Thèse n° 10 257

Deep Generative Models and Applications

Présentée le 7 juillet 2020

à la Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Tatjana CHAVDAROVA

Acceptée sur proposition du jury

Prof. P. Frossard, président du jury
Prof. F. Fleuret, directeur de thèse
Prof. L. Denoyer, rapporteur
Prof. S. Lacoste-Julien, rapporteur
Prof. M. Jaggi, rapporteur

École
polytechnique
fédérale
de Lausanne

2020

What I cannot create,
I do not understand.
— Richard Feynman

To my sister Monika.

Acknowledgements

This dissertation would not have been possible without the support and inspiration of many people, to whom I am extremely grateful.

Foremost, I would like to express my gratitude to my supervisor François Fleuret, for his support and guidance throughout my PhD studies, the interesting discussions, as well as allowing me to focus on topics that I am interested in. I would like to thank my thesis committee for their constructive feedback, interest in my research, and for evaluating this thesis.

I am grateful to the Swiss National Science Foundation–grants CRSII2-147693 and 200021-169112, and the Hasler Foundation through the MEMUDE project.

My friends and colleagues from Idiap, EPFL, and UNIL made my Ph.D. journey enjoyable and fun. I sincerely thank my friend Natasha, with whom we shared many of our Ph.D. experiences. I thank my collaborators from Idiap, EPFL, and ETH, including Martin Jaggi, Matteo, Sebastian, Cijo, Andrii, Pierre, Timur, and Louis. It was fun being part of the Machine learning group at Idiap with Olivier, James, Yann, Alain, Suraj, Angelos, Evann, among others. Thanks to my friends Pranay, Gulcan, Rui, Ivana, Skanda, Angel, and many others for all the fun times.

During my PhD, I had the opportunity to do two internships. I thank Yoshua Bengio for accepting me as a research intern at Mila in Montreal, Canada and giving me the opportunity to collaborate with his group. I am particularly thankful to Prof. Simon Lacoste-Julien with whom I had the opportunity to closely collaborate, and from whom I have learned various aspects ranging from technical knowledge to general principles of conducting research and scientific writing. I will never forget my fun memories with my Mila friends, including Linda, Audrey, Gauthier, Hugo, Tristan, Alex, Amy, Jae, Evan, Ankesh, Michael, Isabeau, Florian, and many others. While my internship at DeepMind in London was not directly related to this thesis, it was extremely influential to my research interests and learned tools, for which I especially thank my host and supervisor Irina Higgins, who also introduced me to Hamiltonian Machine Learning. I enjoyed collaborating with Peter Toth, Aleksandar Botev, Danilo J. Rezende, Andrew Jaegle, Sébastien Racanière as well as our discussions with Shakir Mohamed and David Balduzzi. My internship would definitely not have been the same without my friends and colleagues Tatiana, Toni, Kyriacos, Yury, Rugile, Dhurva, Rishub, Tina, among many others.

Finally, while acknowledging the support of our close ones as key to following one's goals and interests, and most importantly providing the confidence necessary to pursue these, I dedicate this thesis to my sister Monika. I have been fortunate enough to have her unconditional love and support. I sincerely thank my father, Mile Chavdarov, who has always been my biggest

idol and inspiration, and whose guidelines, standards, as well as sense of humor I will never forget. Sincere thanks to my mother Snezhana for her devotion and sacrifices for me. I am grateful for staying in touch with my old friends from Macedonia, including Viktoria, Sandra, Ivana, Kristijan, Stefan, George, and Kaliopi. Last but not the least, I thank my family and relatives Mihaela, Bojan, Julia, Jack, Liliana, Sebastian, Dori, Ivan, Simona, Ljupco, among others, for their love and support.

During my PhD, I enjoyed the beautiful hikes and breathtaking landscapes in Switzerland, and I thank all the kind people I met in Lausanne and Martigny.

Lausanne, June 24, 2020 T. Chavdarova.

Abstract

Over the past few years, there have been fundamental breakthroughs in core problems in machine learning, largely driven by advances in deep neural networks. The amount of annotated data drastically increased and supervised deep discriminative models exceeded human-level performances in certain object detection tasks [Russakovsky et al., 2015, He et al., 2015]. The increasing availability in quantity and complexity of unlabelled data also opens up exciting possibilities for the development of unsupervised learning methods.

Among the family of unsupervised methods, deep generative models find numerous applications. Moreover, as real-world applications include high dimensional data, the ability of generative models to automatically learn semantically meaningful subspaces makes their advancement an essential step toward developing more efficient algorithms.

Generative Adversarial Networks (GANs) are a family of unsupervised generative algorithms that have demonstrated impressive performance for data synthesis and are now used in a wide range of computer vision tasks. Despite this success, they gained a reputation for being difficult to train, which results in a time-consuming and human-involved development process to use them. In the first part of this thesis, we focus on improving the stability and the performances of GANs.

Foremost, we consider an alternative training process to the standard one, named *SGAN*, in which several adversarial "local" pairs of networks are trained independently so that a "global" supervising pair of networks can be trained against them. The goal is to train the global pair with the corresponding ensemble opponent for improved performances in terms of mode coverage. Experimental results on both toy and real-world problems demonstrate that this approach outperforms standard training in terms of better mitigating mode collapse, stability while converging and that it surprisingly, increases the convergence speed as well.

Next, to further reduce the computational footprint while maintaining the stability and performance advantages of SGAN, we focus on training single pair of adversarial networks using variance reduced gradient. More precisely, we study the effect of the stochastic gradient noise on the training of generative adversarial networks (GANs) and show that it can prevent the convergence of standard game optimization methods, while the batch version converges. We address this issue with two *stochastic variance-reduced gradient* and *extragradient* optimization algorithms for GANs, named *SVRG-GAN* and *SVRE*, respectively. As batch extragradient is the only method that converges for simple examples of games, our analyses focus on SVRE, which method for a large class of games improves upon the previous convergence rates proposed in the literature. We observe empirically that SVRE performs similarly to a batch method on the MNIST dataset, while being computationally cheaper, and that SVRE yields more stable GAN training on standard datasets.

In the second part of the thesis we present our work on people detection, done prior to the above. People detection methods are highly sensitive to occlusions between pedestrians, which are extremely frequent in many situations where cameras have to be mounted at a limited height. The reduction of camera prices allows for the generalization of static multi–camera set–ups. Using joint visual information from multiple synchronized cameras gives the opportunity to improve detection performance. We address the problem of multi-view people occupancy map estimation using an end–to–end deep learning algorithm called *DeepMCD* that jointly utilizes the correlated streams of visual information. Besides the lack of datasets at the given time, DeepMCD empirically outperformed the classical approaches by large margin. We demonstrate its generalization properties to a small scale dataset called *EPFL-RLC* that we make publicly available. Finally, we present a new large-scale and high-resolution dataset. It has been captured with seven static cameras in a public open area, and unscripted dense groups of pedestrians standing and walking. Together with the camera frames, we provide an accurate joint (extrinsic and intrinsic) calibration, as well as 7 series of 400 annotated frames for detection at a rate of 2 frames per second. This results in over 40 000 bounding boxes delimiting every person present in the area of interest, for a total of more than 300 individuals. We provide a series of benchmark results using baseline algorithms published over the recent months for multi-view detection with deep neural networks, and trajectory estimation using a non-Markovian model.

Résumé

Au cours des dernières années, nous avons assisté à d'importantes avancées concernant les problèmes fondamentaux de l'apprentissage automatique, cela en grande partie grâce aux progrès des réseaux de neurones profonds. La quantité de données annotées a considérablement augmenté et les modèles discriminatifs profonds supervisés ont surpassé les humains dans certaines tâches de détection d'objets [Russakovsky et al., 2015, He et al., 2015]. La disponibilité croissante en quantité et en complexité des données non annotées ouvre également des possibilités intéressantes pour le développement de méthodes d'apprentissage non supervisées.

Parmi ces algorithmes non supervisés, les modèles génératifs profonds trouvent de nombreuses applications. De plus, sachant qu'un scénario réel demande le traitement de données de grande dimensionnalité, la capacité des modèles génératifs à apprendre des sous-espaces sémantiques pertinents font que leur développement est une étape essentielle en vue d'obtenir des algorithmes plus efficaces.

Les réseaux antagonistes génératifs sont une famille d'algorithmes génératifs non supervisés capable de générer des données réalistes, leur succès fait qu'ils sont désormais utilisés dans un large éventail de tâches de vision par ordinateur. Malgré ce succès, ils ont acquis la réputation d'être difficiles à entraîner, ce qui rend leur utilisation complexe et fortement basée sur l'intuition et la supervision de l'expérimentateur. Dans la première partie de cette thèse, nous nous concentrons sur l'amélioration de la stabilité et des performances des réseaux antagonistes génératifs.

Dans un premier temps, nous considérons un processus d'entraînement alternatif, nommé SGAN, dans lequel plusieurs paires de réseaux antagonistes «locaux» sont entraînés indépendamment afin qu'une paire «globale» de réseaux supervisés puisse être entraînée contre eux. L'objectif est d'entraîner la paire globale en l'opposant à l'ensemble des adversaires pour obtenir de meilleures performances en termes de couverture de mode. Les résultats expérimentaux sur des tâches fictives et réelles démontrent que cette approche surpasse les méthodes d'entraînement standard en atténuant l'effondrement des modes, en améliorant la stabilité lors de la convergence, et étonnamment, en augmentant également la vitesse de convergence.

Dans un second temps, pour réduire davantage l'empreinte computationnelle tout en conservant les avantages de stabilité de SGAN, nous nous concentrons sur la formation d'une seule paire de réseaux antagonistes en utilisant un gradient à variance réduite. Plus précisément, nous étudions l'effet du bruit stochastique du gradient sur l'entraînement des réseaux antagonistes génératifs (GAN) et montrons qu'il peut empêcher la convergence des méthodes standard d'optimisation de jeux, tandis que la version utilisant des full-batchs converge. Nous abordons ce problème avec deux algorithmes, un d'optimisation du gradient à variance réduite, et un d'optimisation de l'extragradient pour les GAN, nommés SVRG-GAN et SVRE, respectivement. Étant donné que l'extragradient utilisant des full-batchs est la seule méthode qui converge pour des exemples de jeux simples, nos analyses se concentrent sur SVRE, laquelle améliore les taux de convergence précédemment proposés dans la littérature pour un grand nombre de jeux divers. Nous observons empiriquement que SVRE fonctionne de manière similaire à une méthode utilisant des full-batchs sur le set de données MNIST, tout en étant moins coûteux en termes de calcul. De plus, SVRE permet d'entraîner un GAN de manière plus stable sur des sets de données standards.

Dans la deuxième partie de la thèse, nous présentons nos travaux sur la détection de personnes, réalisés avant ce qui précède. Les méthodes de détection de personnes sont très sensibles aux occlusions entre piétons, qui sont extrêmement fréquentes dans de nombreuses situations où les caméras ne peuvent être montées qu'à une hauteur limitée. La réduction du prix des caméras permet de banaliser les dispositifs à plusieurs caméras statiques. L'utilisation d'informations visuelles provenant de plusieurs caméras permet d'améliorer les performances de détection. Nous abordons le problème de l'estimation de la carte d'occupation des personnes, basé sur des images provenant de plusieurs caméras. Pour ce faire nous utilisons un algorithme d'apprentissage profond appelé DeepMCD qui utilise conjointement les flux corrélés d'informations visuelles. Outre le manque de données disponibles au moment de ce projet, DeepMCD a surpassé empiriquement les approches classiques par un gain de performance significatif. Nous démontrons ses propriétés de généralisation sur un ensemble de données de petite échelle appelé EPFL-RLC que nous mettons à la disposition du public. Enfin, nous présentons un nouvel ensemble de données haute résolution à grande échelle. Il a été créé à partir de sept caméras statiques dans un espace public ouvert, avec des groupes denses de piétons non scénarisés. Joint aux images des caméras, nous fournissons aussi un calibrage précis (extrinsèque et intrinsèque), ainsi que 7 séries de 400 images annotées pour la détection à un taux de 2 images par seconde. Il en résulte plus de 40 000 boîtes de délimitation délimitant chaque personne présente dans la zone d'intérêt, pour un total de plus de 300 individus. Nous fournissons une série de résultats de référence, incluant plusieurs algorithmes de base publiés au cours des derniers mois pour la détection à vues multiples, à base de réseaux de neurones profonds, ou basées sur l'estimation de trajectoire à l'aide d'un modèle non markovien.

Contents

Acknowledgements **i**

Abstract **iii**

Résumé **v**

List of Figures **xi**

List of Tables **xiii**

List of Algorithms **xv**

1 Introduction **1**

1.1 Motivations . . . 1

1.1.1 Improving stability of Generative Adversarial Networks . . . 1

1.1.2 Multi–camera people detection . . . 2

1.2 Summary of contributions . . . 3

1.3 Outline . . . 3

1.4 Notation and Acronyms . . . 4

2 Background **5**

2.1 Maximum likelihood estimation . . . 5

2.2 Deep learning . . . 6

2.2.1 Defining a model: a neural network . . . 6

2.2.2 Loss functions for deep learning . . . 8

2.2.3 Backpropagation algorithm . . . 8

2.2.4 Gradient based optimization . . . 8

2.3 Generative models . . . 9

2.4 Generative Adversarial Networks . . . 10

2.4.1 Evaluating GANs . . . 12

I Deep Generative Models **15**

3 SGAN: An Alternative Training of Generative Adversarial Networks **17**

3.1 Introduction . . . 17

3.2 Related work: Variants of the GAN algorithm 19
3.3 Method 21
3.3.1 Description of SGAN 22
3.4 Experiments 23
3.4.1 Experimental results on toy datasets 25
3.4.2 Experimental results on real-world datasets 28
3.5 Related work: Multi-network GAN methods 29
3.6 Discussion and future directions 32

4 Reducing the Noise of the Stochastic Gradient-based GAN Optimization 35
4.1 Introduction 35
4.2 GANs as a Game and Noise in Games 37
4.2.1 Game theory formulation of GANs 37
4.2.2 Stochasticity Breaks Extragradient 38
4.3 Reducing Noise in Games with Variance Reduced Extragradient 40
4.3.1 Variance Reduced Gradient Methods 41
4.3.2 SVRE: Stochastic Variance Reduced Extragradient 42
4.3.3 Motivating example 44
4.4 Pseudocode for SVRE-GAN and SVRG-GAN 45
4.5 Experiments 46
4.5.1 Results 48
4.6 Related work 50
4.7 Discussion and future directions 50

II Deep Learning Applications in Multi-Camera People Detection 53

5 Deep Multi-Camera People Detection 55
5.1 Introduction 55
5.2 Related work 57
5.2.1 Deep monocular pedestrian detection 57
5.2.2 Multi-view occupancy map estimation methods 57
5.3 Deep Multi-View People Detection 58
5.3.1 Problem definition 58
5.3.2 Method 59
5.4 Experiments 61
5.4.1 datasets and metrics 61
5.4.2 Monocular pedestrian detection 62
5.4.3 Multi-camera people detection 62
5.4.4 Comparison to existing methods on PETS 65
5.5 Conclusion and Future Work 66

6 WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection 67

6.1 Introduction . . . 67
6.2 Related work . . . 69
6.2.1 Datasets . . . 69
6.2.2 Methods . . . 70
6.3 The dataset . . . 72
6.3.1 Hardware and data acquisition . . . 72
6.3.2 Calibration of the cameras . . . 73
6.3.3 Annotation process . . . 74
6.3.4 Statistics . . . 76
6.4 Benchmarks . . . 77
6.4.1 Evaluation protocol . . . 77
6.4.2 Evaluated methods . . . 78
6.4.3 Benchmark results . . . 79
6.5 Discussion . . . 80

7 Concluding Remarks 83
7.1 Summary and contributions . . . 83
7.2 Limitations and future work . . . 84

A Appendix for Chapter 2 87
A.1 Equilibrium of GANs in functional space . . . 87

B Appendix for Chapter 3 89
B.1 Experiments on toy datasets . . . 89
B.1.1 Details on the implementation . . . 89
B.1.2 Experiments . . . 89
B.2 Experiments on real-world datasets . . . 90
B.2.1 Details on the implementation . . . 90
B.3 Different viewpoints of SGAN . . . 91

C Appendix for Chapter 4 101
C.0.1 Why is convergence of the last iterate preferable? . . . 101
C.1 Definitions and Lemmas . . . 101
C.1.1 Smoothness and Monotonicity of the operator . . . 101
C.2 Proof of Theorems . . . 103
C.2.1 Proof of Theorem 1 . . . 103
C.2.2 Proof of Theorem 2 . . . 104
C.3 Details on the SVRE–GAN Algorithm . . . 111
C.3.1 Practical Aspect . . . 111
C.4 Restarted SVRE . . . 112
C.5 Details on the implementation . . . 113
C.5.1 Metrics . . . 113

C.5.2 Architectures & Hyperparameters 114
C.6 Additional Experiments 121
C.6.1 Results on MNIST 121
C.6.2 Results with shallow architectures 121
C.6.3 Results with deeper architectures 124

D Appendix for Chapter 6 129
D.1 Annotation process 129
D.2 Annotations 131
D.2.1 File formats 131
D.2.2 Memory size 132
D.3 Camera calibration 132
D.4 Additional statistics 132
D.5 Recommended splits of the WILDTRACK dataset 133
D.6 Additional tracking benchmarks 133

Bibliography 135

Curriculum Vitae 147

List of Figures

2.1 Example of a feedforward fully connected neural network 7
2.2 Taxonomy of generative models 10
2.3 GAN training scheme 11

3.1 Conceptual illustration of *SGAN* 18
3.2 A toy experiment with vanilla GAN and *SGAN* on mixture of 10 Gaussians . . . 24
3.3 Experiments using 10–S–WGAN on *8-GMM*, *25-GMM*, and the *Swiss Roll* toy datasets. 26
3.4 10–S–WGAN on the 8–GMM toy dataset 27
3.5 Log–likelihood on 8-GMM toy dataset 28
3.6 (5-S-) GAN/WGAN/DRAGAN experiments on MNIST 28
3.7 5-SGAN comparison on CIFAR10. 29
3.8 5-SGAN experiments on CelebA, ImageNet, MNIST and LSUN. 30
3.9 5-S-DCGAN on the STL-10 dataset 31
3.10 Toy experiment of an extension of SGAN on the Swiss Roll dataset. 32

4.1 Intuitive illustration of the discrepancy between games and minimization on simple examples 39
4.2 SVRE benchmark on MNIST with Batch–Extragradient and Stochastic–Extragradient which use Adam 47
4.3 SVRE benchmark on bilinear game using distance to the optimum 48
4.4 SVRG-GAN experiments on MNIST 49
4.5 SVRG-GAN experiments on MNIST and FASHION-MNIST 49

5.1 Illustration of the people occupancy map estimation problem with static cameras 56
5.2 Illustration of the input to DeepMCD 58
5.3 Input–occlusion masks 59
5.4 Illustration of DeepMCD for multi-view joint occupancy estimation 60
5.5 Impact of training with hard-negative multi-view samples, versus the NMS threshold on the EPFL-RLC dataset 63
5.6 ROC and AUC using random forest of 100 trees on the EPFL-RLC dataset 64

6.1 Synchronized corresponding frames from the seven views of WILDTRACK . . . 71
6.2 The overlap between the cameras' fields of view 73

6.3 Percentage of examples that are occluded within certain range 73
6.4 Illustration of the camera calibration precision 74
6.5 GUI of our multi-camera annotation tool 75
6.6 Exapmle of a positive multi–view annotation 76
6.7 Multi-view detectors benchmark on WILDTRACK 81

B.1 5-S-WGAN on the 10-GMM dataset 93
B.2 5-S-DCGAN experiments on the FASHION-MNIST dataset 94
B.3 10-S-DCGAN experiments on the CIFAR10 dataset 94
B.4 5-S-WGAN experiment on the 8-GMM toy dataset 95
B.5 5-S-DRAGAN experiments on the CIFAR10 dataset 96
B.6 5-S-DCGAN experiments on the CelebA dataset 96
B.7 5-S-DRAGAN experiments on the CelebA dataset 97
B.8 5-S-DCGAN experiments on the LSUN-bedroom dataset 97
B.9 5-SW-DRAGAN experiments on the LSUN-bedroom dataset 98
B.10 5-SW-WGAN on the One Billion Word Benchmark dataset 99
B.11 Graphical comparison of the information flow structures of GAN and SGAN. . . 99

C.1 Stochastic, full-batch and variance reduced versions of the extragradient method ran on MNIST 122
C.2 Comparison between SVRE and the *SE–A* baseline on Imagenet, using the *shallow* architectures 122
C.3 Average second moment estimate on SVHN for the Generator and the Discriminator, using the *shallow* architectures 123
C.4 FID scores with different hyperparameters for the SE–A baseline on CIFAR10 and the SG–A baseline on SVHN, using the *deep* architectures 124
C.5 FID scores for SVRE, using the *deep* architectures on SVHN 125
C.6 FID scores for WS–SVRE, using the *deep* architectures on CIFAR-10 125
C.7 Average second moment estimate on CIFAR-10 for the Generator and the Discriminator using *deep* architectures 126

D.1 Graphical User Interface (GUI) of our multi-view labeller 130
D.2 Histogram of the number of frames in which one person appears 133

List of Tables

1.1 Notation 4
1.2 Acronyms 4

3.1 SGAN comparison on the One Billion Word Benchmark dataset 30

4.1 Comparison of variance reduced methods for games for a μ-strongly monotone operator with L_i-Lipschitz stochastic operators. 37
4.2 Methods for solving bilinear games 38
4.3 Best obtained FID scores for the different optimization methods using the *deep* architectures 48

5.1 Test accuracy on the EPFL-RLC dataset 65
5.2 Comparison on the PETS dataset 66

6.1 List of commonly used person datasets with a focus on the multi-camera ones 69
6.2 Tracking results on the WILDTRACK dataset 79
6.3 Benchmark detection results on WILDTRACK 80
6.4 Classification results on the WILDTRACK dataset using ResNet-18 80

C.1 Description of the DCGAN architectures used for experiments on MNIST 115
C.2 Description of the layers of the self–attention block used in the SAGAN architectures 117
C.3 Description of the *Shallow* SAGAN architectures for experiments on SVHN and CIFAR-10 118
C.4 Description of the *Shallow* SAGAN architectures for experiments on ImageNet 119
C.5 Description of the ResNet blocks used for the ResNet architectures 120
C.6 Description of the *Deep* ResNet architectures used for experiments on SVHN and CIFAR-10 120
C.7 Best obtained IS and FID scores for the different optimization methods, using *shallow* architectures 121

D.1 Additional tracking results on the WILDTRACK dataset 134

List of Algorithms

1 Pseudocode for *alternating* GAN. 12
2 Pseudocode for SGAN 21
3 Pseudocode for SVRE. 37
4 Pseudocode for SVRE-GAN 45
5 Pseudocode for SVRG-GAN 46
6 Pseudocode for Restarted SVRE 113

1 Introduction

Contrary to explicitly defining detailed rules to solve a particular prediction or inference task such as classification or clustering, *machine learning* is the study of algorithms that can extract statistical patterns directly from the data to accomplish such tasks. These algorithms are particularly useful for problems where the former approach is impractical, such as object detection given raw pixels, decision making, or training agents able to generalize their knowledge to new tasks, among others.

Deep Learning algorithms utilize layer-wise processing of the input data by using *Neural Networks* to solve machine learning tasks. Each layer of the network contains parameters that are trained iteratively using the Backpropagation algorithm and a non-linear mapping. Such stacked layers allow for learning hierarchical representations, as each layer's input is the output of the previous layer. This allows for combining the learned patterns from that preceding layer into more complex or abstract patterns of the input data. For details, we refer the reader to § 2.2.

1.1 Motivations

This thesis was inspired from two separate problems in machine learning learning.

1.1.1 Improving stability of Generative Adversarial Networks

A longstanding aim of machine learning researchers is obtaining artificial agents that despite being trained on a limited number of tasks, can generalize and perform well on unseen tasks. Developing such algorithms is considerably more challenging than specialized algorithms, where agents are trained on a specific task. Due to the high dimensionality of the raw input data of real-world tasks, a critical approach for efficiency in terms of fast optimization and manageable memory footprint for such algorithms is learning a good semantic subspace in which we map the input datapoints. Generative models share the same underlying goal of learning representations of input data, and these models represent a probability distribution over a set of random variables either *explicitly* or *implicitly*. In the latter case, the model is capturing an underlying probability distribution that we do not have direct access to, but we can sample according to it, see § 2.3. Deep generative models (DGM) are generative models that rely on deep neural networks (DNN).

Another application of generative models is for *planning* in Reinforcement learning [Sutton and Barto, 2018], where agents simulate sequences of outcomes to determine which action to take [*e.g.* Kurutach et al., 2018]. Generative models are also used in physics, for example for generating plausible trajectory of a particle that follows Hamiltonian dynamics [Greydanus et al., 2019, Botev et al., 2020], among others. Other applications include image to image translation (*e.g.* edges to images [Isola et al., 2017]), super-resolution [Ledig et al., 2016], image *inpainting* and text to image generation [Zhang et al., 2017].

Generative Adversarial Networks (GANs, [Goodfellow et al., 2014]) are a family of *implicit* generative algorithms that are fast to sample from. Training is formulated as *minimax* optimization (see § 2.4), by defining competing objectives of a *generator* and a *discriminator* deep neural network. The discriminator–D and the Generator–G aim at distinguishing real from generated samples, and "fooling" D that the generated samples are real (by mapping random noise to samples), respectively. The networks are updated in an alternating fashion–by fixing the parameters of one of the networks while updating the other, and *vice versa*, see § 2.4.

GANs have found numerous applications in deep learning, particularly in computer vision. However, theoretical insights on how the minimax optimization differs from the well studied single objective optimization are still lacking, and SGD based algorithms–well developed for supervised tasks, are also used for GAN training. Hence, the architectural or optimization settings essential for successful minimax optimization are mostly empirically driven [Radford et al., 2016]. Moreover, convergence failures, poor performance (referred to as "mode collapse", see § 3.1) or different optimal hyperparameters or architectures throughout different real-world datasets are commonly reported.

In the first part of this thesis, we focus on two methods that improve the performance and the stability of their training.

1.1.2 Multi–camera people detection

Pedestrian detection is an important computer vision problem with numerous applications in security, surveillance, robotics, autonomous driving, and crowdsourcing. The variation of pedestrians appearance greatly increases the difficulty of this problem. With the availability of large-scale monocular datasets of annotated pedestrians and the advances in detection algorithms, the accuracy of the pedestrian detectors has improved significantly in the past few years. Moreover, modern detection algorithms using deep learning allow us to learn discriminative features which are transferable across datasets. Impressively, recently developed deep learning based monocular detectors are approaching human-level performance [Zhang et al., 2016] on common benchmark datasets [Du et al., 2017].

However, many situations of practical interest require detection in highly crowded and cluttered scenes. Severe occlusions make monocular pedestrian detection insufficient in these scenarios. Luckily, in real-world applications, image feeds from multiple cameras with overlapping fields of view are often available. Most commonly, the cameras are positioned slightly above the average human height. Hence designing pedestrian detectors by exploiting multiple views and the geometry of the scene will provide reliable detection estimates in crowded scenes.

In the second part of this thesis, we propose: (i) a novel end–to–end deep learning method named DeepMCD which jointly utilizes the synchronized video streams, (ii) a novel camera calibration method based on *bundle adjustment* [146], and (iii) we make publicly available two multi-camera datasets, whose data acquisition process and statistics are elaborated.

1.2 Summary of contributions

The main contributions of this thesis are summarized as follows:

- Motivated by the "mode collapse" problem, we propose a novel method for training GANs named *SGAN*, which uses an ensemble of pairs of adversarial networks whose pairing is fixed. SGAN uses the ensemble to train a single pair of networks hence producing *single generative model*, while maintaining the pairs of the ensemble statistically independent.
- We present two novel optimization methods for GANs that use Variance Reduction: (i) *SVRG–GAN*–which extends for GANs the Stochastic Variance Reduced Gradient (SVRG) method, proposed by Palaniappan and Bach [2016] for single objective optimization; and (ii) *Stochastic Variance Reduced Extragradient (SVRE)*–which combines SVRG with the *Extragradient* method [proposed by Korpelevich, 1976] .
- We propose the first end–to–end deep learning method for multi-camera people detection, which jointly utilizes the correlated video streams to resolve the occlusions among the people.
- We propose a novel multi-camera calibration method which uses *bundle adjustment* 146.
- We make publicly available two multi-camera datasets: (i) the *EPFL–RLC* dataset; as well as (ii) the *WILDTRACK* dataset which uses the proposed camera calibration method.

1.3 Outline

This thesis is organized as follows. In Chapter 2 we review machine learning concepts relevant for the rest of the thesis, such as: (i) deep neural networks, (ii) generative models with focus

Table 1.1 – Notation.

$\mathbb{R}$	The set of real numbers
$\mathcal{N}(\mu,\sigma)$	Normal distribution with mean μ and standard deviation σ
$f: x \mapsto y$	mapping f from $x \in \mathbb{R}^a$ to $y \in \mathbb{R}^b$, where $a, b \in \mathbb{R}$ are specified where needed
$f(\cdot;\theta)$	mapping f parameterized with $\theta \in \mathbb{R}^d$, where $d \in \mathbb{R}$ is specified where needed
$\mathbb{D}_{KL}$	Kullback–Leibler divergence
$\mathbb{D}_{JS}$	Jensen–Shannon divergence
p_d	real data distribution
n	number of samples in the given dataset
θ	model's parameters

on GANs, as well as (iii) elaborating the problem of multi-camera people detection.

We separate the thesis into two parts: (i) in the first part, we describe SGAN and SVRE in Chapters 3 and 4, respectively, and (ii) in the second part, we focus on multi-camera people detection, where we describe the proposed method called *DeepMCD* and the WILDTRACK dataset, in Chapters 5 and 6, respectively. Chapter 7 concludes our work.

1.4 Notation and Acronyms

Table 1.1 summarizes the notation that used throughout the thesis, whereas chapter specific notation is specified where needed. Table 1.2 lists the often used acronyms across the thesis.

Table 1.2 – Acronyms.

(D)NN	(Deep) neural network
FNN	feedforward neural network
CNN	Convolutional neural network
DGM	Deep generative models
GAN	Generative adversarial networks [Goodfellow et al., 2014]
VAE	Variational Autoencoders [Kingma and Welling, 2014]
IS	Inception Score, see § 2.4.1
FID	Fréchet Inception Distance, see § 2.4.1
SVRG	Stochastic Variance Reduced Gradient [Palaniappan and Bach, 2016]
SVRE	Stochastic Variance Reduced Extragradient [Chavdarova et al., 2019]
MLE	Maximum likelihood estimation
GD	batch Gradient Descent
SGD	Stochastic Gradient Descent [Robbins and Monro, 1951]

2 Background

Machine learning algorithms produce a *model* $f(\cdot;\theta)$ parametrized with finite number of parameter(s) θ, which “explains” the observed data x. Such a model proposes a general functional relation between the unknown parameter(s) and the observed data.

In this chapter we provide a brief overview of the concepts relevant for the chapters that follow, whereas for a complete introduction to convex optimization and deep learning we refer the reader to [Boyd and Vandenberghe, 2004] and [Goodfellow et al., 2016], respectively.

Foremost, we discuss *maximum likelihood estimation* in § 2.1 as most commonly used statistical estimator of the unknown model parameters. In § 2.2 we review the most important concepts and steps of building a deep neural network. In § 2.3 we review generative models, and in § 2.4 we focus on GANs and ways to evaluate them.

2.1 Maximum likelihood estimation

The *likelihood function* is the most commonly used statistical estimator of the unknown parameters of the proposed model. Given observed datapoints $x_1,\dots,x_n$ of the random variables $X_1,\dots,X_n$ whose joint density function is $f(X_1,\dots,X_n|\theta)$, the likelihood function of θ is defined as:

$$L(\theta) = L(\theta|x_1,\dots,x_n) = f(x_1,\dots,x_n|\theta). \tag{2.1}$$

Finally, the maximum likelihood estimation (MLE) of θ is defined as:

$$\theta^* = \arg\max_\theta L(\theta|x_1,\dots,x_n) = \arg\max_\theta f(x_1,\dots,x_n|\theta). \tag{2.2}$$

If we assume that the datapoints $x_1,\dots,x_n$ are *i.i.d*, the likelihood simplifies to:

$$\theta^* = \arg\max_\theta \prod_{i=1}^{n} f(x_i|\theta) = \arg\max_\theta \sum_{i=1}^{n} \log f(x_i|\theta). \tag{2.3}$$

where in the last step we used the fact that the logarithm is a monotonically increasing function, obtaining the *maximum log–likelihood function*. Hence, the maximum log–likelihood describes the log–probability that the density function f assigns to all the training data points and adjusts the parameters θ so as to increase it.

2.2 Deep learning

A machine learning algorithm that uses *neural network(s)* (see below) as its model is commonly referred as a "deep learning" algorithm[1].

Given a training dataset, as core components and methods of a deep learning model we enumerate the following: (i) *defining the model*–this includes defining the architecture of the neural network(s), (ii) *defining a loss function* (or several), (iii) the *backpropagation algorithm* [Rumelhart et al., 1986] often called *backprop*, as well as (iv) *selecting an optimization method*. Below we explain these consecutively, while for brevity of this overview we focus on most common or simplistic scenarios, *e.g.* single objective supervised training rather than special cases of having multiple losses or agents.

2.2.1 Defining a model: a neural network

Perceptron & neuron. The units of a neural network were inspired by the *perceptron* algorithm, proposed by Rosenblatt [1958]. A perceptron takes binary input $x_1, \dots, x_d$ where $x_i \in \{0,1\}$ and outputs a binary output $y \in \{0,1\}$. To allow for expressing that each input has different importance, each input has its associated *weight*: $w_i \in \mathbb{R}, i = 1, \dots, d$. Finally, the output is 0 if the sum $\sum_{i=1}^{d} w_i x_i$ is smaller than a predefined threshold $t \in \mathbb{R}$, and 1 otherwise. For clarity we define a *bias* of the perceptron as $b = -t$, and summarize as:

$$f_{perceptron}(x) = \begin{cases} 0, & \text{if } w \cdot x + b \geq 0 \\ 1, & \text{otherwise} \end{cases} .$$

Moreover, m perceptrons can be combined in a layer, where each unit of the first layer takes as input all inputs yielding outputs $y_1, \dots, y_m$. Multiple layers can be further stacked sequentially, allowing for modeling more complex functions that weight the inputs. Neurons are similar in structure to perceptrons, with modification that $x \in \mathbb{R}^d, y \in \mathbb{R}$ and $f_{neuron}(x) = \sigma(w \cdot x + b)$, where $\sigma(\cdot)$ denotes a non–linear mapping known as *activation function* (see below).

Neural networks and types. A network of neurons with connections that do not form a cycle is called a *feedforward neural network* (FNN). The neurons are typically organized into multiple layers, where neurons of i–th layer connect only to neurons of the immediately preceding, *i.e.*

---

[1] Note that the use of term "deep learning" varied over the past years. For example several years ago as "deep neural networks" were considered neural networks with approximately more than five layers.

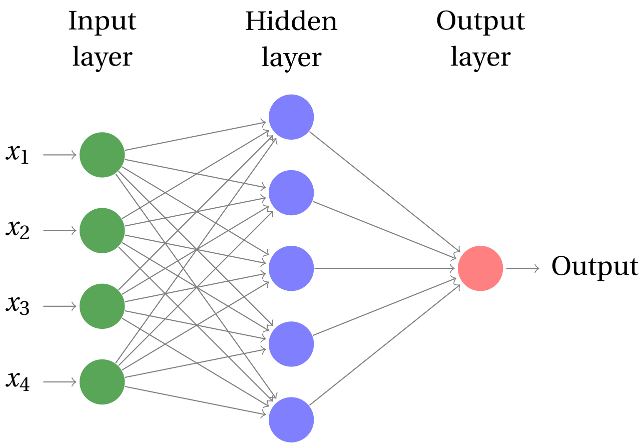


Figure 2.1 – Example of a feedforward fully connected neural network with input layer of four inputs $x_1,\dots,x_4$, one hidden layer, and an output layer, depicted in green, blue, and red, respectively.

$(i-1)$–th layer, and immediately following, *i.e.* $(i+1)$–th layers. The neurons can be connected in different ways such as: (i) *fully connected*–where each neuron at layer i is immediately connected with *all* the neurons of the preceding layer $i-1$, (ii) *convolutional*–special case of fully connected layer with *weight sharing* which in turn decreases the total number of parameters of the network by orders of magnitude, (iii) *pooling*–which layers reduce the dimension of the input data by taking for example the *maximum element* or the *average* over a fixed number of elements of the preceding layer, and others. *Multilayer perceptron* (MLP) is a FNN consisting of solely fully connected layers.

FNN with convolutional layers is called convolutional neural network (CNN) [Fukushima, 1980, LeCun et al., 1999]. CNNs were motivated for computer vision applications, where the input is typically a three dimensional image, with dimensions: number of channels (*e.g.* RGB, HSV), width and height. Convolutional layers typically convolve the three dimensional input with multiple three dimensional matrices of real numbers stacked together, called *filters* or *kernels*, hence yielding four dimensional output called a *feature map*. The use of convolutions allows for *parameter efficiency*–due to weight-sharing, as well as building in prior knowledge into the network of *translational invariance* of the learned features.

Fig. 2.1 depicts an example of a feedforward fully connected neural network with four inputs, one *hidden* layer and one output. The edge between the i–th neuron of the input layer and j–th neuron of the hidden layer depicts the weight $w^{(1)}_{i,j}$. Besides FNNs, networks with connections between neurons in the same or previous layers are known as *recurrent neural networks* discovered by Hopfield [1982].

Neural nets are non–linear mapping. Each neuron of a fully connected or convolutional layer consists of a *non linear* function called *activation function* or *transfer function*. This is *requirement* for a key proof regarding the approximation capacity of a neural network, called the *the universal approximation theorem*, to be valid [Cybenko, 1989, Hornik, 1991]. This theorem states that under certain conditions, any continuous function on a closed bounded subset of $\mathbb{R}^n$ can be approximated by a neural network with one hidden layer and a sufficiently large number of hidden units. The most common activation functions used in deep learning are: (i) *Rectified Linear Unit–ReLU* activation function: $\text{ReLU}(x) = \max(0, x)$, (ii) sigmoid or logistic activation function: $\sigma(x) = \frac{1}{1+e^{-x}}$, (iii) hyperbolic tangent: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, or (iv) *Swish* non-linearity [Ramachandran et al., 2017]: $\text{swish}(x) = x \cdot \sigma(x)$, among others.

2.2.2 Loss functions for deep learning

Loss functions are real–valued functions that provide means to evaluate how well the model fits the given dataset. Let us consider training a classifier $f(x;\theta)$ and let $\hat{y}_1, \dots, \hat{y}_n$ denote the predictions of our model, given samples $x_1, \dots, x_n$, whereas $y_1, \dots, y_n$ denote the corresponding annotated labels.

