistic Models. *arXiv preprint arXiv:2104.05358*.
- Sohl-Dickstein, J., E. Weiss, N. Maheswaranathan, and S. Ganguli (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR.
- Song, J., C. Meng, and S. Ermon (2020). Denoising Diffusion Implicit Models. *arXiv preprint arXiv:2010.02502*.
- Song, Y., C. Durkan, I. Murray, and S. Ermon (2021). Maximum Likelihood Training of Score-Based Diffusion Models. *arXiv e-prints*, arXiv–2101.

- 
- Song, Y. and S. Ermon (2019). Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, pp. 11918–11930.
- Song, Y. and S. Ermon (2020). Improved Techniques for Training Score-Based Generative Models. *Advances in neural information processing systems 33*, 12438–12448.
- Song, Y., S. Garg, J. Shi, and S. Ermon (2020). Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In *Uncertainty in Artificial Intelligence*, pp. 574–584. PMLR.
- Song, Y., J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole (2021). Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.
- Srivastava, A., L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton (2017). VEE-GAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems*, pp. 3308–3318.
- Stili, E. and D. F. Mayers (2003). *An introduction to numerical analysis*. Cambridge university press.
- Sun, R., T. Fang, and A. Schwing (2020). Towards a better global loss landscape of GANs. *Advances in Neural Information Processing Systems 33*.
- Tolstikhin, I., O. Bousquet, S. Gelly, and B. Schoelkopf (2018). Wasserstein Auto-Encoders. In *International Conference on Learning Representations*.
- Tran, N.-T., T.-A. Bui, and N.-M. Cheung (2018). Dist-GAN: An Improved GAN using Distance Constraints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 370–385.
- Vahdat, A., K. Kreis, and J. Kautz (2021). Score-based Generative Modeling in Latent Space. *arXiv preprint arXiv:2106.05931*.
- Vaserstein, L. N. (1969). Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii* 5(3), 64–72.



- 
- Villani, C. (2008). *Optimal transport: old and new*, Volume 338. Springer Science & Business Media.
- Vincent, J. (2019, Apr). Artificial intelligence is helping old video games look like new. <https://www.theverge.com/2019/4/18/18311287/ai-upscaling-algorithms-video-games-mods-modding-esrgan-gigapixel>.
- Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation* 23(7), 1661–1674.
- Wang, X., K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy (2018). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0.
- Welling, M. and Y. W. Teh (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688.
- Wenliang, L. K. (2020). Blindness of score-based methods to isolated components and mixing proportions. *arXiv preprint arXiv:2008.10087*.
- Yu, F., A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao (2015). LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*.
- Zhang, H., I. Goodfellow, D. Metaxas, and A. Odena (2019). Self-Attention Generative Adversarial Networks. In *International Conference on Machine Learning*, pp. 7354–7363. PMLR.
- Zhang, L., L. Zhang, X. Mou, and D. Zhang (2012). A comprehensive evaluation of full reference image quality assessment algorithms. In *2012 19th IEEE International Conference on Image Processing*, pp. 1477–1480. IEEE.
- Zhang, W., J. Sun, and X. Tang (2008). Cat head detection-how to effectively exploit shape and texture features. In *European Conference on Computer Vision*, pp. 802–816. Springer.

- 
- Zhao, H., O. Gallo, I. Frosio, and J. Kautz (2016). Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging* 3(1), 47–57.
- Zhou, S., M. Gordon, R. Krishna, A. Narcomey, L. F. Fei-Fei, and M. Bernstein (2019). HYPE: A Benchmark for Human eYe Perceptual Evaluation of Generative Models. In *Advances in Neural Information Processing Systems*, pp. 3449–3461.

# A

# The relativistic discriminator: a key element missing from standard GAN

---

## A.1 Intuitive and memeful visual representation of RaGANs

See Table A.1.

---


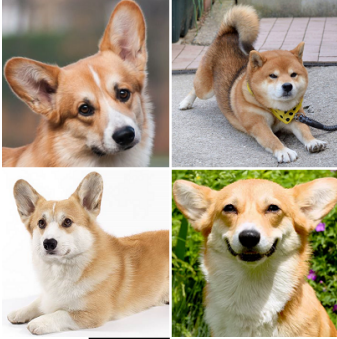



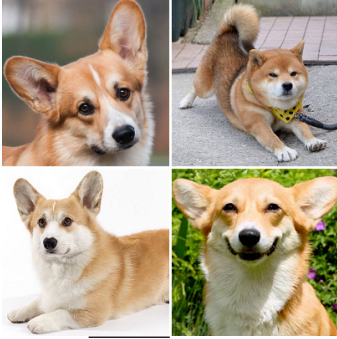
## A.2 Intuition behind RaGANs

Although the relative discriminator provide the missing property that we want in GANs (i.e.,  $G$  influencing  $D(x_r)$ ), its interpretation is different from the standard discriminator. Rather than measuring “the probability that the input data is real”, it is now measuring “the probability that the input data is more realistic than a randomly sampled data of the opposing type (fake if the input is real or real if the input is fake)”. To make the relativistic discriminator act more globally, as in its original definition, our initial idea was the following: average the relativistic discriminator over random samples of data of the opposing type. This can be conceptualized in the following way:

$$\begin{aligned} P(x_r \text{ is real}) &:= \mathbb{E}_{x_f \sim \mathbb{Q}}[P(x_r \text{ is more real than } x_f)] \\ &= \mathbb{E}_{x_f \sim \mathbb{Q}}[\text{sigmoid}(C(x_r) - C(x_f))] \\ &= \mathbb{E}_{x_f \sim \mathbb{Q}}[D(x_r, x_f)], \end{aligned}$$

$$\begin{aligned} P(x_f \text{ is real}) &:= \mathbb{E}_{x_r \sim \mathbb{P}}[P(x_f \text{ is more real than } x_r)] \\ &= \mathbb{E}_{x_r \sim \mathbb{P}}[\text{sigmoid}(C(x_f) - C(x_r))] \\ &= \mathbb{E}_{x_r \sim \mathbb{P}}[D(x_f, x_r)], \end{aligned}$$

**Table A.1** – An illustrative example of the discriminator’s output in standard GAN as traditionally defined ( $P(x_r \text{ is real}) = \text{sigmoid}(C(x_r))$ ) versus the Relativistic average Discriminator (RaD) ( $P(x_r \text{ is real}|\overline{C(x_f)}) = \text{sigmoid}(C(x_r) - \overline{C(x_f)})$ ). Breads represent real images, while dogs represent fake images.

Scenario	Absolute probability (Standard GAN)	Relative probability (Ra Standard GAN)
Real image looks real <b>and</b> fake images look fake	 $C(x_r) = 8$ $P(x_r \text{ is bread}) = 1$	 $\overline{C(x_f)} = -5$ $P(x_r \text{ is bread} \overline{C(x_f)}) = 1$
Real image looks real <b>but</b> fake images look similarly real on average	 $C(x_r) = 8$ $P(x_r \text{ is bread}) = 1$	 $C(x_f) = 7$ $P(x_r \text{ is bread} \overline{C(x_f)}) = .73$
Real image looks fake <b>but</b> fake images look more fake on average	 $C(x_r) = -3$ $P(x_r \text{ is bread}) = .05$	 $\overline{C(x_f)} = -5$ $P(x_r \text{ is bread} \overline{C(x_f)}) = .88$

---

where  $D(x_r, x_f) = \text{sigmoid}(C(x_r) - C(x_f))$ .

Then, the following loss function for  $D$  could be applied:

$$L_D = -\mathbb{E}_{x_r \sim \mathbb{P}} [\log (\mathbb{E}_{x_f \sim \mathbb{Q}} [D(x_r, x_f)])] - \mathbb{E}_{x_f \sim \mathbb{Q}} [\log (1 - \mathbb{E}_{x_r \sim \mathbb{P}} [D(x_f, x_r)])]. \quad (\text{A.1})$$

The main problem with this idea is that it would require looking at all possible combinations of real and fake data in the mini-batch. This would transform the problem from  $\mathcal{O}(m)$  to  $\mathcal{O}(m^2)$  complexity, where  $m$  is the batch size. This is problematic; therefore, we do not use this approach.

Instead, we propose to use the Relativistic average Discriminator (RaD) which compares the critic of the input data to the average critic of samples of the opposite type (See section 4.3). This approach has  $\mathcal{O}(m)$  complexity.

---

## A.3 Gradients

### A.3.1 SGAN

$$\begin{aligned} \nabla_w L_D^{GAN} &= -\nabla_w \mathbb{E}_{x_r \sim \mathbb{P}} [\log D(x_r)] - \nabla_w \mathbb{E}_{x_f \sim \mathbb{Q}_\theta} [\log(1 - D(x_f))] \\ &= -\nabla_w \mathbb{E}_{x_r \sim \mathbb{P}} \left[ \log \left( \frac{e^{C(x_r)}}{e^{C(x_r)} + 1} \right) \right] - \nabla_w \mathbb{E}_{x_f \sim \mathbb{Q}_\theta} \left[ \log \left( 1 - \frac{e^{C(x_f)}}{e^{C(x_f)} + 1} \right) \right] \\ &= -\nabla_w \mathbb{E}_{x_r \sim \mathbb{P}} [C(x_r) - \log(e^{C(x_r)} + 1)] + \nabla_w \mathbb{E}_{x_f \sim \mathbb{Q}_\theta} [\log(e^{C(x_f)} + 1)] \\ &= -\mathbb{E}_{x_r} [\nabla_w C(x_r)] + \mathbb{E}_{x_r} [D(x_r) \nabla_w C(x_r)] + \mathbb{E}_{x_f} [D(x_f) \nabla_w C(x_f)] \\ &= -\mathbb{E}_{x_r \sim \mathbb{P}} [(1 - D(x_r)) \nabla_w C(x_r)] + \mathbb{E}_{x_f \sim \mathbb{Q}_\theta} [D(x_f) \nabla_w C(x_f)] \end{aligned}$$

---


$$\begin{aligned}
\nabla_{\theta} L_G^{GAN} &= -\nabla_{\theta} \mathbb{E}_{z \sim \mathbb{P}_z} [\log D(G(z))] \\
&= -\nabla_{\theta} \mathbb{E}_{z \sim \mathbb{P}_z} \left[ \log \left( \frac{e^{C(G(z))}}{e^{C(G(z))} + 1} \right) \right] \\
&= -\nabla_{\theta} \mathbb{E}_{z \sim \mathbb{P}_z} [C(G(z)) - \log(e^{C(G(z))} + 1)] \\
&= -\mathbb{E}_{z \sim \mathbb{P}_z} \left[ \nabla_x C(G(z)) J_{\theta} G(z) - \left( \frac{e^{C(G(z))}}{e^{C(G(z))} + 1} \right) \nabla_x C(G(z)) J_{\theta} G(z) \right] \\
&= -\mathbb{E}_{z \sim \mathbb{P}_z} [(1 - D(G(z))) \nabla_x C(G(z)) J_{\theta} G(z)]
\end{aligned}$$

### A.3.2 IPM-based GANs

$$\begin{aligned}
\nabla_w L_D^{IPM} &= -\nabla_w \mathbb{E}_{x_r \sim \mathbb{P}} [C(x_r)] + \nabla_w \mathbb{E}_{x_f \sim \mathbb{Q}_{\theta}} [C(x_f)] \\
&= -\mathbb{E}_{x_r \sim \mathbb{P}} [\nabla_w C(x_r)] + \mathbb{E}_{x_f \sim \mathbb{Q}_{\theta}} [\nabla_w C(x_f)]
\end{aligned}$$

$$\begin{aligned}
\nabla_{\theta} L_G^{IPM} &= -\nabla_{\theta} \mathbb{E}_{z \sim \mathbb{P}_z} [C(G(z))] \\
&= -\mathbb{E}_{z \sim \mathbb{P}_z} [\nabla_x C(G(z)) J_{\theta} G(z)]
\end{aligned}$$

---

## A.4 Simplified form of relativistic saturating and non-saturating GANs

The formulation of RGANs can be simplified when we have the following two properties: (1)  $f_2(-y) = f_1(y)$  and (2) the generator assumes a non-saturating loss ( $g_1(y) = f_2(y)$  and  $g_2(y) = f_1(y)$ ). These two properties are observed in standard GAN, LSGAN using symmetric labels (e.g., -1 and 1), IPM-based GANs, etc. With these two properties, RGANs with non-saturating loss can be formulated simply as:

$$L_D^{RGAN*} = \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] \quad (\text{A.2})$$

and

$$L_G^{RGAN*} = \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_f) - C(x_r))] . \quad (\text{A.3})$$

Assuming  $f_2(-y) = f_1(y)$ , we have that

$$\begin{aligned} L_D^{RGAN} &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_2(C(x_f) - C(x_r))] \\ &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] \\ &= 2\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] . \end{aligned}$$

If  $g_1(y) = -f_1(y)$  and  $g_2(y) = -f_2(y)$  (saturating GAN), we have that

$$\begin{aligned} L_G^{RGAN-S} &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [g_1(C(x_r) - C(x_f))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [g_2(C(x_f) - C(x_r))] \\ &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] - \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_2(C(x_f) - C(x_r))] \\ &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] - \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] \\ &= -2\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))] . \end{aligned}$$

If  $g_1(y) = f_2(y)$  and  $g_2(y) = f_1(y)$  (non-saturating GAN), we have that

$$\begin{aligned} L_G^{RGAN-NS} &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [g_1(C(x_r) - C(x_f))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [g_2(C(x_f) - C(x_r))] \\ &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_2(C(x_r) - C(x_f))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_f) - C(x_r))] \\ &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_f) - C(x_r))] + \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_f) - C(x_r))] \\ &= 2\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_f) - C(x_r))] . \end{aligned}$$

---

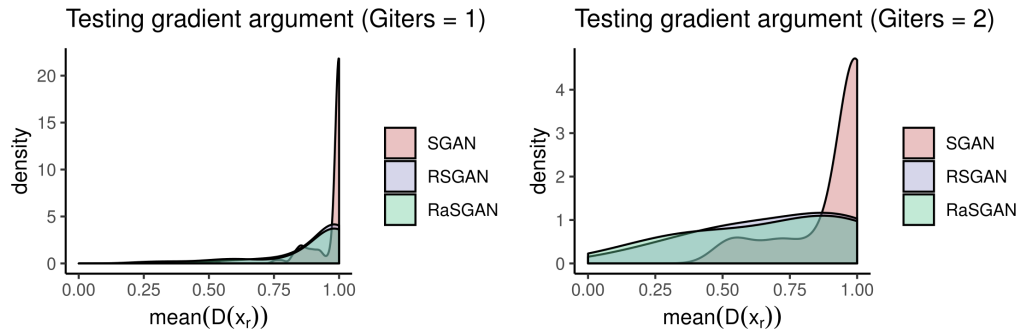
## A.5 Testing the gradient argument

Previously, we argued that SGAN could be equivalent to IPM-GANs under very strict conditions and assumptions. We mentioned that although most assumptions are reasonable, the assumption that the generator is trained to optimality is unrealistic. In which case, SGAN would not be equivalent to IPM-based GANs since  $D(x_r)$  would not reach 0.

As an experiment, we calculated the mini-batch average of  $D(x_r)$  in the first 100 iterations of the training for the CAT dataset in 256x256. Note that SGAN becomes stuck at around 200 iterations and can never go beyond generating noise.

Thus, a difference in the distribution of  $D(x_r)$  could reveal something meaningful about why Relativistic GANs can converge while their non-relativistic counterparts cannot.

In those 100 iterations, we have that the distance between  $\mathbb{P}$  and  $\mathbb{Q}$  is maximal since  $G$  only generate noise. Thus, we can perfectly distinguish real from fake data, which is one of the assumptions. The remaining assumptions were that  $D$  and  $G$  would be trained to optimality. Although we did not train  $D$  more than once, after the discriminator step, we generally had that  $D(x_r) \approx 1$ . What we wanted to verify is whether  $D(x_r) \approx 0$  after the generator step in Relativistic GANs, even though we did not train  $G$  enough to reach optimality.



**Figure A.1** – Density plots of the mini-batch average of  $D(x_r)$  during the first 100 iterations of training on CAT with 256x256 images using only one or two generator updates per discriminator updates. If  $D(x_r) = 0$  in all iterations, this would mean that the loss function would be the same as IPM-based GANs.

Results are shown in Figure 2. We observe that with only one generator update per discriminator update ( $n_G = 1$ ), RSGAN and RaSGAN never reach an average  $D(x_r)$  of 0 but the distribution is much less concentrated around 1 than with SGAN. With  $n_G = 2$ , RSGAN and RaSGAN sometimes reach an average  $D(x_r)$  of 0 and they form an almost uniform distribution around  $[0, 1]$ . This suggests that with the missing property (i.e., using Relativistic GANs), SGAN can be made more similar to IPM-based GANs, but never equivalent. Thus, Relativistic Standard GANs can be seen as having a dynamic in-between SGAN and IPM-based GANs.



---

## A.6 CIFAR-10 Hard/unstable experiments

In these analyses, we compared SGAN, LSGAN, WGAN-GP, RSGAN, RaSGAN, RaLSGAN, and RaHingeGAN with the standard CNN architecture on unstable setups in CIFAR-10. Unless otherwise specified, we used  $lr = .0002$ ,  $\beta_1 = .5$ ,  $\beta_2 = .999$ ,  $n_D = 1$ , and batch norm (Ioffe and Szegedy, 2015) in  $G$  and  $D$ . We tested the following four unstable setups: (1)  $lr = .001$ , (2)  $\beta_1 = .9$ ,  $\beta_2 = .9$ , (3) no batch norm in  $G$  or  $D$ , and (4) all activation functions replaced with Tanh in both  $G$  and  $D$  (except for the output activation function of  $D$ ).

Results are presented in Table 4. We observe that RaLSGAN performed better than LSGAN in all setups. RaHingeGAN performed slightly worse than HingeGAN in most setups. RSGAN and RaSGAN performed better than SGAN in two out of four setups, although differences were small. WGAN-GP generally performed poorly which we suspect is due to the single discriminator update per generator update. Overall, this provide good support for the improved stability of using the relative discriminator with LSGAN, but not with HingeGAN and SGAN. Although results are worse for the relativistic discriminator in some settings, differences are minimal and probably reflect natural variations.

It is surprising to observe low FID for SGAN without batch normalization considering its well-known difficulty with this setting (Arjovsky et al., 2017). Given these results, we suspected that CIFAR-10 may be too easy to fully observe the stabilizing effects of using the relative discriminator. Therefore, in the manuscript, we focused on the more difficult CAT dataset with high resolution pictures.

---

## A.7 Loss functions used in experiments

### A.7.1 SGAN (non-saturating)

$$L_D^{SGAN} = -\mathbb{E}_{x_r \sim \mathbb{P}} [\log (\text{sigmoid}(C(x_r)))] - \mathbb{E}_{x_f \sim \mathbb{Q}} [\log (1 - \text{sigmoid}(C(x_f)))] \quad (\text{A.4})$$

$$L_G^{SGAN} = -\mathbb{E}_{x_f \sim \mathbb{Q}} [\log (\text{sigmoid}(C(x_f)))] \quad (\text{A.5})$$

**Table A.2** – Fréchet Inception Distance (FID) at exactly 100k generator iterations on the CIFAR-10 dataset using unstable setups with different GAN loss functions. Unless otherwise specified, we used  $lr = .0002$ ,  $\beta = (.50, .999)$ ,  $n_D = 1$ , and batch norm (BN) in  $D$  and  $G$ . All models were trained using the same *a priori* selected seed (seed=1).

Loss	$lr = .001$	$\beta = (.9, .9)$	No BN	Tanh
SGAN	154.20	35.29	35.54	59.17
RSGAN	50.95	45.12	37.11	77.21
RaSGAN	55.55	43.46	41.96	54.42
LSGAN	52.27	225.94	38.54	147.87
RaLSGAN	<b>33.33</b>	48.92	<b>34.66</b>	53.07
HingeGAN	43.28	<b>33.47</b>	34.21	58.51
RaHingeGAN	51.05	42.78	43.75	<b>50.69</b>
WGAN-GP	61.97	104.95	85.27	59.94

### A.7.2 RSGAN

$$L_D^{RSGAN} = -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_r) - C(x_f)))] \quad (\text{A.6})$$

$$L_G^{RSGAN} = -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_f) - C(x_r)))] \quad (\text{A.7})$$

### A.7.3 RaSGAN

$$L_D^{RaSGAN} = -\mathbb{E}_{x_r \sim \mathbb{P}} [\log(\tilde{D}(x_r))] - \mathbb{E}_{x_f \sim \mathbb{Q}} [\log(1 - \tilde{D}(x_f))] \quad (\text{A.8})$$

$$L_G^{RaSGAN} = -\mathbb{E}_{x_f \sim \mathbb{Q}} [\log(\tilde{D}(x_f))] - \mathbb{E}_{x_r \sim \mathbb{P}} [\log(1 - \tilde{D}(x_r))] \quad (\text{A.9})$$

$$\tilde{D}(x_r) = \text{sigmoid}(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))$$

$$\tilde{D}(x_f) = \text{sigmoid}(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))$$

### A.7.4 LSGAN

$$L_D^{LSGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [(C(x_r) - 0)^2] + \mathbb{E}_{x_f \sim \mathbb{Q}} [(C(x_f) - 1)^2] \quad (\text{A.10})$$

---


$$L_G^{LSGAN} = \mathbb{E}_{x_f \sim \mathbb{Q}} [(C(x_f) - 0)^2] \quad (\text{A.11})$$

### A.7.5 RaLSGAN

$$\begin{aligned} L_D^{RaLSGAN} &= \mathbb{E}_{x_r \sim \mathbb{P}} [(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f) - 1)^2] \\ &\quad + \mathbb{E}_{x_f \sim \mathbb{Q}} [(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r) + 1)^2] \end{aligned}$$

$$\begin{aligned} L_G^{RaLSGAN} &= \mathbb{E}_{x_f \sim \mathbb{P}} [(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r) - 1)^2] \\ &\quad + \mathbb{E}_{x_r \sim \mathbb{P}} [(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f) + 1)^2] \end{aligned}$$

### A.7.6 HingeGAN

$$L_D^{HingeGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [\max(0, 1 - C(x_r))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [\max(0, 1 + C(x_f))] \quad (\text{A.12})$$

$$L_G^{HingeGAN} = -\mathbb{E}_{x_f \sim \mathbb{Q}} [C(x_f)] \quad (\text{A.13})$$

### A.7.7 RaHingeGAN

$$L_D^{HingeGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [\max(0, 1 - \tilde{D}(x_r))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [\max(0, 1 + \tilde{D}(x_f))] \quad (\text{A.14})$$

$$L_G^{HingeGAN} = \mathbb{E}_{x_f \sim \mathbb{P}} [\max(0, 1 - \tilde{D}(x_f))] + \mathbb{E}_{x_r \sim \mathbb{Q}} [\max(0, 1 + \tilde{D}(x_r))] \quad (\text{A.15})$$

$$\tilde{D}(x_r) = C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f)$$

$$\tilde{D}(x_f) = C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r)$$

### A.7.8 WGAN-GP

$$L_D^{WGAN-GP} = -\mathbb{E}_{x_r \sim \mathbb{P}} [C(x_r)] + \mathbb{E}_{x_f \sim \mathbb{Q}} [C(x_f)] + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} [(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1)^2] \quad (\text{A.16})$$

$$L_G^{WGAN-GP} = -\mathbb{E}_{x_f \sim \mathbb{Q}} [C(x_f)] \quad (\text{A.17})$$

$\mathbb{P}_{\hat{x}}$  is the distribution of  $\hat{x} = \epsilon x_r + (1 - \epsilon)x_f$ , where  $x_r \sim \mathbb{P}$ ,  $x_f \sim \mathbb{Q}$ ,  $\epsilon \sim U[0, 1]$ .

### A.7.9 RSGAN-GP

$$\begin{aligned} L_D^{RSGAN} &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_r) - C(x_f)))] \\ &\quad + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} [(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1)^2] \\ L_G^{RSGAN} &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_f) - C(x_r)))] \end{aligned} \quad (\text{A.18})$$

$\mathbb{P}_{\hat{x}}$  is the distribution of  $\hat{x} = \epsilon x_r + (1 - \epsilon)x_f$ , where  $x_r \sim \mathbb{P}$ ,  $x_f \sim \mathbb{Q}$ ,  $\epsilon \sim U[0, 1]$ .

### A.7.10 RaSGAN-GP

$$\begin{aligned} L_D^{RaSGAN} &= -\mathbb{E}_{x_r \sim \mathbb{P}} [\log(\tilde{D}(x_r))] - \mathbb{E}_{x_f \sim \mathbb{Q}} [\log(1 - \tilde{D}(x_f))] \\ &\quad + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} [(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1)^2] \\ L_G^{RaSGAN} &= -\mathbb{E}_{x_f \sim \mathbb{Q}} [\log(\tilde{D}(x_f))] - \mathbb{E}_{x_r \sim \mathbb{P}} [\log(1 - \tilde{D}(x_r))] \end{aligned} \quad (\text{A.19})$$

$$\tilde{D}(x_r) = \text{sigmoid}(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))$$

$$\tilde{D}(x_f) = \text{sigmoid}(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))$$

$\mathbb{P}_{\hat{x}}$  is the distribution of  $\hat{x} = \epsilon x_r + (1 - \epsilon)x_f$ , where  $x_r \sim \mathbb{P}$ ,  $x_f \sim \mathbb{Q}$ ,  $\epsilon \sim U[0, 1]$ .

---

## A.8 Algorithms

---

**Algorithm 7** Training algorithm for non-saturating RGANs with symmetric loss functions

---

**Require:** The number of  $D$  iterations  $n_D$  ( $n_D = 1$  unless one seeks to train  $D$  to optimality), batch size  $m$ , and functions  $f$  which determine the objective function of the discriminator ( $f$  is  $f_1$  from equation 10 assuming that  $f_2(-y) = f_1(y)$ , which is true for many GANs).

**while**  $\theta$  has not converged **do**

**for**  $t = 1, \dots, n_D$  **do**

    Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}$

    Sample  $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_z$

    Update  $w$  using SGD by ascending with  $\nabla_w \frac{1}{m} \sum_{i=1}^m [f(C_w(x^{(i)}) - C_w(G_\theta(z^{(i)})))]$

    Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}$

    Sample  $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_z$

    Update  $\theta$  using SGD by ascending with  $\nabla_\theta \frac{1}{m} \sum_{i=1}^m [f(C_w(G_\theta(z^{(i)})) - C_w(x^{(i)}))]$

---

---

**Algorithm 8** Training algorithm for non-saturating RaGANs

---

**Require:** The number of  $D$  iterations  $n_D$  ( $n_D = 1$  unless one seek to train  $D$  to optimality), batch size  $m$ , and functions  $f_1$  and  $f_2$  which determine the objective function of the discriminator (see equation 10).

**while**  $\theta$  has not converged **do**

**for**  $t = 1, \dots, n_D$  **do**

    Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}$

    Sample  $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_z$

    Let  $\overline{C_w(x_r)} = \frac{1}{m} \sum_{i=1}^m C_w(x^{(i)})$

    Let  $\overline{C_w(x_f)} = \frac{1}{m} \sum_{i=1}^m C_w(G_\theta(z^{(i)}))$

    Update  $w$  using SGD by ascending with

$$\nabla_w \frac{1}{m} \sum_{i=1}^m \left[ f_1(C_w(x^{(i)}) - \overline{C_w(x_r)}) + f_2(C_w(G_\theta(z^{(i)})) - \overline{C_w(x_f)}) \right]$$

  Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}$

  Sample  $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_z$

  Let  $\overline{C_w(x_r)} = \frac{1}{m} \sum_{i=1}^m C_w(x^{(i)})$

  Let  $\overline{C_w(x_f)} = \frac{1}{m} \sum_{i=1}^m C_w(G_\theta(z^{(i)}))$

  Update  $\theta$  using SGD by ascending with

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^m \left[ f_1(C_w(G_\theta(z^{(i)})) - \overline{C_w(x_r)}) + f_2(C_w(x^{(i)}) - \overline{C_w(x_f)}) \right]$$

---

---

## A.9 Architectures

### A.9.1 Standard CNN

Generator
$z \in \mathbb{R}^{128} \sim N(0, I)$
linear, 128 $\rightarrow$ 512*4*4
Reshape, 512*4*4 $\rightarrow$ 512 x 4 x 4
ConvTranspose2d 4x4, stride 2, pad 1, 512 $\rightarrow$ 256
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, 256 $\rightarrow$ 128
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, 128 $\rightarrow$ 64
BN and ReLU
ConvTranspose2d 3x3, stride 1, pad 1, 64 $\rightarrow$ 3
Tanh

---

Discriminator

---

---

$$x \in \mathbb{R}^{3 \times 32 \times 32}$$

---

Conv2d 3x3, stride 1, pad 1, 3->64

---

LeakyReLU 0.1

---

Conv2d 4x4, stride 2, pad 1, 64->64

---

LeakyReLU 0.1

---

Conv2d 3x3, stride 1, pad 1, 64->128

---

LeakyReLU 0.1

---

Conv2d 4x4, stride 2, pad 1, 128->128

---

LeakyReLU 0.1

---

Conv2d 3x3, stride 1, pad 1, 128->256

---

LeakyReLU 0.1

---

Conv2d 4x4, stride 2, pad 1, 256->256

---

LeakyReLU 0.1

---

Conv2d 3x3, stride 1, pad 1, 256->512

---

Reshape, 512 x 4 x 4 -> 512\*4\*4

---

linear, 512\*4\*4 -> 1

---



---

## A.9.2 DCGAN 64x64

Generator
$z \in \mathbb{R}^{128} \sim N(0, I)$
ConvTranspose2d 4x4, stride 1, pad 0, no bias, 128->512
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 512->256
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 256->128
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 128->64
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 64->3
Tanh

Discriminator
$x \in \mathbb{R}^{3 \times 64 \times 64}$
Conv2d 4x4, stride 2, pad 1, no bias, 3->64
LeakyReLU 0.2
Conv2d 4x4, stride 2, pad 1, no bias, 64->128
BN and LeakyReLU 0.2
Conv2d 4x4, stride 2, pad 1, no bias, 128->256
BN and LeakyReLU 0.2
Conv2d 4x4, stride 2, pad 1, no bias, 256->512
BN and LeakyReLU 0.2
Conv2d 4x4, stride 2, pad 1, no bias, 512->1

---

### A.9.3 DCGAN 128x128

Generator
$z \in \mathbb{R}^{128} \sim N(0, I)$
ConvTranspose2d 4x4, stride 1, pad 0, no bias, 128->1024
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 1024->512
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 512->256
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 256->128
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 128->64
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 64->3
Tanh

---

Discriminator

---

---

$$x \in \mathbb{R}^{3 \times 128 \times 128}$$

---

Conv2d 4x4, stride 2, pad 1, no bias, 3->64

---

LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 64->128

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 128->256

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 256->512

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 512->1024

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 1024->1

---

---

#### A.9.4 DCGAN 256x256

Generator
$z \in \mathbb{R}^{128} \sim N(0, I)$
ConvTranspose2d 4x4, stride 1, pad 0, no bias, 128->1024
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 1024->512
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 512->256
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 256->128
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 128->64
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 64->32
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, no bias, 64->3
Tanh

---

Discriminator (PACGAN2 (Lin et al., 2018))

---

$$x_1 \in \mathbb{R}^{3 \times 256 \times 256}, x_2 \in \mathbb{R}^{3 \times 256 \times 256}$$

---

$$\text{Concatenate } [x_1, x_2] \in \mathbb{R}^{6 \times 256 \times 256}$$

---

Conv2d 4x4, stride 2, pad 1, no bias, 6->32

---

LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 32->64

---

LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 64->128

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 128->256

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 256->512

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 512->1024

---

BN and LeakyReLU 0.2

---

Conv2d 4x4, stride 2, pad 1, no bias, 1024->1

---

---

## A.10 Samples

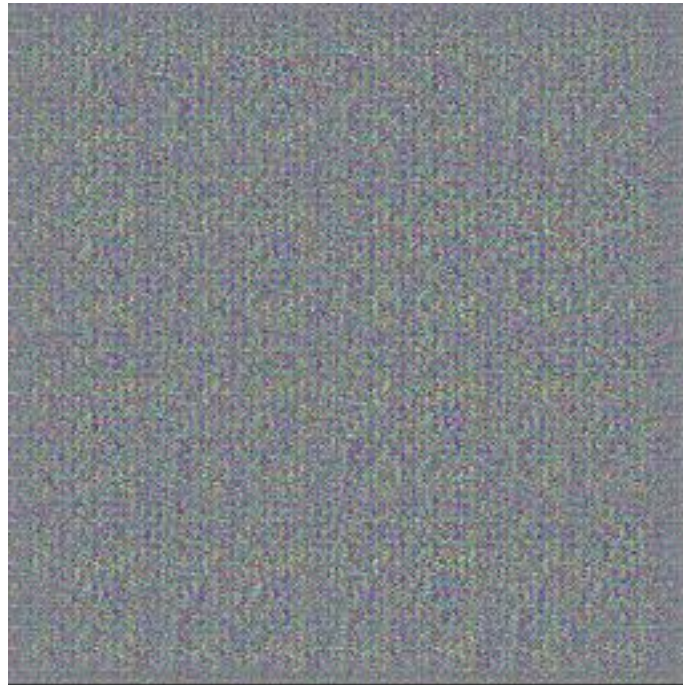
This shows a selection of cats from certain models. Images shown are from the lowest FID registered at every 10k generator iterations. Given space constraint, with cats in high resolution, we show some of the nicer looking cats for each approach, there are evidently some worse looking cats (See [https://github.com/AlexiaJM/RelativisticGAN/tree/master/images/full\\_minibatch](https://github.com/AlexiaJM/RelativisticGAN/tree/master/images/full_minibatch) for all cats of the mini-batch).



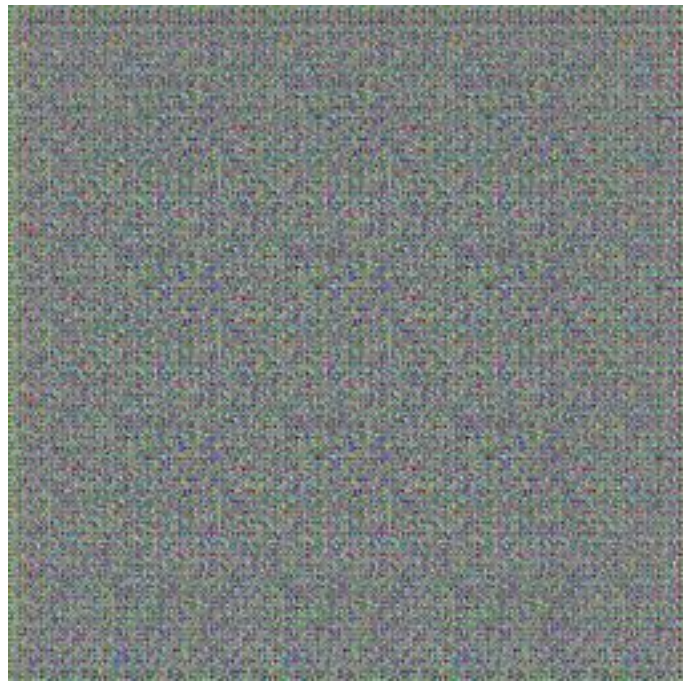
Figure A.2 – 64x64 cats with RaLSGAN (FID = 11.97)



**Figure A.3** – 128x128 cats with RaLSGAN (FID = 15.85)



**Figure A.4** – 256x256 cats with GAN (5k iterations)



**Figure A.5** – 256x256 cats with LSGAN (5k iterations)





**Figure A.6** – 256x256 cats with RaSGAN (FID = 32.11)



**Figure A.7** – 256x256 cats with RaLSGAN (FID = 35.21)



**Figure A.8** – 256x256 cats with SpectralSGAN (FID = 54.73)



**Figure A.9** – 256x256 cats with WGAN-GP (FID > 100)

# B

## On relativistic $f$ -divergences

---

### B.1 Proving that the objective functions are divergences

**Definition B.1.1.** Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions and  $S$  be the set of all probability distributions with common support. A function  $D : (S, S) \rightarrow \mathbb{R}_{>0}$  is a divergence if it respects the following two conditions:

$$\begin{aligned} D(\mathbb{P}, \mathbb{Q}) &\geq 0 \\ D(\mathbb{P}, \mathbb{Q}) = 0 &\iff \mathbb{P} = \mathbb{Q}. \end{aligned}$$

**Definition B.1.2.** A function  $f$  is concave on  $X$  if and only if

**Lemma 3.** Let  $f$  be a concave function on  $X$ , we have that

$$\forall x_1, x_2, x_3 \in X \text{ s.t. } x_1 < x_2 \leq x_3 : \frac{f(x_3) - f(x_1)}{x_3 - x_1} \leq \frac{f(x_2) - f(x_1)}{(x_2 - x_1)}$$

and

$$\forall x_1, x_2, x_3 \in X \text{ s.t. } x_1 \leq x_2 < x_3 : \frac{f(x_3) - f(x_2)}{(x_3 - x_2)} \leq \frac{f(x_3) - f(x_1)}{x_3 - x_1}.$$

*Proof.* Let  $\alpha = \frac{(x_3 - x_2)}{(x_3 - x_1)}$ .

If  $x_1 < x_2 \leq x_3$ , we have that  $\alpha \in [0, 1)$ .

If  $x_1 \leq x_2 < x_3$ , we have that  $\alpha \in (0, 1)$ .

Either way, by concavity, we have that

---


$$\begin{aligned}
f(x_2) &\geq \frac{(x_3 - x_2)}{(x_3 - x_1)}f(x_1) + \left(1 - \frac{(x_3 - x_2)}{(x_3 - x_1)}\right) f(x_3) \\
&= \frac{(x_3 - x_2)}{(x_3 - x_1)}f(x_1) + \frac{(x_2 - x_1)}{(x_3 - x_1)}f(x_3)
\end{aligned}$$

If  $x_1 < x_2 \leq x_3$ , we have that:

$$\begin{aligned}
f(x_2) - f(x_1) &\geq \frac{(x_1 - x_2)f(x_1) + (x_2 - x_1)f(x_3)}{(x_3 - x_1)} \\
\frac{f(x_2) - f(x_1)}{(x_2 - x_1)} &\geq \frac{f(x_3) - f(x_1)}{(x_3 - x_1)}
\end{aligned}$$

If  $x_1 \leq x_2 < x_3$ , we have that:

$$\begin{aligned}
f(x_2) - f(x_3) &\geq \frac{(x_3 - x_2)f(x_1) + (x_2 - x_3)f(x_3)}{(x_3 - x_1)} \\
\frac{f(x_2) - f(x_3)}{(x_3 - x_2)} &\geq \frac{f(x_1) - f(x_3)}{(x_3 - x_1)} \\
\frac{f(x_3) - f(x_2)}{(x_3 - x_2)} &\leq \frac{f(x_3) - f(x_1)}{(x_3 - x_1)}
\end{aligned}$$

□

**Lemma 4.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ . We have that

$$\forall a, b, \nabla \text{ s.t. } b \geq a > 0, \nabla \neq 0 : \frac{f(\nabla b)}{b} \leq \frac{f(\nabla a)}{a}.$$

*Proof.* If  $\nabla > 0$  we have that  $0 < \nabla a \leq \nabla b$ .

By Lemma A.1 , we have that

$$\begin{aligned}
\frac{f(\nabla b) - f(0)}{\nabla(b - 0)} &\leq \frac{f(\nabla a) - f(0)}{\nabla(a - 0)} \\
\iff \frac{f(\nabla b)}{b} &\leq \frac{f(\nabla a)}{a}
\end{aligned}$$

If  $\nabla < 0$ , we have that  $\nabla b \leq \nabla a < 0$ .

By Lemma A.1, we have that

$$\begin{aligned} \frac{f(0) - f(\nabla a)}{\nabla(0 - a)} &\leq \frac{f(0) - f(\nabla b)}{\nabla(0 - b)} \\ \iff \frac{f(\nabla a)}{\nabla a} &\leq \frac{f(\nabla b)}{\nabla b} \\ \iff \frac{f(\nabla a)}{a} &\geq \frac{f(\nabla b)}{b}, \text{ since } \nabla < 0 \end{aligned}$$

Thus, when  $\nabla \neq 0$ , we have that

$$\frac{f(\nabla b)}{b} \leq \frac{f(\nabla a)}{a}$$

□

**Lemma 5.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Let  $L(\nabla) = af(\nabla) + bf(-\nabla)$ , where  $a > 0$ ,  $b > 0$ , and  $a \neq b$ .

If  $a > b$ ,  $\exists \delta > 0$ , s.t.  $\forall \nabla^* \in (0, \delta) : L(\nabla^*) > 0$

If  $a < b$ ,  $\exists \delta > 0$ , s.t.  $\forall \nabla^* \in (-\delta, 0) : L(\nabla^*) > 0$ .

*Proof.* By concavity, for all  $\alpha \in (0, 1]$ , we have  $f(\alpha x^*) \geq \alpha f(x^*) > 0$ .

This means that for any  $\nabla \in (0, x^*]$ , we have that  $f(\nabla) > 0$ .

By concavity, for all  $x$ , we have that  $\frac{1}{2}f(x) + \frac{1}{2}f(-x) \leq f(\frac{1}{2}x - \frac{1}{2}x) = f(0) = 0$ .

Thus, for all  $\nabla \in (0, x^*]$  we have that  $0 < f(\nabla) \leq -f(-\nabla)$ .

This means that  $f(\nabla) > 0$  and  $f(-\nabla) < 0$ .

Let  $R(x) = \frac{g(x)}{f(x)}$ , where  $g(x) = -f(-x)$ .

We can show that:

$$\lim_{x \rightarrow 0} R(x) = \lim_{x \rightarrow 0} \frac{g(x)}{f(x)} \stackrel{H}{=} \lim_{x \rightarrow 0} \frac{g'(x)}{f'(x)} = \lim_{x \rightarrow 0} \frac{f'(-x)}{f'(x)} = \frac{f'(0)}{f'(0)} = 1.$$

If  $\nabla \in (0, x^*]$ , by concavity we have that  $0 < f(\nabla) \leq -f(-\nabla)$ , thus  $R(\nabla) = \frac{-f(-\nabla)}{f(\nabla)} \geq 1$ .

Let  $\epsilon = \frac{(a' - b')}{b'}$ , where  $a' > b' > 0$ .

By the definition of the limit,  $\exists \delta > 0$  s.t.  $\forall x$  s.t.  $0 < |x| < \delta$ , we have

$$|R(x) - 1| < \epsilon.$$

Since this is true for all  $x$  s.t.  $0 < |x| < \delta$ , this is also true for all  $0 < \nabla^* < \min(x^*, \delta)$ .

This means that

$$\begin{aligned}
& |R(\nabla^*) - 1| < \epsilon \\
\implies (R(\nabla^*) - 1) & < \frac{(a' - b')}{b'}, \text{ since } R(\nabla) \geq 1 \text{ for all } \nabla \in (0, x^*] \\
\implies R(\nabla^*) & < \frac{a'}{b'} \\
\implies \frac{-f(-\nabla^*)}{f(\nabla^*)} & < \frac{a'}{b'} \\
\implies a'f(\nabla^*) + b'f(-\nabla^*) & > 0
\end{aligned}$$

If  $a > b$ , let  $a' = a$ ,  $b' = b$ , and we have  $a'f(\nabla^*) + b'f(-\nabla^*) > 0$  for all  $0 < \nabla^* < \min(x^*, \delta)$ .

If  $a < b$ , let  $a' = b$ ,  $b' = a$ , and we have  $a'f(\nabla^*) + b'f(-\nabla^*) > 0$  for all  $-\min(x^*, \delta) < \nabla^* < 0$ .

□

**Theorem 6.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Then, we have that

$$D_f^{Rp}(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))]$$

is a divergence.

*Proof.* Let  $C^w(x) = k \forall x$  (worst possible choice of  $C$ ).

Let  $C^*(x) = \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))]$  (best possible choice of  $C$ ).

#1 Proof that  $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \geq 0$

$$D_f^{Rp}(\mathbb{P}, \mathbb{Q}) = \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C^*(x) - C^*(y))] \geq \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C^w(x) - C^w(y))] = 0.$$



**#2** Proof that  $\mathbb{P} = \mathbb{Q} \implies D_f^{Rp}(\mathbb{P}, \mathbb{Q}) = 0$

$$\begin{aligned}
D_f^{Rp}(\mathbb{P}, \mathbb{Q}) &= \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{P}}} [f(C^*(x) - C^*(y))] \\
&= \mathbb{E}_{x \sim \mathbb{P}} \left[ \mathbb{E}_{y \sim \mathbb{P}} [f(C^*(x) - C^*(y)) | x] \right] \\
&\leq \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( \mathbb{E}_{y \sim \mathbb{P}} [C^*(x) - C^*(y) | x] \right) \right] \\
&= \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C^*(x) - \mathbb{E}_{y \sim \mathbb{P}} [C^*(y)] \right) \right] \\
&= \mathbb{E}_{x \sim \mathbb{P}} [f(C'^*(x))], \text{ where } C'^*(x) = C^*(x) - \mathbb{E}_{y \sim \mathbb{P}} [C^*(y)] \\
&\leq f \left( \mathbb{E}_{x \sim \mathbb{P}} [C'^*(x)] \right), \text{ by Jensen's inequality} \\
&= f(0) \\
&= 0
\end{aligned}$$

Since  $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \geq 0$ , we have that  $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) = 0$ .

**#3** Proof that  $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) = 0 \implies \mathbb{P} = \mathbb{Q}$

We prove this by contraposition (i.e., we prove that  $\mathbb{P} \neq \mathbb{Q} \implies D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \neq 0$ ). To do so, we design a function  $C'$  that is better than the worse option ( $C(x) = k \forall x$ ).

Assume that  $\mathbb{P} \neq \mathbb{Q}$ .

Let  $T = \arg \sup_S \mathbb{P}(S) - \mathbb{Q}(S)$ <sup>1</sup>.

Let  $p = \int_T d\mathbb{P}(x) \implies (1 - p) = \int_{X \setminus T} d\mathbb{P}(x)$ .

Let  $q = \int_T d\mathbb{Q}(y) \implies (1 - q) = \int_{X \setminus T} d\mathbb{Q}(y)$ .

Since  $\mathbb{P} \neq \mathbb{Q}$ , we know that  $T \neq \emptyset$ .

This means that  $p > 0$ ,  $q > 0$ , and  $p > q$ .

$$\text{Let } C'(x) = \begin{cases} \nabla & \text{if } x \in T \\ 0 & \text{else} \end{cases}, \text{ where } \nabla \neq 0.$$

---

1. If  $\mathbb{P}$  and  $\mathbb{Q}$  have probability density functions  $p(x)$  and  $q(x)$  respectively, then  $T = \{x | p(x) > q(x)\}$ .