Examples of loss functions used in deep learning are: (i) minimizing the *cross entropy* loss for binary classification (equivalent to maximizing the likelihood), or more generally the *Negative Log Likelihood* (NLL), $\text{NLL} = -\log \hat{y}$; as well as (ii) *Mean Square Error* (MSE), $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$, among others.

2.2.3 Backpropagation algorithm

The backpropagation method provides a *way of computing the gradient of the cost function w.r.t.* the network's weights and biases. We refer the reader to [Nielsen, 2015, Goodfellow et al., 2016] for details, and below we describe briefly the required steps to point out the computation cost of a parameter update versus inference.

Backprop consists of a *forward* and *backward* pass. In the forward pass, given a sample x_i we compute prediction $\hat{y}_i = f(x_i;\theta)$, as well as its loss $l_i = \mathscr{L}(\hat{y}_i, y_i)$. In the backward pass, we iteratively compute the parameters updates sequentially per each layer, while starting from the last layer to the first. Note that each layer l utilizes the computed updates from its following $l+1$–th layer, to update its parameters $\theta^{(l)}$.

2.2.4 Gradient based optimization

Optimization or *training* in deep learning refers to procedure where we start from initial guess for our model's parameters $f(\cdot;\theta)$, and iteratively update them using *backprop* (§ 2.2.3), so as to minimize a loss function $\mathscr{L}(\cdot)$, see § 2.2.2. For simplicity, in the following, we consider supervised single-objective convex optimization.

More precisely, given a finite set $\mathscr{D}$ of n annotated samples $\mathscr{D}[i] = (x_i, y_i)$, the training of the model's parameters θ aims at finding a vector θ^* which minimizes the (empirical) expectation:

$$\theta^* = \arg\min_{\theta} f(\cdot;\theta) = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} f_i(\theta), \tag{2.4}$$

where $f_i(\theta) := \mathscr{L}(\theta, (x_i, y_i))$, which we assume has L-Lipschitz-continuous gradients.

Batch gradient finds the global minimizer θ^* but each parameter update requires computing $f'(\theta)$ of the full n gradients. As this is prohibtive for large datasets, *Stochastic Gradient Descent* (SGD, Robbins and Monro [1951], Bottou [2010]) is a common alternative. Each parameter update of SGD requires sample (x_i, y_i) where i is chosen uniformly at random. SGD provides unbiased estimate of the gradient $f'(\theta)$ since $\mathbb{E} f_i'(\theta) = f'(\theta)$. Batch gradient descent has linear convergence rate, whereas SGD sub-linear rate of $O(1/t)$. Most popular alternative in deep learning is stochastic steepest descent where each update uses subsample $x_1, \dots, x_B \sim U(\mathscr{D})$, where $B \in [1, n]$ denotes the "mini-batch" size, often referred as *mini-batch SGD* (and typically $B \in [32, 512]$ due to hardware constraints).

The parameters are updated iteratively:

$$\theta = \theta - \eta \Big(\frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta} \mathscr{L}(\theta, \mathscr{D}[\sigma(i)]) \Big),$$

where η is the learning rate, σ is a random permutation of $[1, n]$) and $B = 1$, $B = \sigma$ or $B = N$ for SGD, mini-batch and batch gradient descent, respectively.

2.3 Generative models

The most developed deep learning models are the *discriminative* ones, which given data x aim at predicting its label y, hence modeling the posterior distribution $p(y|x)$. *Generative* models instead model the distribution $p(x)$ defined over the datapoints x, and depending on the type of the generative model we can either evaluate the probability assigned to each datapoint, or sample according to it. The dimensionality of x can be high, *e.g.* high resolution images, and the (implicitly) learned distribution assigns high probability to plausible (as given in the dataset) samples and low otherwise. Moreover, this approach allows for generating samples that do not necessarily exist in the training dataset.

Fig. 2.2 focuses on the generative models that are based on maximum likelihood (see § 2.1), and points out their main differences. The *explicit density* generative models explicitly model the distribution that describes the probability that the model assigns to each datapoint, controlled by parameters θ. These models can be further categorized as: (i) *exact* or *tractable*, and (ii) *approximate* models. An example of explicit density models are the *fully visible belief networks* [Frey, 1998], which use the chain rule to decompose a probability distribution over a vector x into a product: $p(x) = \prod_{i=1}^{n} p(x_i | x_1, \dots, x_{i-1})$. Examples of popular model of this class are *PixelCNN* [van den Oord et al., 2016] and *WaveNet* [van den Oord et al., 2016]. Approximate

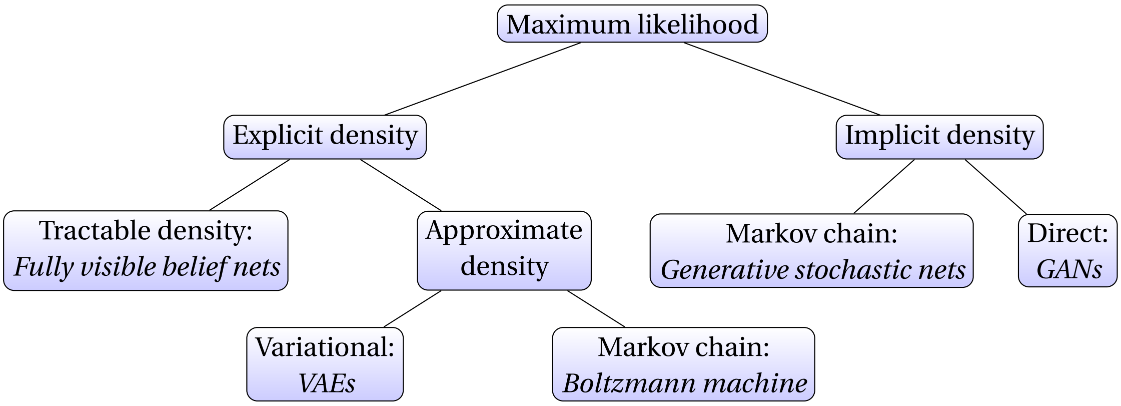


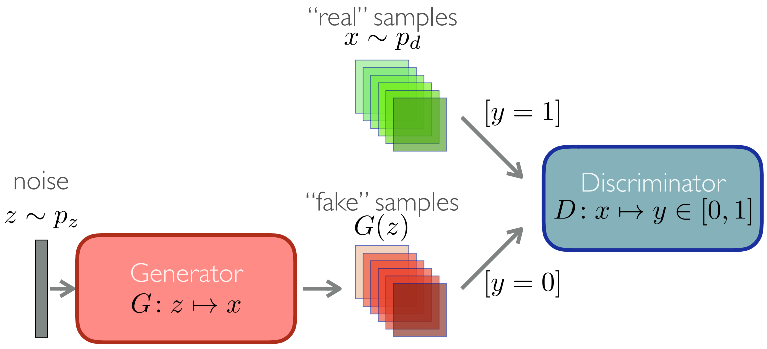
Figure 2.2 – Taxonomy of *generative models that are based on maximum likelihood* [Goodfellow, 2016]. *VAE* stands for Variational Autoencoders [Kingma and Welling, 2014].

models either: (i) maximize a lower bound of the log–likelihood, called *variational* models, or (ii) use Markov Chain to estimate the log–likelihood function or its gradient. A Boltzmann machine [Hinton et al., 2006] is a network of units that make stochastic decisions about whether to be on or off. They are defined by an energy function, and the probability of being in a particular state is proportional to the exponential of the value of the energy. To obtain a probability this value is normalized by dividing with the sum over all states which sum is approximated with Markov Carlo methods. Rather than calculating the likelihood, *implicit density models* instead provide a way to draw samples. This can be done using a Markov Chain, as it is done in *Generative Stochastic Networks*. Finally, we can have implicit density models from which we can directly obtain samples, such as GANs or deep moment–matching networks.

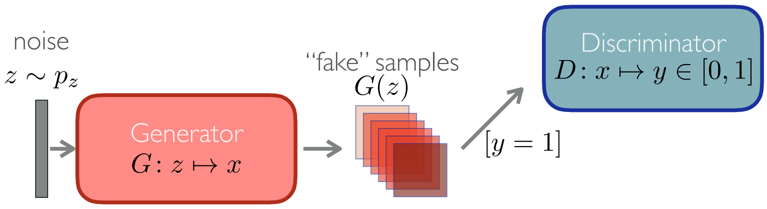
Variational Autoencoders (VAEs), proposed by Kingma and Welling [2014], are one of the most widely used generative models in deep learning. VAEs consist of an encoder and a decoder neural networks, where the encoder maps an input x to a lower-dimensional vector z of latent variables. VAEs use a variational approximation which introduces a parametric distribution $q_\lambda(z|x)$ over the latent variables z, and training aims at making the posterior distribution closer to the true posterior $p(z|x)$ over the latent variables, by using the Kullback–Leibler divergence between the two $\mathbb{D}_{KL}(q_\lambda(z|x)||p(z|x))$ as an additional loss.

2.4 Generative Adversarial Networks

Different from traditional generative models, a GAN generator represents a mapping $G\colon z \mapsto x$, such that if z follows a known distribution p_z, then x follows the distribution p_d of the data. Notably, this approach omits an explicit representation of $p_g(x)$ (see also § 2.3), or the ability to apply directly a maximum-likelihood maximization for training (see 2.1). This is aligned with the practical need, which is that we do not need an explicit formulation of $p_g(x)$, but rather a mean to sample from it, preferably in a computationally efficient manner. The training of



(a) Discriminator: mapping $D\colon x \mapsto y \in [0,1]$, where y is an estimated probability that $x \sim p_d$. The discriminator "distinguishes" *real* Vs. *fake* samples (the latter for example labelled as $y=0$).



(b) Generator: mapping $G\colon z \mapsto x$, such that if $z \sim p_z$, then $x \sim p_d$. Intuitively, the generator aims at "fooling" the discriminator that its samples are real, hence the label switches to $y=1$.

Figure 2.3 – A scheme of Vanilla–GAN training. With p_d and p_z we denote the "real" data distribution and the "noise" distribution, respectively, where the latter is a known distribution that we can sample from, *e.g.* $\mathcal{N}(0,1)$. Subfigures a and b depict the training of the discriminator and of the generator, respectively. Intuitively, an equilibrium for such a zero-sum game would be when $p_g = p_d$ and D outputs probability 0.5 for any input Goodfellow et al. [2014], see App. A.1.

the generator involves a discriminative model $D\colon x \mapsto y \in [0,1]$, whose output represents an estimated probability that x originates from the dataset.

More precisely, the algorithm consists of two training steps, see Alg. 1. Given that there is a probability 0.5 that the input x originates from the dataset and 0.5 that it was generated by G, the discriminator D is trained to distinguish between "real" and "fake" inputs, respectively. On the other hand, G is trained to "fool" D by generating synthetic samples indistinguishable from the real ones, as illustrated in Fig. 2.3.

Formally, the two competing models play the following two-player minimax–alternatively zero-sum–game:

$$\min_G \max_D \mathop{\mathbb{E}}_{x\sim p_d}[\log D(x)] + \mathop{\mathbb{E}}_{z\sim p_z}[\log(1-D(G(z)))]. \tag{2.5}$$

The two models are parametrized differentiable functions $G(z;\theta_g)$ and $D(x;\theta_d)$, implemented with neural networks, whose parameters θ_g and θ_d are optimized iteratively. In functional space, the competing models are guaranteed to reach a Nash Equilibrium, in particular under the assumptions that we optimize directly p_g instead of θ_g and that the two networks have enough capacity, see proof in App. A.1. At this equilibria point, D outputs probability 0.5 for any input, see App. A.1.

Algorithm 1 Pseudocode for *alternating* GAN.

```
Input: dataset $\mathscr{D}$, batch size $B$, known distribution $p_z$,
    stopping iteration $T$, learning rate $\eta$, generator loss $\mathscr{L}^G$,
    discriminator loss $\mathscr{L}^D$
Initialize: $D(\cdot;\theta_D)$, $G(\cdot;\theta_G)$
for $e = 0$ to $T - 1$ do
    $x_1 \dots x_B \sim U(\mathscr{D})$
    $z_1 \dots z_B \sim p_z$
    $\theta_D = \theta_D - \eta \nabla_{\theta_D} \mathscr{L}^D(D, G, \mathbf{x}, \mathbf{z})$
    $z_1 \dots z_B \sim p_z$
    $\theta_G = \theta_G - \eta \nabla_{\theta_G} \mathscr{L}^G(D, G, \mathbf{z})$
end for
Output: $\theta_G, \theta_D$
```

2.4.1 Evaluating GANs

Evaluating GANs is a difficult problem because it requires access to the distribution that generated our finite set of training samples, which for real–world datasets we do not have access to. In practice, besides that each has drawbacks two metrics are most commonly used by researchers: (i) the **Inception score** (IS, Salimans et al., 2016), and (ii) the **Fréchet Inception distance** (FID, Heusel et al., 2017).

Inception Score

Given an image x, the Inception Score (IS) uses the softmax output a pretrained network $p(y|x)$ on task with C classes, which represents the probability that x is of class $c_i, i \in 1 \dots C$, i.e., $p(y|x) \in [0,1]^C$. It then computes the marginal class distribution $p(y) = \int_x p(y|x) p_g(x)$. IS measures the Kullback–Leibler divergence $\mathbb{D}_{KL}$ between the predicted conditional label distribution $p(y|x)$ and the marginal class distribution $p(y)$. More precisely, it is computed as follows:

$$IS(G) = \exp\big(\mathbb{E}_{x\sim p_g}[\mathbb{D}_{KL}(p(y|x)||p(y))]\big) = \exp\big(\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{C} p(y_c|x_i)\log\frac{p(y_c|x_i)}{p(y_c)}\big). \tag{2.6}$$

It aims at estimating (i) if the samples look realistic i.e., $p(y|x)$ should have low entropy, and (ii) if the samples are diverse (from different classes) i.e., $p(y)$ should have high entropy. As these are combined using the Kullback–Leibler divergence, the higher the score is, the better the performance.

Fréchet Inception Distance

Contrary to IS, FID aims at comparing the synthetic samples $x \sim p_g$ with those of the training dataset $x \sim p_d$ in a feature space. The samples are embedded using the first several layers of the Inception network. It then estimates the means $\boldsymbol{m}_g$ and $\boldsymbol{m}_d$ and covariances C_g and C_d, respectively for p_g and p_d in that feature space. Finally, FID is computed as:

$$\mathbb{D}_{\text{FID}}(p_d, p_g) \approx d^2((\boldsymbol{m}_d, C_d), (\boldsymbol{m}_g, C_g)) = ||\boldsymbol{m}_d - \boldsymbol{m}_g||_2^2 + Tr(C_d + C_g - 2(C_d C_g)^{\frac{1}{2}}), \tag{2.7}$$

where d^2 denotes the Fréchet Distance and assuming that p_g and p_d are multivariate normal distributions. Note that as this metric is a distance, the lower it is, the better the performance.

Deep Generative Models Part I

3 SGAN: An Alternative Training of Generative Adversarial Networks

This chapter presents an alternative algorithm for training GANs, named “SGAN”. It differs from the standard training process due to the use of an ensemble of “local” adversarial pairs trained independently to help a “global” pair to be trained against them. By training the global pair in such a way we hope to improve mode coverage of the data distribution. A key feature of SGAN is to maintain statistical independence between the pairs of the ensemble. By doing this, we allow the different pairs of the ensemble to explore a different mode of the target distribution hence reducing the chances of the global pair being trapped in an unsatisfactory local minimum, or to face oscillations often observed in practice. To guarantee the latter, the global pair never affects the local ones.

We show that this novel training procedure outperforms standard methods by improved mode coverage, stability and convergence speed, on both artificial and real-world datasets.

The content of this chapter is related to the following publication:

- T. Chavdarova and F. Fleuret, *SGAN: An Alternative Training of Generative Adversarial Networks*, in Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR), 2018.

3.1 Introduction

An important research effort has recently focused on improving the convergence analysis of the Generative Adversarial Networks (GANs), proposed by Goodfellow et al. [2014] and described in § 2.4. This family of unsupervised learning algorithms provides powerful generative models, and have found numerous and diverse applications [Isola et al., 2017, Ledig et al., 2016, Nguyen et al., 2016, Zhang et al., 2017].

In practice, GANs are difficult to optimize, and practitioners have amassed numerous techniques to improve stability of the training process [see Radford et al., 2016]. However, the neglected inherited problems of the neural networks such as the lack of convexity, the numeri-

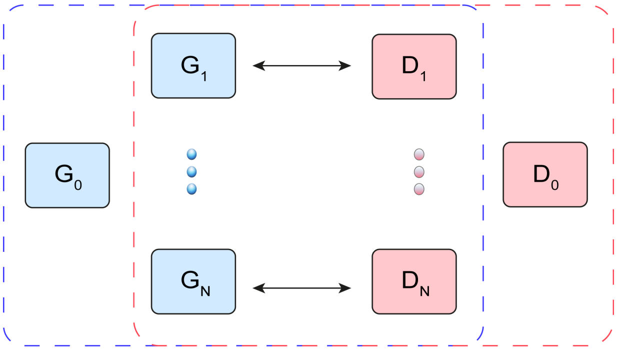


Figure 3.1 – Conceptual illustration of SGAN. There are $N+1$ pairs, of which the pair (G_0, D_0) is not trained directly. D_0 is trained with G_i, $i=1,\ldots,N$, and G_0 is trained with D_i, $i=1,\ldots,N$, as illustrated with the dashed line rectangles.

cal instabilities of some of the involved operations, the limited representation capacity, as well as the problem of vanishing and exploding gradients often emerge in practice.

Current state-of-the-art GAN variants [Arjovsky et al., 2017, Gulrajani et al., 2017] eliminate gradient instabilities, which mitigated the discrepancy between the theoretical requirement that D should be trained up to convergence before updating G, and the practical procedures of vanilla GAN for which this is not the case. These results are important, as vanishing or exploding gradients results in G to produce samples of noise.

However, oscillations between noisy patterns and samples starting to look like real data while the algorithm is converging, as well as failures of capturing p_d, are not resolved. In addition, in practical applications, it is very difficult to assess the diversity of the generated samples. “Fake” samples may look realistic but could be similar to each other – indicating that the modes of p_d have been only partially “covered” by p_g. This is a problem that arises and is referred as *mode collapse*. As a result, a golden rule remains that one does multiple trials of combinations of hyperparameters, architectural and optimization choices, and variants of GANs. As under different choices the performances vary [see large–scale empirical study by Lucic et al., 2017], this is followed by tedious and subjective assessments of the quality of the generated samples in order to select a generator.

As a summary, what made GAN distinctly powerful is the opponent-wise engagement of two networks belonging to an already outperforming class of algorithms. Such a framework–the discriminator being a deep neural network–allows for time-efficient training of the generative model and directly formulates what we aim at–to generate samples that resemble those we have, in contrast to memorizing these. Intuitively, the more we enforce constraints, either architectural or functional, the better the coherency of the learning dynamics. On the other hand, this may result in reduced sample quality or increased convergence time at the minimum. A question arises if we can improve training stability, guarantees of “successful” training, and performances, without imposing restrictions on the architectures of G or D.

We propose a novel way of training a *global* pair (G_0, D_0), such that the optimization process will make use of the "flow of information" generated by training an ensemble of N adversarial pairs $(G_1, D_1), \ldots, (G_N, D_N)$, as sketched in Fig. 3.1.

The rules of this game are as follows: G_0 and D_0 can solely be trained with $\{D_1, \ldots D_N\}$ and $\{G_1, \ldots G_N\}$, respectively, and local pairs do not have access to outputs or gradients from G_0 and D_0.

The most prominent advantages of such a training are:

1. if the training of a particular pair degrades or oscillates, the global networks continue to learn with higher probability;
2. it is much more likely that training one pair will fail than training all of them, hence the choice of not letting global models to affect the ensemble;
3. if the models' limited capacity is taken into account *i.e.* p_g can capture a limited number of modes of p_d (which increases with the number of training iterations), and under the assumption that each mode of p_d has a non-zero probability of being captured, then the modeled distribution by the ensemble is closer to p_d in some metric space due to the statistical averaging; and conveniently
4. large chunks of the computation can be carried out in parallel making the time overhead negligible.

In what follows, we first review in § 3.2 GAN variants we use in the experimental evaluation of our SGAN algorithm, which to the best of our knowledge are the current state-of-the-art methods. We then describe SGAN in detail in § 3.3, and present thorough experimental evaluation in § 3.4 as well as in App. B. We then overview GAN methods that use multiple discriminators or generators in § 3.5.

3.2 Related work: Variants of the GAN algorithm

Optimizing Eq. 2.4 amounts to minimizing the Jensen-Shannon divergence between the data and the model distribution $JS(p_d, p_g)$ [Goodfellow et al., 2014]. More generally, GANs learn p_d by minimizing a particular f-divergence between the real samples and the generated samples [Nowozin et al., 2016].

With a focus on generating images, Radford et al. [2016] propose specific architectures of the two models, named Deep Convolutional Generative Adversarial Networks–**DCGAN**. Radford et al. [2016] also enumerate a series of practical guidelines, critical for the training to succeed. Up to this point, when the architecture and the hyper-parameters are empirically selected, DCGAN demonstrates outperforming results both in terms of quality of generated samples and convergence speed.

To ensure usable gradient for optimization, the mapping $\theta_d \mapsto p_d$ should be differentiable, and to have a non-zero gradient everywhere. As the JS divergence does not take into account the Euclidean structure of the space, it may fail to make the optimization move distributions closer to each other if they are "too far apart" [Arjovsky et al., 2017]. Hence, Arjovsky et al. [2017] suggest the use of the Wasserstein distance, which precisely accounts for the Euclidean structure. Through the Kantorovich Rubinstein duality principle [Villani, 2008], this boils down to having a K-Lipschitz discriminator.

From a purely practical standpoint, this means that strongly regularizing the discriminator prevents the gradient from vanishing through it, and helps the optimization of the generator by providing it with a long-range influence that translates into a non-zero gradient.

In **WGAN** [Arjovsky et al., 2017] the Lipschitz continuity is forced through weight clipping, which may make the optimization of D harder–as it makes the gradient with respect to D's parameters vanish–and often leads to degrading the overall convergence. It was later proposed to enforce the Lipschitz constraint smoothly by adding a term in the loss which penalizes gradients whose norm is higher than one–**WGAN-GP** [Gulrajani et al., 2017].

Motivated by game theory principles, Kodali et al. [2017] derive combined solution of vanilla GAN and WGAN with gradient penalty. In particular, the authors aim at smoothing the value function via regularization by minimizing the regret over the training period, so as to mitigate the existence of the multiple saddle points. Finally, while building on vanilla GAN, the proposed algorithm named **DRAGAN**–Deep Regret Analytic GAN–forces the constraint on the gradients of $D(x)$ solely in local regions around real samples.

Algorithm 2 Pseudocode for SGAN.

```
Input: $\mathscr{X}_{inf}$, $N$, I, $I_D$.
$\mathscr{G}, \mathscr{D}$ = init($N$)
$G_0, D_0$ = init(1)
for $i = 1$ to $I$ do
    for $n = 1$ to $N$ do
        for $j = 1$ to $I_D$ do
            zeroGradients($\mathscr{D}[n]$)
            backprop($\mathscr{G}[n]$,$\mathscr{D}[n]$,$\mathscr{X}_{inf}$)
            updateParameters($\mathscr{D}[n]$)
        end for
        zeroGradients($\mathscr{G}[n]$)
        backprop($\mathscr{G}[n]$,$\mathscr{D}[n]$,$\mathscr{X}_{inf}$)
        updateParameters($\mathscr{G}[n]$)
    end for
    $\mathscr{D}^{msg}$ = copy($\mathscr{D}$)
    for $n = 1$ to $N$ do
        for $j = 1$ to $I_D$ do
            zeroGradients($\mathscr{D}^{msg}[n]$)
            backprop($G_0$,$\mathscr{D}^{msg}[n]$,$\mathscr{X}_{inf}$)
            updateParameters($\mathscr{D}^{msg}[n]$)
        end for
    end for
    zeroGradients($G_0$)
    for $n = 1$ to $N$ do
        backprop($G_0$,$\mathscr{D}^{msg}[n]$,$\mathscr{X}_{inf}$)
    end for
    updateParameters($G_0$)
    zeroGradients($D_0$)
    for $n = 1$ to $N$ do
        backprop($\mathscr{G}[n]$, $D_0$,$\mathscr{X}_{inf}$)
    end for
    updateParameters($D_0$)
end for
Output: $G_0, D_0$.
```

3.3 Method

Structure. We use a set $\mathscr{G} = \{G_1, \dots G_N\}$ of N generators, a set $\mathscr{D} = \{D_1 \dots D_N\}$ of N discriminators, and a global generator-discriminator pair (G_0, D_0), as sketched in Fig. 3.1.

Summary of a simplified-SGAN implementation. The pairs (G_n, D_n), $n = 1, \dots N$ are trained individually in a standard approach. In parallel to their training, D_0 is optimized to detect samples generated by any of the local generators $G_1, \dots, G_N$, and similarly G_0 is optimized to fool all of the local discriminators $D_1, \dots, D_N$.

Note that, to satisfy the theoretical analyses of minimizing the Wasserstein distance and the Jensen-Shannon divergence for WGAN and GAN, respectively, the above procedure of training implies that each $\{D_1 \dots D_N\}$ *should be trained with* G_0 at each iteration of SGAN. Solely by following such a procedure G_0 follows the principles of the GAN framework [Goodfellow et al., 2014], which trains it with gradients "meaningful" for it.

Introducing "messengers" discriminators for improved guarantees. To prevent that one of the network pairs "influences" the ensemble, and thus keep the guarantees of successful training, we propose to train G_0 against herein referred as "messengers" discriminators $D_1^{msg}, \dots, D_N^{msg}$, which at re-created at every iteration as clones of $D_1, \dots, D_N$, optimized against G_0.

We empirically observed that this strategy helps consecutive steps to be more coherent, and improves drastically the convergence. It is worth noting that, despite the increased complexity in terms of obtaining the theoretical analyses, such an approach is practically convenient since it allows for training G_0 in parallel to the local pairs.

3.3.1 Description of SGAN

More formally, let $\mathscr{X}_{inf}$ be a sampling operator over the dataset, which provides mini-batches of i.i.d. samples $x \sim p_d$.

Let *backprop* be a function that given a pair G and D, buffers the updates of the networks' parameters, *updateParameters* that actually updates the parameters using these buffers, and *zeroGradients* resets these buffers. Also, let *init* be a function that initializes a given number of pairs of G and D. Let N be the number of pairs to be used. The algorithm iterates for a given number of iterations I, and depending on the used GAN variant, each discriminator network is updated either once or several times, hence the I_D input parameter.

At each iteration, foremost the local models are being updated (lines 5 - 14).

Then, to obtain meaningful gradients for G_0, without affecting the local models, we first make a copy of the latter (line 15) into the "messenger discriminators" $\mathscr{D}^{msg}$, and update them against G_0 (lines 16 - 22). We then update G_0 jointly versus all of the discriminators (lines 23 - 27).

As D_0 does not affect generators it is trained with, it is directly updated jointly versus all of the local generators (lines 28 - 32).

Note that for clarity in Alg. 2 we present SGAN sequentially. However, each iteration of the training can be parallelized since G_0 is trained with a copy of $\mathscr{D}$, and the local pairs can be trained independently. In addition, Alg. 2 can be used with different GAN variants.

SGAN can also be implemented with weight-sharing (see § 3.4.1) since low-level features can be learned jointly across the networks. This motivates the training of D_0 in Alg. 2. In addition, whether the discriminator can be made use of is not a closed topic. In fact, a some works answer in the affirmative [Lai et al., 2017].

3.4 Experiments

Datasets. As toy problems in $\mathbb{R}^2$ we used (i) mixtures of M Gaussians (M-GMM) whose means are regularly positioned either on a circle or a grid, with $M = 8, 10$, or 25, and (ii) the classical Swiss Roll toy dataset [Marsland, 2009]. In the former case, we manually generate such datasets, by using a mixture of M Gaussians with modes that are uniformly distributed in a circle or in a grid. With such an evaluation, we follow related works–for *e.g* [Gulrajani et al., 2017, Kodali et al., 2017, Tolstikhin et al., 2017] since GANs in prior work often failed to converge even on such simplistic datasets [Metz et al., 2017].

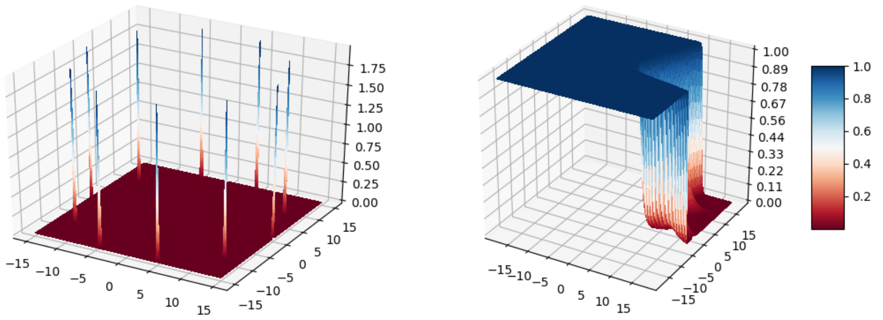
To assess SGAN or real world applications, we used:

1. small scale datasets: CIFAR10 [Krizhevsky, 2009, chapter 3], STL-10 [Coates et al., 2011], MNIST [Lecun and Cortes, 1998], as well as the recent FASHION-MNIST [Xiao et al., 2017];
2. large scale datasets: CelebA [Liu et al., 2015], LSUN [Yu et al., 2015] using its “bedroom” class, and ImageNet [Russakovsky et al., 2015]; as well as
3. large language corpus of text in English, known as One Billion Word Benchmark [Jozefowicz et al., 2016].

Methods. As WGAN with gradient penalty [Gulrajani et al., 2017] outperformed WGAN with weight clipping [Arjovsky et al., 2017] in our experiments, herein as “WGAN” we refer to the former. Regarding vanilla GAN, instead of minimizing $\log(1-D(G(z)))$, we train G to maximize $\log(D(G(z)))$, as it is recommended by Goodfellow et al. [2014], and done in practice [Goodfellow et al., 2014, Radford et al., 2016]. For conciseness, let us adopt the following notation regarding SGAN: we prefix the type of GAN with N-S, where N is the number of local pairs being used. For example, SGAN with 5 WGAN local pairs and one global WGAN pair would be denoted as 5-S-WGAN.

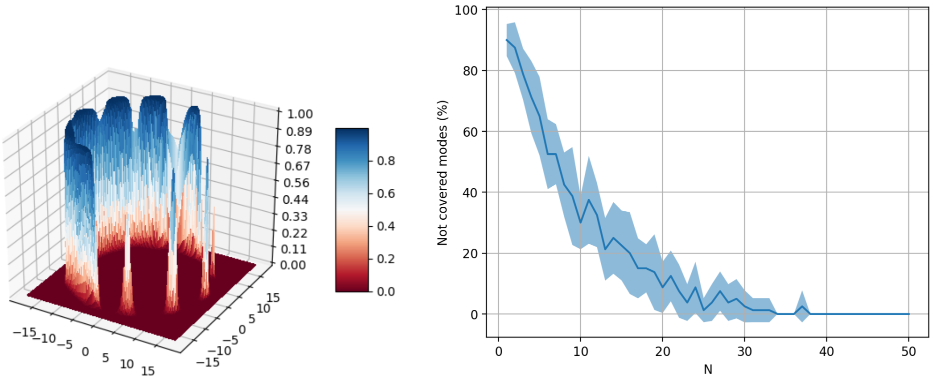
Implementation. For experiments on toy datasets, we used separate $2\times(N+1)$ neural networks. Regarding experiments on real-world datasets, we experimented with two implementations: using separate networks, as well as sharing parameters. In the latter case, we used approximately half of the parameters to be shared among the generators, and analogously same quantity among the discriminators. For further details on our implementation, see App. B.

As a deep learning framework we used PyTorch [Paszke et al., 2017].



(a) Real data (10-GMM)

(b) Discriminator output



(c) S-Discriminator output

(d) Not-covered modes (%)

Figure 3.2 – Figures (a-c) depict a toy experiment with vanilla GAN. Figure (d) depicts the percentage of not covered modes (y-axis) by the generators, as more pairs are used (x-axis). See text for details, § 3.4.1.

Metrics. A serious limitation to improve GANs is the lack of a proper means of evaluating them [Lucic et al., 2017]. When dealing with images, most commonly used measure is the **Inception score** (IS), proposed by Salimans et al. [2016], described in § 2.4.1.

Using real data images of ImageNet, LSUN-bedroom, CIFAR10, and CelebA, we obtain the following Inception scores: 46.99 (3.547), 2.37 (0.082), 10.38 (0.502), 2.50 (0.082), respectively. The high variance across the datasets, suggests that training the model on the dataset at hand may improve the estimate. Hence, we adopt it as is for CIFAR10–as it turned into a standard GAN metric, whereas for MNIST we use a classifier specifically trained on this dataset. In the former case, we use the original implementation of it [Salimans et al., 2016] and a sample of

p_g of size $50{\cdot}10^3$, whereas for the latter we use our own implementation in PyTorch [Paszke et al., 2017].

The **Fréchet Inception Distance** (FID) [Heusel et al., 2017] uses the Inception model [Szegedy et al., 2015] to embed samples into a "good" feature space. We describe this metric in detail in § 2.4.1.

For experiments on MNIST, using a separately trained classifier, we also plot the **entropy** of the generated samples' mode distribution, as well as the **total variation** between the class distribution of the generated samples and a uniform one. For toy experiments on mixtures of Gaussians, we also used the log-likelihood of the generated samples.

For more results and details on our implementation, see the App B.

3.4.1 Experimental results on toy datasets

Independently trained ensemble of GANs. To motivate the idea of favoring information from the independent ensemble to train a single pair, we conduct the following experiment. We train in parallel few pairs of networks, as well as *two* additional pairs: (i) **SGAN**–trained with the local independent pairs, as well as (ii) **GAN**–a regularly trained pair. In addition to training these two pairs with equal frequency, we used the *identical real-data and noise samples*.

Fig. 3.2 depicts vanilla-GAN experiment on the 10-GMM dataset (Fig. 3.2a). We recall that the only difference between the two discriminators is that the GAN discriminator is trained with fake samples from his tied single opponent (Fig. 3.2b), whereas the one of SGAN is trained with fake samples from the ensemble (Fig. 3.2c).

Fig. 3.2d depicts that the probability that a mode will not be covered (y-axis) by the ensemble, at a random iteration, goes down exponentially with the number of pairs (x-axis). For this experiment, we used the 8-GMM toy dataset and the vanilla-GAN algorithm.

Performance of the global pair in SGAN. In Fig. 3.3 we use WGANs, where each network is a multilayer perceptron (MLP) of 4 fully connected layers (see App. B.1). The first column depicts the 10 local pairs: generators' samples and discriminators' contours (level sets) are displayed in varying and transparent colors, respectively. The rightmost column depicts the 10-S-WGAN pair (trained with the networks of the first column): samples from G_0 are drawn in green, whereas the illustrated contours are from D_0. We observe that S-WGAN exhibits higher stability and faster convergence speed. Fig. 3.3 also depicts samples from a generator updated N-fold times more (middle column), what indicates that an SGAN generator is comparable with these.

Fig. 3.4 depicts 10-S-WGAN experiment on the 8-GMM toy dataset. At each iteration, after

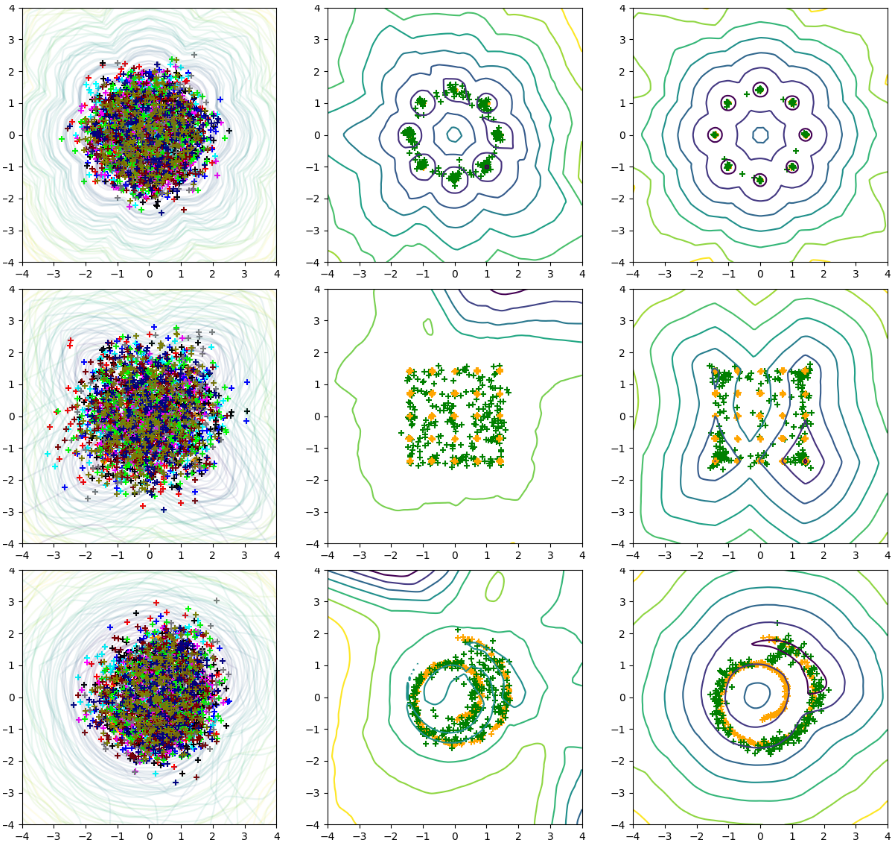


Figure 3.3 – (10-S-)WGAN on (top to bottom row): circle 8-GMM, grid 25-GMM, Swiss Roll (best seen in color). Real data-points are shown in orange. The level sets of the output of the discriminator(s) are shown with yellow to purple contours which denote low and high, respectively. See § 3.4.1 for more details.

training the local pairs, the global generator is trained with the local discriminators, samples of which are displayed on the left and right columns, respectively. We observe that in the earlier iterations samples from the global generator may lie in $\mathbb{R}^2$ regions distinct from the real data samples. Nonetheless, it converges notably quicker, and through the early iterations, its generated samples lie in regions which often do not intersect with those of the local generators (jointly) at the same iteration.