---

Let  $L(\nabla) = \mathbb{E} [f(C'(x) - C'(y))]$ ,  
 $\begin{matrix} x \sim \mathbb{P} \\ y \sim \mathbb{Q} \end{matrix}$

We have that

$$\begin{aligned}
L(\nabla) &= \int_{\mathcal{X}} \int_{\mathcal{X}} f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) \\
&= \int_T \int_T f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) + \\
&\quad \int_T \int_{\mathcal{X} \setminus T} f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) + \\
&\quad \int_{\mathcal{X} \setminus T} \int_T f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) + \\
&\quad \int_{\mathcal{X} \setminus T} \int_{\mathcal{X} \setminus T} f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) \\
&= (1) + (2) + (3) + (4)
\end{aligned}$$

$$\begin{aligned}
(1) \quad &\int_T \int_T f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) \\
&= \int_T \int_T f(\nabla - \nabla) d\mathbb{P}(x)d\mathbb{Q}(y) = 0
\end{aligned}$$

$$\begin{aligned}
(2) \quad &\int_T \int_{\mathcal{X} \setminus T} f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) \\
&= f(\nabla) \int_T d\mathbb{P}(x) \int_{\mathcal{X} \setminus T} d\mathbb{Q}(y) = f(\nabla)p(1 - q)
\end{aligned}$$

$$\begin{aligned}
(3) \quad &\int_{\mathcal{X} \setminus T} \int_T f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) \\
&= f(-\nabla) \int_{\mathcal{X} \setminus T} d\mathbb{P}(x) \int_T d\mathbb{Q}(y) = f(-\nabla)q(1 - p)
\end{aligned}$$

$$\begin{aligned}
(4) \quad &\int_{\mathcal{X} \setminus T} \int_{\mathcal{X} \setminus T} f(C'(x) - C'(y)) d\mathbb{P}(x)d\mathbb{Q}(y) \\
&= \int_{\mathcal{X} \setminus T} \int_{\mathcal{X} \setminus T} f(0 - 0) d\mathbb{P}(x)d\mathbb{Q}(y) = 0
\end{aligned}$$

This means that  $L(\nabla) = af(\nabla) + bf(-\nabla)$ , where  $a = p(1 - q) > 0$  and  $b = q(1 - p) > 0$ .

We know that  $a = p(1 - q) > q(1 - p) = b$ .

Thus, by Lemma A.4, we have that  $\exists \nabla^* > 0$  s.t.  $L(\nabla^*) > 0$ .

---

Thus, if we let  $\nabla = \nabla^*$ , we have that

$$D_f^{Rp}(\mathbb{P}, \mathbb{Q}) = \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C^*(x) - C^*(y))] \geq \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C'(x) - C'(y))] > 0.$$

□

**Theorem 7.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Then, we have that

$$D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right]$$

is a divergence.

*Proof.* Let  $C^w(x) = k \forall x$  (worst possible choice of  $C$ ).

Let  $C^*(x) = \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right]$  (best possible choice of  $C$ ).

#1 Proof that  $D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) \geq 0$

$$\begin{aligned} D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) &= \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C^*(x) - \mathbb{E}_{y \sim \mathbb{Q}} C^*(y) \right) \right] \\ &\geq \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C^w(x) - \mathbb{E}_{y \sim \mathbb{Q}} C^w(y) \right) \right] \\ &= \mathbb{E}_{x \sim \mathbb{P}} [f(k - k)] \\ &= 0. \end{aligned}$$

**#2** Proof that  $\mathbb{P} = \mathbb{Q} \implies D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) = 0$

$$\begin{aligned}
D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right] \\
&= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{P}} C(y) \right) \right], \text{ since } \mathbb{P} = \mathbb{Q} \\
&= \sup_{\substack{C': \mathcal{X} \rightarrow \mathbb{R} \\ \text{s.t. } \mathbb{E}[C'(x)] = 0}} \mathbb{E}_{x \sim \mathbb{P}} [f(C'(x))] \\
&= \mathbb{E}_{x \sim \mathbb{P}} [f(C'^*(x))], \text{ where } C'^* = \arg \sup_{\substack{C': \mathcal{X} \rightarrow \mathbb{R} \\ \text{s.t. } \mathbb{E}[C'(x)] = 0}} \mathbb{E}_{x \sim \mathbb{P}} [f(C'(x))] \\
&\leq f \left( \mathbb{E}_{x \sim \mathbb{P}} [C'^*(x)] \right), \text{ by Jensen's inequality} \\
&= f(0) \\
&= 0
\end{aligned}$$

Since  $D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) \geq 0$ , we have that  $D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) = 0$ .

**#3** Proof that  $D_f^{Ra}(\mathbb{P}, \mathbb{Q}) = 0 \implies \mathbb{P} = \mathbb{Q}$

We prove this by contraposition (i.e., we prove that  $\mathbb{P} \neq \mathbb{Q} \implies D_f^{Ra}(\mathbb{P}, \mathbb{Q}) \neq 0$ ). To do so, we design a function  $C'$  that is better than the worse option ( $C(x) = k \forall x$ ).

Assume that  $\mathbb{P} \neq \mathbb{Q}$ .

Let  $T = \arg \sup_S \mathbb{P}(S) - \mathbb{Q}(S)$ .

Let  $p = \int_T d\mathbb{P}(x) \implies (1-p) = \int_{\mathcal{X} \setminus T} d\mathbb{P}(x)$ .

Let  $q = \int_T d\mathbb{Q}(y) \implies (1-q) = \int_{\mathcal{X} \setminus T} d\mathbb{Q}(y)$ .

Since  $\mathbb{P} \neq \mathbb{Q}$ , we know that  $T \neq \emptyset$ .

This means that  $p > 0$ ,  $q > 0$ , and  $p > q$ .

Let  $C'(x) = \begin{cases} \nabla & \text{if } x \in T \\ 0 & \text{else} \end{cases}$ , where  $\nabla \neq 0$ .

Let  $L(\nabla) = \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C'(x) - \mathbb{E}_{y \sim \mathbb{Q}} C'(y) \right) \right]$ .

We have that

$$\begin{aligned}
L(\nabla) &= \int_{\mathcal{X}} f \left( C'(x) - \mathbb{E}_{y \sim \mathbb{Q}} C'(y) \right) d\mathbb{P}(x) \\
&= \int_{\mathcal{X}} f \left( C'(x) - \int_T \nabla d\mathbb{Q}(y) \right) d\mathbb{P}(x) \\
&= \int_{\mathcal{X}} f(C'(x) - \nabla q) d\mathbb{P}(x) \\
&= \int_T f(\nabla - \nabla q) d\mathbb{P}(x) + \int_{\mathcal{X} \setminus T} f(0 - \nabla q) d\mathbb{P}(x) \\
&= pf(\nabla(1-q)) + (1-p)f(-\nabla q)
\end{aligned}$$

Case 1: If  $q < (1-q)$ , by Lemma A.3, we have that:

$$\begin{aligned}
\frac{f(-\nabla(1-q))}{(1-q)} &\leq \frac{f(-\nabla q)}{q} \\
\implies f(-\nabla q) &\geq \frac{q}{(1-q)} f(-\nabla(1-q))
\end{aligned}$$

Thus,  $L(\nabla) \geq pf(\nabla(1-q)) + \frac{(1-p)q}{(1-q)} f(-\nabla(1-q))$ .

Knowing that  $p > q$  and  $(1-p) < (1-q)$ , we have that  $p > q > \frac{q(1-p)}{(1-q)}$ .

Thus, by Lemma A.4, we have that  $\exists \nabla^* > 0$  s.t.  $L(\nabla^*) > 0$ .

Case 2: If  $q \geq (1-q)$ , by Lemma A.3, we have that:

$$\begin{aligned}
\frac{f(\nabla q)}{q} &\leq \frac{f(\nabla(1-q))}{(1-q)} \\
\implies f(\nabla(1-q)) &\geq \frac{(1-q)}{q} f(\nabla q)
\end{aligned}$$

Thus,  $L(\nabla) \geq \frac{p(1-q)}{q} f(\nabla q) + (1-p)f(-\nabla q)$ .

Knowing that  $p > q$  and  $(1-p) < (1-q)$ , we have that  $(1-p) < (1-q) < \frac{(1-q)p}{q}$ .

Thus, by Lemma A.4, we have that  $\exists \nabla^* > 0$  s.t.  $L(\nabla^*) > 0$ .

Thus, if we let  $\nabla = \nabla^*$ , we have that

$$D_f^{Ralf}(\mathbb{P}, \mathbb{Q}) = \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C^*(x) - \mathbb{E}_{y \sim \mathbb{Q}} C^*(y) \right) \right] \geq \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C'(x) - \mathbb{E}_{y \sim \mathbb{Q}} C'(y) \right) \right] > 0.$$

□

---

**Theorem 8.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Then, we have that

$$D_f^{Ra}(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C(x) - C(y) \right) \right]$$

is a divergence.

*Proof.* Let  $C^w(x) = k \forall x$  (worst possible choice of  $C$ ).

Let  $C^*(x) = \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C(x) - C(y) \right) \right]$   
(best possible choice of  $C$ ).

**#1** Proof that  $D_f^{Ra}(\mathbb{P}, \mathbb{Q}) \geq 0$

$$\begin{aligned} D_f^{Ra}(\mathbb{P}, \mathbb{Q}) &= \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C^*(x) - \mathbb{E}_{y \sim \mathbb{Q}} C^*(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C^*(x) - C^*(y) \right) \right] \\ &\geq \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C^w(x) - \mathbb{E}_{y \sim \mathbb{Q}} C^w(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C^w(x) - C^w(y) \right) \right] \\ &= \mathbb{E}_{x \sim \mathbb{P}} [f(k - k)] + \mathbb{E}_{x \sim \mathbb{Q}} [f(k - k)] \\ &= 0. \end{aligned}$$

**#2** Proof that  $\mathbb{P} = \mathbb{Q} \implies D_f^{Ra}(\mathbb{P}, \mathbb{Q}) = 0$

Let  $C'(x) = C(x) - \mathbb{E}_{x \sim \mathbb{P}} C(x)$

$$\begin{aligned} D_f^{Ra}(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} C(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C(x) - C(y) \right) \right] \\ &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{P}} C(y) \right) \right] + \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C(y) - C(x) \right) \right] \\ &= \sup_{\substack{C': \mathcal{X} \rightarrow \mathbb{R} \\ \text{s.t. } \mathbb{E}[C'(x)] = 0}} \mathbb{E}_{x \sim \mathbb{P}} [f(C'(x)) + f(-C'(x))] \\ &\leq 2 \sup_{\substack{C': \mathcal{X} \rightarrow \mathbb{R} \\ \text{s.t. } \mathbb{E}[C'(x)] = 0}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( \frac{1}{2} C'(x) - \frac{1}{2} C'(x) \right) \right], \text{ by concavity} \\ &= 2 \sup_{\substack{C': \mathcal{X} \rightarrow \mathbb{R} \\ \text{s.t. } \mathbb{E}[C'(x)] = 0}} \mathbb{E}_{x \sim \mathbb{P}} [f(0)] \\ &= 0 \end{aligned}$$

Since  $D_f^{Ra}(\mathbb{P}, \mathbb{Q}) \geq 0$ , we have that  $D_f^{Ra}(\mathbb{P}, \mathbb{Q}) = 0$ .

**#3** Proof that  $D_f^{Ra}(\mathbb{P}, \mathbb{Q}) = 0 \implies \mathbb{P} = \mathbb{Q}$

We prove this by contraposition (i.e., we prove that  $\mathbb{P} \neq \mathbb{Q} \implies D_f^{Ra}(\mathbb{P}, \mathbb{Q}) \neq 0$ ). To do so, we design a function  $C'$  that is better than the worse option ( $C(x) = k \forall x$ ).

Assume that  $\mathbb{P} \neq \mathbb{Q}$ .

Let  $T = \arg \sup_S \mathbb{P}(S) - \mathbb{Q}(S)$ .

Let  $p = \int_T d\mathbb{P}(x) \implies (1-p) = \int_{\mathcal{X} \setminus T} d\mathbb{P}(x)$ .

Let  $q = \int_T d\mathbb{Q}(y) \implies (1-q) = \int_{\mathcal{X} \setminus T} d\mathbb{Q}(y)$ .

Since  $\mathbb{P} \neq \mathbb{Q}$ , we know that  $T \neq \emptyset$ .

This means that  $p > 0$ ,  $q > 0$ , and  $p > q$ .

Let  $C'(x) = \begin{cases} \nabla & \text{if } x \in T \\ 0 & \text{else} \end{cases}$ , where  $\nabla \neq 0$ .

Let  $L(\nabla) = \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C'(x) - \mathbb{E}_{y \sim \mathbb{Q}} C'(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C'(x) - C'(y) \right) \right]$ .

We have that

$$\begin{aligned}
L(\nabla) &= \int_{\mathcal{X}} f \left( C'(x) - \mathbb{E}_{y \sim \mathbb{Q}} C'(y) \right) d\mathbb{P}(x) + \int_{\mathcal{X}} f \left( \mathbb{E}_{x \sim \mathbb{P}} C'(x) - C'(y) \right) d\mathbb{Q}(y) \\
&= \int_{\mathcal{X}} f \left( C'(x) - \int_T \nabla d\mathbb{Q}(y) \right) d\mathbb{P}(x) + \int_{\mathcal{X}} f \left( \int_T \nabla d\mathbb{P}(x) - C'(y) \right) d\mathbb{Q}(y) \\
&= \int_{\mathcal{X}} f (C'(x) - \nabla q) d\mathbb{P}(x) + \int_{\mathcal{X}} f (\nabla p - C'(y)) d\mathbb{Q}(y) \\
&= \int_T f (\nabla(1-q)) d\mathbb{P}(x) + \int_{\mathcal{X} \setminus T} f (-\nabla q) d\mathbb{P}(x) + \\
&\quad \int_T f (\nabla(p-1)) d\mathbb{Q}(y) + \int_T f (\nabla p) d\mathbb{Q}(y) \\
&= pf (\nabla(1-q)) + (1-p)f (-\nabla q) + qf (\nabla(p-1)) + (1-q)f (\nabla p) \\
&= pf (\nabla(1-q)) + (1-p)f (-\nabla q) + qf (-\nabla(1-p)) + (1-q)f (\nabla p)
\end{aligned}$$

---

Case 1: If  $(1 - q) \geq p$ , by Lemma A.3, we have that:

$$\begin{aligned} \frac{f(\nabla(1 - q))}{(1 - q)} &\leq \frac{f(\nabla p)}{p} \\ \implies f(\nabla p) &\geq \frac{p}{(1 - q)} f(\nabla(1 - q)) \end{aligned}$$

Also, we have that  $(1 - p) \geq q$ , thus, by Lemma A.3, we have that:

$$\begin{aligned} \frac{f(-\nabla(1 - p))}{(1 - p)} &\leq \frac{f(-\nabla q)}{q} \\ \implies f(-\nabla q) &\geq \frac{q}{(1 - p)} f(-\nabla(1 - p)) \end{aligned}$$

Also,  $q < p \implies (1 - q) > (1 - p)$ , thus, by Lemma A.3, we have that:

$$\begin{aligned} \frac{f(-\nabla(1 - q))}{(1 - q)} &\leq \frac{f(-\nabla(1 - p))}{(1 - p)} \\ \implies f(-\nabla(1 - p)) &\geq \frac{(1 - p)}{(1 - q)} f(-\nabla(1 - q)) \end{aligned}$$

Thus,

$$\begin{aligned} L(\nabla) &= pf(\nabla(1 - q)) + (1 - p)f(-\nabla q) + qf(-\nabla(1 - p)) + (1 - q)f(\nabla p) \\ &\geq pf(\nabla(1 - q)) + qf(-\nabla(1 - p)) + qf(-\nabla(1 - p)) + pf(\nabla(1 - q)) \\ &= 2pf(\nabla(1 - q)) + 2qf(-\nabla(1 - p)) \\ &\geq 2pf(\nabla(1 - q)) + 2\frac{q(1 - p)}{(1 - q)}f(-\nabla(1 - q)) \end{aligned}$$

Knowing that  $p > q$  and  $(1 - p) < (1 - q)$ , we have that  $2p > 2q > \frac{2q(1 - p)}{(1 - q)}$ .

Thus, by Lemma A.4, we have that  $\exists \nabla^* > 0$  s.t.  $L(\nabla^*) > 0$ .

Case 2: If  $p > (1 - q)$ , by Lemma A.3, we have that:

$$\begin{aligned} \frac{f(\nabla p)}{p} &\leq \frac{f(\nabla(1 - q))}{(1 - q)} \\ \implies f(\nabla(1 - q)) &\geq \frac{(1 - q)}{p} f(\nabla p) \end{aligned}$$



---

Also, we have that  $q > (1 - p)$ , thus, by Lemma A.3, we have that:

$$\begin{aligned} \frac{f(-\nabla q)}{q} &\leq \frac{f(-\nabla(1-p))}{(1-p)} \\ \implies f(-\nabla(1-p)) &\geq \frac{(1-p)}{q} f(-\nabla q) \end{aligned}$$

Also,  $p > q$ , thus, by Lemma A.3, we have that:

$$\begin{aligned} \frac{f(-\nabla p)}{p} &\leq \frac{f(-\nabla q)}{q} \\ \implies f(-\nabla q) &\geq \frac{q}{p} f(-\nabla p) \end{aligned}$$

Thus,

$$\begin{aligned} L(\nabla) &= pf(\nabla(1-q)) + (1-p)f(-\nabla q) + qf(-\nabla(1-p)) + (1-q)f(\nabla p) \\ &\geq (1-q)f(\nabla p) + (1-p)f(-\nabla q) + (1-p)f(-\nabla q) + (1-q)f(\nabla p) \\ &= 2(1-q)f(\nabla p) + 2(1-p)f(-\nabla q) \\ &\geq 2(1-q)f(\nabla p) + 2\frac{q(1-p)}{p}f(-\nabla p) \end{aligned}$$

Knowing that  $p > q$  and  $(1-p) < (1-q)$ , we have that  $2(1-q) > 2(1-p) > 2\frac{q(1-p)}{p}$ . Thus, by Lemma A.4, we have that  $\exists \nabla^* > 0$  s.t.  $L(\nabla^*) > 0$ .

Thus, if we let  $\nabla = \nabla^*$ , we have that

$$\begin{aligned} D_f^{Ra}(\mathbb{P}, \mathbb{Q}) &= \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C^*(x) - \mathbb{E}_{y \sim \mathbb{Q}} C^*(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C^*(x) - C^*(y) \right) \right] \\ &\geq \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C'(x) - \mathbb{E}_{y \sim \mathbb{Q}} C'(y) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} C'(x) - C'(y) \right) \right] \\ &> 0. \end{aligned}$$

□

**Theorem 9.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Let  $\mathbb{M} = \frac{1}{2}\mathbb{P} + \frac{1}{2}\mathbb{Q}$ . Then, we

have that

$$D_f^{Rc}(\mathbb{P}, \mathbb{Q}) = \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{m \sim \mathbb{M}} C(m) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{m \sim \mathbb{M}} C(m) - C(y) \right) \right]$$

is a divergence.

*Proof.* Let  $C^w(x) = k \forall x$  (worst possible choice of  $C$ ).

Let  $C^*(x) = \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{m \sim \mathbb{M}} C(m) \right) \right] + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{m \sim \mathbb{M}} C(m) - C(y) \right) \right]$   
(best possible choice of  $C$ ).

**#1** Proof that  $D_f^{Rc}(\mathbb{P}, \mathbb{Q}) \geq 0$

Same proof as theorem A.6 #1.

**#2** Proof that  $\mathbb{P} = \mathbb{Q} \implies D_f^{Rc}(\mathbb{P}, \mathbb{Q}) = 0$

Same proof as theorem A.6 #2.

**#3** Proof that  $D_f^{Rc}(\mathbb{P}, \mathbb{Q}) = 0 \implies \mathbb{P} = \mathbb{Q}$

We prove this by contraposition (i.e., we prove that  $\mathbb{P} \neq \mathbb{Q} \implies D_f^{Rc}(\mathbb{P}, \mathbb{Q}) \neq 0$ ).

To do so, we design a function  $C'$  that is better than the worse option ( $C(x) = k \forall x$ ).

Assume that  $\mathbb{P} \neq \mathbb{Q}$ .

Make the same assumptions as theorem A.6 #2. The only thing that changes is  $L(\nabla)$ .

We instead have that

$$\begin{aligned} L(\nabla) &= pf(\nabla(1-c)) + (1-p)f(-\nabla c) + qf(-\nabla(1-c)) + (1-q)f(\nabla c) \\ &= L_1(\nabla) + L_2(\nabla), \end{aligned}$$

where  $c = \frac{1}{2}p + \frac{1}{2}q$ ,

$$L_1(\nabla) = pf(\nabla(1-c)) + qf(-\nabla(1-c)),$$

$$L_2(\nabla) = (1-q)f(\nabla c) + (1-p)f(-\nabla c).$$

Knowing that  $p > q$  and  $(1-q) > (1-p)$ , we can use Lemma A.4 to show that  $\exists \delta_1 > 0$ , s.t.  $\forall \nabla_1^* \in (0, \delta_1) : L_1(\nabla_1^*) > 0$  and  $\exists \delta_2 > 0$ , s.t.  $\forall \nabla_2^* \in (0, \delta_2) : L_2(\nabla_2^*) > 0$ .

Thus, let  $\delta = \min(\delta_1, \delta_2)$ . We have that  $\forall \nabla^* \in (0, \delta) : L_1(\nabla^*) > 0$  and  $L_2(\nabla^*) > 0$ .

This means that  $L(\nabla) = L_1(\nabla^*) + L_2(\nabla^*) > 0$

□

---

## B.2 Inequalities between Relativistic Divergences

To prove that  $D_1$  is weaker than  $D_2$ , we can just show that  $D_1(\mathbb{P}, \mathbb{Q}) \leq D_2(\mathbb{P}, \mathbb{Q})$  since we have that:

$$D_1(\mathbb{P}_n, \mathbb{P}) \leq D_2(\mathbb{P}_n, \mathbb{P}) \rightarrow 0 \implies D_1(\mathbb{P}_n, \mathbb{P}) \rightarrow 0.$$

**Theorem 10.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a concave function such that  $f(0) = 0$ ,  $f$  is differentiable at 0,  $f'(0) \neq 0$ ,  $\sup_x f(x) = M > 0$ , and  $\arg \sup_x f(x) > 0$ . Let  $\mathbb{P}$  and  $\mathbb{Q}$  be probability distributions with support  $\mathcal{X}$ . Then, we have that

- $D^S(\mathbb{P}, \mathbb{Q}) \leq D_f^{Rp}(\mathbb{P}, \mathbb{Q})$
- $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \leq D_f^{Ral f}(\mathbb{P}, \mathbb{Q})$  and  $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \leq D_f^{Ra}(\mathbb{P}, \mathbb{Q})$

*Proof.* Showing that  $D^S(\mathbb{P}, \mathbb{Q}) \leq D_f^{Rp}(\mathbb{P}, \mathbb{Q})$ :

Let

$$C_S^*(x) = \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} [f(C(x))] + \mathbb{E}_{z \sim \mathbb{Q}} [f(-C(y))]$$

and

$$C_{Rp}^*(x) = \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))].$$

$$\begin{aligned} D^S(\mathbb{P}, \mathbb{Q}) &= \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} [f(C(x))] + \mathbb{E}_{z \sim \mathbb{Q}} [f(-C(y))] \\ &= 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} \left[ \frac{1}{2} f(C_S^*(x)) + \frac{1}{2} f(-C_S^*(y)) \right] \\ &\leq 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} \left[ f \left( \frac{1}{2} C_S^*(x) - \frac{1}{2} C_S^*(y) \right) \right] \\ &= 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C'(x) - C'(y))], \text{ where } C'(x) = \frac{1}{2} C_S^*(x) \\ &\leq \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))] \\ &= D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \end{aligned}$$

Showing that  $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \leq D_f^{Ral f}(\mathbb{P}, \mathbb{Q})$ :

---


$$\begin{aligned}
D_f^{Rp}(\mathbb{P}, \mathbb{Q}) &= \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))] \\
&= 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C_{Rp}^*(x) - C_{Rp}^*(y))] \\
&= 2 \mathbb{E}_{x \sim \mathbb{P}} \left[ \mathbb{E}_{y \sim \mathbb{Q}} [f(C_{Rp}^*(x) - C_{Rp}^*(y)) | x] \right] \\
&\leq 2 \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( \mathbb{E}_{y \sim \mathbb{Q}} [C_{Rp}^*(x) - C_{Rp}^*(y) | x] \right) \right] \\
&= 2 \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C_{Rp}^*(x) - \mathbb{E}_{y \sim \mathbb{Q}} [C_{Rp}^*(y)] \right) \right] \\
&\leq \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} 2 \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} [C(y)] \right) \right] \\
&= D_f^{Ral}(\mathbb{P}, \mathbb{Q})
\end{aligned}$$

Showing that  $D_f^{Rp}(\mathbb{P}, \mathbb{Q}) \leq D_f^{Ra}(\mathbb{P}, \mathbb{Q})$ :

$$\begin{aligned}
D_f^{Rp}(\mathbb{P}, \mathbb{Q}) &= \arg \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C(x) - C(y))] \\
&= 2 \mathbb{E}_{\substack{x \sim \mathbb{P} \\ y \sim \mathbb{Q}}} [f(C_{Rp}^*(x) - C_{Rp}^*(y))] \\
&= \mathbb{E}_{x \sim \mathbb{P}} \left[ \mathbb{E}_{y \sim \mathbb{Q}} [f(C_{Rp}^*(x) - C_{Rp}^*(y)) | x] \right] \\
&\quad + \mathbb{E}_{y \sim \mathbb{Q}} \left[ \mathbb{E}_{x \sim \mathbb{P}} [f(C_{Rp}^*(x) - C_{Rp}^*(y)) | y] \right] \\
&\leq \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( \mathbb{E}_{y \sim \mathbb{Q}} [C_{Rp}^*(x) - C_{Rp}^*(y) | x] \right) \right] \\
&\quad + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} [C_{Rp}^*(x) - C_{Rp}^*(y) | y] \right) \right] \\
&= \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C_{Rp}^*(x) - \mathbb{E}_{y \sim \mathbb{Q}} [C_{Rp}^*(y)] \right) \right] \\
&\quad + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} [C_{Rp}^*(x)] - C_{Rp}^*(y) \right) \right] \\
&\leq \sup_{C: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{x \sim \mathbb{P}} \left[ f \left( C(x) - \mathbb{E}_{y \sim \mathbb{Q}} [C(y)] \right) \right] \\
&\quad + \mathbb{E}_{y \sim \mathbb{Q}} \left[ f \left( \mathbb{E}_{x \sim \mathbb{P}} [C(x)] - C(y) \right) \right] \\
&= D_f^{Ra}(\mathbb{P}, \mathbb{Q})
\end{aligned}$$

□

---

### B.3 Bias in RalfGANs, RaGANs, and RcGANs

Note that we refer to the second term in RaGANs as "RaGAN2". When possible, we calculate the bias for RalfGANs, RaGAN2s, RaGANs, and RcGANs.

Let

$$\begin{aligned}
\mathbb{E}_{x \sim \mathbb{P}} [C(x)] &= \mu_x, \\
\text{Var}_{x \sim \mathbb{P}} [C(x)] &= \sigma_x^2, \\
\mathbb{E}_{x \sim \mathbb{P}} [C(x)^2] &= \sigma_x^2 + \mu_x^2,
\end{aligned}$$

---


$$\begin{aligned}\mathbb{E}_{y \sim \mathbb{Q}}[C(y)] &= \mu_y, \\ \text{Var}_{y \sim \mathbb{Q}}[C(y)] &= \sigma_y^2, \\ \mathbb{E}_{y \sim \mathbb{Q}}[C(y)^2] &= \sigma_y^2 + \mu_y^2.\end{aligned}$$

In a minibatch of size  $k$ , we have that  $x_1, \dots, x_k$  and  $y_1, \dots, y_k$  are iid. Thus,  $C(x_1), \dots, C(x_k)$  and  $C(y_1), \dots, C(y_k)$  are also iid.

This means that:

$$\begin{aligned}\mathbb{E}[C(x_i)C(x_j)] &= \mathbb{E}[C(x_i)]\mathbb{E}[C(x_j)] = \mu_x^2 \quad \forall i \neq j, \\ \mathbb{E}[C(y_i)C(y_j)] &= \mathbb{E}[C(y_i)]\mathbb{E}[C(y_j)] = \mu_y^2 \quad \forall i \neq j.\end{aligned}$$

### B.3.1 SGAN

$$f(x) = \log(\text{sigmoid}(x)) + \log(2) = -\log(1 + e^{-x}) + \log(2)$$

$$\begin{aligned}\text{Bias}^{\text{RaSGAN}}(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} \left[ f \left( C(x) - \frac{1}{k} \sum_{i=1}^k C(y_i) \right) - f(C(x) - \mu_y) \right] \\ &= \mathbb{E} \left[ -\log \left( 1 + e^{\frac{1}{k} \sum_{i=1}^k C(y_i) - C(x)} \right) + \log \left( 1 + e^{\mu_y - C(x)} \right) \right] \\ &= \mathbb{E} \left[ \log \left( \frac{1 + e^{\mu_y - C(x)}}{1 + e^{\frac{1}{k} \sum_{i=1}^k C(y_i) - C(x)}} \right) \right] \\ &= \mathbb{E} \left[ \log \left( \frac{e^{C(x)} + e^{\mu_y}}{e^{C(x)} + e^{\frac{1}{k} \sum_{i=1}^k C(y_i)}} \right) \right] \\ &= \mathbb{E} \left[ \log \left( e^{C(x)} + e^{\mu_y} \right) - \log \left( e^{C(x)} + e^{\frac{1}{k} \sum_{i=1}^k C(y_i)} \right) \right] \\ &\approx \mathbb{E} \left[ C(x) + e^{\mu_y - C(x)} - C(x) - e^{\frac{1}{k} \sum_{i=1}^k C(y_i) - C(x)} \right] \\ &= \mathbb{E} \left[ \frac{e^{\mu_y} - e^{\frac{1}{k} \sum_{i=1}^k C(y_i)}}{e^{C(x)}} \right]\end{aligned}$$

We cannot find a close form for the bias.

### B.3.2 (Ra) LSGAN

$$f(x) = -(x - 1)^2 + 1$$

$$\begin{aligned}
\widehat{\text{Div}}_1(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} \left[ \frac{1}{k} \sum_{i=1}^k f \left( C(x_i) - \frac{1}{k} \sum_{j=1}^k C(y_j) \right) \right] \\
&= \mathbb{E} \left[ \frac{1}{k} \sum_{i=1}^k \left( - \left( C(x_i) - \frac{1}{k} \sum_{j=1}^k C(y_j) - 1 \right)^2 + 1 \right) \right] \\
&= \frac{1}{k} \sum_{i=1}^k \left( -\mathbb{E} [C(x_i)^2] + \frac{2}{k} \sum_{j=1}^k \mathbb{E} [C(x_i)] \mathbb{E} [C(y_j)] + 2\mathbb{E} [C(x_i)] \right. \\
&\quad \left. - 2\frac{1}{k} \sum_{j=1}^k \mathbb{E} [C(y_j)] - \frac{1}{k^2} \sum_{j=1}^k \mathbb{E} [C(y_j)^2] - \frac{1}{k^2} \sum_{\substack{r=1 \\ r \neq j}}^k \sum_{j=1}^k \mathbb{E} [C(y_j)] \mathbb{E} [C(y_r)] \right) \\
&= \frac{1}{k} \sum_{i=1}^k \left( -\sigma_x^2 - \mu_x^2 + 2\mu_x \mu_y + 2\mu_x - 2\mu_y - \frac{1}{k} (\sigma_y^2 + \mu_y^2) - \frac{(k-1)}{k} \mu_y^2 \right) \\
&= -\sigma_x^2 - \mu_x^2 + 2\mu_x \mu_y + 2\mu_x - 2\mu_y - \frac{1}{k} \sigma_y^2 - \mu_y^2
\end{aligned}$$

---


$$\begin{aligned}
\widehat{\text{Div}}_2(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} \left[ \frac{1}{k} \sum_{j=1}^k f \left( \frac{1}{k} \sum_{i=1}^k C(x_i) - C(y_j) \right) \right] \\
&= \mathbb{E} \left[ \frac{1}{k} \sum_{j=1}^k \left( - \left( \frac{1}{k} \sum_{i=1}^k C(x_i) - C(y_j) - 1 \right)^2 - 1 \right) \right] \\
&= \frac{1}{k} \sum_{j=1}^k \left( -\mathbb{E} [C(y_j)^2] + \frac{2}{k} \sum_{i=1}^k \mathbb{E} [C(x_i)] \mathbb{E} [C(y_j)] - 2\mathbb{E} [C(y_j)] \right. \\
&\quad \left. + 2\frac{1}{k} \sum_{i=1}^k \mathbb{E} [C(x_i)] - \frac{1}{k^2} \sum_{i=1}^k \mathbb{E} [C(x_i)^2] - \frac{1}{k^2} \sum_{\substack{r=1 \\ r \neq i}}^k \sum_{i=1}^k \mathbb{E} [C(x_i)] \mathbb{E} [C(x_r)] \right) \\
&= \frac{1}{k} \sum_{j=1}^k \left( -\sigma_y^2 - \mu_y^2 + 2\mu_x \mu_y - 2\mu_y + 2\mu_x - \frac{1}{k} (\sigma_x^2 + \mu_x^2) - \frac{(k-1)}{k} \mu_x^2 \right) \\
&= -\sigma_y^2 - \mu_y^2 + 2\mu_x \mu_y - 2\mu_y + 2\mu_x - \frac{1}{k} \sigma_x^2 - \mu_x^2
\end{aligned}$$

$$\begin{aligned}
\text{Div}_1(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} [f(C(x) - \mu_y)] \\
&= \mathbb{E} [-(C(x) - \mu_y - 1)^2 - 1] \\
&= \mathbb{E} [-C(x)^2 + 2C(x)\mu_y + 2C(x) - 2\mu_y - \mu_y^2] \\
&= -\sigma_x^2 - \mu_x^2 + 2\mu_x \mu_y + 2\mu_x - 2\mu_y - \mu_y^2
\end{aligned}$$

$$\begin{aligned}
\text{Div}_2(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} [f(\mu_x - C(y))] \\
&= \mathbb{E} [-(\mu_x - C(y) - 1)^2 - 1] \\
&= \mathbb{E} [-\mu_x^2 + 2C(y)\mu_x - 2C(y) + 2\mu_x - C(y)^2] \\
&= -\sigma_y^2 - \mu_y^2 + 2\mu_x \mu_y - 2\mu_y + 2\mu_x - \mu_x^2
\end{aligned}$$



---


$$\begin{aligned}
\text{Bias}^{\text{RaLSGAN}}(\mathbb{P}, \mathbb{Q}) &= \widehat{\text{Div}}_1(\mathbb{P}, \mathbb{Q}) - \text{Div}_1(\mathbb{P}, \mathbb{Q}) \\
&= -\sigma_x^2 - \mu_x^2 + 2\mu_x\mu_y + 2\mu_x - 2\mu_y - \frac{1}{k}\sigma_y^2 - \mu_y^2 + \sigma_x^2 + \mu_x^2 \\
&\quad - 2\mu_x\mu_y - 2\mu_x + 2\mu_y + \mu_y^2 \\
&= -\frac{1}{k}\sigma_y^2
\end{aligned}$$

$$\begin{aligned}
\text{Bias}^{\text{RaLSGAN}^2}(\mathbb{P}, \mathbb{Q}) &= \widehat{\text{Div}}_2(\mathbb{P}, \mathbb{Q}) - \text{Div}_2(\mathbb{P}, \mathbb{Q}) \\
&= -\sigma_y^2 - \mu_y^2 + 2\mu_x\mu_y - 2\mu_y + 2\mu_x - \frac{1}{k}\sigma_x^2 - \mu_x^2 + \sigma_y^2 + \mu_y^2 \\
&\quad - 2\mu_x\mu_y + 2\mu_y - 2\mu_x + \mu_x^2 \\
&= -\frac{1}{k}\sigma_x^2
\end{aligned}$$

$$\begin{aligned}
\text{Bias}^{\text{RalLSGAN}} &= \text{Bias}^{\text{RaLSGAN}}(\mathbb{P}, \mathbb{Q}) + \text{Bias}^{\text{RaLSGAN}^2}(\mathbb{Q}, \mathbb{P}) \\
&= -\frac{1}{k}\sigma_y^2 - \frac{1}{k}\sigma_x^2 \\
&= -\frac{1}{k}(\sigma_x^2 + \sigma_y^2)
\end{aligned}$$

Let

$$\begin{aligned}
\hat{\sigma}_x^2 &= \frac{1}{(k-1)} \sum_{i=1}^k \left( C(x_i) - \frac{1}{k} \sum_{i=1}^k C(x_j) \right), \\
\hat{\sigma}_y^2 &= \frac{1}{(k-1)} \sum_{i=1}^k \left( C(y_i) - \frac{1}{k} \sum_{i=1}^k C(y_j) \right).
\end{aligned}$$

We know that  $\hat{\sigma}_x^2$  and  $\hat{\sigma}_y^2$  are unbiased estimators of  $\sigma_x^2$  and  $\sigma_y^2$  respectively. Thus, if we add  $\frac{1}{k}\hat{\sigma}_y^2$  to the objective function of RalLSGAN and  $\frac{1}{k}(\hat{\sigma}_x^2 + \hat{\sigma}_y^2)$  to the objective function of RaLSGAN, we have that the new objective functions are unbiased.

### B.3.3 (Rc) LSGAN

$$\begin{aligned}
\widehat{\text{Div}}_1(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} \left[ \frac{1}{k} \sum_{i=1}^k f \left( C(x_i) - \frac{1}{2k} \sum_{j=1}^k (C(x_j) + C(y_j)) \right) \right] \\
&= \mathbb{E} \left[ \frac{1}{k} \sum_{i=1}^k \left( - \left( C(x_i) - \frac{1}{2k} \sum_{j=1}^k (C(x_j) + C(y_j)) - 1 \right)^2 + 1 \right) \right] \\
&= \frac{1}{k} \sum_{i=1}^k \left( -\mathbb{E} [C(x_i)^2] + \frac{1}{k} \mathbb{E} [C(x_i)^2] + \frac{1}{k} \sum_{\substack{j=1 \\ j \neq i}}^k \mathbb{E} [C(x_i)] \mathbb{E} [C(x_j)] \right. \\
&\quad + \frac{1}{k} \sum_{j=1}^k \mathbb{E} [C(x_i)] \mathbb{E} [C(y_j)] + 2\mathbb{E} [C(x_i)] - \frac{1}{k} \sum_{j=1}^k \mathbb{E} [C(x_j)] \\
&\quad - \frac{1}{k} \sum_{j=1}^k \mathbb{E} [C(y_j)] - \frac{1}{4k^2} \sum_{j=1}^k \mathbb{E} [(C(x_j) + C(y_j))^2] \\
&\quad \left. - \frac{1}{4k^2} \sum_{\substack{r=1 \\ r \neq j}}^k \sum_{j=1}^k \mathbb{E} [C(x_i) + C(y_i)] \mathbb{E} [C(x_r) + C(y_r)] \right) \\
&= \left( \frac{1}{k} - 1 \right) (\sigma_x^2 + \mu_x^2) + \frac{(k-1)}{k} \mu_x^2 + \mu_x \mu_y + 2\mu_x - \mu_x - \mu_y \\
&\quad - \frac{1}{4k} ((\sigma_x^2 + \mu_x^2) + 2\mu_x \mu_y + (\sigma_y^2 + \mu_y^2)) - \frac{(k-1)}{4k} (\mu_x^2 + 2\mu_x \mu_y + \mu_y^2) \\
&= \frac{(1-k)}{k} \sigma_x^2 + \mu_x \mu_y + \mu_x - \mu_y - \frac{1}{4} \mu_x^2 - \frac{1}{2} \mu_x \mu_y - \frac{1}{4} \mu_y^2 - \frac{1}{4k} \sigma_x^2 - \frac{1}{4k} \sigma_y^2 \\
&= \frac{(.75-k)}{k} \sigma_x^2 - \frac{1}{4k} \sigma_y^2 - \frac{1}{4} \mu_x^2 - \frac{1}{4} \mu_y^2 + \frac{1}{2} \mu_x \mu_y + \mu_x - \mu_y
\end{aligned}$$

$$\begin{aligned}
\widehat{\text{Div}}_1(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} \left[ \frac{1}{k} \sum_{i=1}^k \left( - \left( C(y_i) - \frac{1}{2k} \sum_{j=1}^k (C(x_j) + C(y_j)) + 1 \right)^2 + 1 \right) \right] \\
&= \frac{1}{k} \sum_{i=1}^k \left( -\mathbb{E}[C(y_i)^2] + \frac{1}{k} \mathbb{E}[C(y_i)^2] + \frac{1}{k} \sum_{\substack{j=1 \\ j \neq i}}^k \mathbb{E}[C(y_i)] \mathbb{E}[C(y_j)] \right. \\
&\quad + \frac{1}{k} \sum_{j=1}^k \mathbb{E}[C(x_i)] \mathbb{E}[C(y_j)] - 2\mathbb{E}[C(y_i)] + \frac{1}{k} \sum_{j=1}^k \mathbb{E}[C(x_j)] \\
&\quad \left. - \frac{1}{4k^2} \sum_{j=1}^k \mathbb{E}[(C(x_j) + C(y_j))^2] \right. \\
&\quad \left. + \frac{1}{k} \sum_{j=1}^k \mathbb{E}[C(y_j)] - \frac{1}{4k^2} \sum_{\substack{r=1 \\ r \neq j}}^k \sum_{j=1}^k \mathbb{E}[C(x_i) + C(y_i)] \mathbb{E}[C(x_r) + C(y_r)] \right) \\
&= \left( \frac{1}{k} - 1 \right) (\sigma_y^2 + \mu_y^2) + \frac{(k-1)}{k} \mu_y^2 + \mu_x \mu_y - 2\mu_y + \mu_x + \mu_y \\
&\quad - \frac{1}{4k} ((\sigma_x^2 + \mu_x^2) + 2\mu_x \mu_y + (\sigma_y^2 + \mu_y^2)) - \frac{(k-1)}{4k} (\mu_x^2 + 2\mu_x \mu_y + \mu_y^2) \\
&= \frac{(1-k)}{k} \sigma_y^2 + \mu_x \mu_y + \mu_x - \mu_y - \frac{1}{4} \mu_x^2 - \frac{1}{2} \mu_x \mu_y - \frac{1}{4} \mu_y^2 - \frac{1}{4k} \sigma_x^2 - \frac{1}{4k} \sigma_y^2 \\
&= \frac{(.75-k)}{k} \sigma_y^2 - \frac{1}{4k} \sigma_x^2 - \frac{1}{4} \mu_x^2 - \frac{1}{4} \mu_y^2 + \frac{1}{2} \mu_x \mu_y + \mu_x - \mu_y
\end{aligned}$$

$$\begin{aligned}
\text{Div}_1(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} \left[ f \left( C(x) - \frac{(\mu_x + \mu_y)}{2} \right) \right] \\
&= \mathbb{E} \left[ - \left( C(x) - \frac{(\mu_x + \mu_y)}{2} - 1 \right)^2 - 1 \right] \\
&= \mathbb{E} \left[ -C(x)^2 + C(x)(\mu_x + \mu_y) + 2C(x) - (\mu_x + \mu_y) - \frac{(\mu_x + \mu_y)^2}{4} \right] \\
&= -\sigma_x^2 - \mu_x^2 + \mu_x^2 + \mu_x \mu_y + \mu_x - \mu_y - \frac{1}{4} (\mu_x^2 + \mu_y^2 + 2\mu_x \mu_y) \\
&= -\sigma_x^2 + \frac{1}{2} \mu_x \mu_y + \mu_x - \mu_y - \frac{1}{4} \mu_x^2 - \frac{1}{4} \mu_y^2
\end{aligned}$$

---


$$\begin{aligned}
\text{Div}_2(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} [f(C(x) - \mu_y)] \\
&= \mathbb{E} \left[ - \left( C(y) - \frac{(\mu_x + \mu_y)}{2} + 1 \right)^2 - 1 \right] \\
&= \mathbb{E} \left[ -C(y)^2 + C(y)(\mu_x + \mu_y) - 2C(y) + (\mu_x + \mu_y) - \frac{(\mu_x + \mu_y)^2}{4} \right] \\
&= -\sigma_y^2 - \mu_y^2 + \mu_x^2 + \mu_x \mu_y + \mu_x - \mu_y - \frac{1}{4}(\mu_x^2 + \mu_y^2 + 2\mu_x \mu_y) \\
&= -\sigma_x^2 + \frac{1}{2}\mu_x \mu_y + \mu_x - \mu_y - \frac{1}{4}\mu_x^2 - \frac{1}{4}\mu_y^2
\end{aligned}$$

$$\begin{aligned}
\text{Bias}^{\text{RcLSGAN}}(\mathbb{P}, \mathbb{Q}) &= \widehat{\text{Div}}_1(\mathbb{P}, \mathbb{Q}) - \text{Div}_1(\mathbb{P}, \mathbb{Q}) \\
&= \frac{3}{4k}\sigma_x^2 - \frac{1}{4k}\sigma_y^2
\end{aligned}$$

$$\begin{aligned}
\text{Bias}^{\text{RcLSGAN}^2}(\mathbb{P}, \mathbb{Q}) &= \widehat{\text{Div}}_2(\mathbb{P}, \mathbb{Q}) - \text{Div}_2(\mathbb{P}, \mathbb{Q}) \\
&= \frac{3}{4k}\sigma_y^2 - \frac{1}{4k}\sigma_x^2
\end{aligned}$$

Let

$$\begin{aligned}
\hat{\sigma}_x^2 &= \frac{1}{(k-1)} \sum_{i=1}^k \left( C(x_i) - \frac{1}{k} \sum_{i=1}^k C(x_j) \right)^2, \\
\hat{\sigma}_y^2 &= \frac{1}{(k-1)} \sum_{i=1}^k \left( C(y_i) - \frac{1}{k} \sum_{i=1}^k C(y_j) \right)^2.
\end{aligned}$$

We know that  $\hat{\sigma}_x^2$  and  $\hat{\sigma}_y^2$  are unbiased estimators of  $\sigma_x^2$  and  $\sigma_y^2$  respectively.

Thus, if we subtract  $\frac{1}{2k}(\hat{\sigma}_x^2 + \hat{\sigma}_y^2)$  to the objective function of RcLSGAN, we have that the new objective functions are unbiased.

### B.3.4 HingeGAN

$$f(x) = -\max(0, 1 - x) + 1$$

For simplicity:

Let  $x' = C(x)$ ,  $y'_i = C(y_i)$ ,  $p(x)$  and  $q(x)$  be the probability density functions of  $x'$

and  $y'_i$ .

$$\begin{aligned} \text{Div}(\mathbb{P}, \mathbb{Q}) &= \mathbb{E} \left[ f \left( C(x) - \frac{1}{k} \sum_{i=1}^k C(y_i) \right) \right] \\ &= \mathbb{E} \left[ -\max \left( 0, 1 + \frac{1}{k} \sum_{i=1}^k y'_i - x' \right) + 1 \right] \end{aligned}$$

This is non-linear and we cannot derive a close-form.