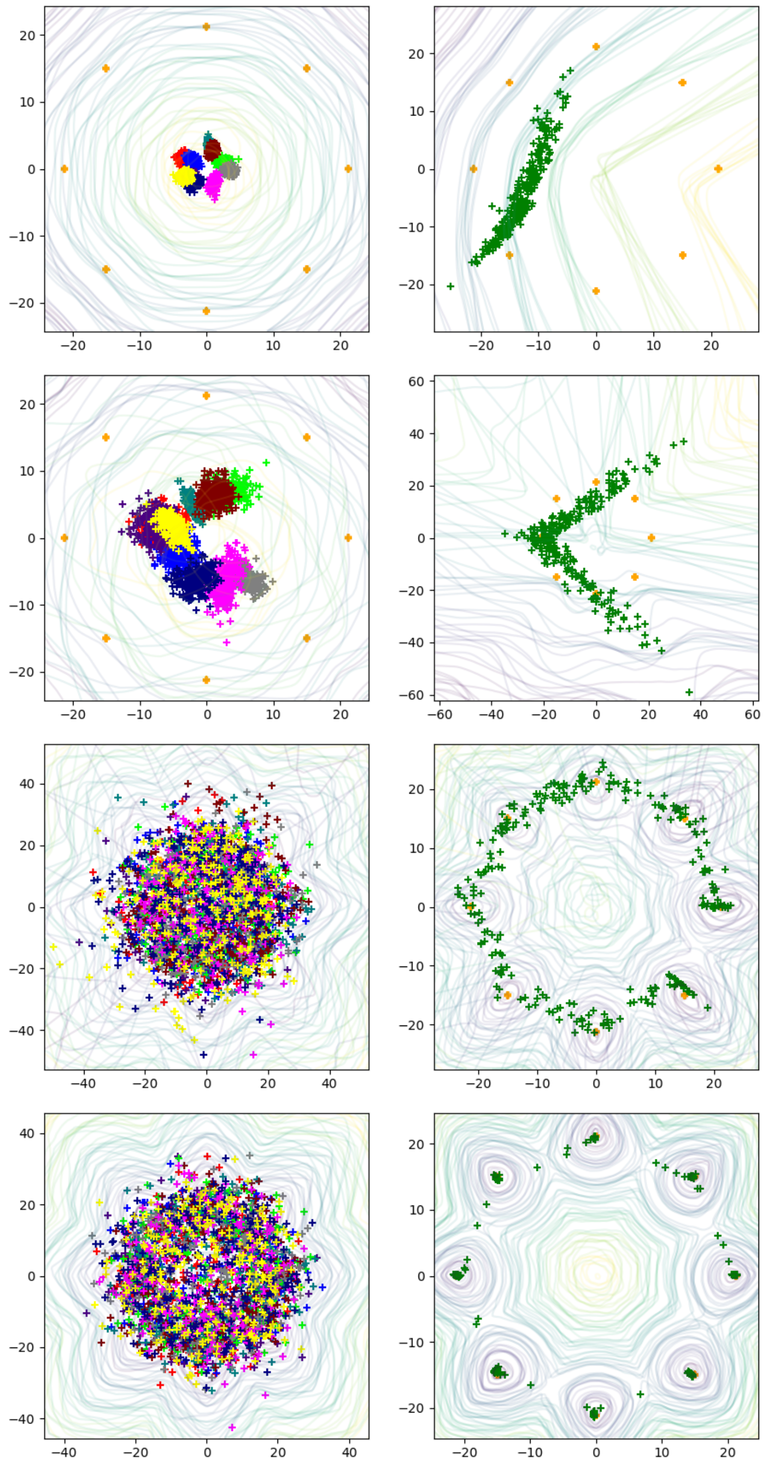


Figure 3.4 – 10-S-WGAN on the 8-GMM toy dataset (best seen in color). Samples from p_d are displayed in orange. Each row is a particular iteration (top to bottom): 5th, 10th, 100th, and 400th iteration. Samples from the local generators and the global one are illustrated on the left (in separate colors) and right (in green), respectively. The displayed contours represent the level sets of $\mathscr{D}$ and $\mathscr{D}^{msg}$–illustrated on the left and right, respectively, where yellow is low and purple is high.

Value of N. Fig. 3.5 depicts the log-likelihood on 8-GMM for different values of N, more precisely the log-probability of the generated samples, under the real-data distribution (8-GMM). “Simplified-SGAN” denotes the SGAN variant without the messengers discriminators (see § 3.3). We observe that increasing N helps, but that the trade-off performance gain versus computation resources starts to saturate compared to the gain obtained of SGAN relative to regular training. Moreover, we observe that “Simplified-SGAN” does not improve notably the performances of the WGAN baseline, which empirically justifies the approach of maintaining statistical independence of the local adversarial pairs.

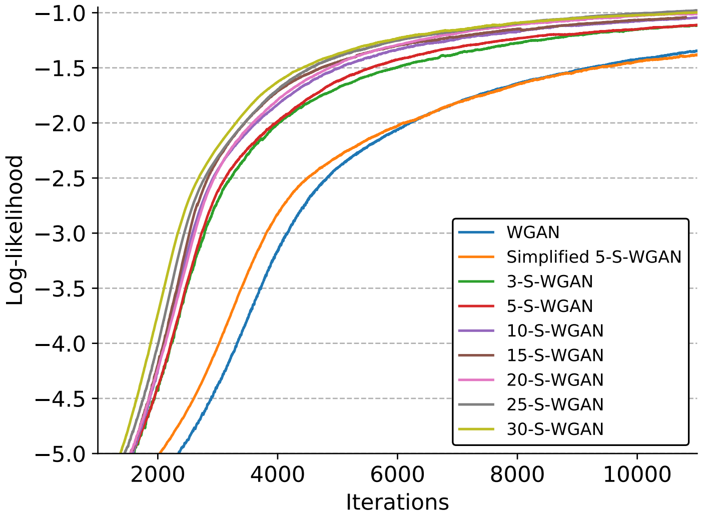
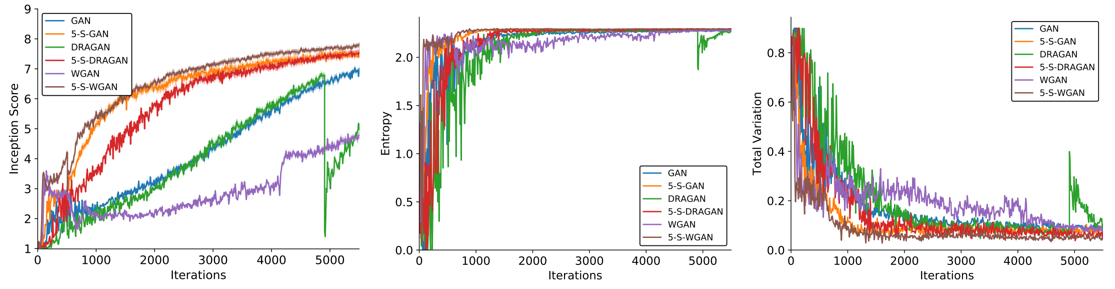


Figure 3.5 – Log-likelihood on 8-GMM toy dataset (see § 3.4.1).

3.4.2 Experimental results on real-world datasets



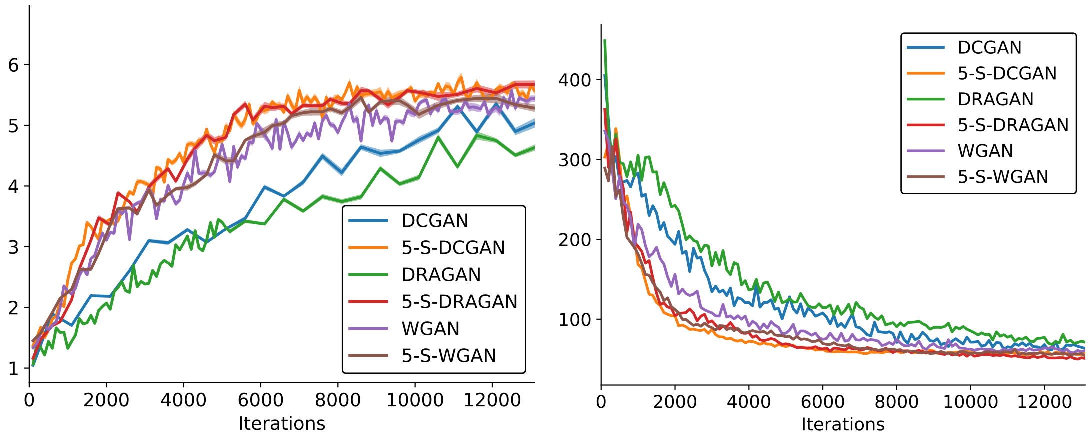
(a) IS [131] (higher is better) (b) Entropy (higher is better) (c) Total variation (lower is better)

Figure 3.6 – Results on MNIST using (5-S-) GAN/WGAN/DRAGAN (best seen in color).

In Figures 3.6 and 3.8 we show experimental results on image datasets. In the latter, samples are taken at a random iteration, *prior to final convergence*, so as the difference in the quality of the samples is clearer. Fig. 3.7 depicts quantitative results on CIFAR10 using the Inception Score (Fig. 3.7a) and the Fréchet Inception Distance (Fig. 3.7b), under identical hyperparameter setup and architecture choice (see App. B.2 for details on our implementation).

In Table 3.1 we show fake samples of the global generators of S-WGAN on the One Billion Word Benchmark dataset. The output of a standard WGAN training is *a single character (white space) for all of the first 660 iterations*, what indicates slower than N-fold convergence speed compared to an N-S-Generator. In addition, it is interesting to observe similar behavior as on toy datasets: at the first iteration (first row in Table 3.1) the SGAN generators are pushed far from the modes of the real data samples, as they generate non-commonly used letters.

Fig. 3.9 depicts samples taken from each generator in 5-S-DCGAN at the $100{\cdot}10^3$-*th* iteration, as well as samples from the generator of a separately trained *DCGAN* pair at $100{\cdot}10^3$-*th* and $500{\cdot}10^3$-*th* iterations. Besides that the global generator of SGAN shows no visible mode collapse for the human eye (compared to samples taken from the rest of the generators), we also observed that its performance did not oscillate through the iterations.



(a) IS [Salimans et al., 2016] (higher is better) (b) FID [Heusel et al., 2017] (lower is better)

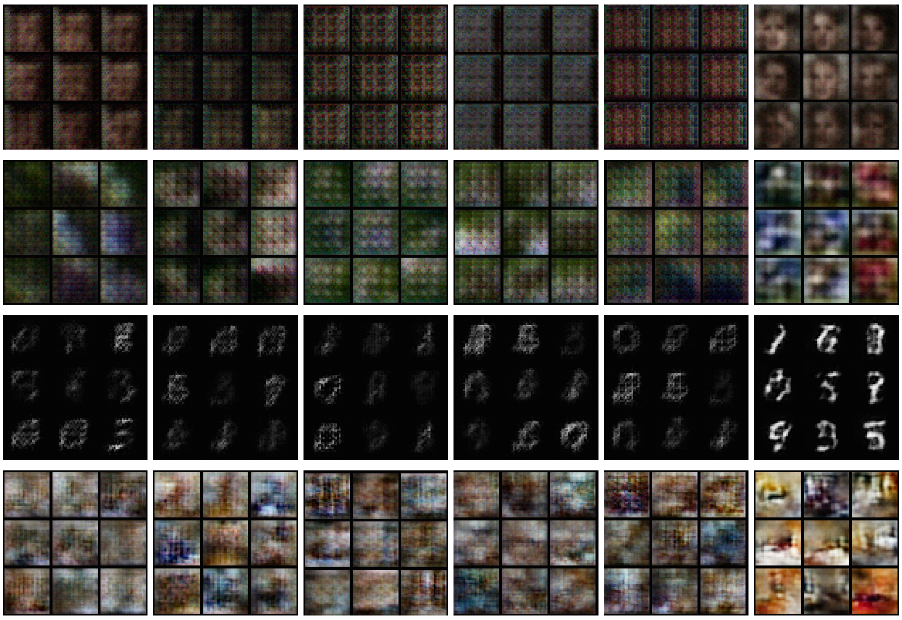
Figure 3.7 – 5-SGAN comparison on CIFAR10 using the number of iterations as the x-axis. Note that the x-axis is *not* normalized with the number of passes and instead depicts the performance gain (in terms of IS–left, and FID–right) if SGAN is implemented in parallel.

3.5 Related work: Multi-network GAN methods

Independently, Boosted Generative Models [Grover and Ermon, 2017] and AdaGAN [Tolstikhin et al., 2017] propose the iterative boosting algorithm to solve the mode collapse problem. At each step, a new component is added into a mixture of models, by updating the samples' weights, while using the vanilla GAN algorithm.

With the similar motivation of increasing the mode coverage Ghosh et al. [2017] and Hoang et al. [2017] propose to instead train multiple generators versus a single discriminator.

In [Ghosh et al., 2017] the discriminator is trained against N generators which share parameters



(a) G_1 (b) G_2 (c) G_3 (d) G_4 (e) G_5 (f) G_0

Figure 3.8 – **5-SGAN**. In the top to bottom rows we use DCGAN on CelebA, DRAGAN on ImageNet, WGAN on MNIST and DCGAN on LSUN, respectively. Each of the above samples are taken at the earlier iterations, in particular at the 100-*th*, 500-*th*, 500-*th* and the 1000-*th* iteration, respectively for each row. In columns (a-e) we show samples from the local generators, whereas in (f) from the global generator. We used separate networks and real data space of 32×32.

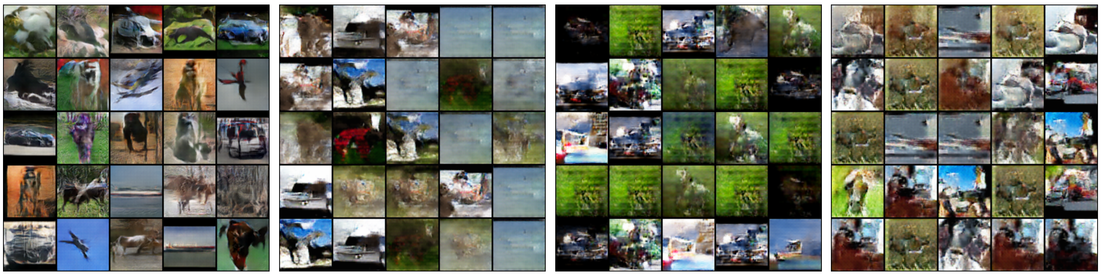
Table 3.1 – Output snippets of the global generators trained on the One Billion Word Benchmark. The top to bottom rows depict the 1-*st*, 100-*th* and the 200-*th* iteration. See § 3.4.2.

5-S-WGAN	10-S-WGAN

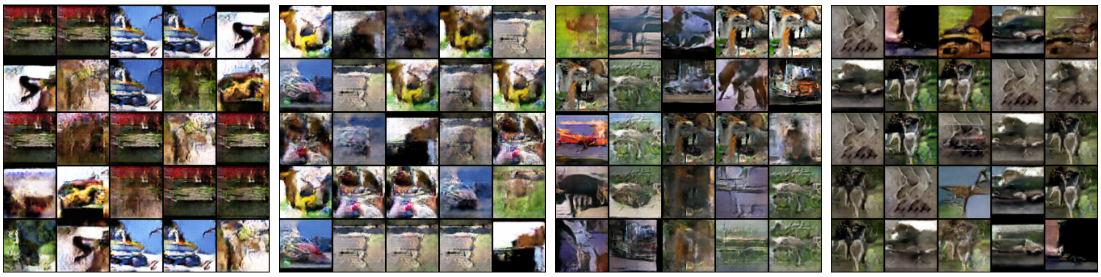
`aaa aaaaaaaaaaaaaa aaaa aaaaaaaa`
`a   aaaaaa aaaaaaaa aaaaa aaaaaa`
` aaaaaa a aaaaaaaaa aaaaa aaaaaa`
` aaa aaaaaaaa aaaaaaaaaaaaaaaa a`
`a aaaaaaaaaaa aaaaaaaaa aaaaaaaa`
`aaaaaa  aaaa a aaaaaaaaaa aaaaaa`
`a  aaa  aa aaaa aaaaaaaaaaaaaaa`
	`ii iii   iiiii   iiiii  i iiiii`
	`ii  ii iiiii ii iiii i iiiii ii`
	`iiiii ii ii iii ii iiii i   iiii`
	`iiii iii i  i  iiii iiiiii  i`
	`iii iiiii   i  iii i i ii i iii`
	`ii ii     iii i iii ii ii iii ia`
	`iiiiii   i i  iiiii     i i i ii`

`hieq  as ieq aa  dhhie  as ie  t`
`eq shiq as heq aaa hheq  asheq t`
`ieqq aasheq dsheq aa dd dhhie  t`
`iq ddq as ie  sie  ashieq as e t`
`eq as hheq  aas ie  s heq as e t`
`q hheq aas ieq dshieq as hie  at`
`eq asid ddd as hieq as heq diq t`
	`SAeS Aer areS SnSSSharSonS Soe`
	`AS SSSSer oeS SarSonSSS Ss ShS i`
	`S tes SrSsne SoerhaS SsnS Soar o`
	`MSSSS S Sha tos ShS aoS as Sha i`
	`s She tAeS SsS SsnSSSoes ShS Son`
	`eS es S SoS Ss SoSSS Ss SSS Son`
	`ASSSS tSa SoarSonS Ssne Soar osn`

in all layers except the last one, and it outputs probability estimate for $N+1$ classes representing whether the input is a real sample, or by whom of the generators it originates. To enforce diversity between the generated samples, a penalty term is added with a user-defined similarity



Global generator *Local generator #1* *Local generator #2* *Local generator #3*



Local generator #4 *Local generator #5* *DCGAN,* $100{\cdot}10^3$ *iter.* *DCGAN,* $500{\cdot}10^3$ *iter.*

Figure 3.9 – Samples of the generators of 5-S-DCGAN on the STL-10 dataset at the $100{\cdot}10^3$-*th* iteration (rows 1–3), as well as separately trained DCGAN at the $100{\cdot}10^3$ and $100{\cdot}10^3$-*th* iteration (bottom row). Using 64×64 data space.

based function.

Similarly, Hoang et al. [2017] propose multiple generators that share parameters versus single discriminator whose output is fake versus real, as well as training an additional model that classifies by whom of the generators a given fake input was generated. The output of the classifier is used in an additional penalty term that forces diversity between the generators. Durugkar et al. [2017] proposesutilizing multiple discriminators versus one generator, in aim to stabilize the training.

Ghosh et al. [2016] propose multiple generators versus single discriminator, where the generators communicate through two types of messages. Namely, there are both co-operation and competing objectives. The former ensures the other generator to generate images better than itself, and the latter encourages each generator to generate better samples than its counterpart. Motivated by the observed oscillations, in [Wang et al., 2016] a so-called “self-ensembles” is proposed. Non-traditionally, this self-ensemble is built out of copies of the generator taken from a different iteration while training a single pair.

Hence, SGAN depicts different structures and solutions to the problem of training GANs. Regarding the former, none of the above methods utilizes explicitly multiple pairs trained independently. Instead, most commonly a structure of one-to-many is used, either for the generator or for the discriminator. Compared to AdaGAN, SGAN is applicable to any GAN variant, runs in parallel, and produces a single generator. Concerning the latter, SGAN uses “supervising” models and prevents an influence of one pair towards all.

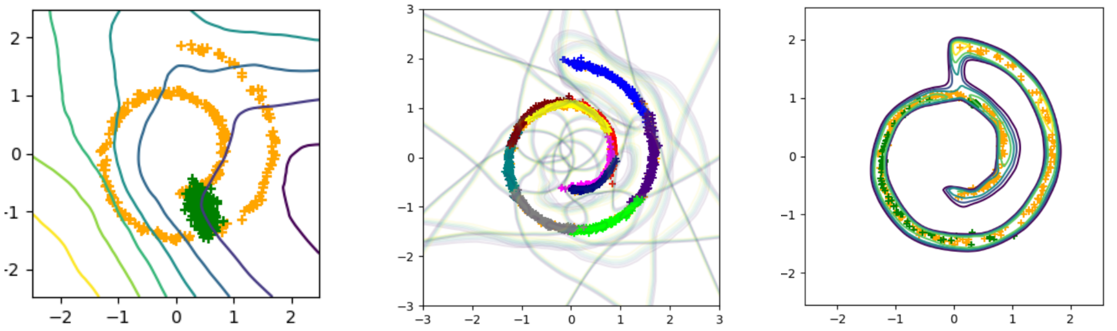
3.6 Discussion and future directions

We proposed a general framework dubbed SGAN for training GANs, applicable to any variant of this algorithm. It consists of training several adversarial pairs of networks independently and uses them to train a global pair that combines the multiple learned representations.

A key idea in our approach is maintaining the statistical independence between the individual pairs, by preventing any flow of information between them, in particular through the global pairs it aims at training eventually. Maintaining this makes the probability of a failure to go down exponentially with the number of pairs involved in the process.

The motivations of such a training methodology originate from the discrepancy between the theoretical justifications being derived in functional space, and the fact that we optimize the parameters of the deep neural networks [Goodfellow et al., 2014]. More precisely, training the generator finally produced by SGAN in such a way aims at addressing if the limited representational capacity (more prominent at the early iterations) affects the trajectories taken during the optimization procedure, which could itself be an important factor causing the training difficulties. The presented empirical evaluation indicates that it is indeed the case, as it was shown that such a training follows different trajectories and that for realistic datasets, it eliminates oscillations observed with a standard training under an identical set-up.

Experimental results on diverse datasets demonstrate systematic improvements upon classical algorithms as well as increased stability of the framework regarding real-world applications. Furthermore, SGAN is convenient for many applications, as it produces a single generator.



(a) Single pair training, real and fake samples are shown in yellow and green, resp.

(b) Local pairs of SGAN, fake samples are shown in varying colors per generator.

(c) SGAN pair, real and fake samples are shown in yellow and green, resp.

Figure 3.10 – Toy experiment of an extension of SGAN on the Swiss Roll dataset, where we enforce that the local generators model different distributions, see text § 3.6. All samples were taken at an equal number of iterations. The lines in Fig. 3.10b depict the contour lines of the local discriminators of SGAN (made transparent, for clarity of the illustration).

One possible future extensions of SGAN is improving the covering behavior by enforcing that the modeled distributions of the local pairs focus on disjoint parts of the support of the real data distribution p_d. We illustrate this idea in Fig. 3.10 where we use an additional classifier C, trained to classify from which generator a sample comes from. More precisely, given a sample $x \sim p_{g_i}, i \in \{1, \ldots, N\}$ of a local generator G_i, the classifier $C : x \mapsto \mathbb{R}^N$ aims at predicting i. Apart from fooling D_i each local generator G_i is trained to reduce the error of the classifier C.

4 Reducing the Noise of the Stochastic Gradient-based GAN Optimization

In this chapter we focus on the optimization algorithm used to train GANs. Motivated by the empirically observed reduced variance of SGAN and also the improved performances as the batch size increases [Chavdarova et al., 2018, Brock et al., 2019], we investigate the impact of noise of the stochastic gradient to GAN training. We show that it can prevent the convergence of standard game optimization methods, while the batch version converges. We address this issue with two *stochastic variance-reduced gradient* and *extragradient* optimization algorithms for GANs, named *SVRG-GAN* and *SVRE*, respectively.

Related publications:

- T. Chavdarova, G. Gidel, F. Fleuret and S. Lacoste-Julien, *Reducing Noise in GAN Training with Variance Reduced Extragradient*, in Proceedings of the international conference on Neural Information Processing Systems (NeurIPS), 2019.
- T. Chavdarova, S. Stich, M. Jaggi and F. Fleuret, *Stochastic Variance Reduced Gradient Optimization of Generative Adversarial Networks*, in Proceedings of Theoretical Foundations and Applications of Deep Generative Models Workshop, ICML workshop, 2018.

4.1 Introduction

Many empirical risk minimization algorithms rely on gradient-based optimization methods. These iterative methods handle large-scale training datasets by computing gradient estimates on a subset of it, a *mini-batch*, instead of using all the samples at each step, the *full batch*, resulting in a method called *stochastic gradient descent* (SGD, Robbins and Monro [1951], Bottou [2010]).

SGD methods are known to efficiently minimize *single* objective loss functions, such as cross-entropy for classification or squared loss for regression. Some algorithms go beyond such training objective and define multiple agents with different or competing objectives. The associated optimization paradigm requires a multi-objective joint minimization. An example of such a class of algorithms are the generative adversarial networks (GANs, Goodfellow et al., 2014), which aim at finding a Nash equilibrium of a two-player *minimax* game, where the players are deep neural networks (DNNs).

As of their success on supervised tasks, SGD based algorithms have been adopted for GAN training as well. Recently, Gidel et al. [2019] proposed to use an optimization technique coming from the variational inequality literature called *extragradient* [Korpelevich, 1976] with provable convergence guarantees to optimize games (see § 4.2). However, convergence failures, poor performance (sometimes referred to as "mode collapse"), or hyperparameter susceptibility are more commonly reported compared to classical supervised DNN optimization.

We question naive adoption of such methods for game optimization so as to address the reported training instabilities. We argue that as of the two player setting, noise impedes drastically more the training compared to single objective one. More precisely, we point out that the noise due to the stochasticity may break the convergence of the extragradient method, by considering a simplistic stochastic bilinear game for which it provably does *not* converge.

The theoretical aspect we present in this chapter is further supported empirically, since using larger mini-batch sizes for GAN training has been shown to considerably improve the quality of the samples produced by the resulting generative model: Brock et al. [2019] report a relative improvement of 46% of the Inception Score metric (see § 4.5) on ImageNet if the batch size is increased 8–fold. This notable improvement raises the question if noise reduction optimization methods can be extended to game settings. In turn, this would allow for a principled training method with the practical benefit of omitting to empirically establish this multiplicative factor for the batch size.

In this chapter, we investigate the interplay between noise and multi-objective problems in the context of GAN training. Our contributions can be summarized as follows: (i) we show in a motivating example how the noise can make stochastic extragradient fail (see § 4.2.2). (ii) we propose a new method "stochastic variance reduced extragradient" (SVRE) that combines variance reduction and extrapolation (see Alg. 3 and § 4.3.2) and show experimentally that it effectively reduces the noise. (iii) we prove the convergence of SVRE under local strong convexity assumptions, improving over the known rates of competitive methods for a large class of games (see § 4.3.2 for our convergence result and Table 4.1 for comparison with standard methods). (iv) we test SVRE empirically to train GANs on several standard datasets, and observe that it can improve SOTA deep models in the late stage of their optimization (see § 4.5).

Method	Complexity	μ-adaptivity
SVRG	$\ln(\frac{1}{\epsilon}) \times (n + \frac{\bar{L}^2}{\mu^2})$	no
Acc. SVRG	$\ln(\frac{1}{\epsilon}) \times (n + \sqrt{n}\frac{\bar{L}}{\mu})$	no
SVRE §4.3.2	$\ln(\frac{1}{\epsilon}) \times (n + \frac{\bar{\ell}}{\mu})$	if $\bar{\ell} = O(\bar{L})$

Table 4.1 – Comparison of variance reduced methods for games for a μ-strongly monotone operator with L_i-Lipschitz stochastic operators. Our result makes the assumption that the operators are ℓ_i-cocoercive. Note that $\ell_i \in [L_i, L_i^2/\mu]$, more details and a tighter rate are provided in §4.3.2. The SVRG variants are proposed by Palaniappan and Bach [2016]. *μ-adaptivity* indicates if the hyper-parameters that guarantee convergence (step size & epoch length) depend on the strong monotonicity parameter μ: if not, the algorithm is adaptive to local strong monotonicity. Note that in some cases the constant ℓ may depend on μ but SVRE is adaptive to strong convexity when $\bar{\ell}$ remains close to $\bar{L}$ (see for instance Proposition 2).

Algorithm 3 Pseudocode for SVRE.

```
Input: Stopping time $T$, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\varphi}}$, initial weights $\boldsymbol{\theta}_0$, $\boldsymbol{\varphi}_0$. $t = 0$
while $t \leq T$ do
    $\boldsymbol{\varphi}^{\mathcal{S}} = \boldsymbol{\varphi}_t$ and $\boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\varphi}} \mathcal{L}_i^D(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$
    $\boldsymbol{\theta}^{\mathcal{S}} = \boldsymbol{\theta}_t$ and $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i^G(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$
    $N \sim \mathrm{Geom}\big(1/n\big)$                    (Sample epoch length)
    for $i = 0$ to $N-1$ do {Beginning of the epoch}
        Sample $i_{\boldsymbol{\theta}}, i_{\boldsymbol{\varphi}} \sim \pi_{\boldsymbol{\theta}}, \pi_{\boldsymbol{\varphi}}$, do extrapolation:
        $\tilde{\boldsymbol{\varphi}}_t = \boldsymbol{\varphi}_t - \eta_{\boldsymbol{\varphi}} \boldsymbol{d}_{i_{\boldsymbol{\varphi}}}^D(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$        ▷ (4.5)
        $\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_t - \eta_{\boldsymbol{\theta}} \boldsymbol{d}_{i_{\boldsymbol{\theta}}}^G(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$        ▷ (4.5)
        Sample $i_{\boldsymbol{\theta}}, i_{\boldsymbol{\varphi}} \sim \pi_{\boldsymbol{\theta}}, \pi_{\boldsymbol{\varphi}}$ and do update:
        $\boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t - \eta_{\boldsymbol{\varphi}} \boldsymbol{d}_{i_{\boldsymbol{\varphi}}}^D(\tilde{\boldsymbol{\theta}}_t, \tilde{\boldsymbol{\varphi}}_t, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$        ▷ (4.5)
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_{\boldsymbol{\theta}} \boldsymbol{d}_{i_{\boldsymbol{\theta}}}^G(\tilde{\boldsymbol{\theta}}_t, \tilde{\boldsymbol{\varphi}}_t, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$        ▷ (4.5)
        $t \leftarrow t + 1$
    end for
end while
Output: $\boldsymbol{\theta}_T$, $\boldsymbol{\varphi}_T$
```

4.2 GANs as a Game and Noise in Games

4.2.1 Game theory formulation of GANs

The models in a GAN are a generator G, that maps an embedding space to the signal space, and should eventually map a fixed noise distribution to the training data distribution, and a discriminator D whose purpose is to allow the training of the generator by classifying genuine samples against generated ones. At each iteration of the algorithm, the discriminator D is

Method	Gradient method	Extragradient
Batch	$\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\| \to \infty$	$\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\| \to 0$
Stochastic	No hope for convergence	$\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\| \to \infty$

Table 4.2 – Methods for solving bilinear games. As batch-gradient does not converge for bilinear games [Mescheder et al., 2017], there is no hope for convergence proof of stochastic-gradient. Moreover, we show that the stochastic counterpart of extragradient [Korpelevich, 1976] also diverges for this class of games, see § 4.2.2.

updated to improve its "real vs. generated" classification performance, and the generator G to degrade it.

From a game theory point of view, GAN training is a differentiable two-player game where the generator $G_{\boldsymbol{\theta}}$ and the discriminator $D_{\boldsymbol{\varphi}}$ aim at minimizing their own cost function $\mathscr{L}^G$ and $\mathscr{L}^D$, resp.:

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathscr{L}^G(\boldsymbol{\theta}, \boldsymbol{\varphi}^*) \qquad \text{and} \qquad \boldsymbol{\varphi}^* \in \arg\min_{\boldsymbol{\varphi}\in\Phi} \mathscr{L}^D(\boldsymbol{\theta}^*, \boldsymbol{\varphi})\,. \tag{2P-G}$$

When $\mathscr{L}^D = -\mathscr{L}^G =: \mathscr{L}$ this game is called a *zero-sum game* and (2P-G) is a minimax problem:

$$\min_{\boldsymbol{\theta}\in\Theta} \max_{\boldsymbol{\varphi}\in\Phi} \mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \tag{SP}$$

The gradient method does not converge for some convex-concave examples [Mescheder et al., 2017, Gidel et al., 2019]. To address this, Korpelevich [1976] proposed to use the *extragradient* method[1] which performs a lookahead step in order to get signal from an *extrapolated* point:

$$\text{Extrapolation:} \begin{cases} \tilde{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathscr{L}^G(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) \\ \tilde{\boldsymbol{\varphi}}_t = \boldsymbol{\varphi}_t - \eta \nabla_{\boldsymbol{\varphi}} \mathscr{L}^D(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) \end{cases} \quad \text{Update:} \begin{cases} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathscr{L}^G(\tilde{\boldsymbol{\theta}}_t, \tilde{\boldsymbol{\varphi}}_t) \\ \boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t - \eta \nabla_{\boldsymbol{\varphi}} \mathscr{L}^D(\tilde{\boldsymbol{\theta}}_t, \tilde{\boldsymbol{\varphi}}_t) \end{cases} \tag{EG}$$

Note how $\boldsymbol{\theta}_t$ and $\boldsymbol{\varphi}_t$ are updated with a gradient from a different point, the *extrapolated* one. In the context of a zero-sum game, for any *convex-concave* function $\mathscr{L}$ and any closed convex sets Θ and Φ, the extragradient method converges [Harker and Pang, 1990, Thm. 12.1.11].

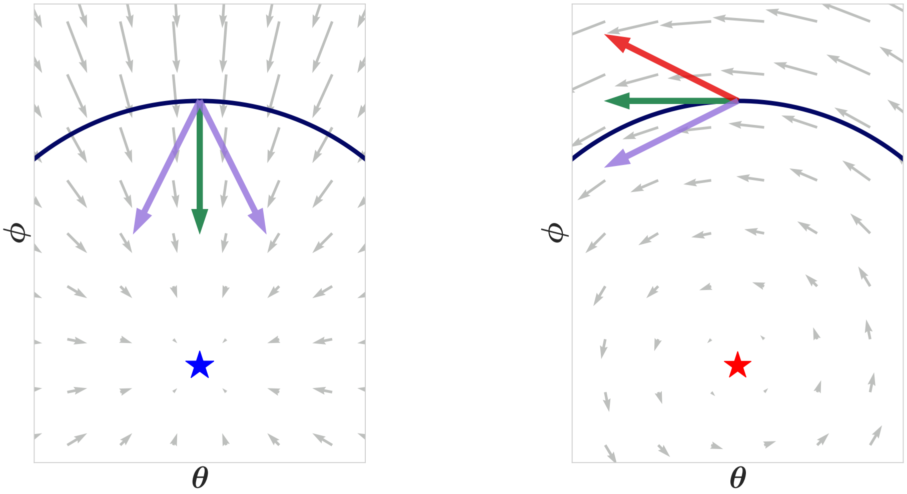
As batch–gradient does not converge for bilinear game, neither does stochastic–gradient. Therefore, in the section that follows we focus on the extragradient method, and we investigate if its stochastic counterpart converges, see Tab. 4.2,

4.2.2 Stochasticity Breaks Extragradient

As the (EG) converges for some examples for which gradient methods do not, it is reasonable to expect that so does its stochastic counterpart (at least to a neighborhood). However, the resulting noise in the gradient estimate may interact in a problematic way with the oscillations

---

[1] For simplicity, we focus on *unconstrained* setting where $\Theta = \mathbb{R}^d$. For the *constrained* case, a Euclidean projection on the constraints set should be added at every update of the method.



(a) **Minimization:** up to a neighborhood, the noisy gradient always points to a direction that make the iterate closer to the minimum (★).

(b) **Game:** the noisy gradient may point to a direction (red arrow) that push the iterate away from the Nash Equilibrium (★).

Figure 4.1 – Illustration of the discrepancy between games and minimization on simple examples:

$$\textit{minimization:} \min_{\theta,\phi\in\mathbb{R}} \theta^2+\phi^2\,, \qquad \textit{game:} \min_{\theta\in\mathbb{R}}\max_{\phi\in\mathbb{R}} \theta\cdot\phi\,.$$

due to the *adversarial component* of the game. We depict this phenomenon in Fig. 4.1, where we show the direction of the noisy gradient on single objective minimization example and contrast it with a multi-objective one.

We present a simplistic example where the extragradient method *converges linearly* [Gidel et al., 2019, Corollary 1] using the full gradient but *diverges geometrically* when using stochastic estimates of it. Note that standard gradient methods, both batch and stochastic, diverge on this example.

In particular, we show that: (i) if we use standard stochastic estimates of the gradients of $\mathscr{L}$ with a simple finite sum formulation, then the iterates $\boldsymbol{\omega}_t := (\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ produced by the stochastic extragradient method (SEG) diverge geometrically, and on the other hand (ii) the full-batch extragradient method does converge to the Nash equilibrium $\boldsymbol{\omega}^*$ of this game [Harker and Pang, 1990, Thm. 12.1.11].

Theorem 1 (Noise may induce divergence)**.** *For any $\epsilon \geq 0$ There exists a zero-sum $\frac{\epsilon}{2}$-strongly monotone stochastic game such that if $\boldsymbol{\omega}_0 \neq \boldsymbol{\omega}^*$, then for any step-size $\eta > \epsilon$, the iterates $(\boldsymbol{\omega}_t)$ computed by the stochastic extragradient method diverge geometrically, i.e., there exists $\rho > 0$, such that $\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|^2] > \|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|^2(1+\rho)^t$.*

Proof sketch. All detailed proofs can be found in § C.2 of the appendix. We consider the following stochastic optimization problem (with $d = n$):

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\epsilon}{2}\theta_i^2 + \boldsymbol{\theta}^\top \boldsymbol{A}_i \boldsymbol{\varphi} - \frac{\epsilon}{2}\varphi_i^2 \quad \text{where} \quad [\boldsymbol{A}_i]_{kl} = 1 \text{ if } k = l = i \text{ and } 0 \text{ otherwise.} \tag{4.1}$$

Note that this problem is a simple dot product between $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ with an (ϵ/n)-ℓ_2 norm penalization, thus we can compute the batch gradient and notice that the Nash equilibrium of this problem is $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*) = (\mathbf{0}, \mathbf{0})$. However, as we shall see, this simple problem *breaks* with standard stochastic optimization methods.

Sampling a mini-batch without replacement $I \subset \{1, \dots, n\}$, we denote $\boldsymbol{A}_I := \sum_{i \in I} \boldsymbol{A}_i$. The extragradient update rule can be written as:

$$\begin{cases} \boldsymbol{\theta}_{t+1} = (1 - \eta \boldsymbol{A}_I \epsilon)\boldsymbol{\theta}_t - \eta \boldsymbol{A}_I((1 - \eta \boldsymbol{A}_J \epsilon)\boldsymbol{\varphi}_t + \eta \boldsymbol{A}_J \boldsymbol{\theta}_t) \\ \boldsymbol{\varphi}_{t+1} = (1 - \eta \boldsymbol{A}_I \epsilon)\boldsymbol{\varphi}_t + \eta \boldsymbol{A}_I((1 - \eta \boldsymbol{A}_J \epsilon)\boldsymbol{\theta}_t - \eta \boldsymbol{A}_J \boldsymbol{\varphi}_t), \end{cases} \tag{4.2}$$

where I and J are the mini-batches sampled for the update and the extrapolation step, respectively. Let us write $N_t := \|\boldsymbol{\theta}_t\|^2 + \|\boldsymbol{\varphi}_t\|^2$. Noticing that $[\boldsymbol{A}_I \boldsymbol{\theta}]_i = [\boldsymbol{\theta}]_i$ if $i \in I$ and 0 otherwise, we have,

$$\mathbb{E}[N_{t+1}] = \left(1 - \tfrac{|I|}{n}(2\eta\epsilon - \eta^2(1 + \epsilon^2)) - \tfrac{|I|^2}{n^2}(2\eta^2 - \eta^4(1 + \epsilon^2))\right)\mathbb{E}[N_t]. \tag{4.3}$$

Consequently, if the mini-batch size is smaller than half of the dataset size, i.e. $2|I| \le n$, we have that $\forall \eta > \epsilon, \exists \rho > 0,\ s.t.,\ \mathbb{E}[N_t] > N_0(1 + \rho)^t$. For the theorem statement, we set $n = 2$ and $|I| = 1$.