---

## B.4 Architecture

Generator
$z \in \mathbb{R}^{128} \sim N(0, I)$
linear, 128 -> 512*4*4
Reshape, 512*4*4 -> 512 x 4 x 4
ConvTranspose2d 4x4, stride 2, pad 1, 512->256
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, 256->128
BN and ReLU
ConvTranspose2d 4x4, stride 2, pad 1, 128->64
BN and ReLU
ConvTranspose2d 3x3, stride 1, pad 1, 64->3
Tanh

---

Discriminator

---

---

$$x \in \mathbb{R}^{3 \times 32 \times 32}$$

---

Conv2d 3x3, stride 1, pad 1, 3->64

---

LeakyReLU 0.1

---

Conv2d 4x4, stride 2, pad 1, 64->64

---

LeakyReLU 0.1

---

Conv2d 3x3, stride 1, pad 1, 64->128

---

LeakyReLU 0.1

---

Conv2d 4x4, stride 2, pad 1, 128->128

---

LeakyReLU 0.1

---

Conv2d 3x3, stride 1, pad 1, 128->256

---

LeakyReLU 0.1

---

Conv2d 4x4, stride 2, pad 1, 256->256

---

LeakyReLU 0.1

---

Conv2d 3x3, stride 1, pad 1, 256->512

---

Reshape, 512 x 4 x 4 -> 512\*4\*4

---

linear, 512\*4\*4 -> 1

---



# Adversarial score matching and improved sampling for image generation

---

## C.1 Broader Impact

Unfortunately, these improvements in image generation come at a very high computational cost, meaning that the ability to generate high-resolution images is constrained by the availability of large computing resources (TPUs or clusters of 8+ GPUs). This is mainly due to the architectures used in this paper, while adding a discriminator further adds to the training computational load.

---

## C.2 Experiments details

**Table C.1** – Sampling learning rates for non-adversarial ( $\epsilon$ ) and adversarial ( $\epsilon_{adv}$ ) score matching

Network architecture	Dataset	Consistent	$n_\sigma$	$\epsilon$	$\epsilon_{adv}$
Song et al. (2020)	CIFAR-10	No	1	1.8e-5	1.725e-5
Song et al. (2020)	CIFAR-10	No	5	3.6e-6	3.7e-6
Song et al. (2020)	CIFAR-10	Yes	1	5.6e-6	5.55e-6
Song et al. (2020)	CIFAR-10	Yes	5	1.1e-6	1.05e-6
Song et al. (2020)	LSUN-Churches	No	1	4.85e-6	4.85e-6
Song et al. (2020)	LSUN-Churches	No	5	9.7e-7	9.7e-7
Song et al. (2020)	LSUN-Churches	Yes	1	2.8e-6	2.8e-6
Song et al. (2020)	LSUN-Churches	Yes	5	4.5e-7	4.5e-7
Ho et al. (2020)	CIFAR-10	No	1	1.6e-5	1.66e-05
Ho et al. (2020)	CIFAR-10	No	5	4.0e-6	4.25e-6
Ho et al. (2020)	CIFAR-10	Yes	1	5.45e-6	5.6e-6
Ho et al. (2020)	CIFAR-10	Yes	5	1.05e-6	1e-6
Song et al. (2020)	3-StackedMNIST	Yes	1	5.0e-6	5.0e-6

Following the recommendations from [Song and Ermon \(2020\)](#), we chose  $\sigma_1 = 50$  and  $L = 232$  on CIFAR-10, and  $\sigma_1 = 140$ ,  $L = 788$  on LSUN-Churches, both with  $\sigma_L = 0.01$ . We used a batch size of 128 in all models. We first swept summarily the training checkpoint (saved at every 2.5k iterations), the Exponential Moving Average (EMA) coefficient from  $\{.999, .9999\}$ , and then swept over the sampling step size  $\epsilon$  with approximately 2 significant number precision. The values reported in [Table 8.1](#) correspond to the sampling step size that minimized the denoised FID for every  $n_\sigma$  (See [Table C.1](#)). We used the same sampling step sizes for adversarial and non-adversarial. Empirically, the optimal sampling step size is found for a certain  $n_\sigma$  and is extrapolated to other precision levels by solving  $\beta_{n'_\sigma} = \beta_{n_\sigma} \sqrt{n_\sigma/n'_\sigma}$  for the consistent algorithm. In the non-consistent algorithm, we found the best sampling step size at  $n_\sigma = 1$  and divided by 5 to obtain a starting point to find the optimal value at  $n_\sigma = 5$ . The best EMA values were found to be .9999 in CIFAR-10



---

and .999 in LSUN-churches. The number of score network training iterations was 300k on CIFAR-10 and 200k on LSUN-Churches.

Of note, Song and Ermon (2020) used non-denoised non-consistent sampling with  $n_\sigma = 5$  and  $n_\sigma = 4$  for CIFAR-10 and LSUN-Churches respectively. However, they did not use the same learning rates (we tuned ours more precisely) and they used bigger images for LSUN-Churches.

Regarding the adversarial approach, we swept for the GAN loss function, the number of discriminator steps per score network steps ( $n_D \in \{1, 2\}$ ), the Adam optimizer (Kingma and Ba, 2014) parameters, and the hyperparameter  $\lambda$  (see Eq. 8.5) based on quick experiments on CIFAR-10. LSGAN (Mao et al., 2017) yielded the best FID scores among all other loss functions considered, namely the original GAN loss function (Goodfellow et al., 2020), HingeGAN (Lim and Ye, 2017), as well as their relativistic counterparts (Jolicœur-Martineau, 2019, 2020). Note that the saturating variant (see Goodfellow et al. (2020)) on LSGAN worked as well as its non-saturating version; we did not use it for simplicity. Following the trend towards zero or negative momentum (Gidel et al., 2019), we used the Adam optimizer with hyperparameters  $(\beta_1, \beta_2) = (-.5, .9)$  for the discriminator and  $(\beta_1, \beta_2) = (0, .9)$  for the score network. These values were found by sweeping over  $\beta_1 \in \{.9, .5, 0, -.5\}$  and  $\beta_2 \in \{.9, .999\}$ . We found the simple setting  $\lambda = 1$  to perform comparatively better than more complex weighting schemes. We used  $n_D = 2$  on CIFAR-10 and  $n_D = 1$  on LSUN-Churches.

The 3-Stacked MNIST experiment was conducted with an arbitrary EMA of .999. The sampling step size was broadly swept upon. Following (Song and Ermon, 2020) hyperparameter recommendations, we obtained  $\sigma_1 = 50$ ,  $L = 200$ .

For the synthetic 2D experiments, we used Langevin sampling with  $n_\sigma = 10$ ,  $\epsilon = .2$ , and  $\sigma = .1$ .

---

### C.3 Supplementary experiments

Due to limited computing resources (4 V100 GPUs), the training of models on FFHQ (70k images) in 256x256 (Karras et al., 2019), with the same setting as previously done by Song and Ermon (2020), was impossible. Using a reduced model yielded very poor results. The adversarial version performed worse than the other;

we suspect this was the case due to the mini-batch of size 32, our computational limit, while the BigGAN architecture normally assumes very large batch sizes of 2048 when working with images of that size (256x256 or higher).

---

## C.4 Optimal unconditional score function

Recall the loss from Eq. 8.1 where  $s(\tilde{\mathbf{x}}, \sigma) = s(\tilde{\mathbf{x}})/\sigma$ , meaning  $s$  is no longer conditioned on the noise level:

$$\mathcal{L}[s] = \frac{1}{L} \sum_{i=1}^L \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}), \mathbf{x} \sim p(\mathbf{x})} \left[ \frac{1}{2} \left\| s(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma_i} \right\|_2^2 \right] \quad (\text{C.1})$$

$$= \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x}), \mathbf{x} \sim p(\mathbf{x}), \sigma \sim p(\sigma)} \left[ \frac{1}{2} \left\| s(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma} \right\|_2^2 \right], \quad (\text{C.2})$$

where  $p(\sigma)$  is chosen to be a discrete uniform distribution over a specific set of values. We use this expectation formulation over  $\sigma$  to obtain a more general result; the choice of  $p(\sigma)$  is not important for this derivation.

According to the Euler-Lagrange equation, a function  $s$  that minimizes the denoising score matching objective  $\mathcal{L}[s]$  will satisfy

$$\frac{\partial F(\tilde{\mathbf{x}}, s)}{\partial s} = 0 \quad \text{where}$$

$$\mathcal{L}[s] = \mathbb{E}_{q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})p(\mathbf{x})p(\sigma)} \left[ \frac{1}{2} \left\| s(\tilde{\mathbf{x}}) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma} \right\|_2^2 \right] = \int_{\tilde{\mathbf{x}}} F(\tilde{\mathbf{x}}, s) d\tilde{\mathbf{x}} \quad (\text{C.3})$$

Solving for  $s$ , we get

$$\begin{aligned}
\frac{\partial F(\tilde{\mathbf{x}}, s)}{\partial s} &= \frac{1}{2} \frac{\partial}{\partial s} \int_{\sigma} \int_{\mathbf{x}} q(\tilde{\mathbf{x}}|\mathbf{x}, \sigma) p(\mathbf{x}) p(\sigma) \left\| s(\tilde{\mathbf{x}}) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma} \right\|_2^2 d\mathbf{x} d\sigma = 0 \\
\iff \int_{\sigma} \int_{\mathbf{x}} q(\tilde{\mathbf{x}}|\mathbf{x}, \sigma) p(\mathbf{x}) p(\sigma) \left( s(\tilde{\mathbf{x}}) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma} \right) d\mathbf{x} d\sigma &= 0 \\
\iff s(\tilde{\mathbf{x}}) q(\tilde{\mathbf{x}}) &= \int_{\sigma} \frac{1}{\sigma} \int_{\mathbf{x}} q(\tilde{\mathbf{x}}|\mathbf{x}, \sigma) p(\mathbf{x}) p(\sigma) (\mathbf{x} - \tilde{\mathbf{x}}) d\mathbf{x} d\sigma \\
\iff s(\tilde{\mathbf{x}}) &= \int_{\sigma} \frac{q(\tilde{\mathbf{x}}|\sigma) p(\sigma)}{q(\tilde{\mathbf{x}})} \frac{1}{\sigma} \int_{\mathbf{x}} q(\mathbf{x}|\tilde{\mathbf{x}}, \sigma) (\mathbf{x} - \tilde{\mathbf{x}}) d\mathbf{x} d\sigma \\
\iff s(\tilde{\mathbf{x}}) &= \int_{\sigma} q(\sigma|\tilde{\mathbf{x}}) \frac{1}{\sigma} \int_{\mathbf{x}} q(\mathbf{x}|\tilde{\mathbf{x}}, \sigma) (\mathbf{x} - \tilde{\mathbf{x}}) d\mathbf{x} d\sigma \\
\iff s(\tilde{\mathbf{x}}) &= \mathbb{E}_{\sigma \sim q(\sigma|\tilde{\mathbf{x}})} \left[ \frac{\mathbb{E}_{\mathbf{x} \sim q(\mathbf{x}|\tilde{\mathbf{x}}, \sigma)}[\mathbf{x}] - \tilde{\mathbf{x}}}{\sigma} \right]. \tag{C.4}
\end{aligned}$$

---

## C.5 Optimal conditional score function

As the explicit optimal score function is obtained in Appendix C.4 for the unconditional case, a similar result can be obtained for the conditional case. Recall the loss from Eq. 8.1

$$\mathcal{L}[s] = \frac{1}{L} \sum_{i=1}^L \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}), \mathbf{x} \sim p(\mathbf{x})} \left[ \frac{\sigma_i^2}{2} \left\| s(\tilde{\mathbf{x}}, \sigma_i) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma_i^2} \right\|_2^2 \right]$$

Then, the minimizer  $s^*$  of  $\mathcal{L}[s]$  would be such that  $s(\cdot, \sigma_i)$  minimises

$$\mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}), \mathbf{x} \sim p(\mathbf{x})} \left[ \frac{\sigma_i^2}{2} \left\| s(\tilde{\mathbf{x}}, \sigma_i) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma_i^2} \right\|_2^2 \right] \quad \forall i \in \{1, \dots, L\}.$$

Applying the same steps that in Appendix C.4, one gets that the minimizer is such that

$$s^*(\tilde{\mathbf{x}}, \sigma) = \frac{\mathbb{E}_{\mathbf{x} \sim q_{\sigma}(\mathbf{x}|\tilde{\mathbf{x}})}[\mathbf{x}] - \tilde{\mathbf{x}}}{\sigma^2}.$$

Let us finally demonstrate the equivalence between this term and the score of the corrupted distribution of standard deviation  $\sigma$ :

---


$$\begin{aligned}
\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) &= \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \nabla_{\tilde{\mathbf{x}}} \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x} \\
&= \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \int p(\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x} \\
&= \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x} \\
&= \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma^2} d\mathbf{x} \\
&= \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \frac{1}{\sigma^2} \left( \int q_\sigma(\mathbf{x}|\tilde{\mathbf{x}}) q_\sigma(\tilde{\mathbf{x}}) \mathbf{x} d\mathbf{x} - \tilde{\mathbf{x}} q_\sigma(\tilde{\mathbf{x}}) \right) \\
&= \frac{\mathbb{E}_{\mathbf{x} \sim q_\sigma(\mathbf{x}|\tilde{\mathbf{x}})}[\mathbf{x}] - \tilde{\mathbf{x}}}{\sigma^2}
\end{aligned}$$

making use of the fact that  $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 \mathbf{I})$ ,  $q_\sigma(\tilde{\mathbf{x}}) \triangleq \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}$  and that  $q_\sigma(\mathbf{x}|\tilde{\mathbf{x}}) = \frac{p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{q_\sigma(\tilde{\mathbf{x}})}$

---

## C.6 ALS Non-geometric proof

**Proposition 1.** *Let  $s^*$  be the optimal score function from Eq. 8.3. Following the sampling described in Algorithm 4, the variance of the noise component in the sample  $\mathbf{x}$  will remain greater than  $\sigma_t^2$  at every step  $t$ .*

*Proof.* Assume at the start of an iteration of Langevin Sampling that the point  $\mathbf{x}$  is comprised of an *image component* and of a *noise component* denoted  $v_0 \mathbf{z}_0$  for  $\mathbf{z}_0 \sim \mathcal{N}(0, \mathbf{I})$ . We assume from the proposition statement that the Langevin Sampling is performed at the level of variance  $\sigma_t^2$ , meaning the update rule is as follows:

$$\mathbf{x} \leftarrow \mathbf{x} + \eta \sigma_t^2 s^*(\mathbf{x}, \sigma_t) + \sigma_t \sqrt{2\eta} \mathbf{z}$$

for  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$  and  $0 < \eta < 1$ . From Eq. 8.3, we get:

$$\begin{aligned}
\mathbf{x} &\leftarrow \mathbf{x} + \eta (\mathbb{E}_{\mathbf{x}' \sim q_{v_0}(\mathbf{x}'|\mathbf{x})}[\mathbf{x}'] - \mathbf{x}) + \sigma_t \sqrt{2\eta} \mathbf{z} \\
&= (1 - \eta) \mathbf{x} + \eta \mathbb{E}_{\mathbf{x}' \sim q_{v_0}(\mathbf{x}'|\mathbf{x})}[\mathbf{x}'] + \sigma_t \sqrt{2\eta} \mathbf{z}.
\end{aligned}$$

---

The noise component of  $(1 - \eta)\mathbf{x}$  and  $\sigma_t\sqrt{2\eta}\mathbf{z}$  can then be summed as:

$$(1 - \eta)v_0\mathbf{z}_0 + \sigma_t\sqrt{2\eta}\mathbf{z}.$$

Making use of the fact that both sources of noise follow independent normal distributions, the sum will be normally distributed, centered and of variance  $v_1^2$ , with

$$v_1^2 = v_0^2(1 - \eta)^2 + 2\eta\sigma_t^2.$$

Applying the same steps allows us to examine the variance after multiple Langevin Sampling iterations

$$\begin{aligned} v_2^2 &= v_1^2(1 - \eta)^2 + 2\eta\sigma_t^2 \\ &= v_0^2(1 - \eta)^4 + 2\eta\sigma_t^2 + 2\eta\sigma_t^2(1 - \eta)^2 \\ &\vdots \\ v_n^2 &= v_0^2(1 - \eta)^{2n} + 2\eta\sigma_t^2 \sum_{i=0}^{n-1} (1 - \eta)^{2i} \\ &= v_0^2(1 - \eta)^{2n} + \frac{2\sigma_t^2}{2 - \eta}(1 - (1 - \eta)^{2n}). \end{aligned}$$

From there, the two following statements can be obtained:

- (1)  $\lim_{n \rightarrow \infty} v_n = \sigma_t \sqrt{\frac{2}{2 - \eta}} > \sigma_t$
- (2)  $\frac{dv_n}{dn} < 0 \iff \eta < 2 - \frac{2\sigma_t^2}{v_0^2}.$

From these observations, we understand that  $v_n^2$  is monotonically decreasing (under conditions generally respected in practice) but converges to a point superior to  $\sigma_t^2$  after an infinite number of Langevin Sampling steps. We then conclude that for all  $n$ , the variance of the noise component in the sample will always exceed  $\sigma_t^2$ . We also note that this will be true across the full sampling, at every step  $t$ .

In the particular case where  $\sigma_t$  corresponds to a geometrically decreasing series, it means that even given an optimal score function, the standard deviation of the noise component cannot follow its prescribed schedule.

□

---

## C.7 CAS Geometric proof

**Proposition 2.** *Let  $s^*$  be the optimal score function from Eq. 8.3. Following the sampling described in Algorithm 5, the variance of the noise component in the sample  $\mathbf{x}$  will consistently be equal to  $\sigma_t^2$  at every step  $t$ .*

*Proof.* Let us first define  $\beta$  to be equal to  $\sqrt{1 - (1 - \eta)^2/\gamma^2}$  with  $\eta = \epsilon/\sigma_L^2$  and  $\gamma$  defined as Eq. 8.2. At the start of Algorithm 5, assume that  $\mathbf{x}$  is comprised of an *image component* and a *noise component* denoted  $\sigma_0\mathbf{z}_0$ , where  $\mathbf{z}_0 \sim \mathcal{N}(0, \mathbf{I})$  and  $\sigma_0 = \sigma_1/\gamma$ . We will proceed by induction to show that the noise component at step  $t$  will be a Gaussian of variance  $\sigma_t^2$  for every  $t$ .

The first induction step is trivial. Assume the noise component of  $\mathbf{x}_t$  to be  $\sigma_t\mathbf{z}_t$ , where  $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$ . Following Algorithm 5, the update step will be:

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \eta\sigma_t^2 s^*(\mathbf{x}_t, \sigma_t) + \sigma_{t+1}\beta\mathbf{z},$$

with  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$  and  $0 < \eta < 1$ . From Eq. 8.3, we get

$$\begin{aligned} \mathbf{x}_{t+1} &\leftarrow \mathbf{x}_t + \eta(\mathbb{E}_{\mathbf{x} \sim q_{\sigma_t}(\mathbf{x}|\mathbf{x}_t)}[\mathbf{x}] - \mathbf{x}_t) + \sigma_{t+1}\beta\mathbf{z} \\ &= (1 - \eta)\mathbf{x}_t + \eta\mathbb{E}_{\mathbf{x} \sim q_{\sigma_t}(\mathbf{x}|\mathbf{x}_t)}[\mathbf{x}] + \sigma_{t+1}\beta\mathbf{z}. \end{aligned}$$

The noise component from  $(1 - \eta)\mathbf{x}_t$  and  $\sigma_{t+1}\beta\mathbf{z}$  can then be summed as:

$$\sigma_t(1 - \eta)\mathbf{z}_t + \sigma_{t+1}\beta\mathbf{z}.$$

Making use of the fact that both sources of noise follow independent normal distributions, the sum will be normally distributed, centered and of variance:

$$\sigma_t^2(1 - \eta)^2 + \sigma_{t+1}^2\beta^2 = \sigma_{t+1}^2 \left[ \left( \frac{1 - \eta}{\gamma} \right)^2 + \beta^2 \right] = \sigma_{t+1}^2.$$

By induction, the noise component of the sample  $\mathbf{x}$  will follow a Gaussian distribution of variance  $\sigma_i^2 \forall i \in \{0, \dots, L\}$ . In the particular case where  $\sigma_i$  corresponds to a geometrically decreasing series, it means that given an optimal score function, the standard deviation of the noise component will follow its prescribed schedule.  $\square$

---

## C.8 Update rule

**Proposition 3.** *Given a noise-conditional score function, the update rules from Algorithm 4 and Algorithm 5 are equivalent to the respective following update rules:*

$$\begin{aligned}\mathbf{x} &\leftarrow (1 - \eta)\mathbf{x} + \eta H(\mathbf{x}, \sigma_i) + \sqrt{2\eta}\sigma_i \mathbf{z} && \text{for } \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \text{ and } \eta = \frac{\epsilon}{\sigma_L^2} \\ \mathbf{x} &\leftarrow (1 - \eta)\mathbf{x} + \eta H(\mathbf{x}, \sigma_i) + \beta\sigma_{i+1}\mathbf{z}\end{aligned}$$

*Proof.* Recall from Algorithm 4 that  $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_L^2} = \eta\sigma_i^2$ . Then, the update rule is as follows:

$$\begin{aligned}\mathbf{x} &\leftarrow \mathbf{x} + \alpha_i s(\mathbf{x}, \sigma_i) + \sqrt{2\alpha_i} \mathbf{z} \\ &= \mathbf{x} + \eta\sigma_i^2 \left( \frac{H(\mathbf{x}) - \mathbf{x}}{\sigma_i^2} \right) + \sqrt{2\alpha_i} \mathbf{z} \\ &= (1 - \eta)\mathbf{x} + \eta H(\mathbf{x}) + \sqrt{2\eta}\sigma_i \mathbf{z}\end{aligned}$$

The same thing can be proven for Algorithm 5 in the very same way.  $\square$

---

## C.9 Inception Score (IS)

While the FID is improved by applying the EDS to image samples, the Inception Score is not. Convolutional neural networks suffer from texture bias (Geirhos et al., 2019). Since the IS is built upon convolution layers, this flaw is also strongly present in the metric. Designed to answer the question of how easy it is to recover the class of an image, it tends to bias towards within-class texture similarity (Barratt and Sharma, 2018).

Since we denoise the final image, we are evaluating the expected lower level of details across all classes. Therefore, the denoiser will confound the textures used by the IS to distinguish between classes, invariably worsening the score. Since the FID has already been shown to be more consistent with the level of noise than the IS (Heusel et al., 2017), and since ALS methods are particularly prone to inject

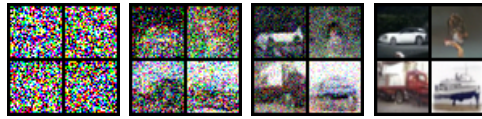
---

class-specific imperceptible noise, we would recommend against its use to compare within and between score matching models.