This result may seem contradictory with the standard result on SEG [Juditsky et al., 2011] saying that the average of the iterates computed by SEG does converge to the Nash equilibrium of the game. However, an important assumption made by Juditsky et al. is that the iterates are projected onto a compact set and that estimator of the gradient has finite variance. These assumptions break in this example since the variance of the estimator is proportional to the norm of the (unbounded) parameters. Note that constraining the optimization problem (4.1) to bounded domains Θ and Φ, would make the finite variance assumption from Juditsky et al. [2011] holds. Consequently, the averaged iterate $\bar{\boldsymbol{\omega}}_t := \frac{1}{t}\sum_{s=0}^{t-1}\boldsymbol{\omega}_s$ would converge to $\boldsymbol{\omega}^*$. In § C.0.1, we explain why in a *non-convex setting*, the convergence of the *last iterate* is preferable.

4.3 Reducing Noise in Games with Variance Reduced Extragradient

One way to reduce the noise in the estimation of the gradient is to use mini-batches of samples instead of one sample. However, mini-batch stochastic extragradient fails to converge on (4.1) if the mini-batch size is smaller than half of the dataset size (see § C.2.1). In order to get an estimator of the gradient with a vanishing variance, the optimization literature proposed to take advantage of the finite-sum formulation that often appears in machine learning [Schmidt et al., 2017, and references therein].

4.3.1 Variance Reduced Gradient Methods

Let us assume that the objective in (2P-G) can be decomposed as a finite sum such that[2]

$$\mathcal{L}^G(\boldsymbol{\omega}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_i^G(\boldsymbol{\omega}) \quad \text{and} \quad \mathcal{L}^D(\boldsymbol{\omega}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_i^D(\boldsymbol{\omega}) \quad \text{where} \quad \boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi})\,. \tag{4.4}$$

Johnson and Zhang [2013] propose the “stochastic variance reduced gradient” (SVRG) as an *unbiased* estimator of the gradient with a smaller variance than the vanilla mini-batch estimate. The idea is to occasionally take a snapshot $\boldsymbol{\omega}^{\mathcal{S}}$ of the current model’s parameters, and store the full batch gradient $\boldsymbol{\mu}^{\mathcal{S}}$ at this point. Computing the full batch gradient $\boldsymbol{\mu}^{\mathcal{S}}$ at $\boldsymbol{\omega}^{\mathcal{S}}$ is an expensive operation but not prohibitive if done infrequently (for instance once every dataset pass).

Assuming that we have stored $\boldsymbol{\omega}^{\mathcal{S}}$ and $\boldsymbol{\mu}^{\mathcal{S}} := (\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}}, \boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}})$, the *SVRG estimates* of the gradients are:

$$\boldsymbol{d}_i^G(\boldsymbol{\omega}) := \frac{\nabla \mathcal{L}_i^G(\boldsymbol{\omega}) - \nabla \mathcal{L}_i^G(\boldsymbol{\omega}^{\mathcal{S}})}{n\pi_i} + \boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}}\,, \quad \boldsymbol{d}_i^D(\boldsymbol{\omega}) := \frac{\nabla \mathcal{L}_i^D(\boldsymbol{\omega}) - \nabla \mathcal{L}_i^D(\boldsymbol{\omega}^{\mathcal{S}})}{n\pi_i} + \boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}}\,. \tag{4.5}$$

These estimates are unbiased: $\mathbb{E}[\boldsymbol{d}_i^G(\boldsymbol{\omega})] = \frac{1}{n}\sum_{i=1}^{n} \nabla \mathcal{L}_i^G(\boldsymbol{\omega}) = \nabla \mathcal{L}^G(\boldsymbol{\omega})$, where the expectation is taken over i, picked with probability π_i. The non-uniform sampling probabilities π_i are used to bias the sampling according to the Lipschitz constant of the stochastic gradient in order to sample more often gradients that change quickly. This strategy has been first introduced for variance reduced methods by Xiao and Zhang [2014] for SVRG and has been discussed for saddle point optimization by Palaniappan and Bach [2016].

Originally, SVRG was introduced as an epoch based algorithm with a *fixed epoch size*: in Alg. 3, one epoch is an inner loop of size N (Line 6). However, Hofmann et al. [2015] proposed instead to *sample* the size of each epoch from a geometric distribution, enabling them to analyze SVRG the same way as SAGA under a unified framework called q-memorization algorithm. We generalize their framework to handle the extrapolation step (EG) and provide a convergence proof for such q-memorization algorithms for games in § C.2.2.

One advantage of Hofmann et al. [2015]’s framework is also that the sampling of the epoch size does not depend on the condition number of the problem, whereas the original proof for SVRG had to consider an epoch size larger than the condition number (see [Leblond et al., 2018, Corollary 16] for a detailed discussion on the convergence rate for SVRG). Thus, this new version of SVRG with a random epoch size becomes *adaptive to the local strong convexity* since none of its hyper-parameters depend on the strong convexity constant.

However, because of some new technical aspects when working with monotone operators, Palaniappan and Bach [2016]’s proofs (both for SAGA and SVRG) require a step-size

---

[2] The “noise dataset” in a GAN is not finite though; see § C.3.1 for details on how to cope with this in practice.

(and epoch length for SVRG) that depends on the strong monotonicity constant making these algorithms not adaptive to local strong monotonicity. This motivates the proposed SVRE algorithm, which may be adaptive to local strong monotonicity, and is thus more appropriate for non-convex optimization.

4.3.2 SVRE: Stochastic Variance Reduced Extragradient

We describe our proposed algorithm called stochastic variance reduced extragradient (SVRE) in Alg. 3. In an analogous manner to how Palaniappan and Bach [2016] combined SVRG with the gradient method, SVRE combines SVRG estimates of the gradient (4.5) with the *extragradient method* (EG).

With SVRE we are able to improve the convergence rates for variance reduction for a large class of stochastic games (see Table 4.1 and Thm. 2), and we show in § 4.3.3 that it is the only method which empirically converges on the simple example of § 4.2.2.

We now describe the theoretical setup for the convergence result. A standard assumption in convex optimization is the assumption of strong convexity of the function. However, in a game, the operator,

$$\boldsymbol{v} : \boldsymbol{\omega} \mapsto \left[\nabla_{\boldsymbol{\theta}} \mathscr{L}^G(\boldsymbol{\omega}) \,,\, \nabla_{\boldsymbol{\varphi}} \mathscr{L}^D(\boldsymbol{\omega}) \right]^\top , \tag{4.6}$$

associated with the updates is no longer the gradient of a single function. To make an analogous assumption for games the optimization literature considers the notion of *strong monotonicity.*

Definition 1. *An operator* $F : \boldsymbol{\omega} \mapsto (F_{\boldsymbol{\theta}}(\boldsymbol{\omega}), F_{\boldsymbol{\varphi}}(\boldsymbol{\omega})) \in \mathbb{R}^{d+p}$ *is said to be* $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$*-strongly monotone if for all* $\boldsymbol{\omega}, \boldsymbol{\omega}' \in \mathbb{R}^{p+d}$ *we have*

$$\Omega((\boldsymbol{\theta}, \boldsymbol{\varphi}), (\boldsymbol{\theta}', \boldsymbol{\varphi}')) := \mu_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 + \mu_{\boldsymbol{\varphi}} \|\boldsymbol{\varphi} - \boldsymbol{\varphi}'\|^2 \leq (F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') , \tag{4.7}$$

where we write $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \mathbb{R}^{d+p}$*. A* monotone operator *is a* $(0,0)$*-strongly monotone operator.*

This definition is a generalization of strong convexity for operators: if f is μ-strongly convex, then ∇f is a μ-monotone operator. Another assumption is the γ regularity assumption,

Definition 2. *An operator* $F : \boldsymbol{\omega} \mapsto (F_{\boldsymbol{\theta}}(\boldsymbol{\omega}), F_{\boldsymbol{\varphi}}(\boldsymbol{\omega})) \in \mathbb{R}^{d+p}$ *is said to be* $(\gamma_{\boldsymbol{\theta}}, \gamma_{\phi})$*-regular if,*

$$\gamma_{\boldsymbol{\theta}}^2 \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 + \gamma_{\boldsymbol{\varphi}}^2 \|\boldsymbol{\varphi} - \boldsymbol{\varphi}'\|^2 \leq \|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|^2 , \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \mathbb{R}^{p+d} . \tag{4.8}$$

Note that an *operator* is always $(0,0)$-regular. This assumption originally introduced by Tseng [1995] has been recently used [Azizian et al., 2019] to improve the convergence rate of extragradient. For instance for a full rank bilinear matrix problem γ is its smallest singular value. More generally, in the case $\gamma_{\boldsymbol{\theta}} = \gamma_{\boldsymbol{\varphi}}$, the regularity constant is a lower bound on the minimal singular value of the Jacobian of F [Azizian et al., 2019].

One of our main assumptions is the cocoercivity assumption, which implies the Lipchitzness of the operator in the unconstrained case. We use the cocoercivity constant because it provides a tighter bound for general strongly monotone and Lipschitz games (see discussion following Theorem 2).

Definition 3. *An operator* $F : \boldsymbol{\omega} \mapsto (F_{\boldsymbol{\theta}}(\boldsymbol{\omega}), F_{\boldsymbol{\varphi}}(\boldsymbol{\omega})) \in \mathbb{R}^{d+p}$ *is said to be* $(\ell_{\boldsymbol{\theta}}, \ell_{\boldsymbol{\varphi}})$*-cocoercive, if for all* $\boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega$ *we have*

$$\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|^2 \le \ell_{\boldsymbol{\theta}} (F_{\boldsymbol{\theta}}(\boldsymbol{\omega}) - F_{\boldsymbol{\theta}}(\boldsymbol{\omega}'))^\top (\boldsymbol{\theta} - \boldsymbol{\theta}') + \ell_{\boldsymbol{\varphi}} (F_{\boldsymbol{\varphi}}(\boldsymbol{\omega}) - F_{\boldsymbol{\varphi}}(\boldsymbol{\omega}'))^\top (\boldsymbol{\varphi} - \boldsymbol{\varphi}') . \tag{4.9}$$

Note that for a L-Lipschitz and μ-strongly monotone operator, we have $\ell \in [L, L^2/\mu]$ [Facchinei and Pang, 2003]. For instance, when F is the gradient of a convex function, we have $\ell = L$. More generally, when $F(\boldsymbol{\omega}) = (\nabla f(\boldsymbol{\theta}) + M\boldsymbol{\varphi}, \nabla g(\boldsymbol{\varphi}) - M^\top \boldsymbol{\theta})$, where f and g are μ-strongly convex and L smooth we have that $\gamma = \sigma_{\min}(M)$ and $\|M\|^2 = O(\mu L)$ is a sufficient condition for $\ell = O(L)$ (see §C.1). Under this assumption on each cost function of the game operator, we can define a cocoercivity constant adapted to the non-uniform sampling scheme of our stochastic algorithm:

$$\bar{\ell}(\pi)^2 := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{n\pi_i} \ell_i^2 . \tag{4.10}$$

The standard *uniform sampling scheme* corresponds to $\pi_i := \frac{1}{n}$ and the optimal *non-uniform* sampling scheme corresponds to $\tilde{\pi}_i := \frac{\ell_i}{\sum_{i=1}^n \ell_i}$. By Jensen's inequality, we have: $\bar{\ell}(\tilde{\pi}) \le \bar{\ell}(\pi) \le \max_i \ell_i$.

For our main result, we make strong convexity, cocoercivity and regularity assumptions.

Assumption 1. *For* $1 \le i \le n$*, the gradients* $\nabla_{\boldsymbol{\theta}} \mathcal{L}_i^G$ *and* $\nabla_{\boldsymbol{\varphi}} \mathcal{L}_i^D$ *are respectively* $\ell_i^{\boldsymbol{\theta}}$ *and* $\ell_i^{\boldsymbol{\varphi}}$*-cocoercive and* $(\gamma_i^{\boldsymbol{\theta}}, \gamma_i^{\boldsymbol{\varphi}})$*-regular. The operator* (4.6) *is* $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$*-strongly monotone.*

We now present our convergence result for SVRE with non-uniform sampling (to make our constants comparable to those of Palaniappan and Bach [2016]), but note that we have used uniform sampling in all our experiments (for simplicity).

Theorem 2. *Under Assumption 1, after t iterations, the iterate* $\boldsymbol{\omega}_t := (\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ *computed by SVRE (Alg. 3) with step-size* $\eta_{\boldsymbol{\theta}} \le (40\bar{\ell}_{\boldsymbol{\theta}})^{-1}$ *and* $\eta_{\boldsymbol{\varphi}} \le (40\bar{\ell}_{\boldsymbol{\varphi}})^{-1}$ *and sampling scheme* $(\tilde{\pi}_{\boldsymbol{\theta}}, \tilde{\pi}_{\boldsymbol{\varphi}})$ *verifies:*

$$\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] \le \left(1 - \frac{1}{2}\min\left\{\eta_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}} + \frac{9\eta_{\boldsymbol{\theta}}^2 \bar{\gamma}_{\boldsymbol{\theta}}^2}{10}, \eta_{\boldsymbol{\varphi}}\mu_{\boldsymbol{\varphi}} + \frac{9\eta_{\boldsymbol{\varphi}}^2 \bar{\gamma}_{\boldsymbol{\varphi}}^2}{10}, \frac{4}{5n}\right\}\right)^t \mathbb{E}[\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2] ,$$

where $\bar{\ell}_{\boldsymbol{\theta}}(\pi_{\boldsymbol{\theta}})$ *and* $\bar{\ell}_{\boldsymbol{\varphi}}(\pi_{\boldsymbol{\varphi}})$ *are defined in* (4.10). *Particularly, for* $\eta_{\boldsymbol{\theta}} = \frac{1}{40\bar{\ell}_{\boldsymbol{\theta}}}$ *and* $\eta_{\boldsymbol{\varphi}} = \frac{1}{40\bar{\ell}_{\boldsymbol{\varphi}}}$ *we get*

$$\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] \le \left(1 - \frac{1}{2}\min\left\{\tfrac{1}{40}\Big(\frac{\mu_{\boldsymbol{\theta}}}{\bar{\ell}_{\boldsymbol{\theta}}} + \frac{\bar{\gamma}_{\boldsymbol{\theta}}^2}{45\bar{\ell}_{\boldsymbol{\theta}}^2}\Big), \tfrac{1}{40}\Big(\frac{\mu_{\boldsymbol{\varphi}}}{\bar{\ell}_{\boldsymbol{\varphi}}} + \frac{\bar{\gamma}_{\boldsymbol{\varphi}}^2}{45\bar{\ell}_{\boldsymbol{\varphi}}^2}\Big), \frac{4}{5n}\right\}\right)^t \mathbb{E}[\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2] .$$

We prove this theorem in § C.2.2. We can notice that the respective *condition numbers* of

$\mathscr{L}^G$ and $\mathscr{L}^D$ defined as $\kappa_{\boldsymbol{\theta}} := \frac{\mu_{\boldsymbol{\theta}}}{\bar{\ell}_{\boldsymbol{\theta}}} + \frac{\bar{\gamma}^2_{\boldsymbol{\theta}}}{\bar{\ell}^2_{\boldsymbol{\theta}}}$ and $\kappa_{\boldsymbol{\varphi}} := \frac{\mu_{\boldsymbol{\varphi}}}{\bar{\ell}_{\boldsymbol{\varphi}}} + \frac{\bar{\gamma}^2_{\boldsymbol{\varphi}}}{\bar{\ell}^2_{\boldsymbol{\varphi}}}$ appear in our convergence rate. The cocoercivity constant ℓ belongs to $[L, L^2/\mu]$, thus our rate may be significantly faster[3] than the convergence rate of the (non-accelerated) algorithm of Palaniappan and Bach [2016] that depends on the product $\frac{\mu_{\boldsymbol{\theta}}}{\bar{L}_{\boldsymbol{\theta}}} \frac{\mu_{\boldsymbol{\varphi}}}{\bar{L}_{\boldsymbol{\varphi}}}$. They avoid a dependence on the maximum of the condition numbers squared, $\max\{\kappa^2_{\boldsymbol{\varphi}}, \kappa^2_{\boldsymbol{\theta}}\}$, by using the weighted Euclidean norm $\Omega(\boldsymbol{\theta}, \boldsymbol{\varphi})$ defined in (4.7) and rescaling the functions $\mathscr{L}^G$ and $\mathscr{L}^D$ with their strong-monotonicity constant. However, this rescaling trick suffers from two issues: (i) we do not know in practice a good estimate of the strong monotonicity constant, which was *not* the case in Palaniappan and Bach [2016]'s application; and (ii) the algorithm does not adapt to local strong-monotonicity. This property is important in non-convex optimization since we want the algorithm to exploit the (potential) local stability properties of a stationary point.

4.3.3 Motivating example

The example (4.1) for $\epsilon = 0$ seems to be challenging in the stochastic setting since all the standard methods and even the stochastic extragradient method fails to find its Nash equilibrium (note that this example is *not* strongly monotone). We set $n = d = 100$, and draw $[\boldsymbol{A}_i]_{kl} = \delta_{kli}$ and $[\boldsymbol{b}_i]_k, [\boldsymbol{c}_i]_k \sim \mathcal{N}(0, 1/d)$, $1 \leq k, l \leq d$, where $\delta_{kli} = 1$ if $k = l = i$ and 0 otherwise. Our optimization problem is:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \max_{\boldsymbol{\varphi} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^\top \boldsymbol{b}_i + \boldsymbol{\theta}^\top \boldsymbol{A}_i \boldsymbol{\varphi} + \boldsymbol{c}_i^\top \boldsymbol{\varphi}). \tag{4.11}$$

We compare variants of the following algorithms (with uniform sampling and average our results over 5 different seeds): (i) AltSGD: the standard method to train GANs–stochastic gradient with alternating updates of each player. (ii) SVRE: Alg. 3. The AVG prefix correspond to the *uniform average* of the iterates, $\bar{\boldsymbol{\omega}} := \frac{1}{t} \sum_{s=0}^{t-1} \boldsymbol{\omega}_s$. We observe in Fig. 4.3 that AVG-SVRE converges sublinearly (whereas AVG-AltSGD fails to converge).

This motivates a new variant of SVRE based on the idea that even if the averaged iterate converges, we do not compute the gradient at that point and thus we do not benefit from the fact that this iterate is closer to the optimums (see § C.0.1). Thus the idea is to occasionally restart the algorithm, i.e., consider the averaged iterate as the new starting point of our algorithm and compute the gradient at that point. Restart goes well with SVRE as we already occasionally stop the inner loop to recompute $\boldsymbol{\mu}^{\mathscr{S}}$, at which point we decide (with a probability p to be fixed) whether or not to restart the algorithm by taking the snapshot at point $\bar{\boldsymbol{\omega}}_t$ instead of $\boldsymbol{\omega}_t$. This variant of SVRE is described in Alg. 6 in § C.4 and the variant combining VRAd in § C.3.1.

In Fig. 4.3 we observe that the only method that converges is SVRE and its variants. We do not

---

[3] Particularly, when F is the gradient of a convex function (or close to it) we have $\ell \approx L$ and thus our rate recovers the standard $\ln(1/\epsilon)L/\mu$, improving over the accelerated algorithm of Palaniappan and Bach [2016]. More generally, under the assumptions of Proposition 2, we also recover $\ln(1/\epsilon)L/\mu$.

Algorithm 4 Pseudocode for SVRE-GAN.

```
Input: dataset $\mathscr{D}$, noise dataset $\mathscr{Z}$ ($|\mathscr{Z}| = |\mathscr{D}| = n$), stopping iteration $T$, learning rates $\eta_D, \eta_G$, generator loss $\mathscr{L}^G$, discriminator loss $\mathscr{L}^D$, mini-batch size B.
Initialize: $\varphi, \theta$
for $e = 0$ to $T-1$ do
    $\varphi^{\mathscr{S}} = \varphi$ and $\boldsymbol{\mu}_\varphi = \frac{1}{n}\sum_{i=1}^n \sum_{j=1}^n \nabla_\varphi \mathscr{L}^D(\theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, \mathscr{D}_j, \mathscr{Z}_i)$
    $\theta^{\mathscr{S}} = \theta$ and $\boldsymbol{\mu}_\theta = \frac{1}{n}\sum_{i=1}^n \nabla_\theta \mathscr{L}^G(\theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, \mathscr{Z}_i)$
    $N \sim \text{Geom}\big(B/n\big)$                                  (length of the epoch)
    for $i = 0$ to $N-1$ do
        Sample mini-batches $(n_d, n_z)$; do extrapolation:
        $\tilde{\varphi} = \varphi - \eta_D \boldsymbol{d}_\varphi(\theta, \varphi, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, n_z)$                ▷ (4.14)
        $\tilde{\theta} = \theta - \eta_G \boldsymbol{d}_\theta(\theta, \varphi, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, n_d, n_z)$           ▷ (4.15)
        Sample new mini-batches $(n_d, n_z)$; do update:
        $\varphi = \varphi - \eta_D \boldsymbol{d}_\varphi(\tilde{\theta}, \tilde{\varphi}, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, n_z)$                ▷ (4.14)
        $\theta = \theta - \eta_G \boldsymbol{d}_\theta(\tilde{\theta}, \tilde{\varphi}, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, n_d, n_z)$           ▷ (4.15)
    end for
end for
Output: $\varphi, \theta$
```

provide convergence guarantees for Alg. 6 and leave its analysis for future work. However, it is interesting that, to our knowledge, this algorithm is the only stochastic algorithm (excluding batch extragradient as it is not stochastic) that converge for (4.1). Note that we tried all the algorithms presented in Fig. 3 from [Gidel et al., 2019] on this *unconstrained* problem and that all of them diverge.

4.4 Pseudocode for SVRE-GAN and SVRG-GAN

In order to cope with the issues introduced by the stochastic game formulation of the GAN models, we proposed the SVRE algorithm, described in Alg. 3, which combines SVRG and extragradient method. We refer to the method of applying SVRE to train GANs as the *SVRE-GAN* method, and we describe it in detail in Alg. 4 (generalizing it with mini-batching, but using uniform probabilities). Assuming that we have $\mathscr{D}[n_d]$ and $\mathscr{Z}[n_z]$, respectively two mini-batches of size B of the true dataset and the noise dataset, we compute $\nabla_\varphi \mathscr{L}^D(\theta, \varphi, \mathscr{D}[n_d], \mathscr{Z}[n_z])$ and $\nabla_\theta \mathscr{L}^G(\theta, \varphi, \mathscr{Z}[n_z])$ the respective mini-batches gradient of the discriminator and the generator:

$$\nabla_\varphi \mathscr{L}^D(\theta, \varphi, \mathscr{D}[n_d], \mathscr{Z}[n_z]) := \frac{1}{|n_z|}\frac{1}{|n_d|}\sum_{i \in n_z}\sum_{j \in n_d} \nabla_\varphi \mathscr{L}^D(\theta, \varphi, \mathscr{D}_j, \mathscr{Z}_i) \tag{4.12}$$

$$\nabla_\theta \mathscr{L}^G(\theta, \varphi, \mathscr{Z}[n_z]) := \frac{1}{|n_z|}\sum_{i \in n_z} \nabla_\theta \mathscr{L}^G(\theta, \varphi, \mathscr{Z}_i)\,, \tag{4.13}$$

where $\mathscr{Z}_i$ and $\mathscr{D}_j$ are respectively the i^{th} example of the noise dataset and the j^{th} of the true dataset. Note that n_z and n_d are lists and thus that we allow repetitions in the summations

Algorithm 5 Pseudocode for SVRG-GAN.

```
1:  Input: dataset $\mathscr{D}$, noise dataset $\mathscr{Z}$ ($|\mathscr{Z}| = |\mathscr{D}| = n$), stopping iteration $T$, learning rates $\eta_D, \eta_G$, generator loss $\mathscr{L}^G$, discriminator loss $\mathscr{L}^D$, mini-batch size B.
2:  Initialize: $\varphi, \theta$
3:  for $e = 0$ to $T-1$ do
4:      $\varphi^{\mathscr{S}} = \varphi$ and $\boldsymbol{\mu}_\varphi = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\nabla_\varphi\mathscr{L}^D(\theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, \mathscr{D}_j, \mathscr{Z}_i)$
5:      $\theta^{\mathscr{S}} = \theta$ and $\boldsymbol{\mu}_\theta = \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta\mathscr{L}^G(\theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, \mathscr{Z}_i)$
6:      $N \sim \text{Geom}\left(B/n\right)$                                    (length of the epoch)
7:      for $i = 0$ to $N-1$ do
8:          Sample new mini-batches $(n_d, n_z)$; do update:
9:          $\varphi = \varphi - \eta_D \boldsymbol{d}_\varphi(\theta, \varphi, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, n_z)$              ▷ (4.14)
10:         $\theta = \theta - \eta_G \boldsymbol{d}_\theta(\theta, \varphi, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, n_d, n_z)$           ▷ (4.15)
11:     end for
12: end for
13: Output: $\varphi, \theta$
```

over n_z and n_d. The variance reduced gradient of the SVRG method are thus given by:

$$\boldsymbol{d}_\varphi(\theta, \varphi, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}) := \boldsymbol{\mu}_\varphi + \nabla_\varphi\mathscr{L}^D(\theta, \varphi, \mathscr{D}[n_d], \mathscr{Z}[n_z]) - \nabla_\varphi\mathscr{L}^D(\theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, \mathscr{D}[n_d], \mathscr{Z}[n_z]) \tag{4.14}$$

$$\boldsymbol{d}_\theta(\theta, \varphi, \theta^{\mathscr{S}}, \varphi^{\mathscr{S}}) := \boldsymbol{\mu}_\theta + \nabla_\theta\mathscr{L}^G(\theta, \varphi, \mathscr{Z}[n_z]) - \nabla_\theta\mathscr{L}^G(\theta^{\mathscr{S}}, \varphi^{\mathscr{S}}, \mathscr{Z}[n_z]), \tag{4.15}$$

where $G^{\mathscr{S}}$ and $D^{\mathscr{S}}$ are the snapshots and $\boldsymbol{\mu}_\varphi$ and $\boldsymbol{\mu}_\theta$ their respective gradients.

Alg. 4 summarizes SVRE–GAN. To obtain that $\mathbb{E}\left[\nabla_{\boldsymbol{\Theta}^{\mathscr{S}}}\mathscr{L}(\boldsymbol{\theta}^{\mathscr{S}}, \boldsymbol{\varphi}^{\mathscr{S}}, \cdot) - \boldsymbol{\mu}\right]$ vanishes, when updating $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ where the expectation is over samples of $\mathscr{D}$ and $\mathscr{Z}$ respectively, we use the snapshot networks $\boldsymbol{\theta}^{\mathscr{S}}$ and $\boldsymbol{\varphi}^{\mathscr{S}}$ for the second term in lines 9, 10, 12 and 13. Moreover, the noise dataset $\mathscr{Z} \sim p_z$, where $|\mathscr{Z}| = |\mathscr{D}| = n$, is fixed. Empirically we observe that directly sampling from p_z (contrary to fixing the noise dataset and re-sampling it before computing $\boldsymbol{\mu}_D$ and $\boldsymbol{\mu}_G$) does not impact the performance, as $|\mathscr{Z}|$ is usually high.

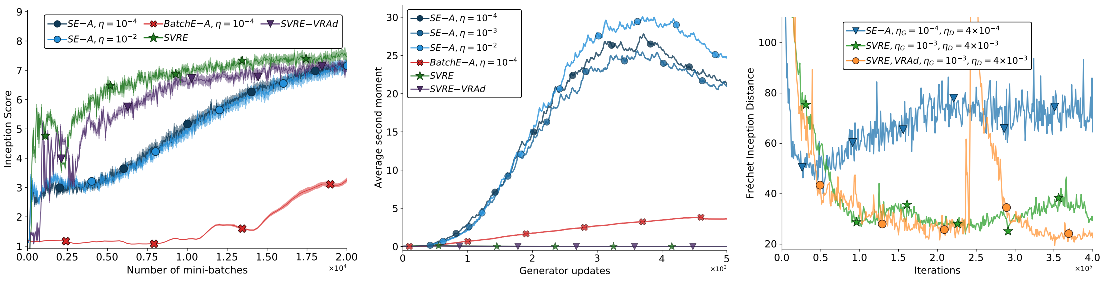
Note that the double sum in Line 4 can be written as two sums because of the separability of the expectations in typical GAN objectives. Thus the time complexity for calculating μ_φ is still $O(n)$ and not $O(n^2)$ which would be prohibitively expensive.

Alg. 5 summarizes the SVRG optimization extended to GAN, which we refer to as SVRG-GAN. Relative to SVRE–GAN, SVRG-GAN does not the compute the extrapolate point $\tilde{\theta}, \tilde{\varphi})$ of Alg. 4

4.5 Experiments

In this section, we investigate the empirical performance of SVRE and SVRG for *GAN training*. Note, however, that our theoretical analysis does not hold for games with non-convex objectives such as GANs.

Datasets. We used the following datasets: (i) **MNIST** [Lecun and Cortes, 1998] and **FASH-**



(a) IS (higher is better), **MNIST** (b) Generator–SME, **MNIST** (c) FID (lower is better), **SVHN**

Figure 4.2 – **Figures a & b.** Stochastic, full-batch and variance reduced extragradient optimization on **MNIST**. We used $\eta = 10^{-2}$ for SVRE. *SE–A* with $\eta = 10^{-3}$ achieves similar IS performances as $\eta = 10^{-2}$ and $\eta = 10^{-4}$, omitted from Fig. a for clarity. **Figure c.** FID on **SVHN**, using *shallow* architectures. See § 4.5 and § C.5 for naming of methods and details on the implementation, respectively.

ION-MNIST [Xiao et al., 2017], (ii) **CIFAR-10** [Krizhevsky, 2009, §3], (iii) **SVHN** [Netzer et al., 2011], and (iv) **ImageNet** ILSVRC 2012 [Russakovsky et al., 2015], using 28×28, 3×32×32, 3×32×32, and 3×64×64 resolution, respectively.

Metrics. We used the **Inception score** (IS, Salimans et al., 2016) and the **Fréchet Inception distance** (FID, Heusel et al., 2017) as performance metrics for image synthesis. To gain insights if SVRE indeed reduces the variance of the gradient estimates, we used the **second moment estimate–SME** (uncentered variance), computed with an exponentially moving average. See § C.5.1 for details. For **MNIST** we also used the **entropy** and the **total variation** metrics, which are described in § 3.4.

DNN architectures. For experiments on **MNIST**, we used the DCGAN architectures [Radford et al., 2016], described in § C.5.2. For real-world datasets, we used two architectures (see § C.5.2 for details and § C.5.2 for motivation): (i) SAGAN [Zhang et al., 2018], and (ii) ResNet, replicating the setup of Miyato et al. [2018], described in detail in § C.5.2 and C.5.2, respectively. For clarity, we refer the former as *shallow*, and the latter as *deep* architectures.

Optimization methods. We conduct experiments using the following optimization methods for GANs: (i) **BatchE:** full–batch extragradient, (ii) **SG:** stochastic gradient (alternating GAN), and (iii) **SE:** stochastic extragradient, and (iv) **SVRE:** stochastic variance reduced extragradient. These can be combined with adaptive learning rate methods such as *Adam* or with parameter averaging, hereafter denoted as **–A** and **AVG–**, respectively. In § C.3.1, we present a variant of Adam adapted to variance reduced algorithms, that is referred to as **–VRAd**. When using the SE–A baseline and *deep* architectures, the convergence rapidly fails at some point of training (cf. § C.6.3). This motivates experiments where we start from a stored checkpoint taken *before* the baseline diverged, and *continue training with SVRE*. We denote these experiments with **WS–SVRE** (warm-start SVRE).

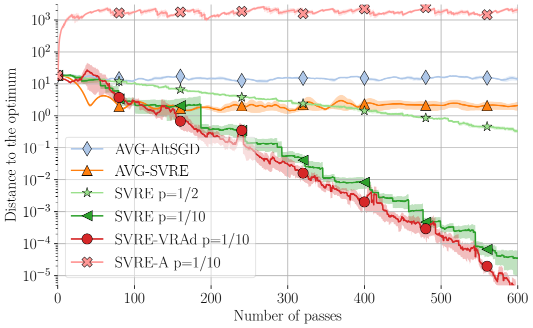


Figure 4.3 – Distance to the optimum of (4.11), see § 4.3.3 for the experimental setup.

	SG-A	SE-A	SVRE	WS-SVRE
CIFAR-10	21.70	18.65	23.56	**16.77**
SVHN	5.66	5.14	**4.81**	**4.88**

Table 4.3 – Best obtained FID scores for the different optimization methods using the *deep* architectures (see Table C.6, § C.5.2). WS–SVRE starts from the best obtained scores of SE–A. See § C.5 and § C.6 for implementation details and additional results, respectively.

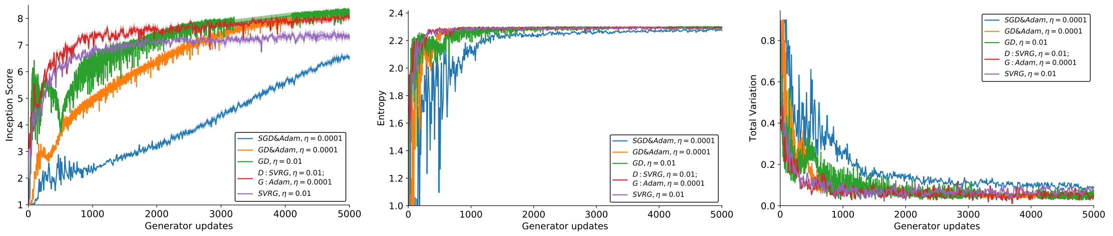
4.5.1 Results

Comparison on MNIST. The **MNIST** common benchmark allowed for comparison with full-batch extragradient, as it is feasible to compute. Fig. 4.2 depicts the IS metric while using either a stochastic, full-batch or variance reduced version of extragradient (see details of SVRE-GAN in § 4.4). We always combine the stochastic baseline (SE) with *Adam*, as proposed by Gidel et al. [2019]. In terms of number of parameter updates, SVRE performs similarly to BatchE–A (see Fig. C.1a, § C.6). Note that the latter requires significantly more computation: Fig. 4.2a depicts the IS metric using the number of mini-batch computations as x-axis (a surrogate for the wall-clock time, see below). We observe that, as SE–A has slower per-iteration convergence rate, SVRE converges faster on this dataset. At the end of training, all methods reach similar performances (IS is above 8.5, see Table C.7, § C.6). Fig. 4.4 compares SVRG-GAN with batch-gradient and stochastic-gradient, and in Fig 4.5 shows samples of SVRG-GAN trained on (Fashion) MNIST.

Computational cost. The relative cost of one pass over the dataset for SVRE versus vanilla SGD is a factor of 5: the full batch gradient is computed (on average) after one pass over the dataset, giving a slowdown of 2; the factor 5 takes into account the extra stochastic gradient computations for the variance reduction, as well as the extrapolation step overhead. However, as SVRE provides less noisy gradient, it may converge faster per iteration, compensating the extra per-update cost. Note that many computations can be done in parallel. In Fig. 4.2a, the x-axis uses an implementation-independent surrogate for wall-clock time that counts the number of mini-batch gradient computations. Note that some training methods for GANs require multiple discriminator updates per generator update, and we observed that to stabilize our baseline when using the *deep* architectures it was required to use 1:5 update ratio of G:D (cf. § C.6.3), whereas for SVRE we used ratio of 1:1 (Tab. 4.3 lists the results).

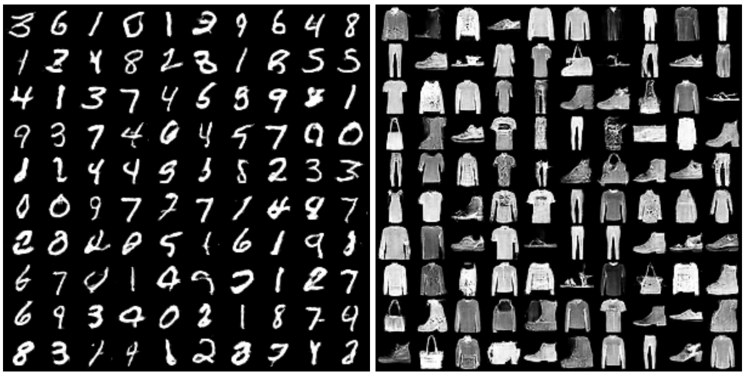
Second moment estimate and Adam. Fig. 4.2b depicts the averaged second-moment estimate for parameters of the Generator, where we observe that SVRE effectively reduces it over the iterations. The reduction of these values may be the reason why Adam combined with SVRE performs poorly (as these values appear in the denominator, see § C.3.1). To our knowledge, SVRE is the first optimization method with a constant step size that has worked empirically for GANs on non-trivial datasets.

Comparison on real-world datasets. In Fig. 4.2c, we compare SVRE with the SE–A baseline on **SVHN**, using *shallow* architectures. We observe that although SE–A in some experiments obtains better performances in the early iterations, SVRE allows for obtaining improved *final* performances. Tab. 4.3 summarizes the results on **CIFAR-10** and **SVHN** with *deep* architectures. We observe that, with deeper architectures, SE–A is notably more unstable, as training collapsed in 100% of the experiments. To obtain satisfying results for SE–A, we used various techniques such as a schedule of the learning rate and different update ratios (see § C.6.3). On the other hand, SVRE *did not collapse in any of the experiments* but took longer time to converge compared to SE–A. Interestingly, although WS–SVRE starts from an iterate point after which the baseline diverges, it continues to improve the obtained FID score and does not diverge. See § C.6 for additional experiments.



(a) IS [131] (higher is better) (b) Entropy (higher is better) (c) Total variation (lower is better)

Figure 4.4 – SVRG-GAN experiments on **MNIST**. “SGD–Adam” denotes stochastic-gradient with Adam, whereas “GD” denotes batch gradient. SVRG denotes full SVRG-GAN (Alg. 5), whereas with “D:SVRG, G:Adam” we denote a variant of it where we use variance reduced gradient only for the discriminator. Excluding batch-gradient, the mini-batch size $B = 50$ for the rest of the experiments.



(a) MNIST (b) Fashion-MNIST

Figure 4.5 – Generated samples of SVRG-GAN trained on **MNIST** and **FASHION-MNIST**.

4.6 Related work

Surprisingly, there exist only a few works on variance reduction methods for monotone operators, namely from Palaniappan and Bach [2016] and Davis [2016]. The latter requires a co-coercivity assumption on the operator and thus only convex optimization is considered. Our work provides a new way to use variance reduction for monotone operators, using the extragradient method [Korpelevich, 1976]. Recently, Iusem et al. [2017] proposed an extragradient method with variance reduction for an *infinite sum* of operators. The authors use mini-batches of growing size in order to reduce the variance of their algorithm and to converge with a constant step-size. However, this approach is prohibitively expensive in our application. Moreover, Iusem et al. are not using the SAGA/SVRG style of updates exploiting the finite sum formulation, leading to sublinear convergence rate, while our method benefits from a linear convergence rate exploiting the finite sum assumption.

Daskalakis et al. [2018] proposed a method called Optimistic-Adam inspired by game theory. This method is closely related to extragradient, with slightly different update scheme. More recently, Gidel et al. [2019] proposed to use extragradient to train GANs, introducing a method called ExtraAdam. This method outperformed Optimistic-Adam when trained on CIFAR-10. Our work is also an attempt to find principled ways to train GANs. Considering that the game aspect is better handled by the extragradient method, we focus on the optimization issues arising from the noise in the training procedure, a disregarded potential issue in GAN training.

In the context of deep learning, despite some very interesting theoretical results on non-convex minimization [Reddi et al., 2016, Allen-Zhu and Hazan, 2016], the effectiveness of variance reduced methods is still an open question, and a recent technical report by Defazio and Bottou [2018] provides negative empirical results on the variance reduction aspect. In addition, two recent large scale studies showed that increased batch size has: (i) only marginal impact on *single objective training* [Shallue et al., 2018] and (ii) a surprisingly large performance improvement on *GAN training* [Brock et al., 2019]. In our work, we are able to show positive results for variance reduction in a real-world deep learning setting. This unexpected difference seems to confirm the remarkable discrepancy, that remains poorly understood, between multi-objective optimization and standard minimization.

4.7 Discussion and future directions

Conclusion. Motivated by a simple bilinear game optimization problem where stochasticity provably breaks the convergence of previous stochastic methods, we proposed the novel SVRE algorithm that combines SVRG with the extragradient method for optimizing games. On the theory side, SVRE improves upon the previous best results for strongly-convex games, whereas empirically, it is the only method that converges for our stochastic bilinear game counter-example.

We empirically observed that SVRE for GAN training obtained convergence speed similar to

Batch-Extragradient on MNIST, while the latter is computationally infeasible for large datasets. For shallow architectures, SVRE matched or improved over baselines on all four datasets. Our experiments with deeper architectures show that SVRE is notably more stable with respect to hyperparameter choice. Moreover, while its stochastic counterpart diverged in all our experiments, SVRE did not.

However, we observed that SVRE took more iterations to converge when using deeper architectures, though notably, we were using constant step-sizes, unlike the baselines which required Adam. As adaptive step-sizes often provide significant improvements, developing such an appropriate version for SVRE is a promising direction for future work. In the meantime, the stability of SVRE suggests a practical use case for GANs as warm-starting it just before the baseline diverges, and running it for further improvements, as demonstrated with the WS–SVRE method in our experiments.

Results with deep nets & future directions. In summary, we observe the following most important advantages of SVRE when using *deep* architectures: (i) consistency of convergence, and improved stability; as well as (ii) reduced number of hyperparameters. Apart from the practical benefit for applications, the former could allow for a more fair comparison of GAN variants. The latter refers to the fact that SVRE omits the tuning of the sensitive (for the stochastic baseline) β_1 hyperparameter (see (C.46)), as well as r and γ–as training converges for SVRE without using different update ratio and step size schedule, respectively. It is important to note that the stochastic baseline does not converge when using constant step size (i.e. when *SGD* is used instead of *Adam*). In our experiments we compared SVRE that uses constant step size, with Adam, making the comparison unfair toward SVRE. Hence, our results indicate that SVRE can be further combined with adaptive step size schemes, so as to obtain both stable GAN performances and fast convergence when using these architectures. Nonetheless, the fact that the baseline either does not start to converge or it diverges later makes SVRE and WS–SVRE a promising approach for practitioners using these *deep* architectures, whereas, for *shallower* ones, SVRE speeds up the convergence and often provides better *final* performances.

Part II

Deep Learning Applications in Multi-Camera People Detection

5 Deep Multi-Camera People Detection

This chapter addresses the problem of multi-view people occupancy map estimation. Existing solutions either operate per-view or rely on a background subtraction pre-processing. Both approaches decrease the detection performance as scenes become more crowded. The former does not exploit joint information, whereas the latter deals with ambiguous input due to the produced foreground blobs becoming increasingly interconnected as the number of targets increases. Although deep learning algorithms have proven to excel on remarkably numerous computer vision tasks, such a method has not been applied yet to this problem. In large part, this is due to the lack of a large-scale multi-camera data-set.

The core of our method is an architecture which makes use of monocular pedestrian dataset, available at a larger scale than the multi-view ones, applies parallel processing to the multiple video streams, and jointly utilizes it. Our end-to-end deep learning method outperforms existing methods by large margins on the commonly used PETS 2009 data-set. Furthermore, we make publicly available a new three-camera dataset.

Related publication:

- T. Chavdarova and F. Fleuret, *Deep Multi-Camera People Detection*, in Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA), 2017

5.1 Introduction

Due to the high demand in applications, pedestrian detection differentiated itself as a separate class from object detection, enjoying separate attention from the research community. In spite of recent advances, performance of monocular methods remains limited due to the occlusions that often occur among the individuals. Multi-camera approaches offer a promising extension to resolve the one-view detection ambiguities, provided that the multi-stream information is used jointly to yield the detection estimation. In this chapter, we confine our discussion to a set-up of either a single calibrated static camera, or several synchronized cameras with

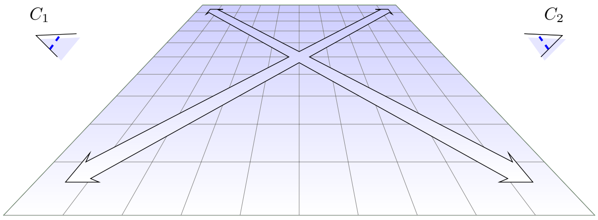


Figure 5.1 – Illustration of the people occupancy map estimation problem in a static camera setup. As an example we depict two cameras and the area is discretized into a 10×10 grid cells, thus following the notation in the text: C=2, G=100.

overlapping fields of view.

Surprisingly, currently only few multi-view methods jointly utilize the information across views, herein referred to as *joint* methods. Each of the existing joint methods performs background subtraction pre-processing of the input images, where the goal is to segment the moving objects out of the background, while taking into account pixel-wise time consistency. There exist a vast catalog of background-subtraction methods, specific to certain applications or even illumination regimes, which are often shown to be difficult to tune, error-prone and noisy. Regarding multi-camera people detection in particular, these segmenting methods introduce ambiguities when the foreground blobs are highly interconnected, hence limiting the success of the joint methods to less-crowded applications. Also, apart from persons, other moving objects are being segmented as well.

On the other hand, state of the art monocular detectors leverage the full signal either by using a deep Convolutional Neural Network (CNN), or by building predictors on top of "deep features" extracted by such a network. Surprisingly, the multi-view occupancy map estimation problem has not been re-visited to incorporate these outperforming methods.

Quite the contrary, the ongoing research on multi-camera people detection incorporates hand-crafted features, whereas to the best of our knowledge real-world industrial applications use monocular CNN pedestrian detector, and yield the final estimation by averaging the separate per-view ones. We presume that this is mostly due to the non-existence of a large-scale multi-camera dataset which would allow for training a multi-input architecture.

As a solution, we propose a method which consists of: (i) fine-tuning a state of the art object detection network on monocular pedestrian detection; (ii) combining several instances of the early layers of that network into a multi-view deep network whose outer layers are trained for multi-view appearance-based joint detection on relatively smaller multi-camera dataset. Our proposed architecture allows for processing the input from the separate views in parallel, and the subsequent interconnection layers allow for automatically learning how features across different views map into each other. In this chapter, we focus on per-frame processing, whereas utilising time consistency could be a further extension.

In summary, we: (i) propose the first full deep learning multi-camera people detector; (ii) we empirically demonstrate the superiority of such a method; and we (iii) provide a new three--view dataset with fairly more accurate calibration in terms of consistency of a projection across all of the views.

The monocular methods overview in § 5.2 is of those methods which utilize deep learning, whereas a complete overview is given for the more closely related methods to this work - the joint multi-camera people detectors. The problem on which this chapter focuses is formally stated in § 5.3, where we also introduce the multi-view architecture that we propose. Our empirical evaluation presented in § 5.4 demonstrates the encouraging superiority of the deep multi-view detection, and also gives insights about practical considerations which arise.

5.2 Related work

5.2.1 Deep monocular pedestrian detection

Applying the R-CNN algorithm [Girshick et al., 2013] to monocular pedestrian detection [Hosang et al., 2015] exceeded the state of the art methods at that time on the Caltech pedestrian dataset [Dollar et al., 2009], being the first demonstration that CNNs are well-suited to the task. Later it was shown that the explicit dealing with pedestrian occlusions, by fine-tuning a separate fully convolutional network for sub-parts of the bounding boxes, provides a 50% relative improvement [Tian et al., 2015]. On top of the part-designated CNNs the authors employ a linear SVM, sparsified to reduce the computational load.

Cai et al. [2015] propose a Boosting algorithm, which merges the proposal generation and the detection steps. The cascade learning is formulated as Lagrangian optimisation of a risk accounting for both accuracy and complexity, and integrates both hand-crafted and convolutional features. The downstream classifier of the extension of R-CNN known as Faster R-CNN [Ren et al., 2015] was recently shown to degrade the performance, and it was proposed to be replaced by boosting forests [Zhang et al., 2016]. The resulting method does not use hand-crafted features and reaches state of the art accuracy.

5.2.2 Multi-view occupancy map estimation methods

The first method which uses multi-view streams jointly is the Probabilistic Occupancy Map (POM), proposed by Fleuret et al. [2008]. Based on a crude generative model, it estimates the probabilities of occupancy through mean-field inference, naturally handling occlusions. Further, it can be combined with a convex max-cost flow optimization to leverage time consistency [Berclaz et al., 2009]. Alahi et al. [2011] re-casted the problem as a linear inverse, regularized by enforcing a sparsity constraint on the occupancy vector. It uses a dictionary whose atoms approximate multi-view silhouettes. To elevate the need of O-Lasso computations, Golbabaee et al. [2014] derived a regression model which includes solely Boolean

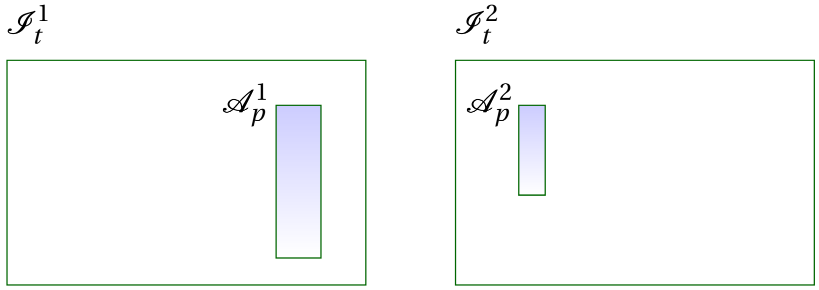


Figure 5.2 – The input to the model when estimating q_p consists of the cropped regions $\mathbf{A}_p$ of $\mathbf{I}_t$ (see § 5.3.1). The illustrated example is in-line with the one in Fig. 5.1.

arithmetic and sustains the sparsity assumption of [Alahi et al., 2011]. In addition, the iterative method is replaced with a greedy algorithm based on set covering.

In [Peng et al., 2015] the occlusions are explicitly modeled per view by a separate Bayesian Network, and a multi-view network is then constructed by combining them, based on the ground locations and the geometrical constraints. Although considering crowd analysis, the multi-view image generation of [Ge and Collins, 2010] is with a stochastic generative process of random crowd configurations, and then maximum a posteriori (MAP) estimate is used to find the best fit with respect to the image observations.

All of these methods utilise background subtraction pre-processing.

5.3 Deep Multi-View People Detection

5.3.1 Problem definition

We discretize the area common to the fields of view of C cameras in a regular grid of G points or interchangeably - cells (Fig. 5.1). To estimate the occupancy of a cell p, we consider a cylinder centred at the pth position, whose height corresponds to the one of the humans' average height. We use the cameras' calibration to obtain the cylinder's projections into the views where it is visible. These rectangular projections yield the cropped regions $\mathbf{A}_p$ of $\mathbf{I}_t$, as illustrated in Fig. 5.2.

At each time step t we are given a set of images $\mathbf{I}_t = \mathscr{I}_t^1, \ldots \mathscr{I}_t^C$ taken synchronously. Hence, given sub-images cropped for a particular position $p = 1, \ldots, G$ in the separate views $\mathbf{A}_p = \left(\mathscr{A}_p^1, \ldots, \mathscr{A}_p^C\right)$, we aim at estimating the probability of it being occupied, that is,

$$q_p = p(X_p = 1 \mid \mathbf{A}_p),$$

where X_p on $\{0, 1\}$ stands for the position p being free and occupied, respectively. In the following, let $\mathscr{I}$ be the set of all possible cropped sub-images.

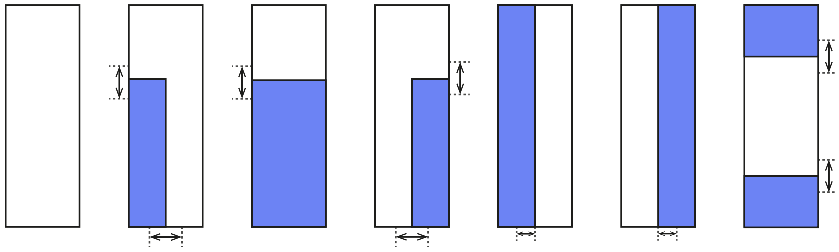


Figure 5.3 – Input occlusion masks. Among these, the first one represents no occlusion. As the arrows indicate, their width and/or height are restrictively randomized within a margin.

5.3.2 Method

Ideally, given a large scale multi-camera dataset, a multi-stream processing model can be directly trained. However, due to the lack of such dataset, as well as the fact that in practice the annotated data is often scarce, the method we propose takes advantage of the existing larger monocular pedestrian dataset Caltech [Dollar et al., 2009] what allows for better generalization.

Monocular fine-tuning with input-dropout

Primarily, to learn discriminative features of the pedestrian class, we train a CNN on monocular pedestrian binary classification. To ease the training procedure we recommend starting from a pre-trained object detection CNN. In particular, in our experiments we used either GoogLeNet [Szegedy et al., 2015] or AlexNet [Krizhevsky et al., 2012], trained on ImageNet [Russakovsky et al., 2015] and we replaced the last fully connected layer with a randomly initialized one with two output units. We then fine-tune the resulting network on the Caltech dataset (see § 5.4.1).

Experimental results on monocular pedestrian detection indicate that state of the art detection accuracy is achieved by explicitly dealing with the occlusions (for example by detecting parts of the body), as explained in § 5.2.1. As we shall see in the following sections, the multi-view architecture finally contains components initialized with the weights of the monocular detector. Hence, it is necessary that the robustness to occlusion of the model is implemented efficiently. Thus, we propose a novel technique dubbed *input dropout*, which consists in augmenting the input data by artificially masking part of the signal. While this procedure is adapted to the morphology of the positive class, it has to be carried out on both the positive and the negative samples while training, so as to avoid providing erroneous discriminative cues.

In our experiments in particular we defined 7 rectangular occluding masks–#1 being "no occlusion at all"–(Fig. 5.3), and for each sample, negative or positive, we pick one of the masks uniformly at random, and replace the pixel it masks with white noise. As for drop-out, this "input drop-out" forces the network toward greater redundancy between representations.

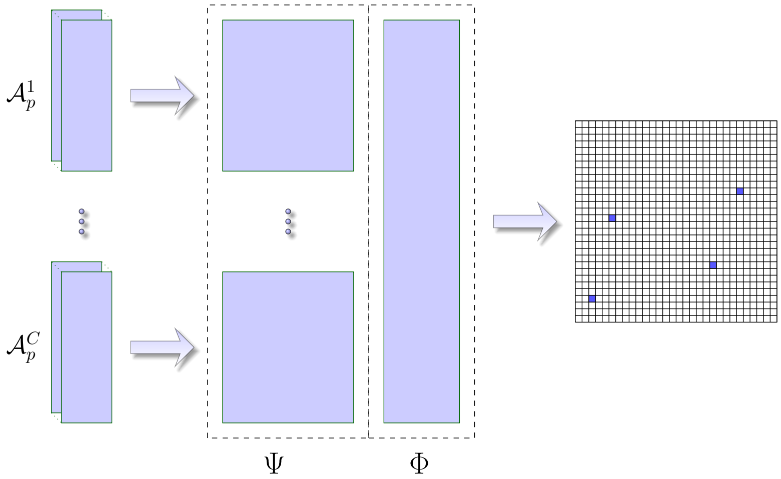


Figure 5.4 – Multi-view joint occupancy estimation. The input is illustrated on the left; followed by the multi-view architecture consisting of two conceptual parts Ψ and Φ (§ 5.3.2); and on the rightmost side a top-view grid of the predictions is plotted where a coloured cell represents a detection.

Multi-camera architecture

Given a CNN monocular detector, we retain d of its layers, resulting in an embedding $\psi : \mathscr{I} \rightarrow \mathbb{R}^Q$, where Q is the number of output units of the dth layer. The concatenation of C such embeddings is a multi-view embedding $\Psi : \mathscr{I}^C \rightarrow \mathbb{R}^{CQ}$, on top of which we train a binary classifier $\Phi : \mathbb{R}^{CQ} \rightarrow \mathbb{R}^2$, as illustrated in Fig. 5.4.

Type of classifier of Φ. Particular object seen from different perspectives/cameras can be discriminated both with features which are consistent across views: *e.g.* color; and with features that differ across the views: for example a line curved to the left would be curved to the right viewed from the opposite side. The former is more likely when the two cameras are quite close to each other, and the latter in the opposite case. The goal of Φ is to learn how features from different views map to each other. Thus, a natural choice for ϕ is a multi-layer perceptron (MLP), in aim to "wire together what fires together". In our experiments the output of Ψ is flattened, and Φ is a MLP, and also for demonstrative analysis Φ is sometimes a (forest of) decision tree(s).

Full or partial fine-tuning. To allow the embedding to capture cues discarded for monocular detection but which are informative for multi-view, the complete multi-stream architecture, that is both Ψ and Φ, can be trained, when the latter is differentiable. On the other hand, keeping Ψ fixed makes the capacity of the predictor family low, and hence prevents over-fitting when the training multi-view dataset is small, *e.g.* in the order of few hundreds of examples. Given the size of the datasets we have used, this approach performed worse than keeping Ψ fixed.

Non Maxima Supression (NMS)

This step selects the final detections so that out of those candidates overlapping more then a predefined threshold in a particular view, only one remains. Differently than standard implementations, we take into account the detections' scores, in such a way that the priority of a detection candidate to be selected is proportional to its detection score.

5.4 Experiments

5.4.1 datasets and metrics

Caltech10x [Dollar et al., 2009]. For monocular training, we use the Caltech-USA pedestrian dataset and its associated benchmarking toolbox[113]. This set consists of fully annotated 30 Hz videos taken from a moving vehicle in a regular traffic, and as in other recent works, we use 10 fps sampling. This increases the training data to $\simeq 2 \cdot 10^4$ detections. We generate proposals with the *SquaresChnFtrs* detector [Benenson et al., 2015], and use as positive examples the proposals with 0.5 overlap threshold.

PETS 2009 S1 L2 dataset [Ferryman and Shahrokni, 2009]. For comparison with existing methods we use the PETS 2009 dataset, which is a sequence of 795 frames recorded with seven static outdoor cameras. As noted by other authors: [Peng et al., 2015, p. 10], or [Ge and Collins, 2010, p. 10], we also observe inconsistencies both in terms of calibration and synchronization.

EPFL-RLC dataset [RLC]. This new corpus was captured with three calibrated HD cameras, with a frame rate of 60 frames per second. Currently the annotations represent a balanced set of 4088 multi-view examples. Note that a negative multi-camera example may contain a pedestrian in one of its views. Thus, for each view the negative samples contain a bit of annotated information wheter it contains a pedestrian or not. This allows the dataset to be used for monocular pedestrian training in which case its size is increased three-fold.

Full ground-truth annotations are provided for the last 300 frames of the sequence, intended to be used for testing while ensuring diversity of the appearance with respect to the training data. As for future work, we also make available the full sequence of 8000 synchronized frames of each view.

Metrics. Apart from standard classification measures, we use the Multiple Object Detection Accuracy (MODA) metric, accounting for the normalized missed detections and false positives, as well as the Multiple Object Detection Precision (MODP) metric, which assesses the localization precision [Kasturi et al., 2009]. We also estimate the empirical precision and recall, calculated by $P{=}TP/(TP{+}FP)$ and $R{=}TP/(TP{+}FN)$ respectively, where: TP, FP, and FN are true positives, false positives and false negatives, respectively.

5.4.2 Monocular pedestrian detection

We conducted our experiments using the Torch framework [Collobert et al., 2011], and mainly GoogLeNet [Szegedy et al., 2015] and AlexNet [Krizhevsky et al., 2012]. The input dimension for these models are respectively 224×224 and 227×227. As a performance measure we use the log average miss rate (MR), as it is widely adopted.

When using GoogLeNet, batch size of 64, learning rate 0.005, SGD with momentum 0.9, as well as a proportion of positive samples per mini-batch of $r = 0.33$, we observe mean MR of 19.81 (σ=0.58) on the test data when the learning starts to saturate. When input dropout is applied the mean MR drops to 17.32 (σ=0.75), and further continues to decrease, although with slight variance. Applying it allowed for reaching MR performances which were never reached with standard training, and the observed variance suggests that combining it with more sophisticated learning rate policy can provide further performance gain. When using AlexNet, the mean MR is 24.04 (σ=0.39), and 22.43 (σ=0.51), without and with input dropout, respectively.

In summary, we consistently observed performance improvement when using: (i) GoogLeNet, (ii) square cropping versus warping the region, (iii) proportion of r=0.33 positive samples per mini-batch, as well as (iv) input dropout (§ 5.3.2). We reduced the MR to 15.61%, which although is higher than the results reported in [Tian et al., 2015], requires six-fold less computation and is simpler to implement and deploy.

5.4.3 Multi-camera people detection

In our experiments, we discretize the ground surface of the EPFL-RLC and PETS datasets to grids of 45×55 and 140×140 positions, respectively. To train the multi-stream model on the PETS dataset we automatically extract negative examples which outnumber the positive ones provided by the annotated detections. During one epoch of training, these negatives are sampled without replacement. Both on EPFL-RLC and PETS datasets we observe improved performances when training with forced ratio of increased negative samples per mini-batch.

Contrary to the monocular training, we observed no performance drop of training Φ if rectangular input is being used, which is why we extract features with reduced input in the width dimension by omitting 50 pixels of both sides. In our implementation, we zero-pad the input of the view for which a particular position is out of its field of view.

Unless otherwise stated, we use MLP with 3 fully connected layers, ReLU non-linearities and log-softmax. Natural consideration is the use of regularization as intuitively the weight vector of the fully connected layers would be sparse. We experimented with the p-norm regularization term: $\|w\|_p = (\sum_{i=1}^{N} |w_i|^p)^{1/p}$, both with $p = 1, 2$, while using SGD optimization, but did not observe a substantial improvement. In fact, by visualizing the weights when training without regularization we observe that major part of the weight vector tends towards zero, and that the training was more stable.

We provide bellow an empirical analysis of the impact on performance of some aspects of our setup.

Training with hard-negative auto-generated examples

Since the EPFL-RLC dataset demonstrates more accurate joint-view calibration then PETS'09 we are able to automatically generate multi-view negative samples which we refer to as *hard negatives* and which force Φ to compare the multi-stream extracted features in a way that generalizes better.

We perform this automatically in two different ways: (i) by shifting the rectangles of a positive detection in one or two of the views, or (ii) by combining positive detections of different persons. Training with these hard negatives allows respectively for: (i) sharper detections in the ground plan, which makes the method less sensitive to the non-maxima suppression post-processing; as well as (ii) forcing the classifier Φ to learn a stronger joint appearance model. The former improve localization and the latter to decrease the false positives of the detector. Note that in the experimental results presented in the following, changing the ratio r implies changing the difficulty of the classification for this dataset and as we shall see hurts the accuracy.

From the results illustrated in Fig. 5.5 we observe that training with such negatives improves the most important MODA metric and reaches almost perfect precision, at the cost of marginally decreasing the MODP metric, and decreased recall due to the slightly increased number of missed detections. However, as the camera calibration of the PETS dataset demonstrates non-negligible joint-camera inconsistency of the projections, we did not experiment with the multi-view hard negatives for it, to avoid increasing the noise level of the input.

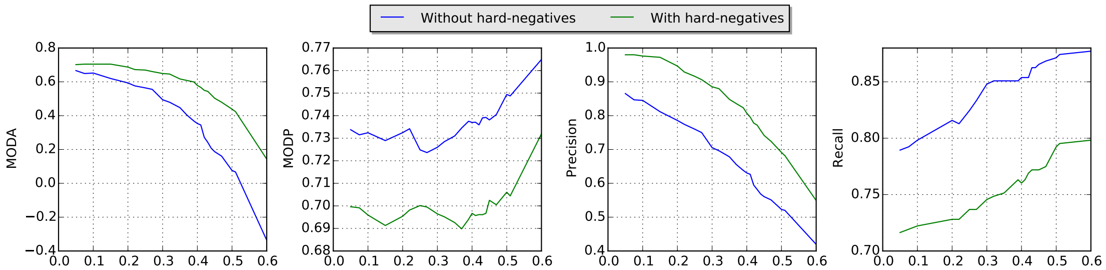


Figure 5.5 – Impact of training with hard-negative multi-view samples, versus the NMS threshold (x-axis) on the EPFL-RLC dataset.

Monocular Vs. Multi-view classification

We compare performance with and without using multiple views. As the performance using a MLP for Φ is potentially highly dependent on the persistence of tuning the hyper-parameters and/or the choice of optimization methods, in this section we use solely random forests for Φ.

For this purpose, we use the accuracy and the ROC while considering all possible (sub)sets of the available views, both on PETS and EPFL-RLC, as well as for different values of the proportion r of positive samples. In all experiments, we consistently observe performance improvement as more views are being added, as illustrated on Fig. 5.6.

Impact of the Selected Layer on the Accuracy

We use GoogLeNet and compare using the first 11, 15, 20 and 23 layers. After flattening the output of each of the layers, we add fully connected layers, each of 1024, 512 and 2 units. We experiment with: (i) mini-batch size of: 32, 64, 128; (ii) optimisation technique: SGD w/o momentum, Adam [Kingma and Ba, 2015], Adadelta [Zeiler, 2012] and RMSPRop [Tieleman and Hinton, 2012]; as well as (iii) different initial values for the learning rate when necessary. We run each experiment for 60 epochs on the EPFL-RLC dataset, and we list in Table 5.1 the best test accuracy at that point for each of the selected layers. In the majority of the experiments we observe best performances using Adadelta or ADAM, and batch size of 64.

Our experiments indicate that the depth of the CNN used as feature extractor does not impact

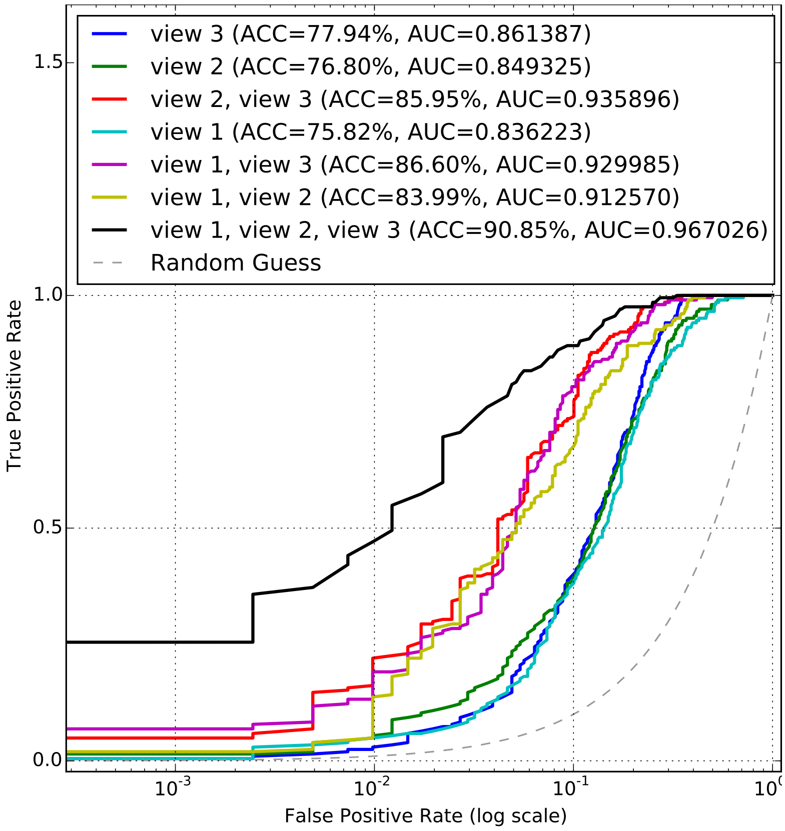


Figure 5.6 – Accuracy, ROC and AUC on the EPFL-RLC test data, using random forest of 100 trees, with $d = 23$, $r = 0.33$ and GoogLeNet (§ 5.4.3).

much the performance. This is encouraging since shallower structures can be used what decreases the computation time when the model is deployed.

Multi-view Information

Herein we focus on the question if multi-view joint information exploitation allows for easier discrimination among the two classes versus monocular classification. To this end, we analyze if features from the different views are used uniformly, or if features from one of the views dominate the decision. The most straightforward way to illustrate this is by using a single decision tree for Φ. Legitimately, we define the importance of a feature in terms of its depth in the decision tree: the higher, or the closest to the root, the more important a feature is.

We used the EPFL-RLC dataset and GoogLeNet with $d = 23$, which for the three views provides us with 3×1024 features. We visit the nodes from the root to the leaves, and count the features from each view. We find that among the 50, 100 and 150 top-nodes, the numbers of features from each view are 16/15/19, 36/32/32, and 55/51/44, respectively. This shows that the classifier exploits the views in quite a balanced manner, what motivates multi-view joint detection.

5.4.4 Comparison to existing methods on PETS

Finally, in Table 5.2 we compare our method with the current state of the art methods. We refer as *ours* when the full PETS sequence is used as testing and the model has never being trained on it; and as *ours-ft* the experiments where we divide the sequence to training for fine-tuning the model, and testing. In the latter we consider all the possible splits to train and test frames, hence we list the mean and the standard deviation of the particular performance measure estimated through these multiple folds.

As the results in Table 5.2 show, without data normalization specific to this dataset and without fine-tuning we outperform the existing methods, what indicates generalization property which is of interest. These results justify the multiple steps of our approach.

We recall the calibration joint inaccuracy of PETS which becomes more prominent as the number of used views of it increases. This homograph mapping deterioration originates in

Table 5.1 – Test accuracy on EPFL-RLC, for d layers, proportion r of positive samples per batch (see § 5.3.2) and with different level of difficulty (see § 5.4.1).

r	$d = 11$	$d = 15$	$d = 20$	$d = 23$
0.50	99.75%	100.00%	99.75%	98.77%
0.33	96.24%	97.05%	96.73%	94.60%
0.25	91.78%	83.57%	83.70%	81.74%

Table 5.2 – Comparison on the PETS dataset (see § 5.4.4). The views' enumeration omits the second view, which was removed from the dataset by its authors.

Method	Views	MODA	MODP	Precision	Recall
[Peng et al., 2015]	1,4,5,7	0.79	0.74	0.92	0.91
[Ge and Collins, 2010]	1,4,5,7	0.75	0.68	0.85	0.89
Ours-ft	1	0.94 (0.03)	0.61 (0.02)	0.97 (0.02)	0.96 (0.02)
Ours	1	0.88	0.75	0.97	0.91
Ours-ft	1,4,5,7	0.90 (0.05)	0.66 (0.02)	0.94 (0.03)	0.96 (0.02)
Ours-ft	1-7	0.94 (0.02)	0.68 (0.02)	0.98 (0.01)	0.97 (0.02)

part from the presence of a slope in the scene [Peng et al., 2015]. To the best of our knowledge researchers avoid using the seven views at the same time. Instead, most often reported is the case of using the views $1,4,5$ and 7 (or $1,5,6,8$ in the original views' enumeration). It is interesting that in our experiments we do not observe performance drop when all of the seven views are used, and that in fact we observe marginal improvement. This could be due to the contextual padding that we recommended earlier. Further, the fact that the MODP metric increases, indicates that Φ is able to learn these calibration inconsistencies.

Our method outperforms the existing state of the art baselines, largely owing to its ability to leverage appearance, and to filter out non-human foreground objects that a background-subtraction procedure may trigger.

5.5 Conclusion and Future Work

Our main contribution is proposing end-to-end deep learning method for the multi-camera people detection problem whose core is an architecture adapted to make use of existing monocular datasets as for improved generalisation, and which jointly leverages the multi-stream deep features. As such it outperforms the existing approaches.

It was shown that: (i) multi-view classification increases the accuracy and the confidence of the classifier over the monocular case, as well as that (ii) appearance features extracted jointly from multiple views provide more easily separable embedding that in turn allows for more accurate classification than the monocular one. We also discussed various implementation insights.

Given larger scale datasets, extensions such as: (i) more explicit multi-view occlusion reasoning; and (ii) training domain-adaptation modules, can further be done. The former would allow for merging the two steps of generating detection candidates and selecting them, whereas the latter would allow for easier adaptation for the task at hand while preserving automatically learnt multi-view features.

6 WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection

In this chapter, we present a new large-scale and high-resolution dataset named *WILDTRACK*. This dataset aims to speed up the development of deep learning methods for the problem of multi-camera people detection with statically positioned cameras. The most notable advantages over the existing datasets are a large number of annotated samples and its precise calibration for which we developed a method to *jointly* obtain the camera calibration parameters. Moreover, we provide a series of benchmark results using baseline algorithms published over the recent months for multi-view detection with deep neural networks, and trajectory estimation using a non-Markovian model. This chapter is based on the following publication:

- T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret, *WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection*, in Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR), 2018.

6.1 Introduction

Pedestrian detection is an important computer vision problem with numerous applications in security, surveillance, robotics, autonomous driving, and crowdsourcing. The variation of pedestrians appearance greatly increases the difficulty of this problem. With the availability of large-scale monocular datasets of annotated pedestrians and the advances in detection algorithms, the accuracy of the pedestrian detectors has improved significantly in the past few years. Moreover, modern detection algorithms using deep learning allow us to learn discriminative features which are transferable across datasets. Impressively, recently developed deep learning based monocular detectors are approaching human-level performance [Zhang et al., 2016] on common benchmark datasets [Du et al., 2017].

However, many situations of practical interest require detection in highly crowded and cluttered scenes. Severe occlusions make monocular pedestrian detection insufficient in these scenarios. Luckily, in real-world applications, image feeds from multiple cameras with overlapping fields of view are often available. Most commonly, the cameras are positioned slightly above the average human height. Hence designing pedestrian detectors by exploiting multiple views and the geometry of the scene will provide reliable detection estimates in crowded scenes.

Surprisingly, to nowadays standards, there is no large scale and good quality public dataset that replicate this setup. The most frequently used one, PETS 2009 [Ferryman and Shahrokni, 2009] is only in the order of several hundreds of frames long. The provided camera calibration is inaccurate which makes it difficult to exploit geometrical constraints jointly. Moreover, it is recorded in a so-called *actor set-up*, meaning that throughout the sequence, a very limited total number of different individuals actually appear. All other standard datasets are either much shorter and scarcely crowded [Fleuret et al., 2008, Xu et al., 2016, Berclaz et al., 2011], have a very constrained scenario [Alameda-Pineda et al., 2016] or use non-overlapping cameras [Ristani et al., 2016].

The lack of such a dataset seriously limits the development of multi-camera detection methods. Recent improvements made by the community call for a more realistic and challenging benchmark that could be used to compare multi-camera detection methods. For example, Chavdarova and Fleuret [2017] show that utilizing the multi-camera input for deep learning based detectors improves both the accuracy and the classification confidence. However, these methods are severely limited by the lack of a large-scale dataset to train on without over-fitting. As a direct consequence, most of the existing joint methods use *ad-hoc* techniques to combine information extracted using pre-trained monocular detectors.

To help in accelerating the research on methods taking advantage of the multi-camera set-ups, we introduce a *large-scale person dataset acquired with seven static cameras, with overlapping fields of view*. It captures a realistic unscripted scenario where pedestrians often occlude each other. We provide a very precise joint (extrinsic and intrinsic) calibration and synchronization of sequences from the 7 views as well as 7 series of 400 annotated frames for detection at a rate of 2 frames per second. This results in over 40 000 bounding boxes delimiting every person present in the area of interest, for a total of more than 300 individuals. The annotations of the individual tracks are provided both as 3D locations on the ground plane and 2D bounding-boxes projected in each of the 7 views. Although our dataset is designed to benchmark 3D multi-camera detection, it can also be used for monocular detection. In the monocular case, the size of our dataset increases seven-fold which makes it comparable to the widely used Caltech-USA dataset [Dollar et al., 2009]. Compared to the latter, our dataset has *much higher image resolution*.

We make the source code of our web-based annotation platform public in order to encourage other researchers to collect and annotate other multi-camera datasets. Finally, we also provide annotations for evaluating camera calibration methods.

This paper is organized as follows: in Section 6.2 we make a review of multi-camera person datasets and related methods. Section 6.3 enumerates details of the new dataset, including our camera calibration procedure. We benchmark several state-of-the-art multi-camera detection methods in Section 4.5 and finally in Section 6.5 we present potential future research directions.

6.2 Related work

6.2.1 Datasets

Table 6.1 – Commonly used person datasets with a focus on the multi-camera ones. *FPS*, *pos* and *imp* stand for frames per second, positive images and image pairs, respectively. Where applicable, with '+' we denote the pre-defined splits to train and test data. See § 6.2.1.