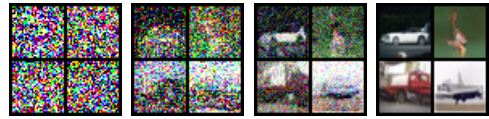


---

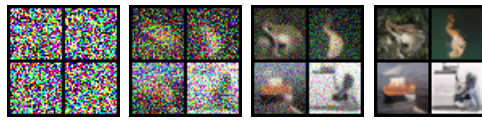
## C.10 Uncurated samples



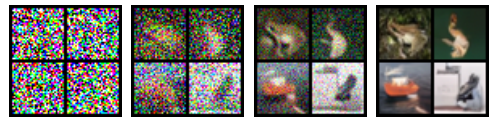
(a) CIFAR-10 Non-adversarial non-consistent  $n_\sigma = 1$



(b) CIFAR-10 Adversarial non-consistent  $n_\sigma = 1$



(c) CIFAR-10 Non-adversarial consistent  $n_\sigma = 1$



(d) CIFAR-10 Adversarial consistent  $n_\sigma = 1$

**Figure C.1** – Denoised sample evolving over time for different methods

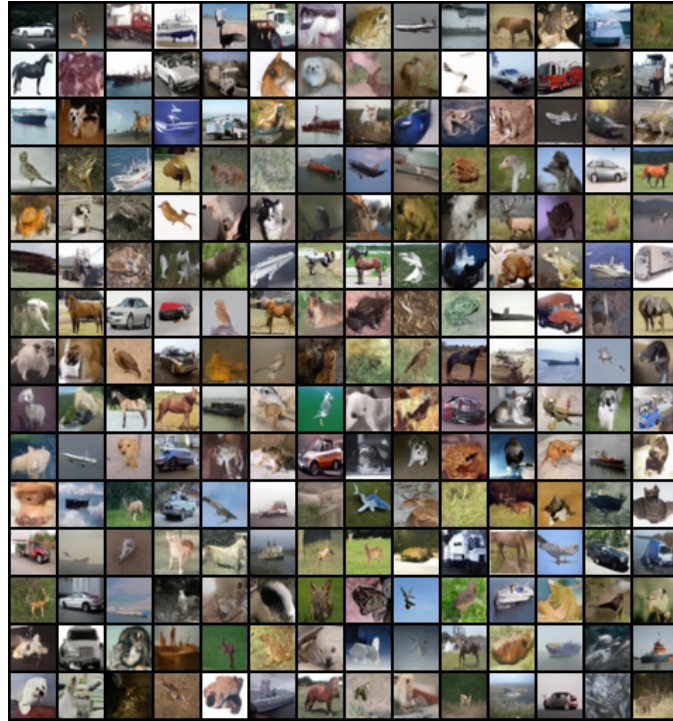


Figure C.2 – CIFAR-10 Non-adversarial non-consistent  $n_\sigma = 1$

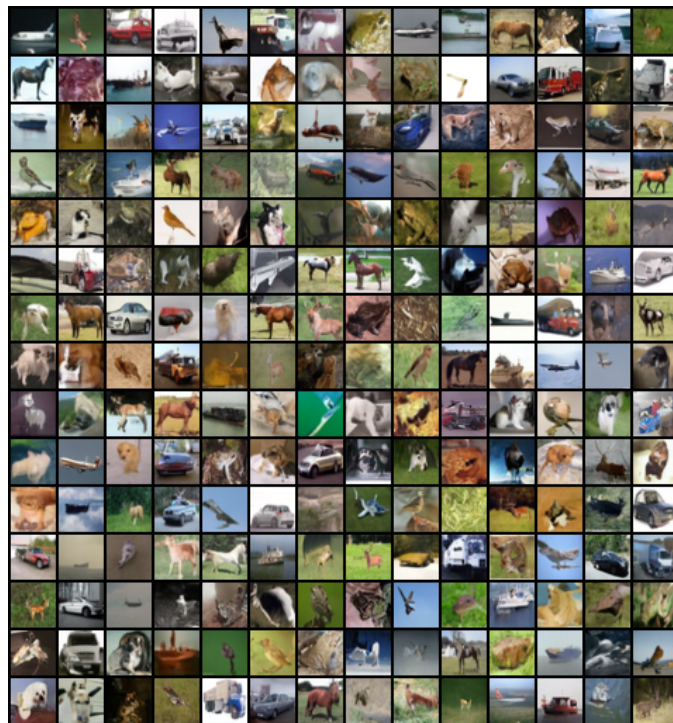


Figure C.3 – CIFAR-10 Adversarial non-consistent  $n_\sigma = 1$

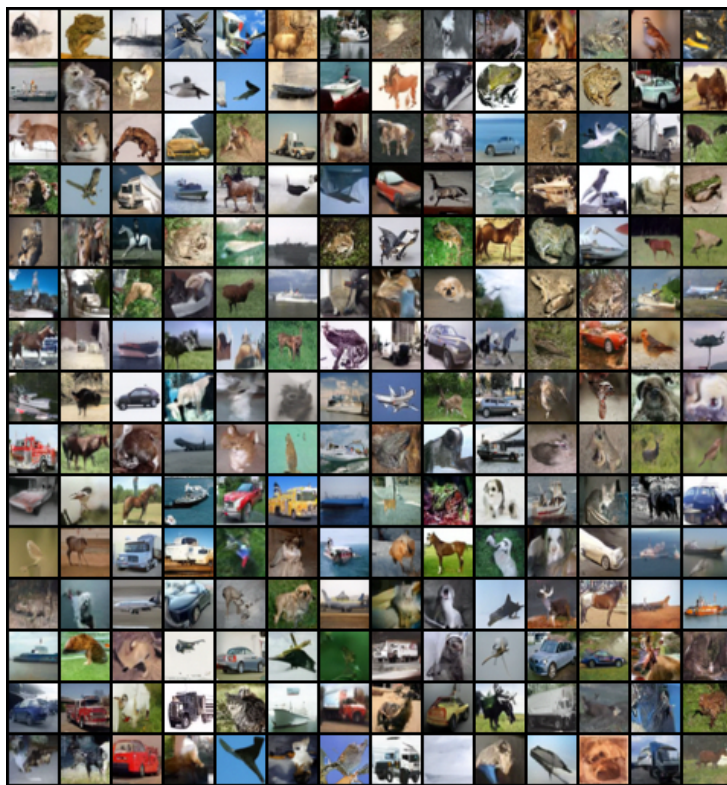
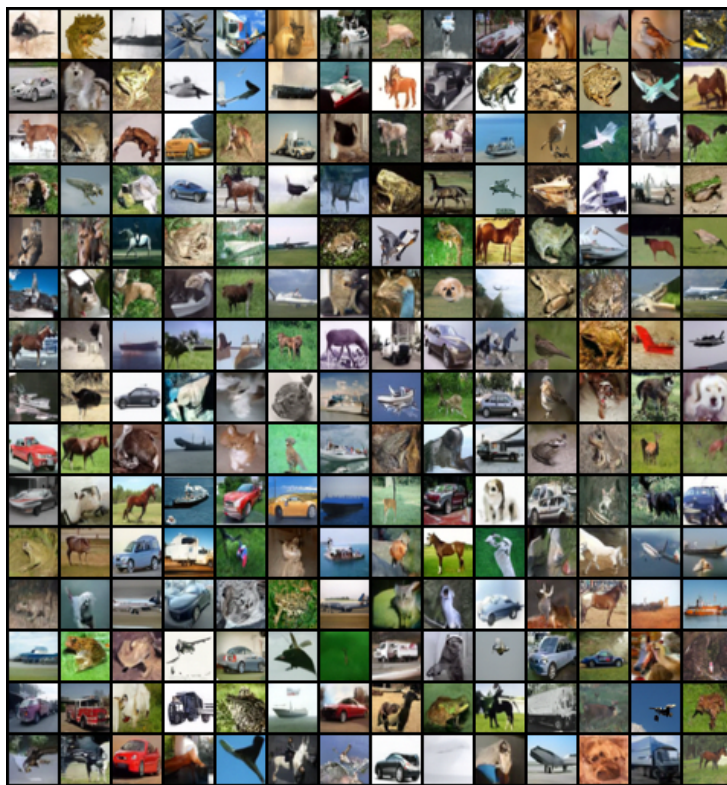
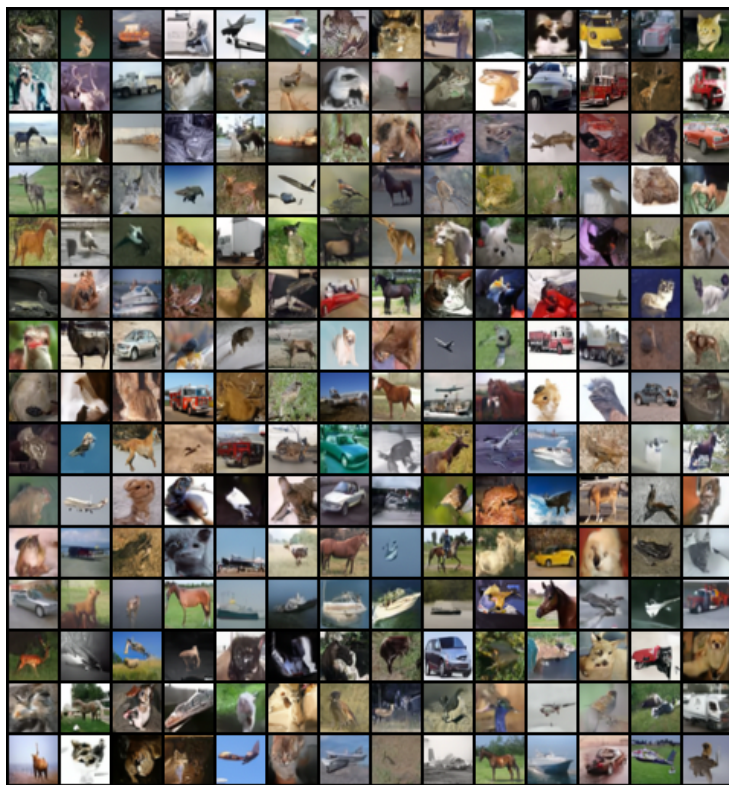


Figure C.4 – CIFAR-10 Non-adversarial non-consistent  $n_\sigma = 5$



**Figure C.5** – CIFAR-10 Adversarial non-consistent  $n_\sigma = 5$



**Figure C.6** – CIFAR-10 Non-adversarial consistent  $n_\sigma = 1$

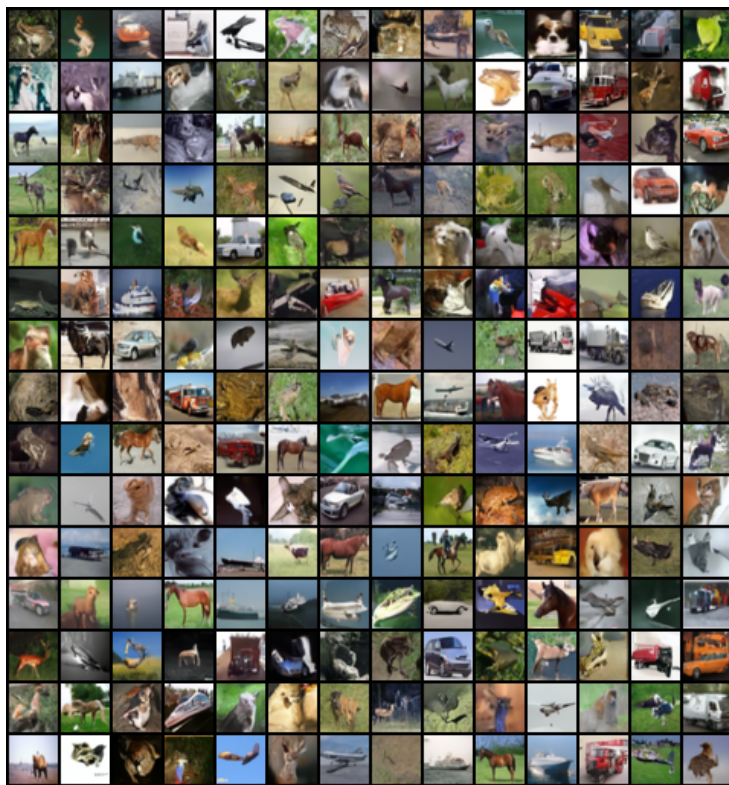
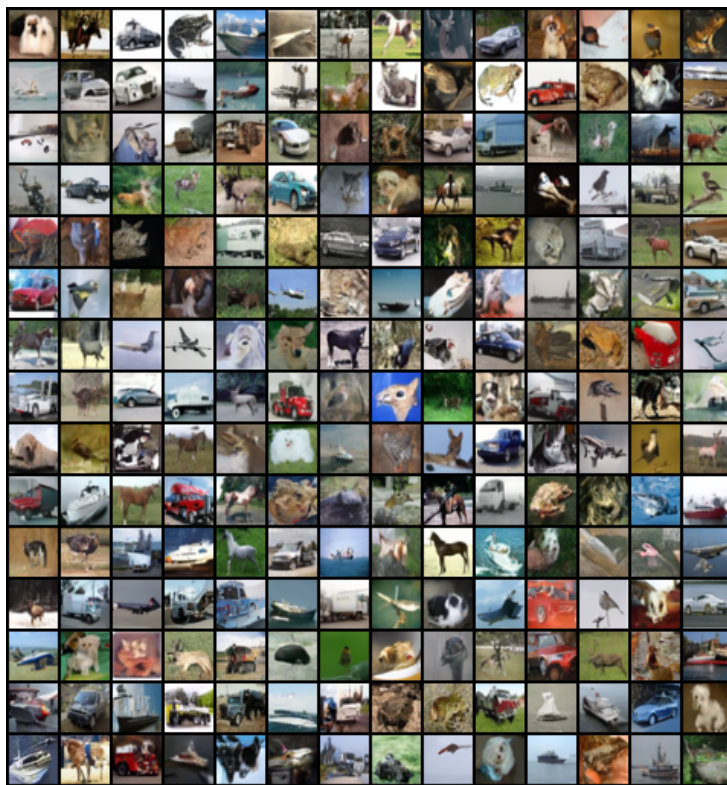


Figure C.7 – CIFAR-10 Adversarial consistent  $n_\sigma = 1$



**Figure C.8** – CIFAR-10 Non-adversarial consistent  $n_\sigma = 5$

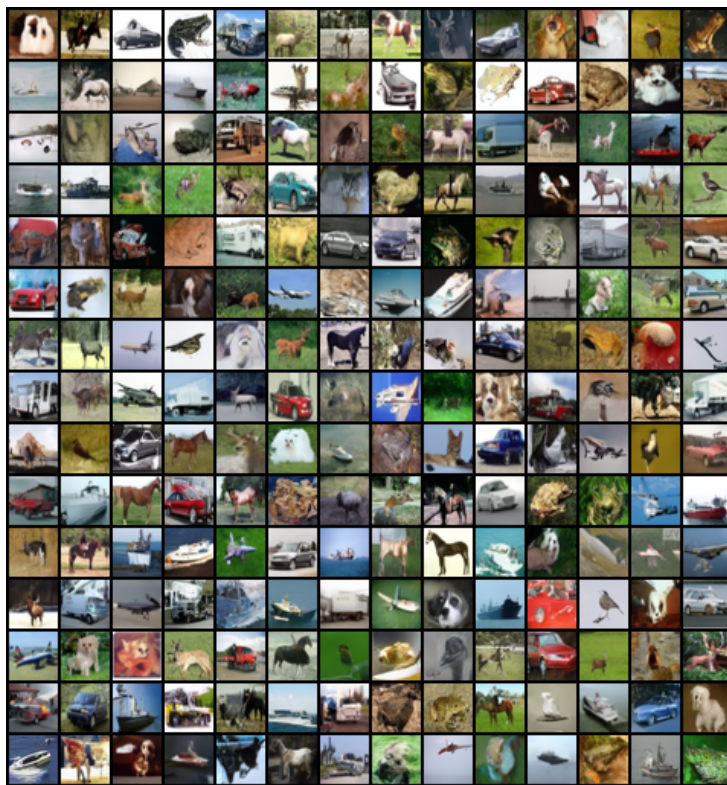


Figure C.9 – CIFAR-10 Adversarial consistent  $n_\sigma = 5$





**Figure C.10** – LSUN-Churches Non-adversarial non-consistent  $n_\sigma = 1$



Figure C.11 – LSUN-Churches Adversarial non-consistent  $n_\sigma = 1$



**Figure C.12** – LSUN-Churches Non-adversarial non-consistent  $n_\sigma = 5$



**Figure C.13** – LSUN-Churches Adversarial non-consistent  $n_\sigma = 5$



Figure C.14 – LSUN-Churches Non-adversarial consistent  $n_\sigma = 1$



**Figure C.15** – LSUN-Churches Adversarial consistent  $n_\sigma = 1$

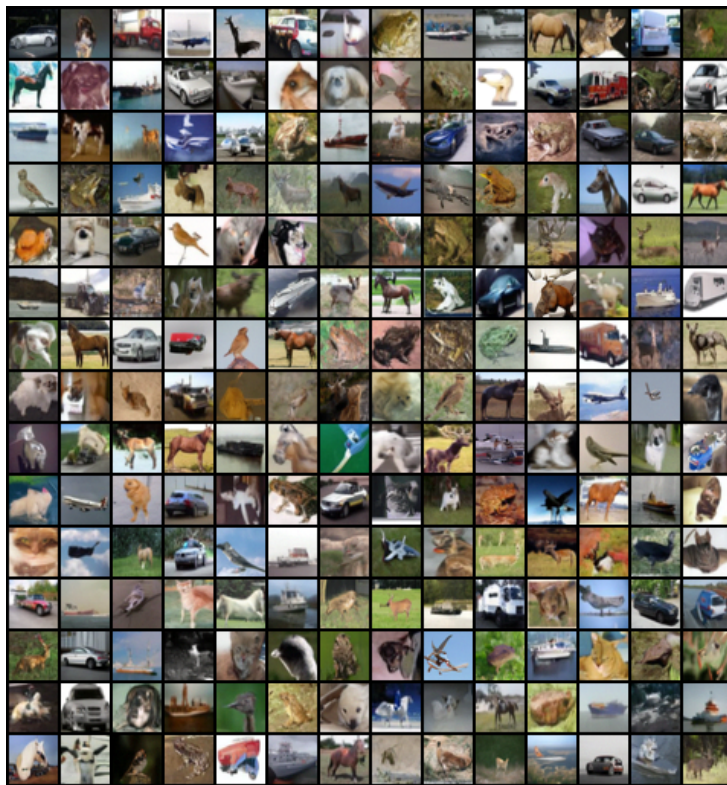


Figure C.16 – LSUN-Churches Non-adversarial consistent  $n_\sigma = 5$

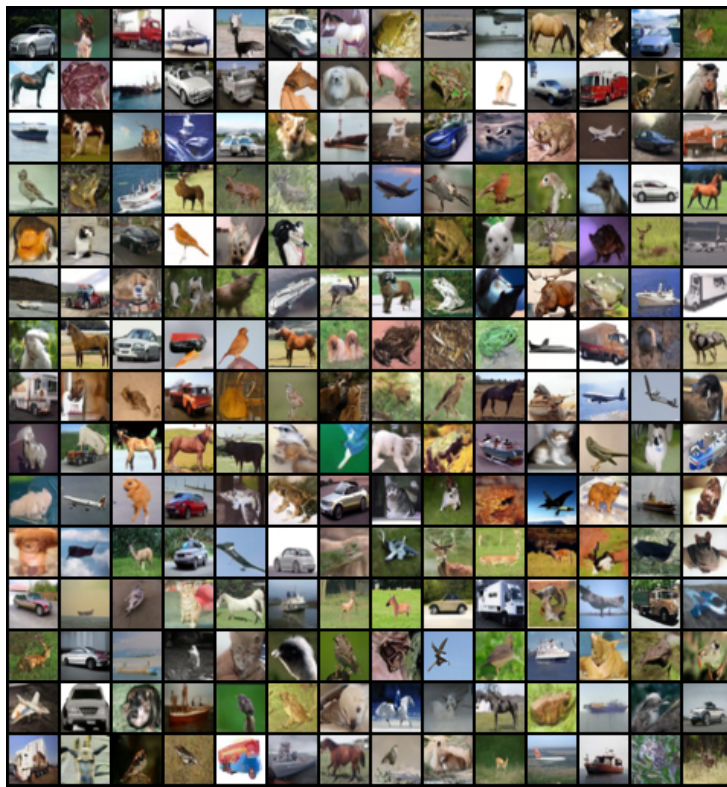


**Figure C.17** – LSUN-Churches Adversarial consistent  $n_\sigma = 5$

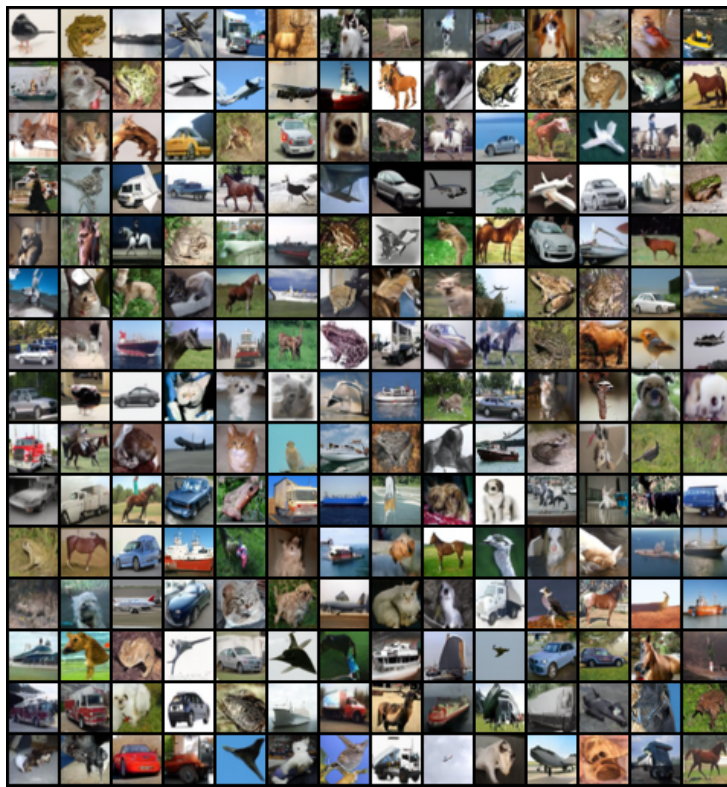




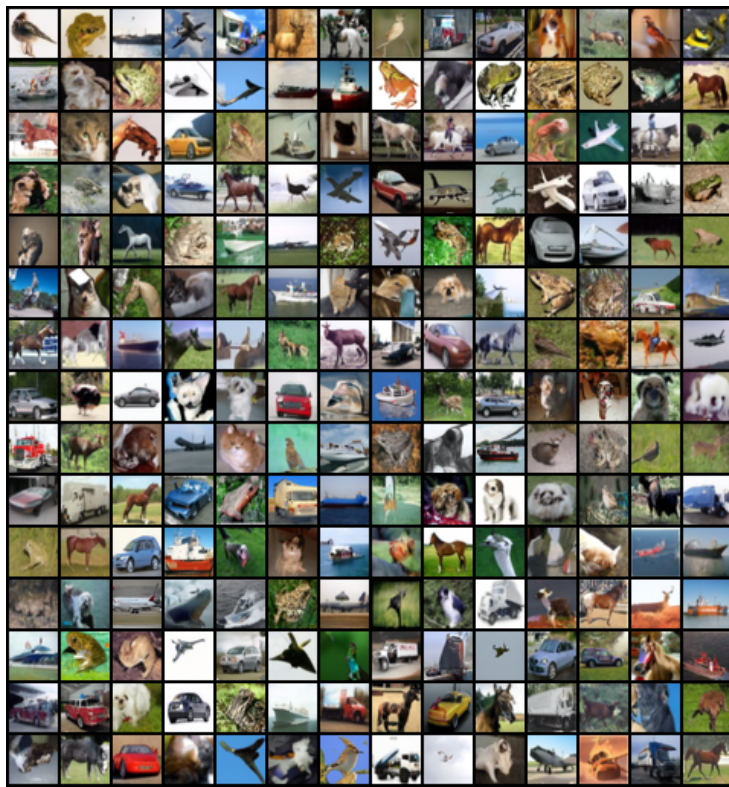
**Figure C.18** – Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial non-consistent  $n_\sigma = 1$