Dataset	Resolution	Cameras	FPS	Mobile/Static	Overlapping	Video	IDs	Annotations	Size/Duration
INRIA [31]	high	n/a	n/a	n/a	n/a	No	-	1200+566	614+288 pos.
ETH [43, 44] (5 seq.)	640×480	1[‡]	15	M	n/a	Yes	-	2853 (~4fps) & 10398 (15fps)	4203 frames

TUD-Brussels [149]	7720×576;
	640×480;
		1	1	M	n/a	Yes	-	1776+1326 (pos.)	1092+508 pos.imp.

Daimler [42]	640×480	1[*]	n/a	M	n/a	No	n/a	2400+1600	-
Daimler-stereo [81]	640×480	1[*‡]	15	M	n/a	Yes	-	3915	15600+56500
Caltech-USA [38]	640×480	1	30	M	n/a	Yes	-	$\sim 2\cdot 10^4$(10fps)+$1\cdot 10^3$(1fps)	~10 hours
KITTI [51]	1392×512	4[†‡]	10	M	n/a	Yes	-	194+195 imp.	7 min.
APIDIS [35]	1600×1200	7	22	S	Yes	Yes	12	86870 (25fps)	1 min.

PETS 2009 [46]	768×576;
	720×576;
		7	7	S	Yes	Yes	19	4650 (7fps)	795 frames

DukeMTMC [123]	1920×1080	8	60	S	No	Yes	~2000	4077132 (60fps); 9668 trajectories.	85 min.
Laboratory [47]	320×240	4	25	S	Yes	Yes	6	476 (1fps)	2.5 min.
Terrace [47]	320×240	4	25	S	Yes	Yes	9	1023 (1fps)	3.5 min.
Passageway [47]	320×240	4	25	S	Yes	Yes	13	226 (1fps)	20 min.
Campus [153]	1920×1080	4	25	S	Yes	Yes	25	240 (1fps)	4×4 min.
SALSA [5]	1024×768	4	15	S	Yes	Yes	18	1200 (0.3fps)	60 min.
EPFL-RLC [23]	1920×1080	3	60	S	Yes	Yes	-	~3×2044a.+300frames	8000 frames
WILDTRACK [25]	1920×1080	7	60	S	Yes	Yes	313	~7×9518 (2fps)	~60 min.

[*] No color channels.
[†] 2 color and 2 grayscale cameras.
[‡] Stereo camera(s).

In Table 6.1 we list the commonly used pedestrian datasets with a focus on the multi-camera ones. For a more exhaustive listing of the monocular datasets, we refer the reader to [Dollar et al., 2012, chap. 2.4]. As *overlapping* we refer to multi-camera datasets whose camera's fields of view strictly overlap. The DukeMTMC [Ristani et al., 2016] dataset does not belong to this category, as only 2 of its camera's fields of view slightly overlap.

The most widely used dataset with an overlapping camera set-up is the PETS 2009 S2.L1 [Ferryman and Shahrokni, 2009] sequence. In part due to the presence of a slope in the scene, the provided calibration poses large homography mapping deterioration and inconsistencies when projecting 3D points across the views (as noted also in [Peng et al., 2015, p. 10], [Ge and Collins, 2010, p. 10], [Chavdarova and Fleuret, 2017, p. 3]). Besides being a small scale dataset, the PETS 2009 S2.L1 is acquired in an actor set-up. Hence it does not allow for good generalization and fair benchmarking of appearance-based methods.

The three sequences shot at the EPFL campus [Fleuret et al., 2008]: Laboratory, Terrace and Passageway, as well as SALSA [Alameda-Pineda et al., 2016] and Campus [Xu et al., 2016] are overlapping multi-camera datasets as well. However, they have a small number of total identities and are relatively sparsely crowded. As we can see from Table 6.1, Laboratory, Terrace and Passageway are of small size and low image quality. In SALSA [Alameda-Pineda et al., 2016], a cocktail party of 30 minutes is recorded, where the people are static most of the time, making this dataset less challenging for tracking. Finally, Campus does not provide the annotations of the $3D$ locations of the people.

The EPFL-RLC [Chavdarova and Fleuret, 2017] dataset demonstrates improved joint-calibration accuracy and synchronization compared to PETS. However, rather than providing a complete ground-truth, this dataset represents a collection of a balanced set of positive and negative multi-view annotations and is used for classification of a position as occupied by a pedestrian or not. Full ground-truth annotations are provided solely for a small subset of the last 300 of the total 8000 frames, originally used for testing [Chavdarova and Fleuret, 2017]. Moreover, in comparison to ours, it is acquired with only three cameras which have a much more limited field of view. This results in a ~10-fold smaller number of detections per frame on average.

Hence, WILDTRACK improves upon other multi-camera person datasets thanks to: (i) the high precision calibration and synchronisation between the cameras (see § 6.3.2), (ii) the large number of annotations that allows for developing deep learning based multi-view detectors. With regard to the state-of-the-art monocular datasets [Dollar et al., 2009]: (i) it exceeds the total number of annotations; and (ii) the regions of interest (ROIs) are of significantly larger resolution.

6.2.2 Methods

Probabilistic Occupancy Map method (POM) [Fleuret et al., 2008] is a generative model which estimates the probabilities of occupancy on the ground plane by exploiting the geometrical constraints from multiple views. POM is formulated in the framework of mean-field inference which naturally handles the occlusions. To leverage time consistency it can be combined with a convex max-cost flow optimization [Berclaz et al., 2009] for tracking.

In [Alahi et al., 2011], multi-view detection is re-casted as a linear inverse problem. Their model is regularized by enforcing a sparsity constraint on the occupancy vector. It uses a dictionary whose atoms approximate multi-view silhouettes. To alleviate the need for the time demanding Lasso-based [Candès et al., 2008] computations, a regression model is derived by [Golbabaee et al., 2014]. This model solely comprises of Boolean arithmetic and sustains the sparsity assumption of [Alahi et al., 2011]. In addition, the iterative method of [Alahi et al., 2011] is replaced with a greedy algorithm based on set covering.

In [Peng et al., 2015] the occlusions are modeled explicitly per view by a separate Bayesian Network. A multi-view network is then constructed by combining them, based on the ground locations and the geometrical constraints.

Considering crowd analysis, in [Ge and Collins, 2010] the multi-view image generation is modeled with a stochastic generative process of random crowd configurations and then maximum a posteriori (MAP) estimate is used to find the best fit with respect to the image observations.

The Deep Multi-camera Detection (DeepMCD) [Chavdarova and Fleuret, 2017] method which integrates CNN features demonstrated state-of-the-art results and showed that the accuracy and the confidence of a CNN classifier increase as more views are used. To mitigate the data requirement problem and improve generalization, the authors first used the larger monocular dataset Caltech [Dollar et al., 2009] to train a base network. The multi-view CNN architecture is adapted so as to use the weights from the base network and produces joint estimates by processing the multi-view streams in parallel. To increase the sharpness and the localization accuracy, [Chavdarova and Fleuret, 2017] also proposes two particular schemes of multi-view hard negative mining to train such a model.

The Deep-Occlusion Reasoning method [Baque et al., 2017] uses a joint CNN-CRF architecture and Mean-Fields inference to produce a Probabilistic Occupancy Map (POM) as in [Fleuret et al., 2008] while leveraging discriminative features extracted by a CNN. It introduces a Higher Order CRF, where unary potentials are produced by ROI pooling CNN [Ren et al., 2015]. Higher-order potentials are computed as a measure of the consistency between pixel labels produced by a fully convolutional network and a generative model which accounts for geometry and occlusions.

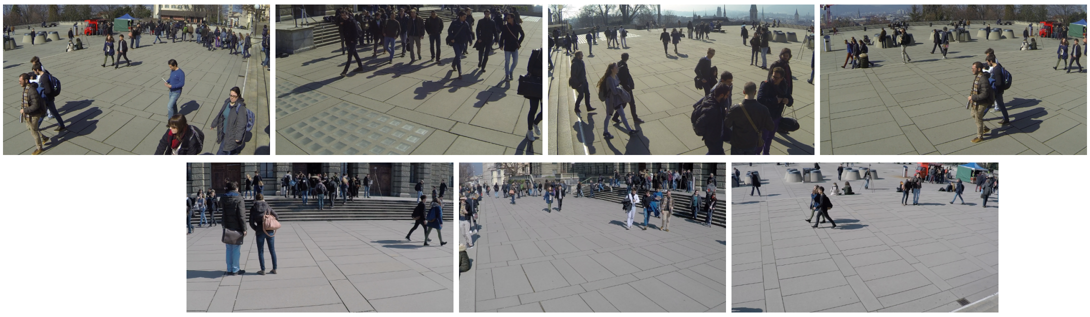


Figure 6.1 – Synchronized corresponding frames from the seven views. Four *GoPro Hero 3* and three *GoPro Hero 4* were used, frames of which are shown in the top and bottom row, respectively.

Tracking multiple objects in multiple overlapping cameras, as well as tracking in a single view, mostly follows the tracking-by-detection paradigm [Andriluka et al., 2008]. However, due to the scarcity of datasets with multiple overlapping cameras, tracking multiple objects in multiple overlapping cameras has received relatively little attention compared to tracking from a single view. For example, 3DMOT2015 benchmark [Leal-Taixé et al., 2015] for 3D tracking lists only a couple methods that exceed a simple linear programming baseline that was proposed with the benchmark. Leal-Taixé et al. [2011] model the motion of people using social forces, and solves a linear program to estimate the tracks over the whole sequences. Klinger et al. [2017] use joint probabilistic data association for online tracking. The approach of [Berclaz et al., 2011] can be used both for regularizing detections and tracking. It formulates tracking as a problem of finding K-Shortest paths in a graph of detections.

Some of the recently proposed approaches, such as [Milan et al., 2017] or [Sadeghian et al., 2017] can be applied for multiple object tracking in 3D. [Milan et al., 2017] learns weights of the Recurrent Neural Network (RNN) to predict the motion of objects for data association. [Sadeghian et al., 2017] learns several RNNs to combine motion, appearance, and social information throughout time for data association. While both of these approaches could be trained for 3D scenes, combining the appearance information from multiple cameras for [Sadeghian et al., 2017] is non-trivial. Training the model of [Milan et al., 2017] requires a lot more annotated data than what is currently available for 3D tracking. A recently proposed approach by Maksai et al. [2017] can be used as a post-processing step to improve the results of other tracking approaches by learning the patterns of human motion in specific scenes and modifying the tracks to follow such patterns.

6.3 The dataset

6.3.1 Hardware and data acquisition

Hardware. The dataset was recorded using seven statically positioned HD cameras. In particular, we used four *GoPro Hero 3* and three *GoPro Hero 4* cameras (Fig. 6.1). All the seven sequences are of resolution 1920×1080 pixels and were recorded with a frame rate of 60 frames per second (fps). The synchronization between the seven sequences was obtained with ~50 ms accuracy (the precision of which can be observed in Fig. 6.4).

Camera placement and layout. The camera layout is highly overlapping as shown in Fig. 6.1, and the cameras are positioned above the humans' average height. In Fig. 6.2 we illustrate from a top view perspective the amount of overlap between the fields of view of the cameras, where the darker the shading, the higher the number of cameras that capture that area. To obtain this illustration, we considered points on the ground plane and counted the number of cameras for which a given point is in their field of view. The circles in Fig. 6.2 denote the position of the cameras.

Data acquisition. The data acquisition took place in front of the main building of the university ETH Zurich in Switzerland, during nice weather conditions.

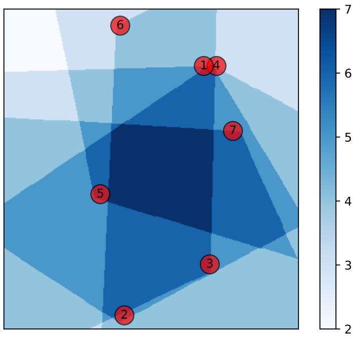


Figure 6.2 – The overlap between the cameras' fields of view (top view). See § 6.3.1.

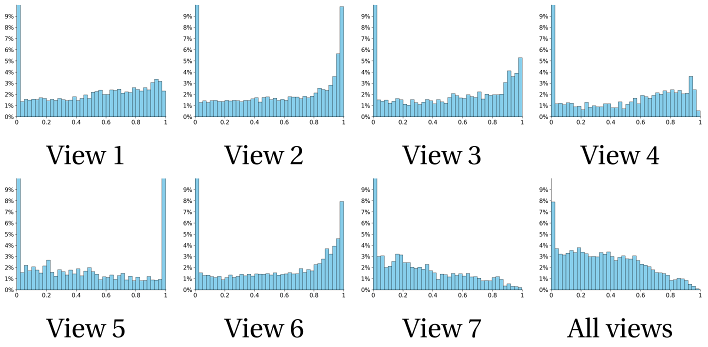


Figure 6.3 – Percentage of examples that are occluded within certain range: x-axis–range of normalized occlusion, y-axis percentage of examples within that range. See § 6.3.4.

6.3.2 Calibration of the cameras

To calibrate the cameras we used the Pinhole camera model [Wikipedia, 2017], due to its widespread usage and support in multiple libraries, including OpenCV [Bradski, 2000]. Primarily, we obtained the intrinsic and the extrinsic parameters, and for the latter, we used points on the ground whose distance was measured by hand.

We put a major focus on providing jointly optimal $3D$ reconstruction between the cameras. To this end, we manually annotated precisely $|\mathcal{D}| = 1398$ $3D$ points by marking corresponding $2D$ points across the seven views and throughout multiple frames; which we used to perform Bundle adjustment [Wikipedia, 2017] as we explain below.

Let **I** and **E** denote the intrinsic and the extrinsic parameters of all of the cameras, respectively. Given a dataset $\mathcal{D}$ of corresponding projections across the views, more precisely: $\mathcal{D} = \{p_i\}$, where $p_i = \{p_i^1 \dots p_i^C\}$, with $p_i^c \in \mathbb{R}^2$ and C denoting the number of cameras, the goal is to find projection matrices P_c whose parameters are contained in $\{\mathbf{I}, \mathbf{E}\}$ and the $3D$ points $M = \{m_i\}$, $m_i \in \mathbb{R}^3$, $i = 1, \dots, |\mathcal{D}|$, such that:

$$\mathbf{I}^*, \mathbf{E}^* = \operatorname*{arg\,min}_{\mathbf{I},\mathbf{E},M} \sum_{i=1}^{|\mathcal{D}|} \sum_{c=1}^{C} w_i^c \, \| p_i - P_c \, m_i \|^2, \tag{6.1}$$

where $\| \cdot \|$ denotes the Euclidean image distance, and w_i^c is the indicator variable equal to 1 when the point p_i is visible in view c, and equal to 0 otherwise. In other words, we formulated the calibration as a non-linear least squares problem, where the error is the squared L_2 norm of the difference between the observed feature location and the projection of the corresponding 3D point on the image plane of the camera.

In accordance with the selected model, the set $\{\mathbf{I}, \mathbf{E}\}$ in our implementation consists of 7×15 parameters for each camera: 3 for rotation, 3 for translation, 2 for focal length (x and y), 2 for the principal point, 3 for radial distortion and 2 for tangential distortion. To optimize Eq. 6.1, we used the open source C++ library *Ceres* [Agarwal et al.] (see App. D.3 for further details on the implementation).

We experimented with fixing one of the sets of parameters and sequentially optimizing each. Nonetheless, jointly optimizing **I** and **E** as formalised in Eq. 6.1 provided best results and this final step significantly improved the joint projection accuracy.

Fig. 6.4 depicts cropped regions taken from synchronized images and of a different view. To illustrate the precision of the camera calibration, we first manually marked a point projections in two of the views, shown in blue color in the left column of Fig. 6.4. Using the provided camera calibration, we then compute the $3D$ location of the point as an intersection between the rays defined by those projections. Finally, the resulting $3D$ point is projected in the remaining views, displayed as red points in Fig. 6.4.

6.3.3 Annotation process

To make sure that our annotations are as precise as possible, we made use of the accurate camera calibration. Namely, we formulated the annotation task as adjusting the position of a $3D$ cylinder on the ground, such that its projections (rectangles) across *all* of the views overlap

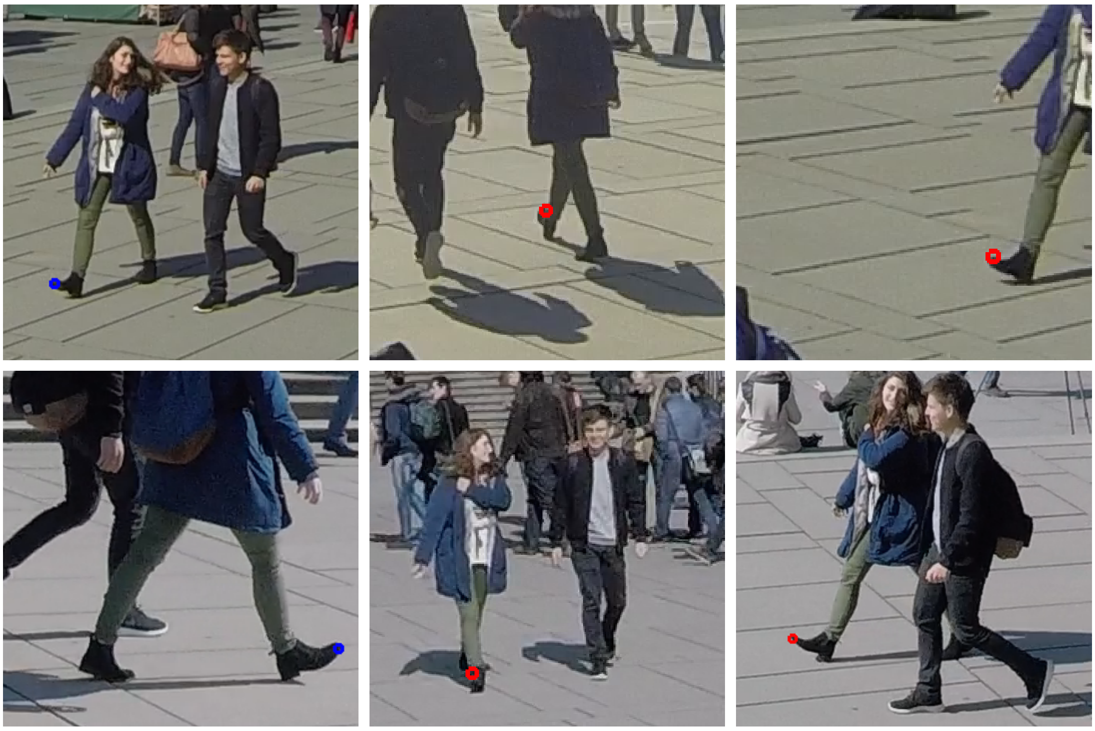
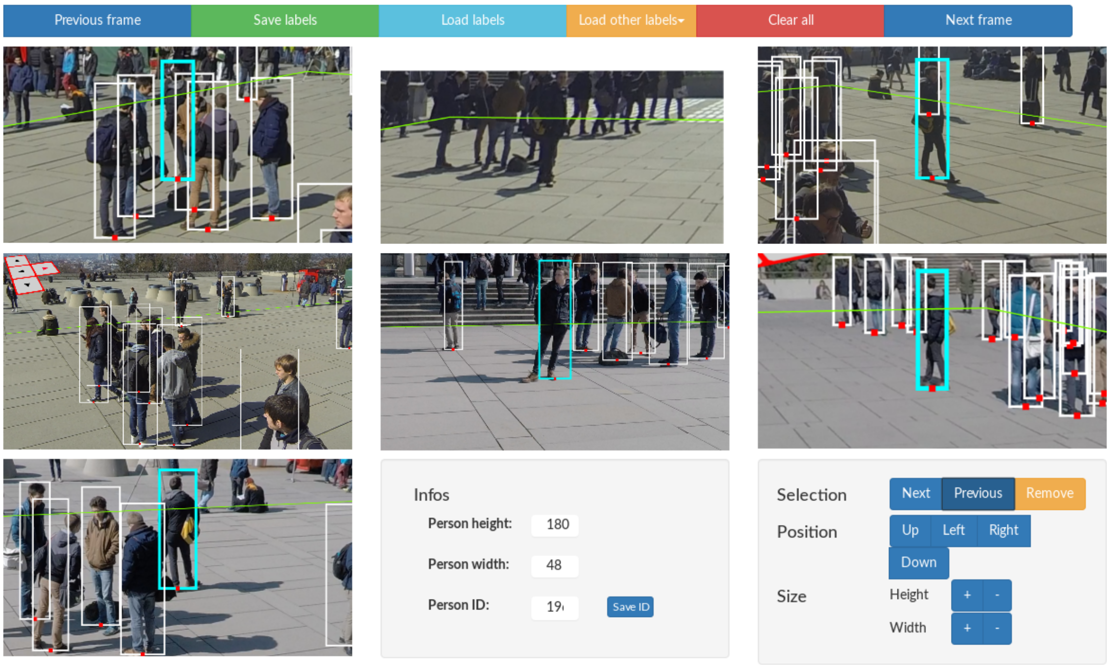


Figure 6.4 – Illustration of the camera calibration precision (best seen in color). See § 6.3.2.

the person being marked by the annotator. In summary, we considered this approach as: (i) more time-effective, as one has to perform one adjustment instead of marking bounding boxes and adjusting each separately; as well as that it allows for obtaining (ii) higher precision, since the position is jointly adjusted. The latter follows from the fact that the best position of a bounding box when annotating in one view is likely to be ambiguous, and in cases of more severe occlusions, it could also be infeasible to guess it.

To this end, we designed a new multi-camera annotation tool, whose Graphical User Interface (GUI) is illustrated in Fig. 6.5. It was written using Python, Django, and Javascript. The tool was deployed through Amazon Mechanical Turk [Buhrmester et al., 2011] so that external people would be remunerated to help us carry out this time-consuming procedure. To ensure that profit is not prioritized over the accuracy and the precision of the annotations, we were highly involved in the process, and particular annotators were carefully selected. For details regarding how the annotation process was carried out, and the file format of the annotations, please refer to App. D.1 and App. D.2, respectively.



Infos

Person height: 180

Person width: 48

Person ID: 19

Figure 6.5 – GUI of our multi-camera annotation tool (best seen in color). The bounding boxes being adjusted (displayed in cyan color) are zoomed-in in those views where the person is visible.

6.3.4 Statistics

Annotated frames. Following practices amassed in monocular pedestrian detection with CNNs, we used frame rate of 10 fps to extract the frames. Herein, as a frame I_t we refer to a set of synchronized images from the seven views (C=7), *i.e.* $\{I_t^1, \dots, I_t^C\}$. We provide annotations for the first 2000 frames. However, in our observations annotating every 5th suffices the speed of movement of the persons, and hence interpolation can be used to further enlarge the dataset. To summarize, this amounts to total of 400 frames at 2 fps, or alternatively 7×400 manually annotated images. Bellow we always list total number of annotations at 2 fps.

Multi-view annotations. There are 9518 multi-view single person annotations in total. In Fig. 6.6, we illustrate an example of a multi-view annotation of our dataset, visible in all of the seven views at the same time. On average, each frame I_t captures 23.8 people. Considering a frame rate of 2 fps, each person is seen in 30.41(47.87) frames, with a mode of 22 frames (see App. D.4).

Monocular annotations. Since a multi-view detection may not be visible in each of the cameras' fields of view, the number of monocular examples is precisely 8731, 7875, 6703, 2239, 3920, 9408, 3731, respectively for each view. This amounts to a total of 42607 detections at 2 fps.

Occlusion levels. As the annotations are obtained in $3D$, the occlusion of each pedestrian in each view can be automatically obtained. Following the work of [Dollar et al., 2009] in Fig. 6.3 we illustrate the level of occlusions per each view separately, by calculating the number of occluded pixels over the number of total pixels for each detection. Similarly, we calculate such normalized occlusion levels for the multi-view examples across all of the views. As can be noticed from Fig. 6.3, if multi-view samples are used, the probability that a person will be completely occluded in all of the views simultaneously significantly goes down.

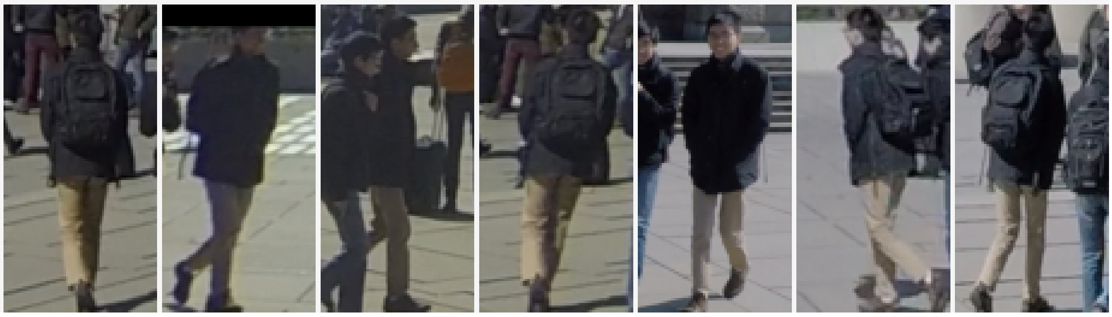


Figure 6.6 – An example of one positive multi view annotation.

6.4 Benchmarks

In the sequel, we evaluate detection and tracking methods. In our experiments, we used a frame rate of 2 fps.

6.4.1 Evaluation protocol

We compute false positives (FP), false negatives (FN) and true positives (TP) by assigning detections to ground truth using Hungarian matching. Since we operate on the ground plane, we impose that a detection can be assigned to a ground truth annotation if and only if it is closer than a distance r. Given FP, FN and TP, we calculate:

- **Multiple Object Detection Accuracy (MODA)** accounting for the normalized missed detections and false positives, as well as the **Multiple Object Detection Precision (MODP)** metric which assesses the localization precision [Kasturi et al., 2009].
- **Precision & Recall**. We estimate the empirical precision and recall, calculated by $P{=}TP/(TP{+}FP)$ and $R{=}TP/(TP{+}FN)$ respectively.

We report MODP, Precision, and Recall for radius $r{=}0.5$m, which roughly corresponds to the width of a human body.

Unless otherwise emphasized, the used metrics are always calculated in terms of the Euclidean distance between the detection and the annotation on the ground plane in world coordinates, or alternatively from top-view. Thus note that such metrics are unforgiving in terms of projection errors as we measure distances on the ground plane, which would not be the case if we evaluated overlap in the image plane as is often done in the monocular case.

Regarding multiple object tracking, we report metrics which are also reported for MOTChallenge benchmark [Leal-Taixé et al., 2015]: a set of CLEAR MOT [Bernardin and Stiefelhagen, 2008] metrics, as well as identity-aware metrics of [Ristani et al., 2016]. For evaluation we used the devkit provided with [Leal-Taixé et al., 2015], and we similarly report metrics for radius $r{=}1$m. Below we briefly review some of the reported tracking metrics:

- **Multiple Object Tracking Accuracy (MOTA)** accounts for missed detections (**False Negatives (FN)**), **False Positives (FP)**, and **Identity Switches (IDs)** between current and next frame of tracking. We also report **Multiple Object Tracking Precision (MOTP)**.
- **Mostly Tracked (MT), Partially Tracked (PT), Mostly Lost (MT)** trajectories, as well as the number of **Fragmentations (FM)**.
- **IDF1** accounts for missed detections, false positives, and identity switches throughout tracking, after finding a global one-to-one assignment between ground truth and pre-

dicted trajectories. It is calculated similarly to F1 score, using the **Identity Precision (IDP)** and **Identity Recall (IDR)** as proxies for precision and recall.

6.4.2 Evaluated methods

We evaluated the following detection methods:

- **POM-CNN.** The multi-camera detector [Fleuret et al., 2008] described in § 6.2.2, uses background subtraction pre-processing and takes such segmented images as input. In its original implementation the input is obtained using traditional algorithms [Ziliani and Cavallaro, 1999, Oliver et al., 2000]. Hence, for a fair comparison reflecting the progress that has occurred since then, we use a CNN-based background subtraction [Long et al., 2015].

- **DeepMCD** (described in § 6.2.2), which is an end-to-end deep learning method. We used its implementation that uses GoogLeNet [Szegedy et al., 2015] by: (i) testing on the WILDTRACK dataset with the provided pre-trained model on the PETS dataset–Pre-DeepMCD; as well as (ii) training solely the top classifier on the WILDTRACK dataset–TopDeepMCD. Its implementation uses monocular Non Maxima Suppression (NMS) and the performance measures are calculated using the first view, rather then the world coordinates.

- **ResNet-DeepMCD & DenseNet-DeepMCD** are our implementations of DeepMCD in PyTorch [Paszke et al., 2017], which use ResNet-18 [He et al., 2015] and DenseNet-121 [Huang et al., 2017], respectively. As WILDTRACK is of larger size, we omitted the step of pre-training on the Caltech dataset. Our implementation uses top-view NMS to select the final detections of the candidates while prioritizing the detections with higher probability estimates, as opposite to [Chavdarova and Fleuret, 2017]. We use NMS threshold of 0.4 and r=0.5m. We used 90% and 10% of the frames for training and testing, respectively and grid density of 60×180. Analogously, we also train a monocular detector ResNet-18 [He et al., 2015] while using samples from the training frames from all of the views, denoted as **ResNet-Monocular**.

- **Deep-Occlusion** (see § 6.2.2), which is a hybrid CNN-CRF method to use information about geometry and calibration while leveraging on the discriminative power of a pre-trained monocular CNNs.

- **RCNN-projected.** The recent work of [Xu et al., 2016] proposes a MCMT tracking framework that relies on a powerful CNN for detection purposes [Ren et al., 2015]. Since the code of [Xu et al., 2016] is not publicly available, we reimplemented their detection methodology *without* the tracking component for a fair comparison with the detection methods that operate on images acquired at the same time. More precisely, we first run the 2D detector proposed by Ren et al. [2015] on each image. We then project the bottom of the 2D bounding box onto the ground reference frame as in [Xu et al., 2016] to get 3D ground coordinates. Finally, we cluster all the detections from all the cameras using 3D proximity to produce the final set of detections.

For multiple object tracking, we provide results of the following methods:

- **KSP** is a simple baseline approach of Berclaz et al. [2011] which finds the most likely sequence of occupancies of ground floor locations of POM given probabilities of occupancies of each location given by a detector. To do so it solves the problem of finding the most likely trajectories given the detections as finding the K-Shortest Paths in a graph. This method does not use appearance information for tracking, but is a suitable baseline as it can be applied out of the box in multi-camera scenario.
- **ptrack** is a recent approach proposed by Maksai et al. [2017] which improves the results of other tracking approaches by learning the motion patterns of the scene and modifying the tracks of people so as to follow those patterns. This is suitable for our scenario both because in our scene there are several clearly identifiable patterns of motion, as well as because it can handle tracking in the ground plane. We therefore use this approach on top of the trajectories found by **KSP**.

6.4.3 Benchmark results

Table 6.2 – Benchmark tracking results on the WILDTRACK dataset.

Method	IDF1	IDP	IDR	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP
DeepOcclusion+KSP	73.2	83.8	65.0	49	79	43	1095	7503	85	92	69.6	61.5
DeepOcclusion+KSP+ptrack	78.4	84.4	73.1	72	74	25	2007	5830	103	95	72.2	60.3

Detection. The results of the enumerated methods in § 6.4.2 are given in Tab. 6.3. We observe that joint utilization of the multiple views largely improves detection performance. More explicit occlusion reasoning further improves the MODA metric, even for small values of r, Fig. 6.7. In Tab. 6.4 we list the classification results obtained when training a monocular ResNet-18 using samples extracted from all of the seven views, as well as multi-view training. The results indicate that even though the size of the dataset is seven-fold larger for the monocular case, using the multiple views increases both the accuracy and the confidence of the classifier.

Tracking.

Tab. 6.2 lists benchmark results of the tracking methods described in § 6.4.2 (see App. D.6 for additional results). Parameters of the methods were optimized for IDF1 metric on the same training data as the appropriate detector was trained on. As shown, the dataset presents a

Table 6.3 – Benchmark detection results on WILDTRACK.

Method	MODA	MODP	Precision	Recall
Deep-Occlusion+KSP	0.752	-	-	-
Deep-Occlusion	0.741	0.538	0.95	0.80
ResNet-DeepMCD	0.678	0.642	0.85	0.82
DenseNet-DeepMCD	0.635	0.666	0.87	0.74
POM-CNN	0.232	0.305	0.75	0.55
RCNN-projected	0.113	0.184	0.68	0.43
Pre-DeepMCD	0.334*	0.528*	0.93	0.36
Top-DeepMCD	0.601*	0.642*	0.80	0.79
ResNet-View 1	-1.823	0.598	0.26	1.00
ResNet-View 2	-1.050	0.607	0.32	0.99
ResNet-View 3	-1.036	0.583	0.32	0.98
ResNet-View 4	-0.251	0.723	0.42	0.71
ResNet-View 5	0.466	0.623	0.67	0.91
ResNet-View 6	-1.841	0.591	0.26	1.00
ResNet-View 7	-0.122	0.701	0.47	0.97

* Monocular calculation of the metric, using the first view.

Table 6.4 – Classification results (accuracy and area under ROC curve) on the test frames of the WILDTRACK dataset using ResNet-18 [He et al., 2015]. The results obtained for monocular classification are averaged over all of the views.

Training	Accuracy (%)	AUC
Monocular	84.57 (1.718)	0.91 (0.028)
Multi-view	95	0.95

significant tracking challenge, with IDF1 metric results lower than those seen on the [Leal-Taixé et al., 2015] benchmark on [Ristani et al., 2016] dataset, where each individual camera mostly observes a separate, and somewhat simpler scene.

6.5 Discussion

The development of new multi-view people tracking methods is hampered by the surprising lack of appropriate datasets. We provide a new large scale, high-resolution, and highly accurately calibrated multi-camera pedestrian dataset, which is more realistic than any of the previously published ones.

Our initial benchmarks show that deep learning person detection indeed largely benefits from a multi-camera set-up and this motivates further work in that direction. Our dataset will motivate and facilitate such research. While performance of monocular pedestrian detectors

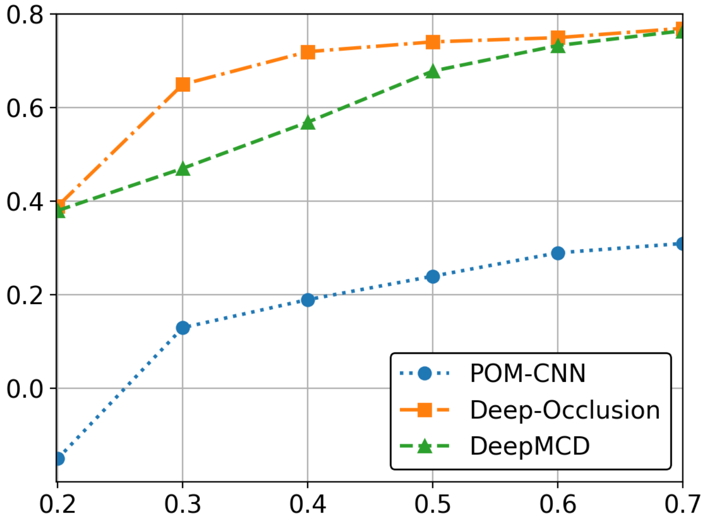


Figure 6.7 – Benchmark of the multi-view detectors using the MODA metric (y-axis) for different radius r (x-axis) on WILDTRACK.

saturates on common benchmarks, our densely crowded realistic set-up, which results in complex dynamics and constant occlusions among persons, and the high resolution, will prove useful for further improving monocular detection, as well as other problem of inference such as tracking, or crowd analysis at large.

Finally, in addition to the large number of individual detections, the WILDTRACK dataset also has a large fraction of unlabelled frames. This will be precious for unsupervised methods, with the possibility to be benchmarked on the annotated portion.

7 Concluding Remarks

The large amounts of data and the expensive procedure of manual labeling by humans are the principal motives for advancing unsupervised learning. Notably, the unsupervised Generative Adversarial Networks–GANs algorithm found applications in a wide variety of domains. At the core of this algorithm, lies finding a Nash equilibrium of a two-player *minimax* game, where the players are Deep Neural Networks. At each iteration of the Stochastic Gradient Descent–SGD based training, the Discriminator–D and the Generator–G aim at distinguishing real from generated samples, and "fooling" D that the generated samples are real (by mapping random noise to samples), respectively. Unlike classical loss functions for training DNNs, such as log-loss or squared error, the function optimized by these networks has no closed form. As a result, training is highly susceptible to hyper-parameter values, optimization method as well as architectural choices.

In large part due to the development of autonomous car driving and security applications, *people detection* is considered as an important problem in the field of computer vision. Current state of the art monocular pedestrian detectors are based on deep learning, which surprisingly had not been extended to the multi-camera setup.

This thesis focuses on *improving Generative Adversarial Nets* as most widely used deep generative models, as well as applying deep learning methods to *multi-camera people detection*.

7.1 Summary and contributions

In this thesis we: (i) presented new algorithms for training GANs, and (ii) focused on bridging the gap between deep learning algorithms and the problem of *multi-camera people detection*, in the first and second parts of the thesis, respectively.

In the third chapter and the first part of the thesis, we proposed a GAN training framework named *SGAN*, applicable to the majority of the existing GAN algorithms. SGAN trains the final pair of networks against an ensemble of adversarial networks while maintaining statistical independence. Hence, it produces a single generative network, making this approach practically convenient at inference time. The presented experimental results on diverse datasets demonstrate systematic improvements upon classical algorithms as well as increased stability of the framework regarding real-world applications. At each parameter update of the generative model, the gradients of the discriminators of the ensemble are averaged, yielding reduced variance over the iterations. The latter motivated variance reduced gradient optimization of a *single* GAN pair, due to the advantage of decreased computation during training, presented next in the thesis.

Motivated by the empirical analyses of SGAN, in chapter 4 of the first part of the thesis we focused on variance reduced gradient optimization of GANs. We considered a simplistic example of game optimization and showed that stochasticity breaks the convergence of existing stochastic methods. We proposed a novel SVRE algorithm which combines SVRG with the extragradient method for optimizing games. SVRE has improved rates over the existing results for a large class of strongly-convex games, whereas empirically it is the only method that converges for the bilinear game example. Interestingly, SVRE empirically matched the convergence speed of Batch-Extragradient on MNIST, while the latter is *infeasible* for large datasets. When using shallow architectures, SVRE matched or improved over baselines on all four datasets, whereas our presented experiments with deeper architectures showed that SVRE is notably more stable with respect to hyperparameter choice. Most importantly, while its stochastic counterpart diverged in all our experiments, SVRE did not.

We start the second part of this thesis with our proposed multi-camera people detector, called DeepMCD, presented in Chapter 5. DeepMCD is an end-to-end deep learning method which jointly leverages the multi-stream deep features. It is suitable for small-scale multi-camera datasets as it makes use of the existing monocular datasets, which also improves its generalization. We showed that DeepMCD outperformed the existing approaches. This motivated us to focus on obtaining a large–scale dataset for this problem.

In the sixth chapter, we presented our seven-view high-resolution WILDTRACK dataset of walking pedestrians. We focused on providing a camera calibration which, for the first time among the existing datasets, would be consistent across the views, rather then done on per-view basis. In particular, after obtaining initial values of the camera calibraion parameters using standard methods, we further optimize these, given annotated corresponding points across the seven views. Our approach to further optimize the calibration parameters based on bundle adjustment provided high precision calibration. Moreover, WILDTRACK captures a densely crowded realistic set-up, which is different from any of the previously published multi-view datasets which often use actors. Finally, we provide benchmarks of relevant existing methods on this dataset.

7.2 Limitations and future work

One disadvantage of SGAN is the redundancy of computation that may occur among the local pairs. Our presented preliminary toy experiments in § 3.6 showed promising results in this direction. Namely, SGAN can be improved by enforcing "diversity" between the local pairs, or more precisely, by forcing the local generators to model different parts of the real data distribution. Other extensions of SGAN include re-casting the analysis in the context of multi-player game theory.

Our discussion in § 4.7 pointed out our observation from our empirical analyses that the convergence speed of SVRE decreases for very deep networks. Note that BatchNorm layers [Ioffe and Szegedy, 2015] break the property of SVRG that its gradient estimates are unbiased (see § 4.3.1). Their impact increases with the depth of the network as typically deeper nets use multiple such layers. Interestingly, a separate line of work focuses on omitting such normalization layers and developing novel parameter initialization methods for single objective optimization [see Dauphin and Schoenholz, 2019, and references therein]. Hence, extending these methods to games and eliminating the use of normalization layers, could be a promising direction for improving SVRE. Moreover, in § 4.7 as stochastic baseline of SVRE we considered Adam, as vanilla SGD does not work for GANs. Note however, that Adam is adaptive step size method, indicating such extensions might be applied to SVRE as well, as discussed in § 4.7. Finally, our work presented in Chapter 4 showed that controlling the variance is *critical* for game optimization. However, variance reduced gradient in games might be achieved in novel ways. Finding novel variance reduction methods suitable for DNN optimization might be a promising direction for GAN optimization.

The WILDTRACK dataset of walking people, presented in the second part of the thesis, opens up wide variety of research directions. In addition to the large number of individual detections, the WILDTRACK dataset also has a large fraction of unlabelled frames. This will be precious for unsupervised methods, with the possibility to be benchmarked on the annotated portion. One of our future directions is applying GANs on this problem.

A Appendix for Chapter 2

A.1 Equilibrium of GANs in functional space

For completeness, we show that in functional space the modelled distribution is equivalent to the data distribution $p_g = p_d$, as shown in Goodfellow et al. [2014].

Proof. The discriminator maximizes:

$$\begin{aligned} V(G,D) &= \int_x p_d(x) \log(D(x))\,\mathrm{d}x + \int_z p_z(z) \log(1 - D(G(z)))\,\mathrm{d}z \\ &= \int_x p_d(x) \log(D(x)) + p_g(x) \log(1 - D(x))\,\mathrm{d}x \end{aligned}$$

Where we used $x = G(z)$, and p_g is the distribution of x.
Hence, the optimal discriminator D^* is: $D^*(x) = \frac{p_d(x)}{p_d(x)+p_g(x)}$.

The generator minimizes:

$$\begin{aligned} V(G,D^*) &= \underset{x\sim p_d}{\mathbb{E}}[\log D^*(x)] + \underset{x\sim p_g}{\mathbb{E}}[\log(1 - D^*(x))] \\ &= \underset{x\sim p_d}{\mathbb{E}}[\log \frac{p_d(x)}{p_d(x)+p_g(x)}] + \underset{x\sim p_g}{\mathbb{E}}[\log \frac{p_g(x)}{p_d(x)+p_g(x)}] \\ &= -\log 4 + \mathbb{D}_{KL}\big(p_d || \frac{p_d+p_g}{2}\big) + \mathbb{D}_{KL}\big(p_g || \frac{p_d+p_g}{2}\big) \\ &= -\log 4 + 2 \cdot \mathbb{D}_{JS}(p_d || p_g) \end{aligned}$$

where we used: $\mathbb{D}_{JS}(p\|q) = \frac{1}{2}\mathbb{D}_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}\mathbb{D}_{KL}(q\|\frac{p+q}{2})$. □

Hence, the optimum is reached when $p_g = p_d$, and the optimal value is $-\log 4$.

B Appendix for Chapter 3

The implementation details, as well as additional results of experiments on toy and realistic datasets, are given in Sections B.1 and B.2. In Section B.3 we discuss two viewpoints of SGAN. We use notation as in Chapter 3.

B.1 Experiments on toy datasets

B.1.1 Details on the implementation

For experiments conducted on toy datasets, we used separate $2\cdot(N+1)$ networks. The architecture and the hyper-parameters are as follows. Each network is a multilayer perceptron (MLP) of 4 fully connected layers and LeakyReLU non-linearity [Maas et al., 2013] with the PyTorch's default value for the negative slope of 0.01 [Paszke et al., 2017]. The number of hidden units for each of the layers is 512, whereas the dimension of the input noise vector for the generator network is 100.

We use learning rate of $1\cdot 10^{-5}$, as well as the *Adam* optimization method [Kingma and Ba, 2015]. Using *RMSProp* [Tieleman and Hinton, 2012] as optimization method did not give obvious improvements in our conducted experiments.

B.1.2 Experiments

Fig. B.1 depicts several image pairs, of: (i) samples generated by the local generators (left); and (ii) samples from the global one (right). The illustrated contours are obtained with GMM Kernel Density Estimation (KDE) [Rosenblatt, 1956], whose bandwidth is cross-validated. We used sample of p_g of size 500 (in Fig. B.1, N denotes the sample size). C in Fig. B.1 denotes the *Coverage* metric [Tolstikhin et al., 2017].

Fig. B.4 depicts experiment in which the parameters of the global pair are updated after each update of a local pair.

B.2 Experiments on real-world datasets

B.2.1 Details on the implementation

Regarding experiments on real-world applications, we considered: (i) using separate $2\cdot(N+1)$ networks; as well as (ii) using parameter sharing of the networks. In the latter, approximately half of the parameters of each network are shared among the corresponding other N networks (discriminators or generators). To distinguish the two, in the sequel we denote the former and the latter case as **N-S-** and **N-SW-**, respectively. For *DCGAN, we recommend using separate networks*, as sharing parameters makes the generators to produce similar samples, thus the performance gain of SGAN can be marginal.

We used learning rate of $1\cdot 10^{-5}$, and a batch size of 50 and 64 for (FASHION)MNIST and the rest of the datasets, respectively. Unless otherwise stated, we used the *Adam* optimizer [Kingma and Ba, 2015] whose hyperparameters (one parameter used for computing running averages of gradient and another for its square) we fixed to 0.5 and 0.999, as in [Radford et al., 2016].

Implementation of the experiments on image datasets. For **MNIST** we did experiments using both MLPs and CNNs for the generators and the discriminators. In the former case, the architectures were almost identical to those used for the toy experiments, except that the first layer was adjusted for input space of 28×28. In the latter case, we used input space of 28×28 and we started with the DCGAN implementation [Radford et al., 2016] and changed it accordingly to the input space. In particular, we reduced the number of 2D transposed convolution layers from 5 to 4 and adjusted the hidden layers' sizes accordingly to the dimensions used for the real data space.

For **CIFAR10** unless otherwise emphasized, we used 32×32 image space. For the rest of the image datasets–unless otherwise stated, we used 64×64 input space and the original DCGAN [Radford et al., 2016] architecture, as provided by the authors. The implementation of DCGAN [Radford et al., 2016] uses Batch Normalization layers [Ioffe and Szegedy, 2015].

Implementation of the experiments on one Billion Word Benchmark. We started from the provided implementation of [Gulrajani et al., 2017] and implemented our method. In particular, the character-level generative language model is implemented as a $1D$ CNN using 4 ResNet blocks [He et al., 2015], which network maps a latent vector into a sequence of one-hot character vectors of dimension 32. The discriminator is also a $1D$ CNN, that takes as input sequences of such character embeddings of size 32.

As optimization method we used *RMSProp* [Tieleman and Hinton, 2012].

Separate networks. In Figure B.3 we show the Inception score [Salimans et al., 2016] (using its original implementation in TensorFlow [Abadi et al., 2015]), of the global generator and the local generators.

In Fig. B.2 we show samples of **5-S-DCGAN** on **FASHION-MNIST** (on the right), as well as of **DCGAN** (on the left). Figures B.5 & B.6 depict samples using **DRAGAN** and **DCGAN**, respectively. We see that the global generator converges much earlier then the local ones.

Shared parameters.

In Fig. B.9 we show samples when training **DRAGAN** and **5-SW-DRAGAN** on **LSUN-bedroom** with input dimension of 64×64. Finally, in Fig. B.10 we show samples when training on the **Billion Word** dataset.

B.3 Different viewpoints of SGAN

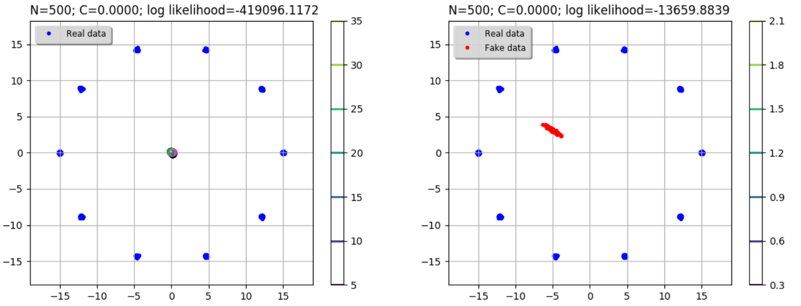
Connecting SGAN to Actor-critic methods. Pfau and Vinyals [2016] argue that at an abstract level GANs find similarities with actor-critic (AC) methods, which are widely used in reinforcement learning. Namely, the two have a feed-forward model which either takes an action (AC) or generates a sample (GAN). This acting/generating model is trained using a second one. The latter model is the only one that has direct access to information from the environment (AC) or the real data (GAN), whereas the former has to learn based on the signals from the latter. We refer the interested reader to [Pfau and Vinyals, 2016] which further elaborates the differences and finds connections that both the methods encounter difficulties in training.

We make use of the graphical illustration proposed in [Pfau and Vinyals, 2016] of the structre of the GAN algorithm illustrated in Fig. B.11a, and we extend it to illustrate how SGAN works, Fig. B.11b, where nodes with index i can be multiple. Empty circles represent models with a distinct loss function. Filled circles represent information from the environment. Diamonds represent fixed functions, both deterministic and stochastic. Solid lines represent the flow of information, while dotted lines represent the flow of gradients used by another model. In SGAN, D_0 is being trained with samples from the multiple generators whose input is in the real-data space. For clarity, we omited D^{msg} in the illustration–used to train G_0, as the arrows already indicate that these two "global" models do not affect the ensemble.

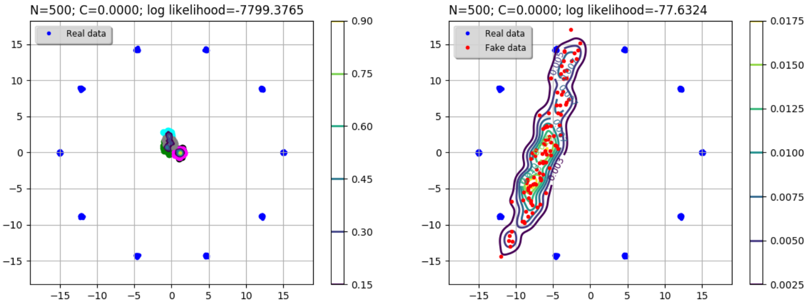
Game theoretic interpretation.

We can define a game that describes the training of G_0 and D_0 in the SGAN framework as follows. Let us consider a tuple $(\mathscr{P}, \mathscr{A}, u)$, where $\mathscr{P} = \{G, D\}$ is the set of new players that we introduce. Let us assume that G and D, at each iteration can select among the elements of $\mathscr{D}$ and $\mathscr{G}$, respectively. Hence, $\mathscr{A} = (A_g, A_d)$ have a finite set of N actions.

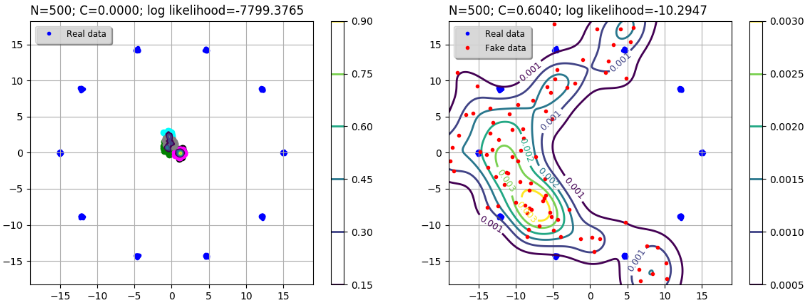
Such "top level players" in SGAN assign uniform distribution over their actions, more precisely both G and D sample from the elements of $\mathscr{D}$ and $\mathscr{G}$ respectively, with uniform probability. To connect to classical training, let us assume that G and D fix their choice to one element of $\mathscr{D}$ and $\mathscr{G}$ respectively, *i.e.* with probability one they sample from a single generator/discriminator. The trained networks G_0 and G_i, as well as D_0 and D_j, with i and j being the selected choice of G and D respectively, are identical in expectation. Finally, rather than predefining the uniform sampling in SGAN, incorporating estimations of the actions' pay-off $u = (u_g, u_d)$ could prove useful for training (G_0, D_0).



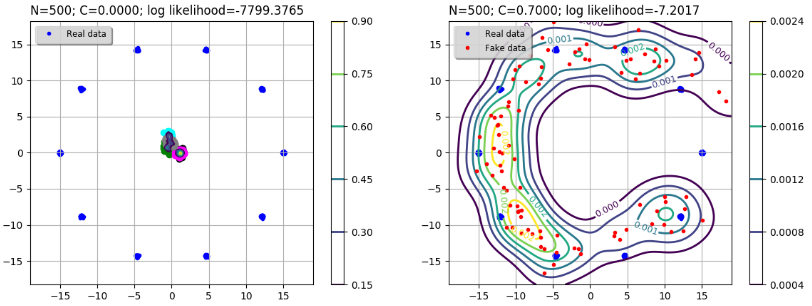
Iteration 1



Iteration 8

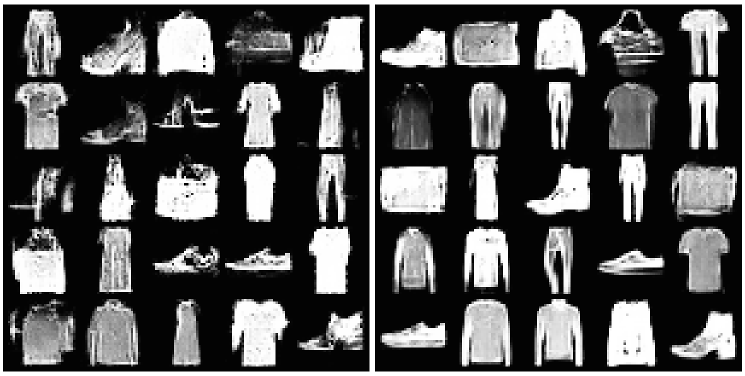


Iteration 40



Iteration 125

Figure B.1 – **5-S-WGAN** on the **10-GMM** dataset. Samples from the five local generators and from the global generator, are shown on the left (in separate color) and on the right (in red color), respectively. See § B.1.2.



(a) Samples using DCGAN. (b) Samples using 5-S-DCGAN.

Figure B.2 – Samples of **DCGAN** and **5-S-DCGAN** on **FASHION-MNIST** taken at the 6000-*th* iteration, on the left and right, respectively. The input dimension is 28×28.

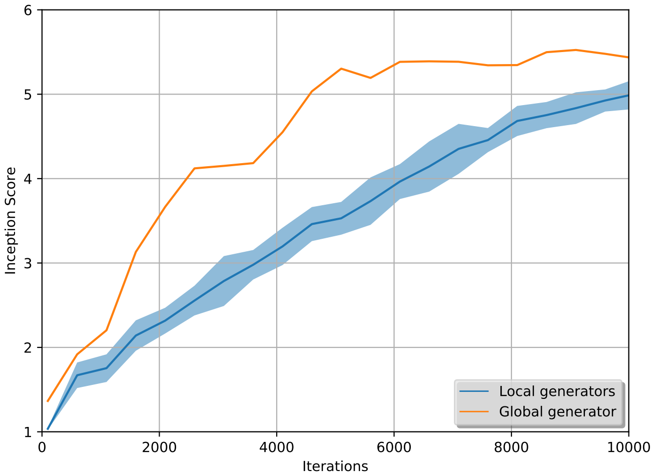
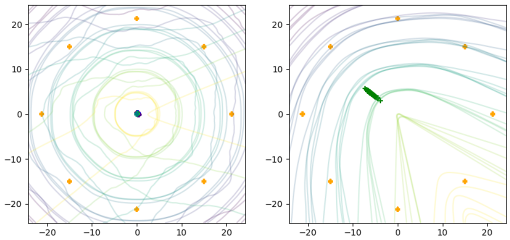
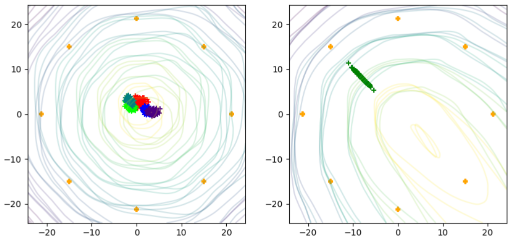


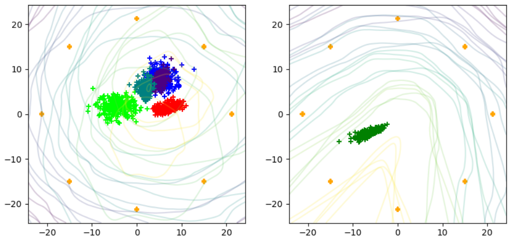
Figure B.3 – **10-S-DCGAN**, on **CIFAR10**. We plot the Inception Score [Salimans et al., 2016] of the global generator (orange) as well as the scores of the local generators (blue). The input dimension is 32×32.



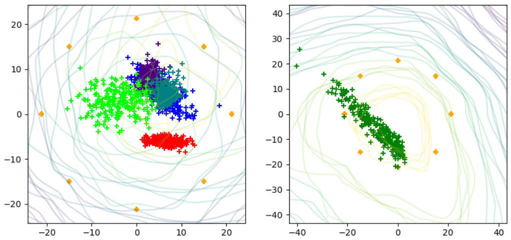
Iteration 1



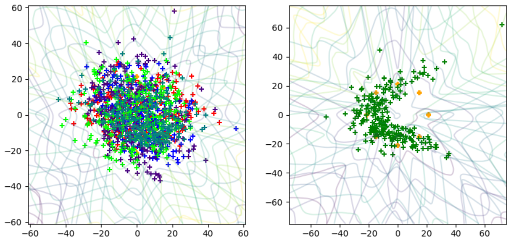
Iteration 5



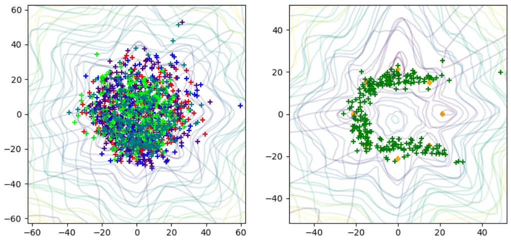
Iteration 10



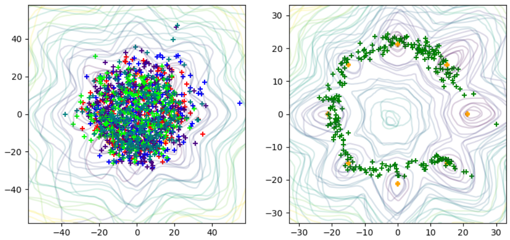
Iteration 15



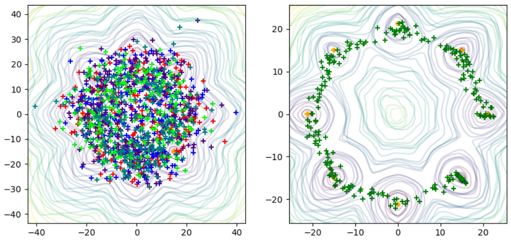
Iteration 60



Iteration 80

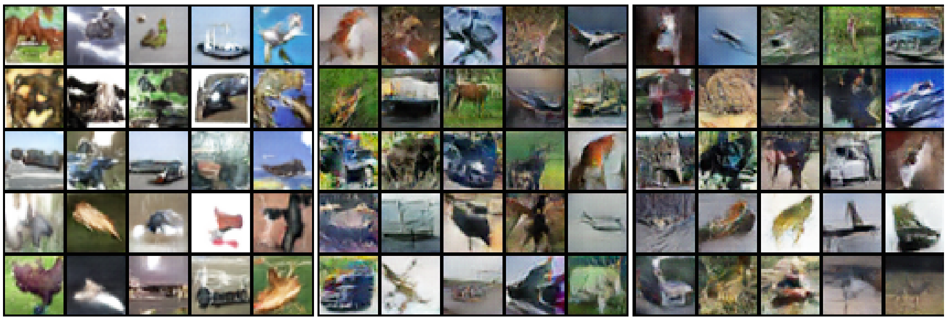


Iteration 100

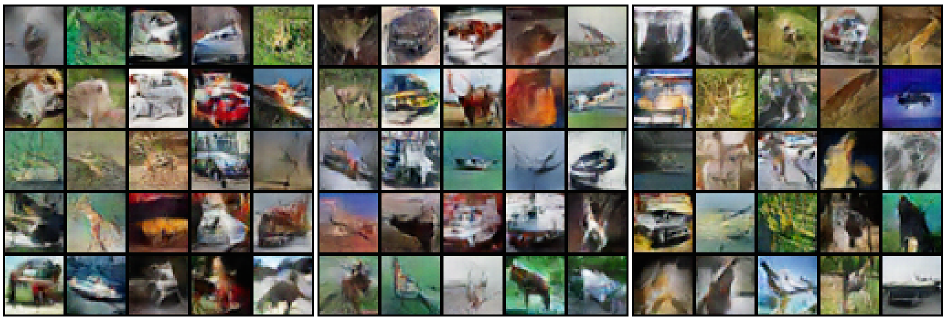


Iteration 300

Figure B.4 – **5-S-WGAN** experiment on the **8-GMM** toy dataset (best seen in color). Real data samples are illustrated in orange. In each image pair, we illustrate samples from the five local generators and from the global generator, on the left (in separate color) and on the right (in green), respectively. The displayed contours represent the level sets of the discriminators $\mathscr{D}$ and $\mathscr{D}^{msg}$–illustrated on the left and right of each image pair, respectively, where yellow is low and purple is high.

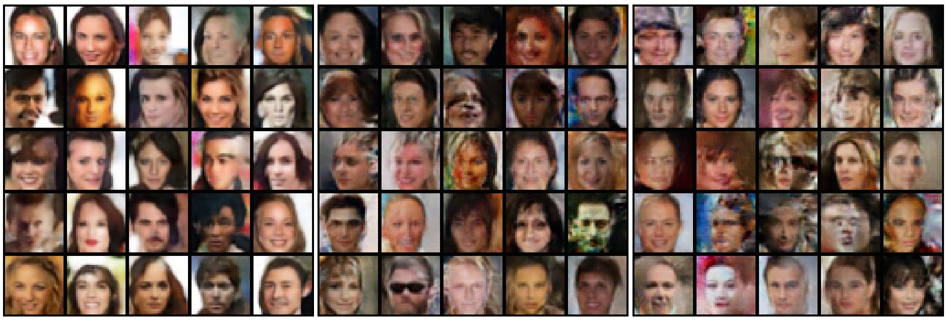


Global generator *Local generator #1* *Local generator #2*

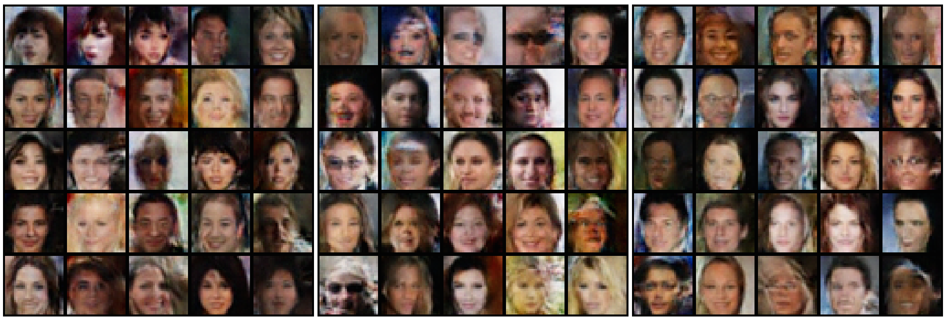


Local generator #3 *Local generator #4* *Local generator #5*

Figure B.5 – **5-S-DRAGAN** on **CIFAR10** at $40{\cdot}10^3$-*th* iteration, and 32×32 real data space.

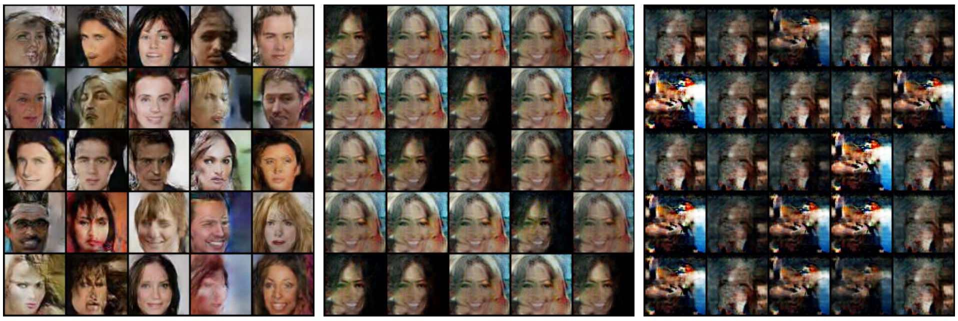


Global generator *Local generator #1* *Local generator #2*

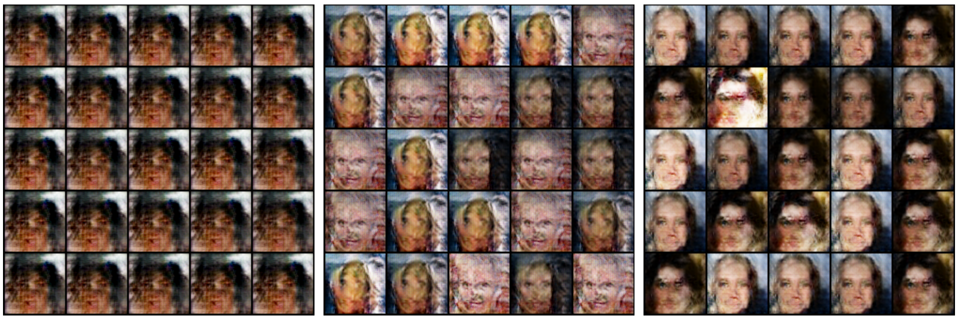


Local generator #3 *Local generator #4* *Local generator #5*

Figure B.6 – **5-S-DCGAN** on **CelebA** at $1{\cdot}10^3$-*th* iteration, and 32×32 real data space.

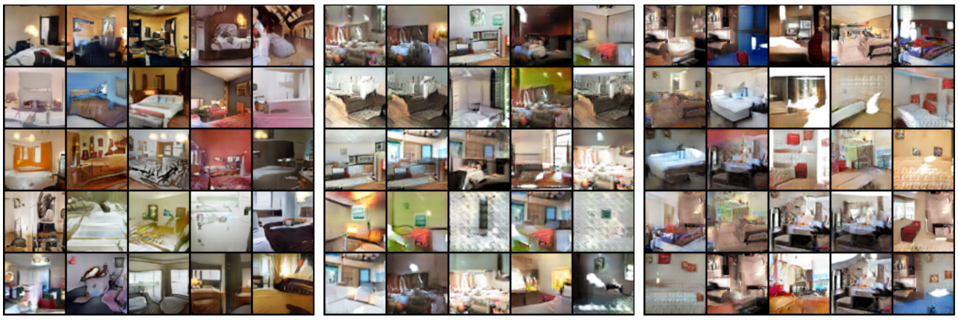


Global generator *Local generator #1* *Local generator #2*

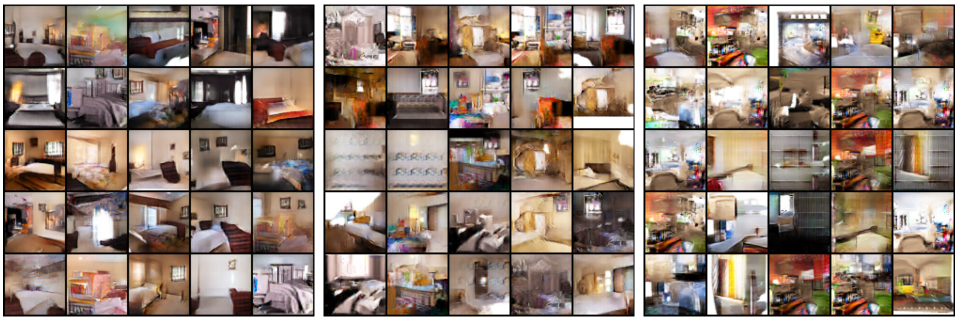


Local generator #3 *Local generator #4* *Local generator #5*

Figure B.7 – Samples of the generators of **5-S-DRAGAN** on the **CelebA** dataset at the $50 \cdot 10^3$-*th* iteration. The input dimension is 64×64.



Global generator *Local generator #1* *Local generator #2*



Local generator #3 *Local generator #4* *Local generator #5*

Figure B.8 – Samples of the generators of **5-S-DCGAN** on the **LSUN-bedroom** dataset at the $100 \cdot 10^3$-*th* iteration. The input dimension is 64×64.

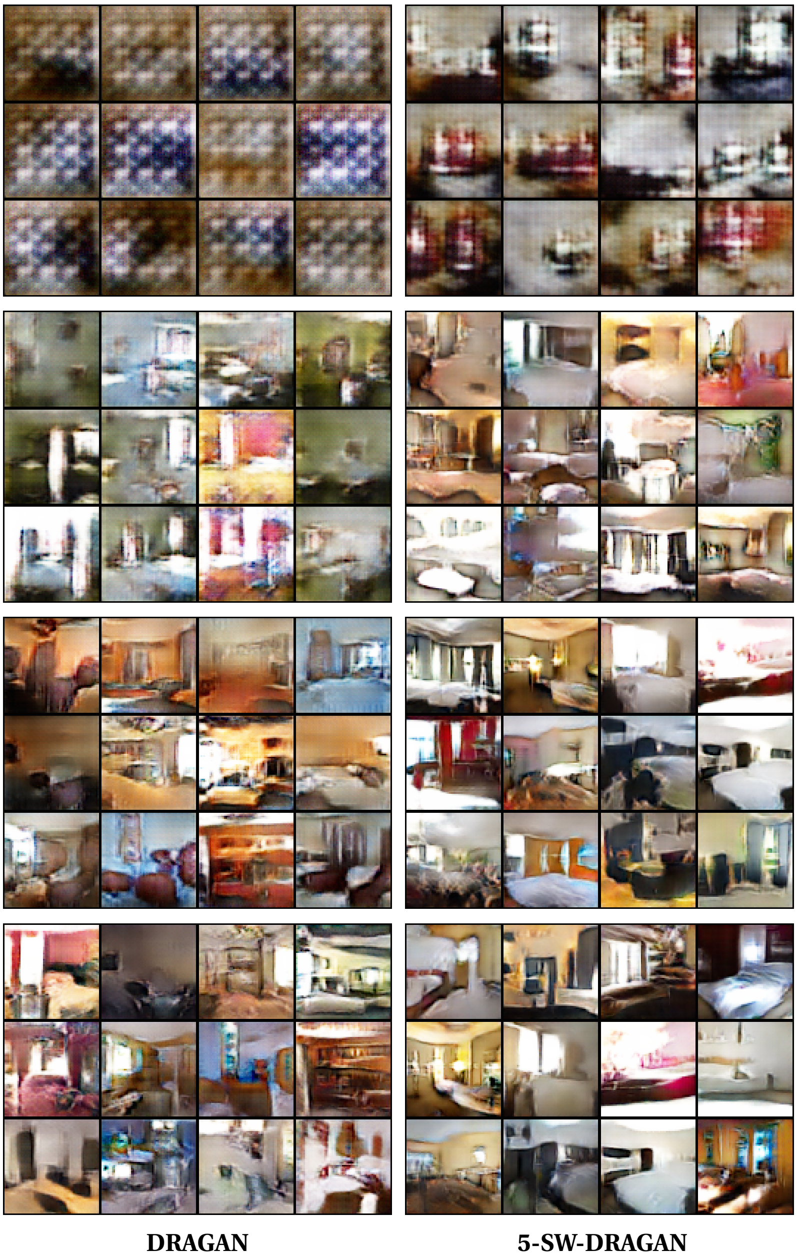


Figure B.9 – **DRAGAN** and **5-SW-DRAGAN** on **LSUN-bedroom** at the 1000-*th*, $5{\cdot}10^3$-*th*, $10{\cdot}10^3$-*th* and $14{\cdot}10^3$-*th* iteration, from top to bottom row, respectively. Using 64×64 real data space.