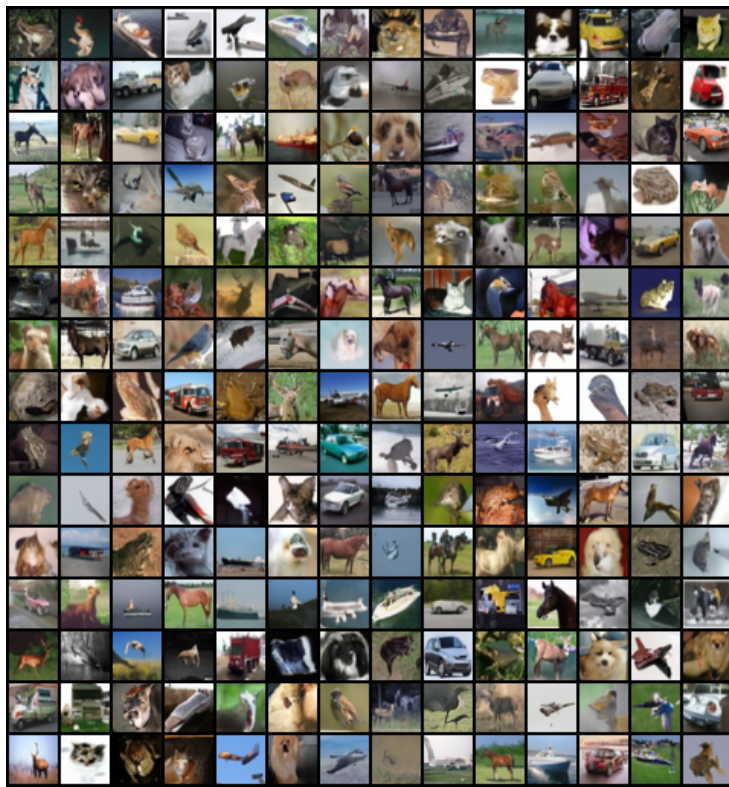
**Figure C.19** – Ho et al. (2020) network architecture with CIFAR-10 Adversarial non-consistent  $n_\sigma = 1$



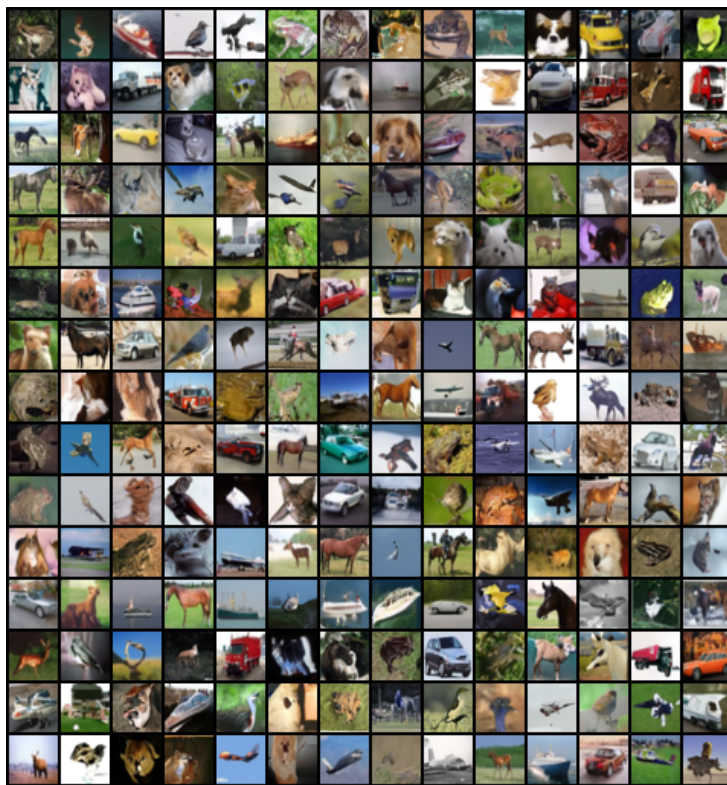
**Figure C.20** – Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial non-consistent  $n_\sigma = 5$



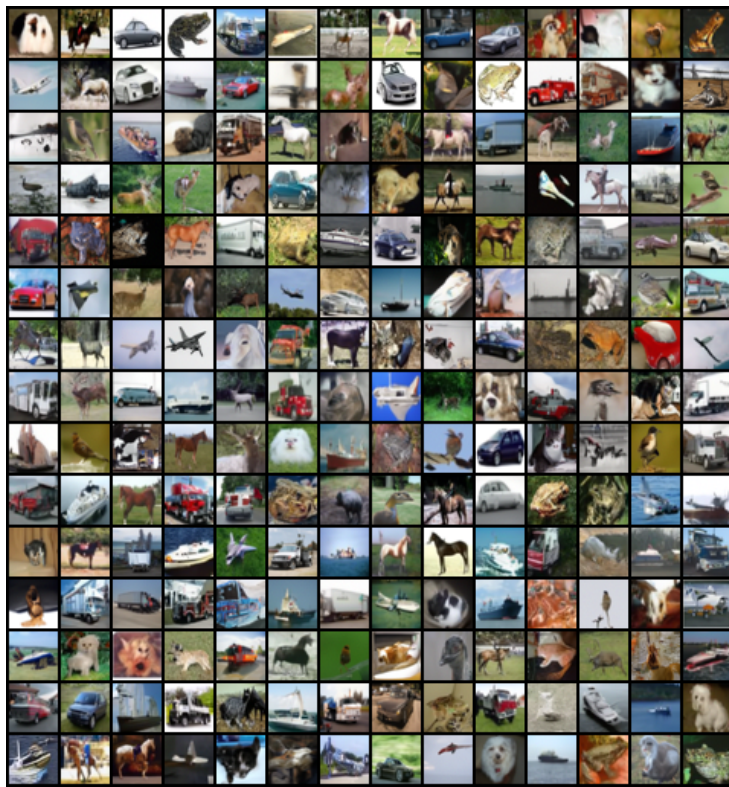
**Figure C.21** – Ho et al. (2020) network architecture with CIFAR-10 Adversarial non-consistent  $n_\sigma = 5$



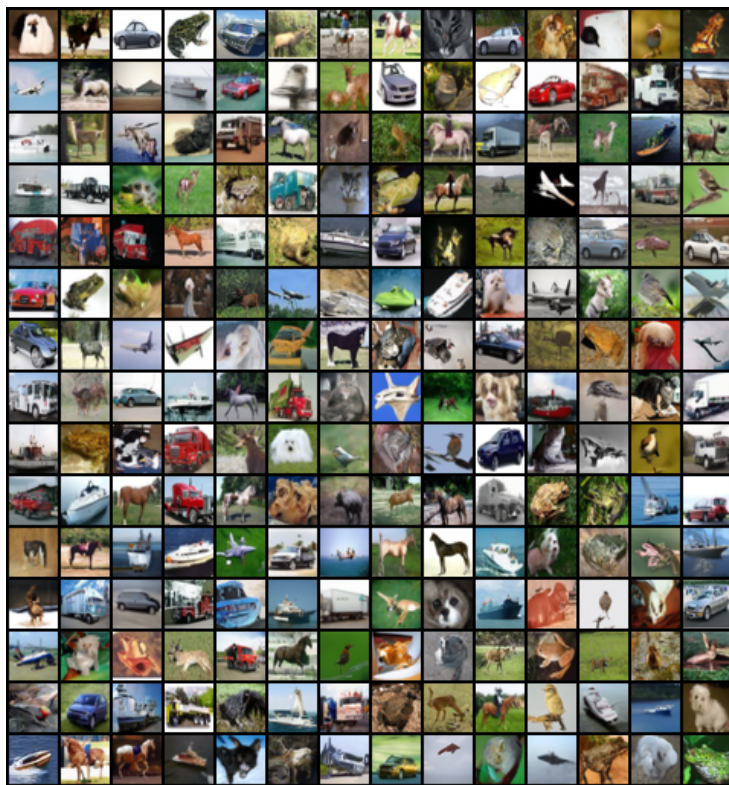
**Figure C.22** – Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial consistent  $n_\sigma = 1$



**Figure C.23** – Ho et al. (2020) network architecture with CIFAR-10 Adversarial consistent  $n_\sigma = 1$



**Figure C.24** – Ho et al. (2020) network architecture with CIFAR-10 Non-adversarial consistent  $n_\sigma = 5$



**Figure C.25** – Ho et al. (2020) network architecture with CIFAR-10 Adversarial consistent  $n_\sigma = 5$



## D

# Gotta Go Fast When Generating Data with Score-Based Models

---

## D.1 DifferentialEquations.jl

**Table D.1** – Short experiments with various SDE solvers from *DifferentialEquations.jl* on the VP model with a small mini-batch.

Method	Strong-Order	Adaptive	Speed
Euler-Maruyama (EM)	0.5	No	Baseline speed
SOSRA (Rößler, 2010)	1.5	Yes	<b>5.92 times slower</b>
SRA3 (Rößler, 2010)	1.5	Yes	<b>6.93 times slower</b>
Lamba EM (default) (Lamba, 2003)	0.5	Yes	Did not converge
Lamba EM (atol=1e-3) (Lamba, 2003)	0.5	Yes	<b>2 times faster</b>
Lamba EM (atol=1e-3, rtol=1e-3) (Lamba, 2003)	0.5	Yes	<b>1.27 times faster</b>
Euler-Heun	0.5	No	<b>1.86 times slower</b>
Lamba Euler-Heun (Lamba, 2003)	0.5	Yes	<b>1.75 times faster</b>
SOSRI (Rößler, 2010)	1.5	Yes	<b>8.57 times slower</b>
RKMil (at various tolerances) (Kloeden and Platen, 1992)	1.0	Yes	Did not converge
ImplicitRKMil (Kloeden and Platen, 1992)	1.0	Yes	Did not converge
ISSEM	0.5	Yes	Did not converge

Here, we report the preliminary experiments we ran with the *DifferentialEquations.jl* Julia package (Rackauckas and Nie, 2017b) before devising our own SDE solver. As can be seen, most methods either did not converge (with warnings of “instability detected”) or converged, but were much slower than Euler-Maruyama. The only promising method was Lamba’s method (Lamba, 2003). Note that an algorithm has strong-order  $p$  when the local error from  $t$  to  $t + h$  is  $\mathcal{O}(h^{p+1})$ .

---

## D.2 Effects of modifying Algorithm 1

As can be seen, most chosen settings lead to better results. However,  $r$  seems to have little impact on the FID. Still, using  $r \in [0.8, 0.9]$  lead to a little bit less score function evaluations and sometimes lead to lower FID.

**Table D.2** – Effect of different settings on the [Inception score (IS) / Fréchet Inception Distance (FID) / Number of score Function Evaluations (NFE)] from 10k samples (with mini-batches of 1k samples) with the VP - CIFAR10 model.

Change(s) in Algorithm 1	IS	FID	NFE
No change [ $q = 2, r = 0.9, \delta(\mathbf{x}', \mathbf{x}'_{prev})$ ]	9.38	4.70	3972
Small modifications			
$\delta(\mathbf{x}')$	9.26	4.69	4166
No Extrapolation (thus, using Euler–Maruyama)	9.58	11.73	3978
$q = \infty$	9.48	4.90	14462
$r = .5$	9.41	4.69	4104
$r = .8$	9.36	4.68	3938
$r = 1$	9.41	4.69	4048
Variations of <a href="#">Lamba (2003)</a> Algorithm			
$r = 0.5$ , Lamba integration	7.80	52.98	1468
$r = 0.5$ , Lamba integration, Extrapolation	7.32	64.65	1438
$r = 0.5$ , Lamba integration, $q = \infty$	9.28	21.09	2360
$r = 0.5$ , Lamba integration, $q = \infty, \theta = 0.8$	9.21	18.82	2346

We notice that using  $q = \infty$  and  $\delta(\mathbf{x}')$  lead to higher NFE as we expect. However, they also generally lead to higher FID, thus lower quality/diversity, which is not expected! We hypothesized that this might be due to the large number of steps taken when using  $q = \infty$  and  $\delta(\mathbf{x}')$ . To test this, we trained the VE and VP models with Euler-Maruyama with 10k steps instead of 1k steps and we indeed obtained higher FIDs. This means that taking too many steps leads to worse performance in score-based models.

Worse quality from taking more steps should typically not happen as more steps should mean a more precise trajectory. We hypothesize this to be caused by the difference between using the actual score function instead of using the pre-trained score-network; given the errors in the score network, it may be that taking too many steps leads to some deviations from the right solution. Alternatively, this could also be due to the metric, as no existing quality/diversity metrics for generative models is truly perfect; but we believe that this hypothesis is less plausible than the increasing errors from using the score-network.

**Table D.3** – Effect of different settings on the [Inception score (IS) / Fréchet Inception Distance (FID) / Number of score Function Evaluations (NFE)] from 10k samples (with mini-batches of 1k samples) with the VE - CIFAR10 model.

Change(s) in Algorithm 1	IS	FID	NFE
No change [ $q = 2, r = 0.9, \delta(\mathbf{x}', \mathbf{x}'_{prev})$ ]	9.39	4.89	8856
Small modifications			
$\delta(\mathbf{x}')$	9.39	4.99	17514
No Extrapolation (thus, using Euler–Maruyama)	9.58	6.57	8802
$q = \infty$	9.41	5.03	39500
$r = 0.5$	9.47	4.87	9594
$r = 0.8$	9.45	4.84	8952
$r = 1$	9.43	4.93	8784
Variations of Lamba (2003) Algorithm			
$r = 0.5$ , Lamba integration	9.08	18.28	2492
$r = 0.5$ , Lamba integration, Extrapolation	3.70	169.78	2252
$r = 0.5$ , Lamba integration, $q = \infty$	9.42	6.80	5886
$r = 0.5$ , Lamba integration, $q = \infty, \theta = 0.8$	9.35	6.20	2970

---

### **D.3 Dynamic step size algorithm for solving any type of SDE (rather than just Reverse Diffusion Processes)**

Assume, we have a Diffusion Process of the form:

$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(\mathbf{x}, t)d\mathbf{w}. \quad (\text{D.1})$$

The algorithm to solve it is represented in Algorithm 9. The differences are:

- it is in forward-time
- the range of time must be given
- The diffusion can depend on  $\mathbf{x}$ , which leads to a slightly more complicated formulation that depends on some random number  $s = \pm 1$  (Roberts, 2012).
- we retain the full trajectory instead of only the ending
- we retain the noise after a rejection to ensure that there is no bias in the rejections

---

**Algorithm 9** Dynamic step size extrapolation for solving arbitrary (forward-time) Diffusion Processes

---

**Require:**  $s_\theta, t_{begin}, t_{end}, \epsilon_{rel}, \epsilon_{abs}, h_{init} = 0.01, r = 0.9, \theta = 0.9$

$t \leftarrow t_{begin}$

$h \leftarrow h_{init}$

Initialize  $\mathbf{x}(t)$

$\mathbf{x}'_{prev} \leftarrow \mathbf{x}$

Draw  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

**while**  $t < t_{end}$  **do**

**if** Stratonovich SDE or  $g(\mathbf{x}, t) = g(\mathbf{x})$  **then**

$s \leftarrow 0$

**else**

$\triangleright$  Itô diffusion

    Draw  $s \sim \text{Uniform}(\{-1, 1\})$

$\mathbf{x}' \leftarrow \mathbf{x}(t) + hf(\mathbf{x}(t), t) + \sqrt{h}g(\mathbf{x}(t), t)(\mathbf{z} - s)$

$\triangleright$  Euler-Maruyama

$\tilde{\mathbf{x}} \leftarrow \mathbf{x}(t) + hf(\mathbf{x}', t + h) + \sqrt{h}g(\mathbf{x}', t + h)(\mathbf{z} + s)$

$\mathbf{x}'' \leftarrow \frac{1}{2}(\mathbf{x}' + \tilde{\mathbf{x}})$

$\triangleright$  Improved Euler (SDE version)

$\boldsymbol{\delta} \leftarrow \max(\epsilon_{abs}, \epsilon_{rel} \max(|\mathbf{x}'|, |\mathbf{x}'_{prev}|))$

$\triangleright$  Element-wise operations

$E_2 \leftarrow \frac{1}{\sqrt{n}} \|(\mathbf{x}' - \mathbf{x}'') / \boldsymbol{\delta}\|_2$

**if**  $E_2 \leq 1$  **then**

$\triangleright$  Accept

$t \leftarrow t + h$

$\mathbf{x}(t) \leftarrow \mathbf{x}''$

$\triangleright$  Extrapolation

$\mathbf{x}'_{prev} \leftarrow \mathbf{x}'$

      Draw  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$h \leftarrow \min(t, \theta h E_2^{-r})$

$\triangleright$  Dynamic step size update

**return**  $\mathbf{x}$

---

---

## D.4 Implementation Details

We started from the original code by Song et al. (2021) but changed a few settings concerning the SDE solving. This creates some very minor difference between their reported results and ours. For the VP and VP-deep models, we obtained 2.55 and 2.49 instead of the original 2.55 and 2.41 for the baseline method (EM). For the VE and VE-deep models, we obtained 2.40 and 2.21 instead of the original 2.38 and 2.20 for the baseline method (Reverse-Diffusion with Langevin).

As done in Song et al. (2021), we used the optimal signal-to-noise ratio of 0.01 for the Langevin corrector.

When solving the SDE, time followed the sequence  $t_0 = 1$ ,  $t_i = t_{i-1} - \frac{1-\epsilon}{N}$ , where  $N = 1000$  for CIFAR-10,  $N = 2000$  for LSUN,  $\epsilon = 1e - 3$  for VP models, and  $\epsilon = 1e - 5$  for VE models.

Meanwhile, the actual step size  $h$  used in the code for Euler-Maruyama (EM) was equal to  $\frac{1}{N}$ . Thus, there was a negligible difference between the step size used in the algorithm ( $h = \frac{1}{N}$ ) and the actual step size implied by  $t$  ( $h = \frac{1-\epsilon}{N}$ ). Note that this has little to no impact.

The bigger issue is at the last predictor step was going from  $t = \epsilon$  to  $t = \epsilon - \frac{1}{N} < 0$ . Thus,  $t$  was made negative. Furthermore the sample was denoised at  $t < 0$  while assuming  $t = \epsilon$ . There are two ways to fix this issue: 1) take only a step from  $t = \epsilon$  to  $t = 0$  and do not denoise (since you cannot denoise with the incorrect  $t$  or with  $t = 0$ ), or 2) stop at  $t = \epsilon$  and then denoise. Since denoising is very helpful, we took approach 2; however, both approaches are sensible.

Finally, denoising was not implemented correctly before. Denoising was implemented as one predictor step (Reverse-Diffusion or EM) without adding noise. This corresponds to:

$$\mathbf{x} \leftarrow \mathbf{x} - h [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})].$$

---

At the last iteration, this incorrect denoising would be:

$$\begin{aligned}
\mathbf{x} &\leftarrow \mathbf{x} + \frac{d[\sigma^2(t)]}{dt} \frac{1}{N} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\
&= \mathbf{x} + \frac{\sigma_{min}}{N} \sqrt{2 \log \left( \frac{\sigma_{max}}{\sigma_{min}} \right)} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\
&\approx \mathbf{x}
\end{aligned}$$

for VE and

$$\begin{aligned}
\mathbf{x} &\leftarrow \mathbf{x} + \frac{\sqrt{\beta_{min}}}{N} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\
&\approx \mathbf{x}
\end{aligned}$$

for VP.

Meanwhile, the correct way to denoise based on Tweedie formula (Efron, 2011) is:

$$\mathbf{x} \leftarrow \mathbf{x} + \text{Var}[\mathbf{x}(t)|\mathbf{x}(0)] \nabla_{\mathbf{x}} \log p_t(\mathbf{x}),$$

where  $\text{Var}[\mathbf{x}(t)|\mathbf{x}(0)]$  is the variance of the transition kernel:  $\text{Var}[\mathbf{x}(t)|\mathbf{x}(0)] = \sigma_{min} = 0.01$  for VE and  $\text{Var}[\mathbf{x}(t)|\mathbf{x}(0)] = 1$ . This means that the correct Tweedie formula corresponds to

$$\begin{aligned}
\mathbf{x} &\leftarrow \mathbf{x} + 0.01^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\
&\approx \mathbf{x}
\end{aligned}$$

for VE and

$$\mathbf{x} \leftarrow \mathbf{x} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

for VP.

As can be seen, denoising has a very small impact on VE so the difference between the correct and incorrect denoising is minor. Meanwhile, for VP the incorrect denoising lead to a tiny change, while the correct denoising lead to a large change. In practice, we observe that changing the denoising method to the correct one does not significantly affect the FID with VE, but lowers down the FID significantly with VP.

---

## D.5 Inception Score on CIFAR-10

Table D.4 – Inception Score on CIFAR-10 (32x32) from 50K samples

Method	VP	VP-deep	VE	VE-deep
Reverse-Diffusion & Langevin	9.94	9.85	9.86	9.83
Euler-Maruyama	9.71	9.73	9.49	9.31
Ours ( $\epsilon_{rel} = 0.01$ )	9.46	9.54	9.50	9.48
Ours ( $\epsilon_{rel} = 0.02$ )	9.51	9.48	9.57	9.50
Ours ( $\epsilon_{rel} = 0.05$ )	9.50	9.61	9.64	9.63
Ours ( $\epsilon_{rel} = 0.10$ )	9.69	9.64	9.87	9.75
Probability Flow (ODE)	9.37	9.33	9.17	9.32

---

## D.6 Stability and Bias of the Numerical Scheme

The following constructions rely on the underlying assumption of the stochastic dynamics being driven by a Wiener process. More so, we also assume that the Brownian motion is time symmetrical. Both assumptions are consistent and widely used in the literature; for example, see (Gardiner, 2009) (Arnold, 1974).

The method described in Algorithm 1 gives us a significant speedup in terms of computing time and actions. Albeit the speed up comes from a piece-wise step in the algorithm combining the traditional Euler Maruyama (EM) with a form of adaptive step size predictor-corrector. Here we show that both the stability and the convergence of the EM scheme are conserved by introducing the extra adaptive stepsize of our new scheme. As a first step, we define the stability and bias in a Stochastic Differential Equation (SDE) numerical solution.

We denote  $\Re(\lambda)$  as the real value of a complex-valued  $\lambda$ .

The linear test SDE is defined in the following way:

$$d\mathbf{x}_t = \lambda \mathbf{x}_t dt + \sigma d\mathbf{w}_t \tag{D.2}$$



---

with its numerical counterpart

$$\mathbf{y}_{n+1} = \Re(h\lambda) \mathbf{y}_n + \mathbf{z}_n,$$

where the  $\mathbf{z}_n$  are random variables that do not depend on  $\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n$  or  $\lambda$  and the EM scheme is

$$\mathbf{y}_{n+1} = (1 + h\lambda) \mathbf{y}_n + \mathbf{z}_n.$$

A numerical scheme is asymptotically unbiased with step size  $h > 0$  if, for a given linear SDE (D.2) driven by a two-sided Wiener process, the distribution of the numerical solution  $\mathbf{y}_n$  converges as  $n \rightarrow \infty$  to the normal distribution with zero mean and variance  $\frac{\sigma^2}{2|\lambda|}$  (Artemiev and Averina, 2011). This stems from the fact that a solution of a linear SDE (D.2) is a Gaussian process whenever the initial condition is Gaussian (or deterministic); thus, there are only two moments that control the bias in the algorithm:

$$\lim_{n \rightarrow \infty} \mathbb{E}[\mathbf{y}_n] = 0, \quad \lim_{n \rightarrow \infty} \mathbb{E}[\mathbf{y}_n^2] = -\frac{\sigma^2}{2|\lambda|}.$$

A numerical scheme with step size  $h$  is numerically stable in mean if the numerical solution  $\mathbf{y}_n^{(h)}$  applied to a linear SDE satisfies

$$\lim_{n \rightarrow \infty} \mathbb{E}[\mathbf{y}_n] = 0,$$

and is stable in mean square (Saito and Mitsui, 1996) if we have that

$$\lim_{h \rightarrow 0} \left( \lim_{n \rightarrow \infty} \mathbb{E}[|\mathbf{y}_n|^2] \right) = \frac{\sigma^2}{2\Re(\lambda)}.$$

In what follows, we will trace the criteria for bias through our algorithm and show that it remains unbiased. By construction, the first EM step remains unbiased, while for the RDP, we write down the time reverse Wiener process as

$$\tilde{\mathbf{y}}_{n+1} = (1 + \lambda h) \tilde{\mathbf{y}}_n + \tilde{\mathbf{z}}_n$$

---

in the reverse time steps  $h$  i.e.,  $t - nh, t - 2nh,$

$$\begin{aligned}
\mathbb{E} [\tilde{\mathbf{y}}_{n+1}] &= (1 + \lambda(t - h)) \mathbb{E} [\tilde{\mathbf{y}}_n] \\
&= (1 + \lambda(t - h)) \mathbb{E} [(1 + \lambda(t - h)) \tilde{\mathbf{y}}_{n-1}] \\
&\quad \vdots \\
&= (1 + \lambda(t - h))^{n+1} \mathbb{E} [\tilde{\mathbf{y}}_0] \\
&= (1 + \lambda(t - h))^{n+1} \mathbb{E} [\mathbf{y}_0].
\end{aligned}$$

Thus, if

$$|1 + \lambda(t - h)| < 1,$$

then

$$\lim_{n \rightarrow \infty} \mathbb{E} [\mathbf{y}_n^{(h)}] = 0.$$

In Algorithm 1, we are performing consecutive steps forward and backwards in time so  $t = 2h$  such that

$$|1 + \lambda h| < 1.$$

Thus, the scheme is both numerically stable and unbiased with respect to the mean.

Next, we focus on the numerical solution in mean square:

$$\begin{aligned}
\mathbb{E} [|\tilde{\mathbf{y}}_{n+1}|^2] &= |1 + \lambda(t - h)|^2 \mathbb{E} [|\tilde{\mathbf{y}}_n|^2] + \sigma^2 h \\
&= |1 + \lambda(t - h)|^2 \left\{ |1 + \lambda(t - h)|^2 \mathbb{E} [|\tilde{\mathbf{y}}_{n-1}|^2] + \sigma^2 h \right\} + \sigma^2 h \\
&\quad \vdots \\
&= |1 + \lambda(t - h)|^{2(n+1)} \mathbb{E} [|\mathbf{y}_0|] + \frac{|1 + \lambda(t - h)|^{2(n+1)} - 1}{2\Re(\lambda) + |\lambda|^2(t - h)} \sigma^2.
\end{aligned}$$

Under the same assumption of consecutive steps, we have that

$$\mathbb{E} [|\tilde{\mathbf{y}}_{n+1}|^2] = |1 + \lambda h|^{2(n+1)} \mathbb{E} [|\mathbf{y}_0|] + \frac{|1 + \lambda h|^{2(n+1)} - 1}{2\Re(\lambda) + |\lambda|^2 h} \sigma^2,$$

$$\lim_{n \rightarrow \infty} \mathbb{E} [|\tilde{\mathbf{y}}_{n+1}|^2] = -\frac{\sigma^2}{2\Re(\lambda) + |\lambda|^2 h},$$

$$\lim_{h \rightarrow 0} \left( \lim_{n \rightarrow \infty} \mathbb{E} [|\tilde{\mathbf{y}}_{n+1}|^2] \right) = -\frac{\sigma^2}{2\Re(\lambda)}.$$

---

Assuming the imaginary part of  $\lambda$  is null, we have that

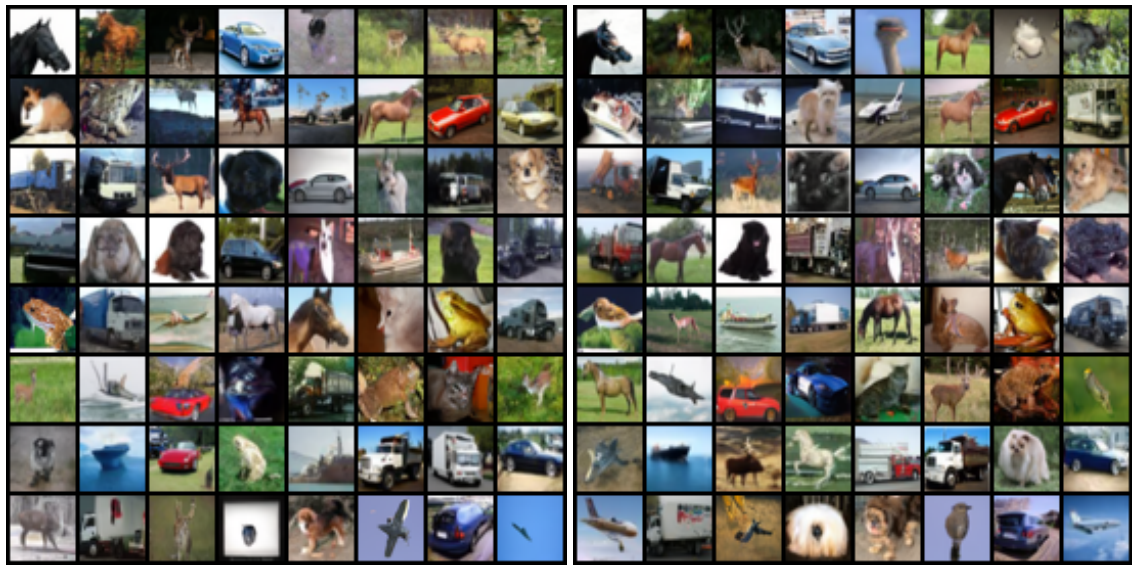
$$\lim_{h \rightarrow 0} \left( \lim_{n \rightarrow \infty} \mathbb{E} \left[ |\tilde{\mathbf{y}}_{n+1}|^2 \right] \right) = -\frac{\sigma^2}{2|\lambda|}.$$

Thus, the numerical scheme is stable and unbiased in the mean square.

Following the two steps for computation of  $\mathbf{x}'$  and  $\tilde{\mathbf{x}}$ , the step size decreases and does not change size; thus, all the above statements hold, and the entire algorithm is stable and unbiased with respect to both the mean and square mean.

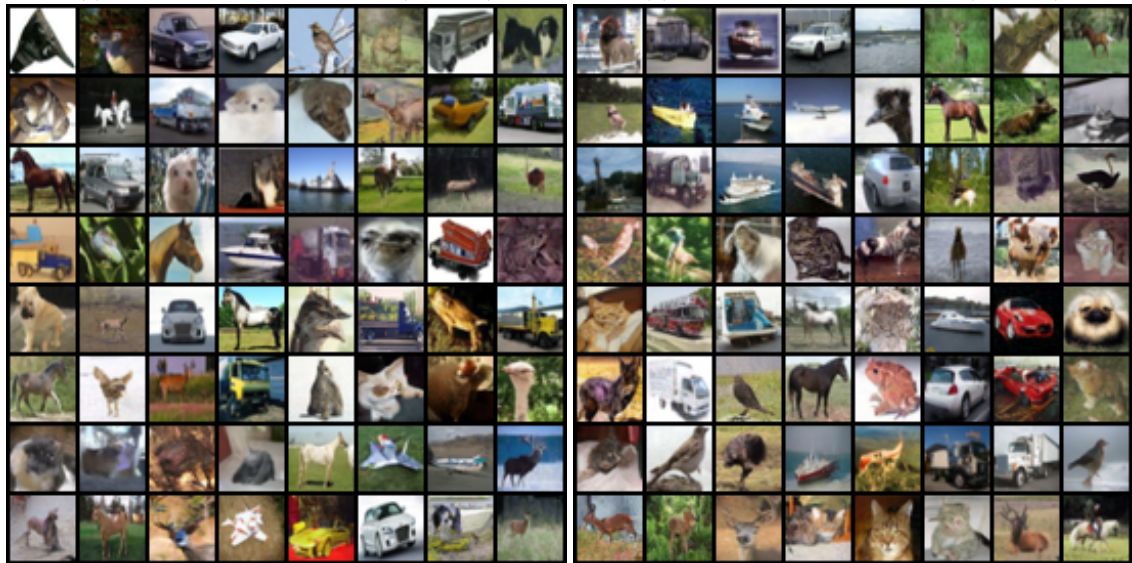
---

## D.7 Samples



(a) Dynamic-step Extrapolation ( $\epsilon = 0.01$ )

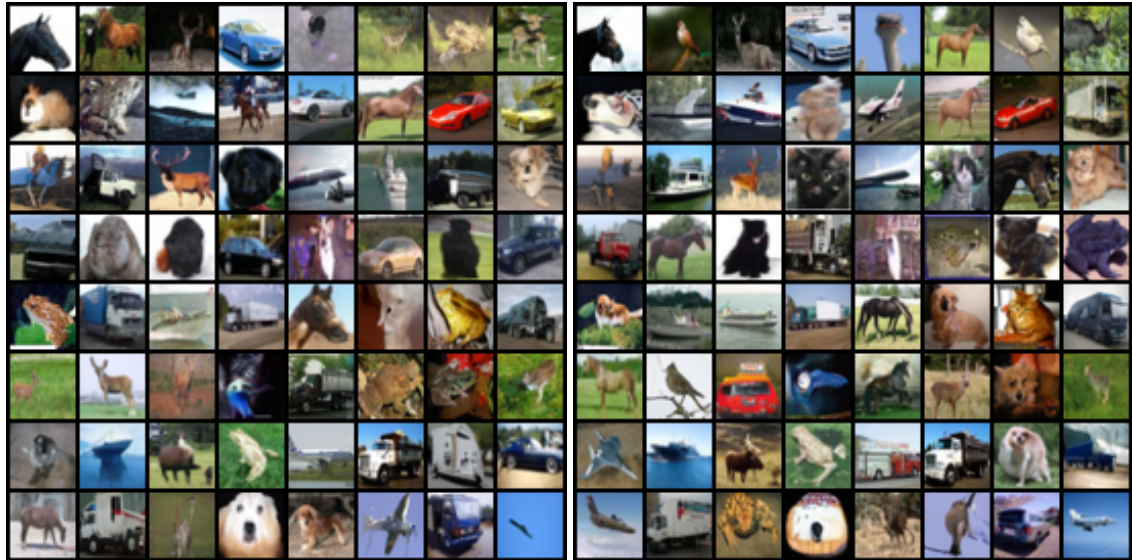
(b) Dynamic-step Extrapolation ( $\epsilon = 0.02$ )



(c) Dynamic-step Extrapolation ( $\epsilon = 0.05$ )

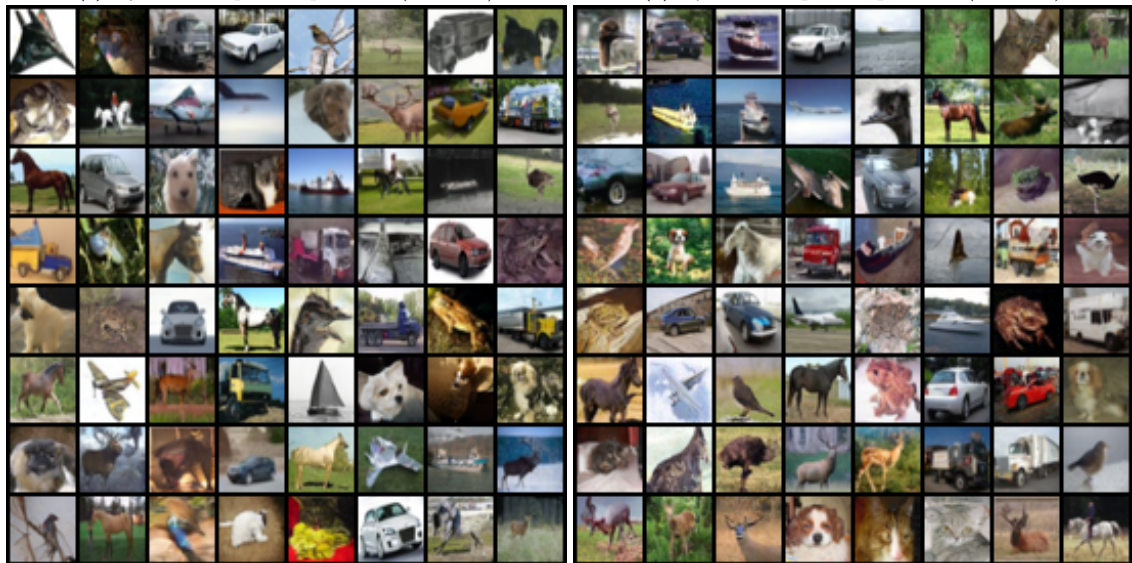
(d) Dynamic-step Extrapolation ( $\epsilon = 0.10$ )

**Figure D.1 – VP - CIFAR10**



(a) Dynamic-step Extrapolation ( $\epsilon = 0.01$ )

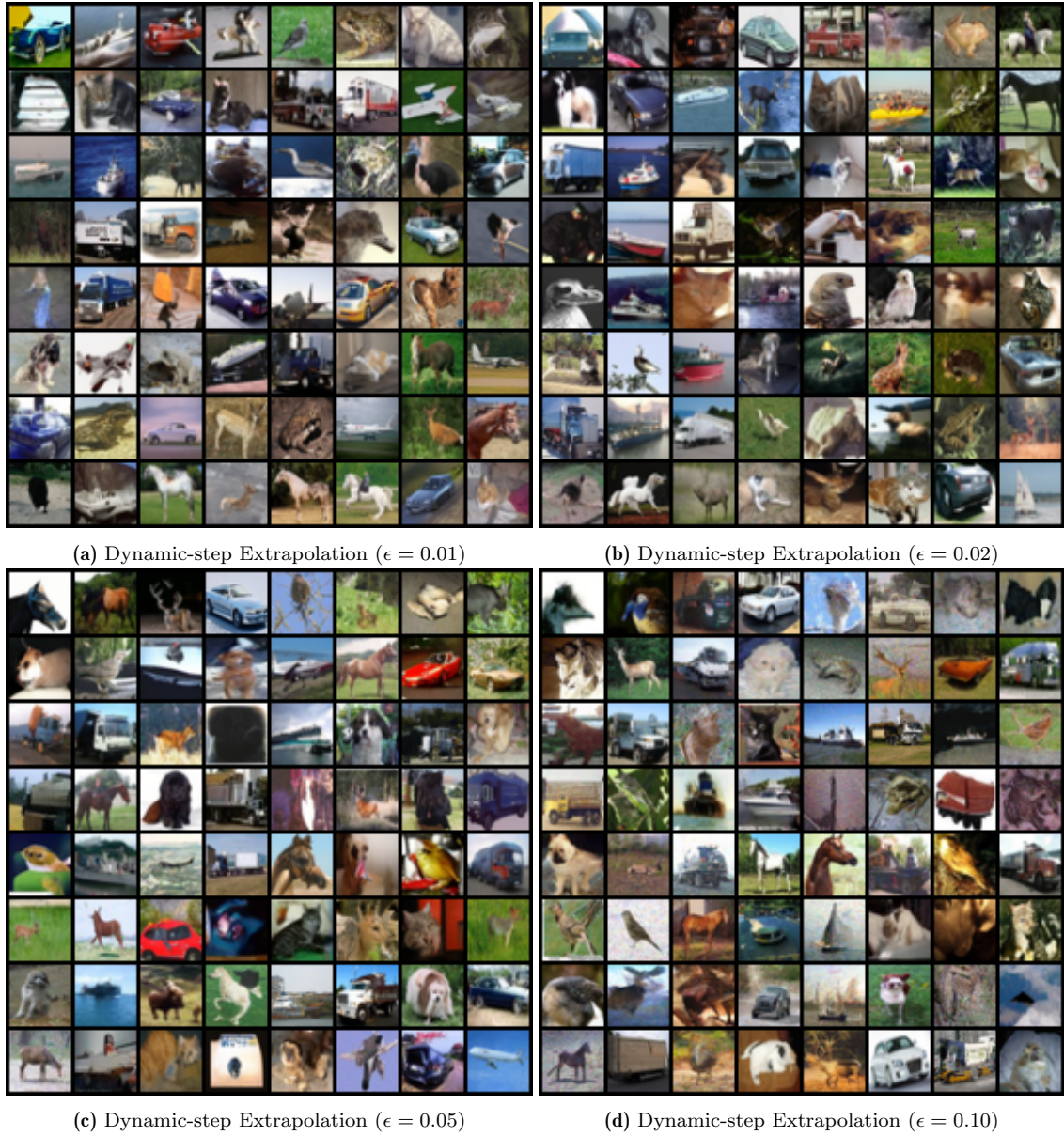
(b) Dynamic-step Extrapolation ( $\epsilon = 0.02$ )



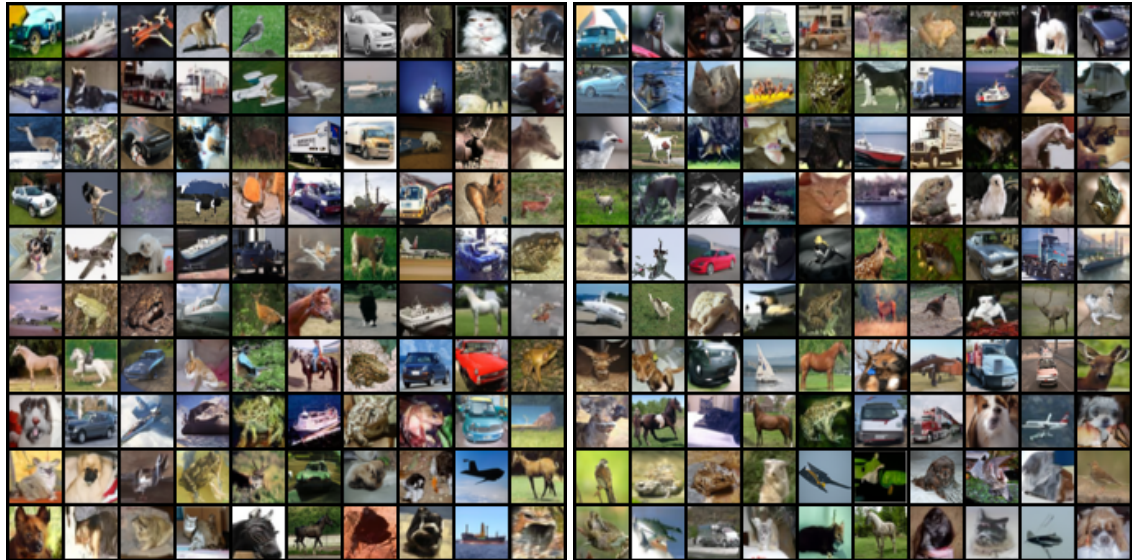
(c) Dynamic-step Extrapolation ( $\epsilon = 0.05$ )

(d) Dynamic-step Extrapolation ( $\epsilon = 0.10$ )

**Figure D.2** – VP-deep - CIFAR10

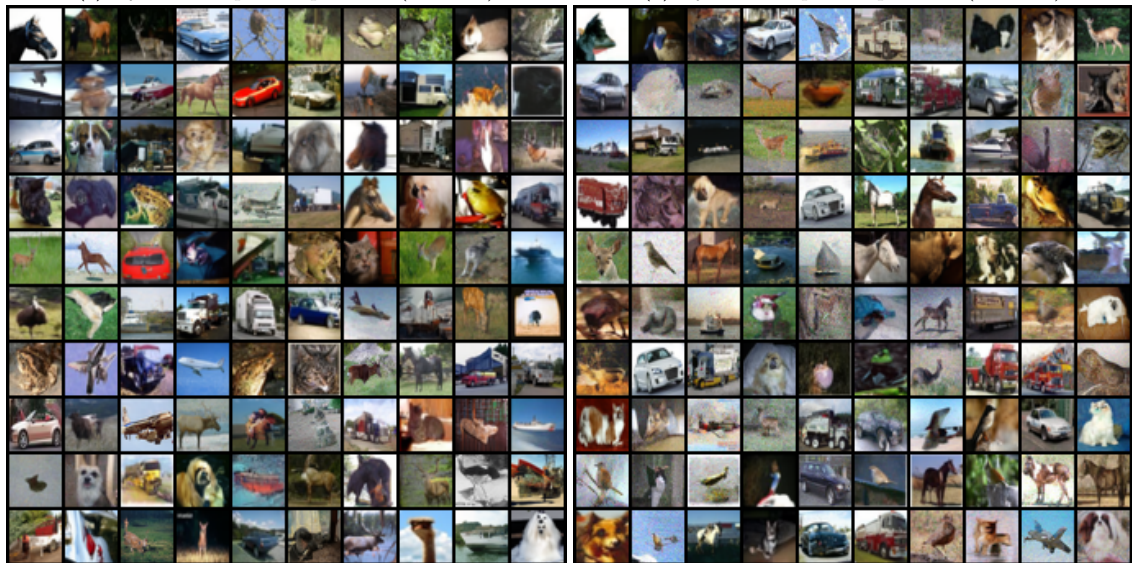


**Figure D.3** – VE - CIFAR10



(a) Dynamic-step Extrapolation ( $\epsilon = 0.01$ )

(b) Dynamic-step Extrapolation ( $\epsilon = 0.02$ )



(c) Dynamic-step Extrapolation ( $\epsilon = 0.05$ )

(d) Dynamic-step Extrapolation ( $\epsilon = 0.10$ )

**Figure D.4** – VE-deep - CIFAR10



(a) Dynamic-step Extrapolation ( $\epsilon = 0.01$ )

(b) Dynamic-step Extrapolation ( $\epsilon = 0.02$ )



(c) Dynamic-step Extrapolation ( $\epsilon = 0.05$ )

(d) Dynamic-step Extrapolation ( $\epsilon = 0.10$ )

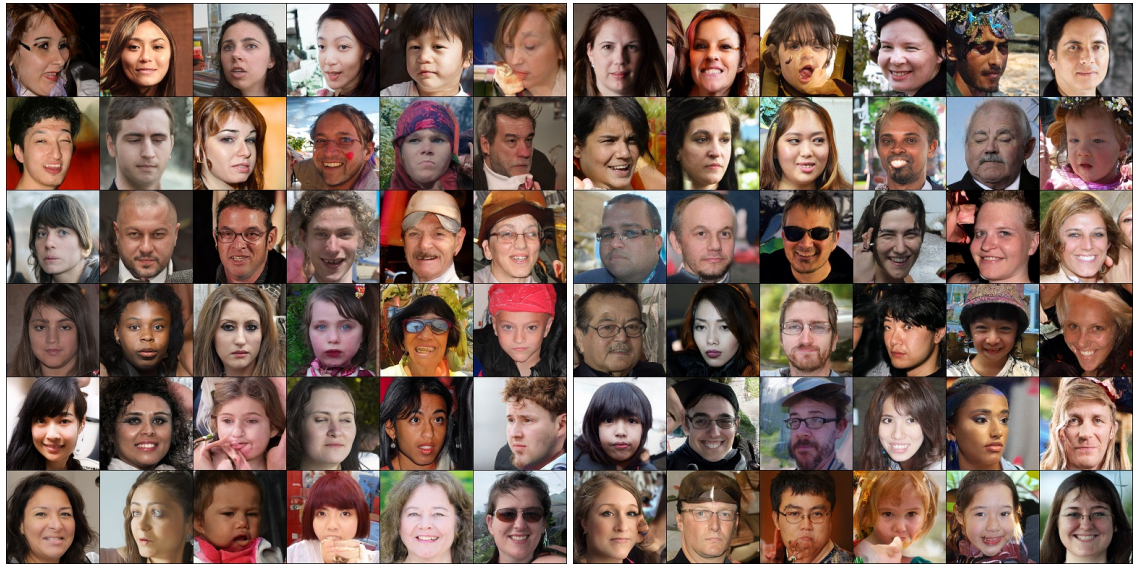
**Figure D.5 – VE - LSUN-Church (256x256)**





(a) Dynamic-step Extrapolation ( $\epsilon = 0.01$ )

(b) Dynamic-step Extrapolation ( $\epsilon = 0.02$ )



(c) Dynamic-step Extrapolation ( $\epsilon = 0.05$ )

(d) Dynamic-step Extrapolation ( $\epsilon = 0.10$ )

**Figure D.6 – VE - FFHQ (256x256)**