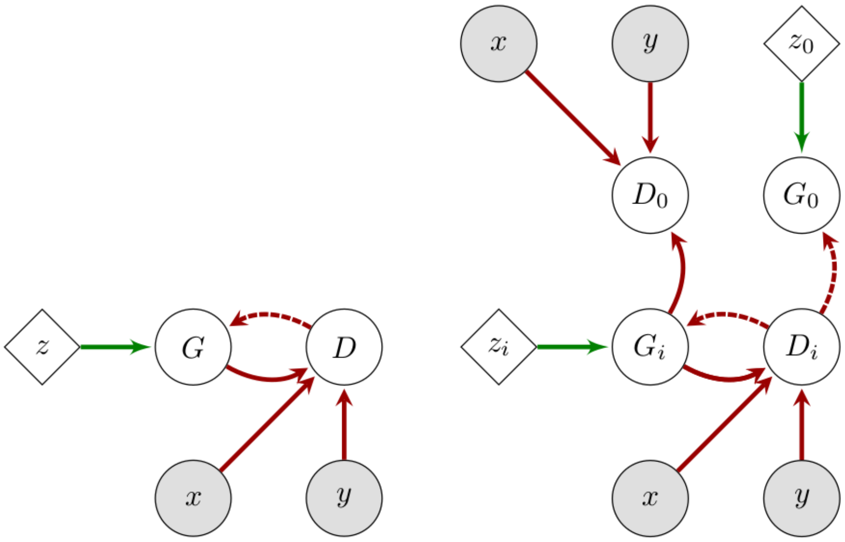
```
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

```
The Larntare F bucnt 1h setirder
The tielirc indian on orabo stir
We dola wan Fobkbomn hrcas and 8
Tod letrocsix r telt car rntr ce
Then on s indent on leWand ghis
Ja candt conteltirt in ald do ce
Gaid nochir Weabsilan of ansany
The ticosgose arc on Mesinntelat
More wucarcanosnped rochusroe t‘
They derato raEyand soalceatecst
De Ths chsrc s aeareP thel ea t
```

```
" S4vvvoFlnls anr ans ffrcinns s
Thon taa forinint ssroso siwfd f
Hothst bffld ’nvlonyoiar" cov sh
Woisi’n fof Monisg dhak N‘f fnv
ThD fas ong fafpn so n wns is of
D" wiay wyd alvriMbnlor M nld ff
DDd4y vooc onl vocfay w s offo f
" c4Df co đ noy soonlono ans war
ln wns bfrfncorfiw Thofv lnnd fo
Dy wad Dld at N fovl dcy fot aor
ThD doDn bacd d vffnonlo anfofin
Th’ toits isg thid st hsgo ffffs
" coracgod Mfopf đlny thisg  aff
```

```
The conareed same ming tay spid
Then the gioncolly the can id co
She  beme lant arecong nelode .
It taecanting the they trehos so
In the later asterol antarlist n
Sion ot tndy ttin an os pomcerer
The pither canned Sblets castery
They BastitiBented tome man angu
I lant taot suncedrthet prourpli
And Biecon eels acecount tre Car
This tain Datertlals comegel yan
Is the Woilg  ate costort thab f
Thin Incin is Inlasar cumand mot
```

Figure B.10 – Snippets from **WGAN** (left) and **5-SW-WGAN** (right) on the **One Billion Word Benchmark**, taken at the 700-*th* and 2500-*th* iteration (top and bottom row, respectively).



(a) GAN training [Pfau and Vinyals, 2016]

(b) SGAN training

Figure B.11 – Graphical representations [Pfau and Vinyals, 2016] of the information flow structures of GAN and SGAN training. See § B.3.

C Appendix for Chapter 4

C.0.1 Why is convergence of the last iterate preferable?

In light of Theorem 1, the behavior of the iterates on the unconstrained version of (4.1) ($\epsilon = 0$):

$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\varphi} \in \Phi} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\theta}^\top \boldsymbol{A}_i \boldsymbol{\varphi} \quad \text{where} \quad [\boldsymbol{A}_i]_{kl} = 1 \text{ if } k = l = i \text{ and } 0 \text{ otherwise.} \tag{C.1}$$

where Θ and Φ are compact and convex sets, is the following: they will diverge until they reach the boundary of Θ and Φ and then they will start to turn around the Nash equilibrium of (C.1) lying on these boundaries. Using convexity properties, we can then show that the averaged iterates will converge to the Nash equilibrium of the problem. However, with an arbitrary large domain, this convergence rate may be arbitrary slow (since it depends on the diameter of the domain).

Moreover, this behavior might be even more problematic in a non-convex framework because even if by chance we initialize close to the Nash equilibrium, we would get away from it and we cannot rely on convexity to expect the average of the iterates to converge.

Consequently, we would like optimization algorithms generating iterates that *stay close to the Nash equilibrium.*

C.1 Definitions and Lemmas

C.1.1 Smoothness and Monotonicity of the operator

Another important property used is the Lipschitzness of an operator.

Definition 4. *A mapping $F : \mathbb{R}^p \to \mathbb{R}^d$ is said to be L-Lipschitz if,*

$$\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|_2 \leq L \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2 , \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega . \tag{C.2}$$

Definition 5. *A differentiable function $f : \Omega \to \mathbb{R}$ is said to be μ-*strongly convex *if*

$$f(\boldsymbol{\omega}) \geq f(\boldsymbol{\omega}') + \nabla f(\boldsymbol{\omega}')^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') + \frac{\mu}{2} \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega \,. \tag{C.3}$$

Definition 6. *A function $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \mapsto \mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\varphi})$ is said convex-concave if $\mathscr{L}(\cdot, \boldsymbol{\varphi})$ is convex for all $\boldsymbol{\varphi} \in \Phi$ and $\mathscr{L}(\boldsymbol{\theta}, \cdot)$ is concave for all $\boldsymbol{\theta} \in \Theta$. An $\mathscr{L}$ is said to be μ-strongly convex concave if $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \mapsto \mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) - \frac{\mu}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{\mu}{2}\|\boldsymbol{\varphi}\|_2^2$ is convex concave.*

Definition 7. *For $\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}} > 0$, an operator $F : \boldsymbol{\omega} \mapsto (F_{\boldsymbol{\theta}}(\boldsymbol{\omega}), F_{\boldsymbol{\varphi}}(\boldsymbol{\omega})) \in \mathbb{R}^{d+p}$ is said to be $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strongly monotone if $\forall\, \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega \subset \mathbb{R}^{p+d}$,*

$$(F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') \geq \mu_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 + \mu_{\boldsymbol{\varphi}} \|\boldsymbol{\varphi} - \boldsymbol{\varphi}'\|^2 \,.$$

where we noted $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \mathbb{R}^{d+p}$.

Definition 8. *An operator $F : (\boldsymbol{\omega}), \in \mathbb{R}^d$ is said to be ℓ-cocoercive, if for all $\boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega$ we have*

$$\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|^2 \leq \ell (F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') \,. \tag{C.4}$$

Proposition 1 (Folklore)**.** *A L-Lipschitz and μ-strongly monotone operator is L^2/μ-cocoercive*

Proof. By applying lipschitzness and strong monotonicity,

$$\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|^2 \leq L^2 \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|^2 \leq L^2/\mu (F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') \tag{C.5}$$

□

Proposition 2. *If $F(\boldsymbol{\omega}) = (\nabla f(\boldsymbol{\theta}) + M\boldsymbol{\varphi}, \nabla g(\boldsymbol{\varphi}) - M^\top \boldsymbol{\theta})$, where f and g are μ-strongly convex and L smooth, then $\|M\|^2 = O(\mu L)$ is a sufficient condition for F to be ℓ-cocoercive with $\ell = O(L)$*

Proof. We rewrite F as the sum of the gradient of convex Lipschitz function F_{grad} and a L-Lipschitz and μ-strongly monotone operator F_{mon}:

$$F_{grad}(\boldsymbol{\omega}) := (\nabla f(\boldsymbol{\theta}) - \mu\boldsymbol{\theta}, \nabla g(\boldsymbol{\varphi}) - \mu\boldsymbol{\varphi}) \quad \text{and} \quad F_{mon} : (M\boldsymbol{\varphi} + \mu\boldsymbol{\theta}, -M^\top \boldsymbol{\theta} + \mu\boldsymbol{\varphi}) \tag{C.6}$$

Then

$$\begin{aligned}
\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|^2 &\leq 2\|F_{grad}(\boldsymbol{\omega}) - F_{grad}(\boldsymbol{\omega}')\|^2 + 2\|F_{mon}(\boldsymbol{\omega}) - F_{mon}(\boldsymbol{\omega}')\|^2 && \text{(C.7)}\\
&\leq 2(L+\mu)(F_{grad}(\boldsymbol{\omega}) - F_{grad}(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') && \text{(C.8)}\\
&\quad + 2(\|M\| + \mu)^2/\mu (F_{mon}(\boldsymbol{\omega}) - F_{mon}(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') && \text{(C.9)}\\
&= O(L)(F_g rad(\boldsymbol{\omega}) - F_g rad(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') && \text{(C.10)}\\
&\quad + O(L)(F_{mon}(\boldsymbol{\omega}) - F_{mon}(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') && \text{(C.11)}\\
&= O(L)(F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') && \text{(C.12)}
\end{aligned}$$

where for the second inequality we used that a $(L+\mu)$-Lipschitz convex function is $(L+\mu)$-cocoercive and Proposition 2. □

C.2 Proof of Theorems

C.2.1 Proof of Theorem 1

Proof. We consider the following stochastic optimization problem,

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\epsilon}{2}\theta_i^2+\boldsymbol{\theta}^\top \boldsymbol{A}_i\boldsymbol{\varphi}-\frac{\epsilon}{2}\varphi_i^2=\frac{1}{n}\sum_{i=1}^{n}\frac{\epsilon}{2}\|\boldsymbol{A}_i\boldsymbol{\theta}\|^2+\boldsymbol{\theta}^\top \boldsymbol{A}_i\boldsymbol{\varphi}-\frac{\epsilon}{2}\|\boldsymbol{A}_i\boldsymbol{\varphi}\|^2 \tag{C.13}$$

where $[\boldsymbol{A}_i]_{kl}=1$ if $k=l=i$ and 0 otherwise. Note that $(\boldsymbol{A}_i)^\top=\boldsymbol{A}_i$ for $1\le i\le n$. Let us consider the extragradient method where to compute an unbiased estimator of the gradients at $(\boldsymbol{\theta},\boldsymbol{\varphi})$ we sample $i\in\{1,\dots,n\}$ and use $[\boldsymbol{A}_i\boldsymbol{\theta},\ \boldsymbol{A}_i\boldsymbol{\varphi}]$ as estimator of the vector flow.

In this proof we note, $\boldsymbol{A}_I:=\sum_{i\in I}\boldsymbol{A}_i$ and $\boldsymbol{\theta}^{(I)}$ the vector such that $[\boldsymbol{\theta}^{(I)}]_i=[\boldsymbol{\theta}]_i$ if $i\in I$ and 0 otherwise. Note that $\boldsymbol{A}_I\boldsymbol{\theta}=\boldsymbol{\theta}^{(I)}$ and that $\boldsymbol{A}_I\boldsymbol{A}_J=\boldsymbol{A}_{I\cap J}$.

Thus the extragradient update rule can be noted as

$$\begin{cases}\boldsymbol{\theta}_{t+1}=(1-\eta\boldsymbol{A}_I\epsilon)\boldsymbol{\theta}_t-\eta\boldsymbol{A}_I((1-\eta\boldsymbol{A}_J\epsilon)\boldsymbol{\varphi}_t+\eta\boldsymbol{A}_J\boldsymbol{\theta}_t)\\ \boldsymbol{\varphi}_{t+1}=(1-\eta\boldsymbol{A}_I\epsilon)\boldsymbol{\varphi}_t+\eta\boldsymbol{A}_I((1-\eta\boldsymbol{A}_J\epsilon)\boldsymbol{\theta}_t-\eta\boldsymbol{A}_J\boldsymbol{\varphi}_t)\end{cases} \tag{C.14}$$

where I is the mini-batch sampled (without replacement) for the update and J the mini-batch sampled (without replacement) for the extrapolation.

We can thus notice that, when $I\cap J=\emptyset$, we have

$$\begin{cases}\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\eta\epsilon\boldsymbol{\theta}_t^{(I)}-\eta\boldsymbol{\varphi}_t^{(I)}\\ \boldsymbol{\varphi}_{t+1}=\boldsymbol{\varphi}_t-\eta\epsilon\boldsymbol{\varphi}_t^{(I)}+\eta\boldsymbol{\theta}_t^{(I)},\end{cases} \tag{C.15}$$

and otherwise,

$$\begin{cases}\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\eta\epsilon\boldsymbol{\theta}_t^{(I)}-\eta\boldsymbol{\varphi}_t^{(I)}-\eta^2(\boldsymbol{\theta}_t^{(I\cap J)}-\epsilon\boldsymbol{\varphi}_t^{(I\cap J)})\\ \boldsymbol{\varphi}_{t+1}=\boldsymbol{\varphi}_t-\eta\epsilon\boldsymbol{\varphi}_t^{(I)}+\eta\boldsymbol{\theta}_t^{(I)}-\eta^2(\boldsymbol{\varphi}_t^{(I\cap J)}+\epsilon\boldsymbol{\theta}_t^{(I\cap J)})\,.\end{cases} \tag{C.16}$$

The intuition is that, on one hand, when $I\cap J=\emptyset$ (which happens with high probability when $|I|<<n$, e.g., when $|I|=1$, $\mathbb{P}(I\cap J=\emptyset)=1-1/n$), the algorithm performs an update that get away from the Nash equilibrium when $2\epsilon\ge\eta$:

$$\text{(C.15)}\ \Rightarrow\ N_{t+1}=N_t+(\eta^2\epsilon^2+\eta^2-2\eta\epsilon)N_t^{(I)}\,, \tag{C.17}$$

where $N_t:=\|\boldsymbol{\theta}_t\|^2+\|\boldsymbol{\varphi}_t\|^2$ and $N_t^{(I)}:=\|\boldsymbol{\theta}_t^{(I)}\|^2+\|\boldsymbol{\varphi}_t^{(I)}\|^2$. On the other hand, The updates that provide improvement only happen when $I\cap J$ is large (which happen with low probability, e.g., when $|I| = 1$, $\mathbb{P}(I \cap J \neq \emptyset) = 1/n$):

$$\text{(C.16)} \Rightarrow N_{t+1} = N_t - N_t^{(I)}(2\eta\epsilon - \eta^2(1+\epsilon^2)) - N_t^{(I\cap J)}(2\eta^2 - \eta^4(1+\epsilon^2)) \tag{C.18}$$

Conditioning on $\boldsymbol{\theta}_t$ and $\boldsymbol{\varphi}_t$, we get that

$$\mathbb{E}[N_t^{(I\cap J)}|\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = \sum_{i=1}^n \mathbb{P}(i \in I \cap J)([\boldsymbol{\theta}_t]_i^2 + [\boldsymbol{\varphi}_t]_i^2) \quad \text{and} \quad \mathbb{P}(i \in I \cap J) = \mathbb{P}(i \in I)\mathbb{P}(i \in J) = \frac{|I|^2}{n^2}. \tag{C.19}$$

Leading to,

$$\mathbb{E}[N_t^{(I\cap J)}|\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = \frac{|I|^2}{n^2}\sum_{i=1}^n ([\boldsymbol{\theta}_t]_i^2 + [\boldsymbol{\varphi}_t]_i^2) = \frac{|I|^2}{n^2} N_t \quad \text{and} \quad \mathbb{E}[N_t^{(I)}|\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = \frac{|I|}{n} N_t. \tag{C.20}$$

Plugging these expectations in (C.18), we get that,

$$\mathbb{E}[N_{t+1}] = \left(1 - \tfrac{|I|}{n}(2\eta\epsilon - \eta^2(1+\epsilon^2)) - \tfrac{|I|^2}{n^2}(2\eta^2 - \eta^4(1+\epsilon^2))\right)\mathbb{E}[N_t]. \tag{C.21}$$

Consequently for $\eta < \epsilon$ we get,

$$\mathbb{E}[N_{t+1}] \geq \left(1 - 2\eta^2\frac{|I|^2}{n^2} + \eta^2\frac{|I|}{n}\right)\mathbb{E}[N_t]. \tag{C.22}$$

To sum-up, if $|I|$ is not large enough (more precisely if $2|I| \leq n$), we have the geometric divergence of the quantity $\mathbb{E}[N_t] := \mathbb{E}[\|\boldsymbol{\theta}_t\|^2 + \|\boldsymbol{\varphi}_t\|^2]$ for any $\eta \geq \epsilon$. □

C.2.2 Proof of Theorem 2

Setting of the Proof. We will prove a slightly more general result than Theorem 2. We will work in the context of monotone operator. Let us consider the *general* extrapolation update rule,

$$\begin{cases} \text{Extrapolation:} & \boldsymbol{\omega}_{t+\frac{1}{2}} = \boldsymbol{\omega}_t - \eta_t \boldsymbol{g}_t \\ \text{Update:} & \boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta_t \boldsymbol{g}_{t+1/2}, \end{cases} \tag{C.23}$$

where $\boldsymbol{g}_t$ depends on $\boldsymbol{\omega}_t$ and $\boldsymbol{g}_{t+1/2}$ depends on $\boldsymbol{\omega}_{t+1/2}$. For instance, $\boldsymbol{g}_t$ can either be $F(\boldsymbol{\omega}_t)$, $F_{i_t}(\boldsymbol{\omega}_t)$ or the SVRG estimate defined in (C.35).

This update rule generalizes (EG) for 2-player games (2P-G) and ExtraSVRG (Alg. 5).

Let us first state a lemma standard in convex analysis (see for instance [Boyd and Vandenberghe, 2004]),

Lemma 1. *Let* $\boldsymbol{\omega} \in \Omega$ *and* $\boldsymbol{\omega}^+ := P_\Omega(\boldsymbol{\omega} + \boldsymbol{u})$ *then for all* $\boldsymbol{\omega}' \in \Omega$ *we have,*

$$\|\boldsymbol{\omega}^+ - \boldsymbol{\omega}'\|_2^2 \leq \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + 2\boldsymbol{u}^\top(\boldsymbol{\omega}^+ - \boldsymbol{\omega}') - \|\boldsymbol{\omega}^+ - \boldsymbol{\omega}\|_2^2. \tag{C.24}$$

Proof of Lemma 1. We start by simply developing,

$$\begin{aligned}\|\boldsymbol{\omega}^+ - \boldsymbol{\omega}'\|_2^2 = \|(\boldsymbol{\omega}^+ - \boldsymbol{\omega}) + (\boldsymbol{\omega} - \boldsymbol{\omega}')\|_2^2 &= \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + 2(\boldsymbol{\omega}^+ - \boldsymbol{\omega})^\top(\boldsymbol{\omega} - \boldsymbol{\omega}') + \|\boldsymbol{\omega}^+ - \boldsymbol{\omega}\|_2^2 \\ &= \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + 2(\boldsymbol{\omega}^+ - \boldsymbol{\omega})^\top(\boldsymbol{\omega}^+ - \boldsymbol{\omega}') - \|\boldsymbol{\omega}^+ - \boldsymbol{\omega}\|_2^2 .\end{aligned}$$

Then since $\boldsymbol{\omega}^+$ is the projection onto the convex set Ω of $\boldsymbol{\omega} + \boldsymbol{u}$ we have that:

$$(\boldsymbol{\omega}^+ - (\boldsymbol{\omega} + \boldsymbol{u}))^\top(\boldsymbol{\omega}^+ - \boldsymbol{\omega}') \leq 0\,,\ \forall\, \boldsymbol{\omega}' \in \Omega$$

leading to the result of the Lemma. □

Lemma 2. *If F is $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strongly monotone for any $\boldsymbol{\omega}, \boldsymbol{\omega}', \boldsymbol{\omega}'' \in \Omega$ we have,*

$$\mu_{\boldsymbol{\theta}}\left(\|\boldsymbol{\theta} - \boldsymbol{\theta}''\|_2^2 - 2\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_2^2\right) + \mu_{\boldsymbol{\varphi}}\left(\|\boldsymbol{\varphi} - \boldsymbol{\varphi}''\|_2^2 - 2\|\boldsymbol{\varphi}' - \boldsymbol{\varphi}\|_2^2\right) \leq 2(F(\boldsymbol{\omega}') - F(\boldsymbol{\omega}''))^\top(\boldsymbol{\omega}' - \boldsymbol{\omega}'')\,, \tag{C.25}$$

where we noted $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi})$.

Proof. By $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strong monotonicity,

$$2\mu_{\boldsymbol{\theta}}\|\boldsymbol{\theta}' - \boldsymbol{\theta}''\|_2^2 + 2\mu_{\boldsymbol{\varphi}}\|\boldsymbol{\varphi}' - \boldsymbol{\varphi}''\|_2^2 \leq 2(F(\boldsymbol{\omega}'') - F(\boldsymbol{\omega}''))^\top(\boldsymbol{\omega}' - \boldsymbol{\omega}'') \tag{C.26}$$

and then we use the inequality $2\|\boldsymbol{a}' - \boldsymbol{a}''\|_2^2 \geq \|\boldsymbol{a} - \boldsymbol{a}''\|_2^2 - 2\|\boldsymbol{a}' - \boldsymbol{a}\|_2^2$ to get the result claimed. □

Using this update rule we can thus deduce the following lemma, the derivation of this lemma is very similar from the derivation of Harker and Pang [1990, Lemma 12.1.10].

Lemma 3. *Considering the update rule* (C.23)*, we have for any $\boldsymbol{\omega} \in \Omega$ and any $t \geq 0$,*

$$2\eta_t \boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 + \eta_t^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2 . \tag{C.27}$$

Proof. By applying Lem. 1 for $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}^+, \boldsymbol{\omega}') = (\boldsymbol{\omega}_t, -\eta_t \boldsymbol{g}_{t+1/2}, \boldsymbol{\omega}_{t+1}, \boldsymbol{\omega})$ and, $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}^+, \boldsymbol{\omega}') = (\boldsymbol{\omega}_t, -\eta_t \boldsymbol{g}_t, \boldsymbol{\omega}_{t+1/2}, \boldsymbol{\omega}_{t+1})$ we get,

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t \boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t\|_2^2\,, \tag{C.28}$$

and

$$\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|_2^2 - 2\eta_t \boldsymbol{g}_t^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 . \tag{C.29}$$

Summing (C.28) and (C.29) we get,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 &- 2\eta_t \boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}) && \text{(C.30)}\\
&- 2\eta_t \boldsymbol{g}_t^\top(\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_{t+1}\|_2^2 && \text{(C.31)}\\
= \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 &- 2\eta_t \boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_{t+1}\|_2^2 \\
&- 2\eta_t(\boldsymbol{g}_t-\boldsymbol{g}_{t+1/2})^\top(\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_{t+1})\,. && \text{(C.32)}
\end{aligned}$$

Then, we can use Young's inequality $-2a^\top b \leq \|a\|_2^2 + \|b\|_2^2$ to get,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 &- 2\eta_t \boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}) + \eta_t^2\|\boldsymbol{g}_t-\boldsymbol{g}_{t+1/2}\|_2^2 \\
&+ \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_{t+1}\|_2^2 - \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_{t+1}\|_2^2 && \text{(C.33)}\\
= \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 &- 2\eta_t \boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}) + \eta_t^2\|\boldsymbol{g}_t-\boldsymbol{g}_{t+1/2}\|_2^2 - \|\boldsymbol{\omega}_{t+1/2}-\boldsymbol{\omega}_t\|_2^2\,. && \text{(C.34)}
\end{aligned}$$

□

Note that if we would have set $\boldsymbol{g}_t = \mathbf{0}$ and $\boldsymbol{g}_{t+1/2}$ any estimate of the gradient at $\boldsymbol{\omega}_t$ we recover the standard lemma for gradient method.

Let us consider *unbiased* estimates of the gradient,

$$\boldsymbol{g}_i(\boldsymbol{\omega}) := \frac{1}{n\pi_i}(F_i(\boldsymbol{\omega}) - \boldsymbol{\alpha}_i) + \bar{\boldsymbol{\alpha}}\,, \tag{C.35}$$

where $\bar{\boldsymbol{\alpha}} := \frac{1}{n}\sum_{j=1}^n \boldsymbol{\alpha}_j$, the index i are (potentially) non-uniformly sampled from $\{1,\dots,n\}$ with replacement according to $\boldsymbol{\pi}$ and $F(\boldsymbol{\omega}) := \frac{1}{n}\sum_{j=1}^n F_i(\boldsymbol{\omega})$. Hence we have that $\mathbb{E}[\boldsymbol{g}_i(\boldsymbol{\omega})] = F(\boldsymbol{\omega})$, where the expectation is taken with respect to the index i sampled from $\boldsymbol{\pi}$.

We will consider a class of algorithm called *uniform memorization algorithms* first introduced by [Hofmann et al., 2015]. This class of algorithms describes a large subset of variance reduced algorithms taking advantage of the finite sum formulation such as SAGA [Defazio et al., 2014], SVRG [Johnson and Zhang, 2013] or q-SAGA and $\mathcal{N}$-SAGA [Hofmann et al., 2015]. In this work, we will use a slightly more general definition of such algorithm in order to be able to handle extrapolation steps:

Definition 9 (Extension of [Hofmann et al., 2015])**.** *A uniform q-memorization algorithm evolves iterates $(\boldsymbol{\omega}_t)$ according to* (C.23)*, with $\boldsymbol{g}_t$ defined in* (C.35) *and selecting in each iteration t a random index set J_t of memory locations to update according to,*

$$\boldsymbol{\alpha}_k^{(0)} := F_k(\boldsymbol{\omega}_0)\,, \quad \boldsymbol{\alpha}_k^{(t+1/2)} := \boldsymbol{\alpha}_k^{(t)},\ \forall k \in \{1,\dots,n\} \quad \textit{and} \quad \boldsymbol{\alpha}_k^{(t+1)} := \begin{cases} F_k(\boldsymbol{\omega}_t) & \textit{if } k \in J_t \\ \boldsymbol{\alpha}_k^{(t)} & \textit{otherwise.}\end{cases} \tag{C.36}$$

such that any k has the same probability q/n to be updated, i.e., $P\{k\} = \sum_{J_t, k\in J_t} P(J_t) = q/n$, $\forall k \in \{1,\dots,n\}$.

In the case of SVRG, either $J_t = \emptyset$ or $J_t = \{1, \ldots, n\}$ (when we update the snapshot).

We have the following lemmas,

Lemma 4. *For any $t \geq 0$, if we consider a q-memorization algorithm we have*

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2] \leq & 10\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_i^{(t)})\|^2] \\ & + 10\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - F_i(\boldsymbol{\omega}_t))\|^2] \\ & + 5\bar{L}^2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2]\,.\end{aligned}$$

Proof. We use an extended version of Young's inequality: $\|\sum_{i=1}^k \boldsymbol{a}_i\|^2 \leq k\sum_{i=1}^k \|\boldsymbol{a}_i\|^2$,

$$\begin{aligned}\|\sum_{i=1}^k \boldsymbol{a}_i\|^2 &= \sum_{i,j=1}^k \boldsymbol{a}_i^\top \boldsymbol{a}_j \\ &\leq \frac{1}{2}\sum_{i,j=1}^k \|\boldsymbol{a}_i\|^2 + \|\boldsymbol{a}_j\|^2 \\ &= k\sum_{i=1}^k \|\boldsymbol{a}_i\|^2\,,\end{aligned}$$

where we used that $2\boldsymbol{a}^\top \boldsymbol{b} \leq +\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2$. We combine Young's inequality with the definition of q-memorization algorithm: $\boldsymbol{g}_t = \frac{1}{n\pi_i}(F_i(\boldsymbol{\omega}_t) - \bar{\boldsymbol{\alpha}}_i^{(t)})$ and $\boldsymbol{g}_{t+1/2} = \frac{1}{n\pi_j}(F_j(\boldsymbol{\omega}_{t+1/2}) - \bar{\boldsymbol{\alpha}}_j^{(t)})$ to get (we omit the t subscript for i and j and we note $\bar{\boldsymbol{\alpha}}_i^{(t)} := \boldsymbol{\alpha}_i^{(t)} - n\pi_i\bar{\boldsymbol{\alpha}}^{(t)}$),

$$\begin{aligned}\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2 &= \|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}_t) - \bar{\boldsymbol{\alpha}}_i^{(t)}) - \tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}_{t+1/2}) - \bar{\boldsymbol{\alpha}}_j^{(t)})\|^2 \\ &= \|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}_t) - \bar{\boldsymbol{\alpha}}_i^{(t)}) + \tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2})) + \tfrac{1}{n\pi_j}(\bar{\boldsymbol{\alpha}}_j^{(t)} - F_j(\boldsymbol{\omega}_t))\|^2 \\ &\leq 5\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \bar{\boldsymbol{\alpha}}_i^{(t)}))\|^2] + 5\mathbb{E}[\|\tfrac{1}{n\pi_j}(\bar{\boldsymbol{\alpha}}_j^{(t)} - F_j(\boldsymbol{\omega}^*))\|^2] \\ &\quad + 5\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - F_i(\boldsymbol{\omega}_t))\|^2] + 5\mathbb{E}[\|\tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}^*) - F_j(\boldsymbol{\omega}_t))\|^2] \\ &\quad + 5\mathbb{E}[\|\tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2}))\|^2]\end{aligned}$$

Notice that since i_t and j_t are independently sampled from the same distribution we have

$$\mathbb{E}[\tfrac{1}{n^2\pi_{j_t}^2}\|F_{j_t}(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_{j_t}^{(t)}\|^2] = \mathbb{E}[\tfrac{1}{n^2\pi_{i_t}^2}\|F_{i_t}(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_{i_t}^{(t)}\|^2]\,. \tag{C.37}$$

Note that we have (using that $\mathbb{E}[F_i(\boldsymbol{\omega}^*)] = 0$ and $\mathbb{E}[\boldsymbol{\alpha}_i^{(t)}] = \bar{\boldsymbol{\alpha}}^{(t)}$),

$$\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \bar{\boldsymbol{\alpha}}_i^{(t)})\|^2] = \mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_i^{(t)})\|^2] - \|\bar{\boldsymbol{\alpha}}^{(t)}\|^2 \leq \mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_i^{(t)})\|^2] \tag{C.38}$$

By assuming that each F_i is L_i-Lipschitz we get,

$$\begin{aligned}
\mathbb{E}[\tfrac{1}{n^2\pi_{j_t}^2}\|F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2})\|^2] &= \frac{1}{n^2}\sum_{j=1}^n \frac{1}{\pi_j}\mathbb{E}[\|F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2})\|^2] && \text{(C.39)}\\
&\leq \frac{1}{n^2}\sum_{j=1}^n \frac{L_j^2}{\pi_j}\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] && \text{(C.40)}\\
&= \bar{L}^2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2, && \text{(C.41)}
\end{aligned}$$

where $\bar{L}^2 := \frac{1}{n^2}\sum_{i=1}^n \frac{L_i^2}{\pi_j}$. Note that $\boldsymbol{\omega}_t$ and $\boldsymbol{\omega}_{t+1/2}$ do not depend on j_t (which is the index sampled for the update step), that is not the case for i (the index for the extrapolation step) since $\boldsymbol{\omega}_{t+1/2}$ is the result of the extrapolation. □

This lemma make appear the quantity $\mathbb{E}[\|\frac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \bar{\boldsymbol{\alpha}}_i^{(t)})\|^2]$ that we need to bound. In order to do that we prove the following lemma,

Lemma 5. *Let $(\boldsymbol{\alpha}_j^{(t)})$ be updated according to the rules of a q-uniform memorization algorithm (Def. 9). Let us note $H_t := \frac{1}{n}\sum_{i=1}^n \frac{1}{n\pi_i}\|F_i(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_i^{(t)}\|^2$. For any $t \in \mathbb{N}$,*

$$\mathbb{E}[H_{t+1}] = \frac{q}{n}\mathbb{E}[\|\tfrac{1}{n\pi_{i_t}}(F_{i_t}(\boldsymbol{\omega}_t) - F_{i_t}(\boldsymbol{\omega}^*))\|^2] + \frac{n-q}{n}\mathbb{E}[H_t]. \qquad \text{(C.42)}$$

Proof. We will use the definition of q-uniform memorization algorithms (saying that $\boldsymbol{\alpha}_i$ is updated at time $t+1$ with probability q/n). We call this event "i updated",

$$\begin{aligned}
\mathbb{E}[H_{t+1}] &:= \mathbb{E}[\frac{1}{n}\sum_{i=1}^n \tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t+1)} - F_i(\boldsymbol{\omega}^*)\|^2]\\
&= \frac{1}{n}\mathbb{E}[\sum_{i \text{ updated}} \tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t+1)} - F_i(\boldsymbol{\omega}^*)\|^2 + \sum_{i \text{ not updated}} \tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t+1)} - F_i(\boldsymbol{\omega}^*)\|^2]\\
&= \frac{1}{n}\mathbb{E}[\sum_{i \text{ updated}} \tfrac{1}{n\pi_i}\|F_i(\boldsymbol{\omega}_t) - F_i(\boldsymbol{\omega}^*)\|^2 + \sum_{i \text{ not updated}} \tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t)} - F_i(\boldsymbol{\omega}^*)\|^2]\\
&= \frac{1}{n}\sum_{i=1}^n \mathbf{P}(i \text{ updated})\tfrac{1}{n\pi_i}\mathbb{E}[\|F_i(\boldsymbol{\omega}_t) - F_i(\boldsymbol{\omega}^*)\|^2 + \frac{1}{n}\sum_{i=1}^n \mathbf{P}(i \text{ not updated})\tfrac{1}{n\pi_i}\mathbb{E}[\|\boldsymbol{\alpha}_i^{(t)} - F_i(\boldsymbol{\omega}^*)\|^2]\\
&= \frac{q}{n}\mathbb{E}[\|\tfrac{1}{n\pi_{i_t}}(F_{i_t}(\boldsymbol{\omega}_t) - F_{i_t}(\boldsymbol{\omega}^*))\|^2] + \frac{n-q}{n}\mathbb{E}[H_t]
\end{aligned}$$

□

Using all these lemmas we can prove our theorem.

Theorem' 2. *Under Assumption 1, after t iterations, the iterate $\boldsymbol{\omega}_t$ computed by a q-memorization algorithm with step-sizes $(\eta_\theta, \eta_\phi) \leq \big((40\bar{\ell}_{\boldsymbol{\theta}})^{-1}, (40\bar{\ell}_{\boldsymbol{\varphi}})^{-1}\big)$ verifies:*

$$\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] \leq \left(1 - \min\left\{\frac{\eta\mu}{2} + \frac{9\eta^2\gamma^2}{10}, \frac{2q}{5n}\right\}\right)^t \mathbb{E}[\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2]. \qquad \text{(C.43)}$$

Proof. In this proof we will consider a constant step-size $\eta_t = (\eta_{\boldsymbol{\theta}}, \eta_{\phi})$. For simplicity of notations we will consider the notation,

$$\bar{L}^2\|\boldsymbol{\omega}\|^2 := \bar{L}_{\boldsymbol{\theta}}^2\|\boldsymbol{\theta}\|^2 + \bar{L}_{\boldsymbol{\varphi}}^2\|\boldsymbol{\varphi}\|^2\,, \quad \eta^2\|\boldsymbol{\omega}\|^2 := \eta_{\boldsymbol{\theta}}^2\|\boldsymbol{\theta}\|^2 + \eta_{\boldsymbol{\varphi}}^2\|\boldsymbol{\varphi}\|^2\,, \quad \mu\|\boldsymbol{\omega}\|^2 := \mu_{\boldsymbol{\theta}}^2\|\boldsymbol{\theta}\|^2 + \mu_{\boldsymbol{\varphi}}^2\|\boldsymbol{\varphi}\|^2$$
$$\eta\mu = (\eta_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\varphi}}\mu_{\boldsymbol{\varphi}})\,, \quad \sigma\bar{L}^2 = (\sigma_{\boldsymbol{\theta}}\bar{L}_{\boldsymbol{\theta}}^2, \sigma_{\boldsymbol{\varphi}}\bar{L}_{\boldsymbol{\varphi}}^2) \quad \text{and} \quad \eta^2\bar{L}^2 = (\eta_{\boldsymbol{\theta}}^2\bar{L}_{\boldsymbol{\theta}}^2, \eta_{\boldsymbol{\varphi}}^2\bar{L}_{\boldsymbol{\varphi}}^2)\,.$$

We start by recalling Lemma 3,

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2 \le \|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2 - 2\eta\boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*) - (1 - 2\eta\mu)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 + \eta^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2\,. \tag{C.44}$$

We can then take the expectation and plug-in the expression of $\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2]$ from Lemma 4,

$$\begin{aligned}
\mathbb{E}[\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2] \le \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] &- 2\eta\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] \\
&- (1 - 2\eta\mu - 5\eta^2\bar{L}^2)]\mathbb{E}[\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2] \\
&+ \eta^2(10\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_i^{(t)})\|^2] + 10\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - F_i(\boldsymbol{\omega}_t))\|^2])\,.
\end{aligned}$$

Let us define $\mathscr{L}_t := \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] + \sigma\mathbb{E}[H_t]$, where $H_t := \frac{1}{n}\sum_{i=1}^n \frac{1}{n\pi_i}\|F_i(\boldsymbol{\omega}^*) - \boldsymbol{\alpha}_i^{(t)}\|^2$. We can combine (C.44) with Lemma 5 multiplied by a constant $\sigma > 0$ that we will set later to get

$$\begin{aligned}
\mathscr{L}_{t+1} &= \mathbb{E}[\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2] + \sigma\mathbb{E}[H_{t+1}] \\
&\le \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - 2\eta\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] - (1 - 2\eta\mu - 5\eta^2\bar{L}^2)\mathbb{E}[\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2] \\
&\quad + (\tfrac{\sigma q}{n} + 10\eta^2)\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - F_i(\boldsymbol{\omega}_t))\|^2] + (\tfrac{10\eta^2}{\sigma} + \tfrac{n-q}{n})\sigma\mathbb{E}[H_t]\,.
\end{aligned}$$

Since i_t and j_t are independently drawn from the same distribution, we have,

$$\mathbb{E}[\|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - F_i(\boldsymbol{\omega}_t))\|^2] = \mathbb{E}[\|\tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}^*) - F_j(\boldsymbol{\omega}_t))\|^2]$$

and thus,

$$\begin{aligned}
\mathscr{L}_{t+1} &\le \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - 2\eta\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] - (1 - 2\eta\mu - 5\eta^2\bar{L}^2)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 \\
&\quad + (\tfrac{\sigma q}{n} + 10\eta^2)\mathbb{E}[\|\tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}^*) - F_j(\boldsymbol{\omega}_t))\|^2 + (\tfrac{10\eta^2}{\sigma} + \tfrac{n-q}{n})\sigma\mathbb{E}[H_t] \\
&\le \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1 - 2\eta\mu - 5\eta^2\bar{L}^2 - 2(\tfrac{\sigma q}{n} + 10\eta^2)\bar{L}^2)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 \\
&\quad - 2\eta\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] + 2(\tfrac{\sigma q}{n} + 10\eta^2)\mathbb{E}[\|\tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}^*) - F_j(\boldsymbol{\omega}_{t+1/2}))\|^2 \\
&\quad + (\tfrac{10\eta^2}{\sigma} + \tfrac{n-q}{n})\sigma\mathbb{E}[H_t] \\
&\le \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1 - 2\eta\mu - 5\eta^2\bar{L}^2 - 2(\tfrac{\sigma q}{n} + 10\eta^2)\bar{L}^2)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 \\
&\quad - 2\eta\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] + (\tfrac{10\eta^2}{\sigma} + \tfrac{n-q}{n})\sigma\mathbb{E}[H_t] \\
&\quad + 2(\tfrac{\sigma q}{n} + 10\eta^2)\mathbb{E}[\tfrac{\ell_j}{n^2\pi_j^2}(F_j(\boldsymbol{\omega}^*) - F_j(\boldsymbol{\omega}_{t+1/2}))^\top(\boldsymbol{\omega}^* - \boldsymbol{\omega}_{t+1/2})]
\end{aligned}$$

where for the second inequality we used Young's inequality and the Lipchitzness of F_j and for the last one we used the co-coercivity of F_j:

$$\|F_j(\boldsymbol{\omega}) - F_j(\boldsymbol{\omega}')\|^2 \leq \ell_i (F_j(\boldsymbol{\omega}') - F_j(\boldsymbol{\omega}))^\top (\boldsymbol{\omega}' - \boldsymbol{\omega}) . \tag{C.45}$$

Thus using $\pi_j = \frac{\ell_j}{\sum_j \ell_j}$, we get

$$\begin{aligned}
\mathscr{L}_{t+1} &\leq \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1 - 2\eta\mu - 5\eta^2 \bar{L}^2 - 2(\tfrac{\sigma q}{n} + 10\eta^2)\bar{L}^2)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 \\
&\quad - 2\eta\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top (\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] + 2(\tfrac{10\eta^2}{\sigma} + \tfrac{n-q}{n})\sigma\mathbb{E}[H_t] \\
&\quad + 2\bar{\ell}(\tfrac{\sigma q}{n} + 10\eta^2)\mathbb{E}[\tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}^*) - F_j(\boldsymbol{\omega}_{t+1/2}))^\top (\boldsymbol{\omega}^* - \boldsymbol{\omega}_{t+1/2})] \\
&= \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1 - 2\eta\mu - 5\eta^2 \bar{L}^2 - 2(\tfrac{\sigma q}{n} + 10\eta^2)\bar{L}^2)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 \\
&\quad - 2\eta\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top (\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] + 2\bar{\ell}(\tfrac{\sigma q}{n} + 10\eta^2)\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top (\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] \\
&\quad + (\tfrac{10\eta^2}{\sigma} + \tfrac{n-q}{n})\sigma\mathbb{E}[H_t]
\end{aligned}$$

where $\bar{\ell} := \frac{1}{n}\sum_i \ell_i$. Now we can set $\frac{20\eta^2}{\sigma} = \frac{q}{n}$ to get

$$\begin{aligned}
\mathscr{L}_{t+1} &\leq \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1 - 2\eta\mu - 65\eta^2 \bar{L}^2)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 \\
&\quad - \eta(2 - 60\bar{\ell}\eta)\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top (\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] + (1 - \tfrac{q}{2n})\sigma\mathbb{E}[H_t] .
\end{aligned}$$

Finally with $\eta \leq \frac{1}{40\bar{\ell}}$ (note that we always have $\bar{\ell} \geq \bar{L}$ because $\ell_i \geq L_i$) we get

$$\mathscr{L}_{t+1} \leq \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - \eta\frac{1}{2}\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top (\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] - \frac{9}{10}\mathbb{E}[\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2] + (1 - \tfrac{q}{2n})\sigma\mathbb{E}[H_t] .$$

We finaly use the projection-type error bound $\|F_i(\boldsymbol{\omega}_t) - F_i(\boldsymbol{\omega}^*)\|^2 \geq \gamma_i^2 \|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|^2$ the same way as [Azizian et al., 2019] to get,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 &= \eta^2 \|\frac{1}{n\pi_i}(F_i(\boldsymbol{\omega}_t) - \bar{\boldsymbol{\alpha}}_i^{(t)})\|^2 \\
&\geq \frac{\eta^2}{2}\|\frac{1}{n\pi_i}(F_i(\boldsymbol{\omega}_t) - F_i(\boldsymbol{\omega}^*))\|^2 - \eta^2\|\frac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \bar{\boldsymbol{\alpha}}_i^{(t)})\|^2 \\
&\geq \frac{\gamma_i^2\eta^2}{2}\|\frac{1}{n\pi_i}(\boldsymbol{\omega}_t - \boldsymbol{\omega}^*)\|^2 - \eta^2\|\frac{1}{n\pi_i}(F_i(\boldsymbol{\omega}^*) - \bar{\boldsymbol{\alpha}}_i^{(t)})\|^2 .
\end{aligned}$$

Thus we have that,

$$\begin{aligned}
\mathscr{L}_{t+1} &\leq \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - \eta\frac{1}{2}\mathbb{E}[F(\boldsymbol{\omega}_{t+1/2})^\top (\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}^*)] \\
&\quad - \frac{\bar{\gamma}^2\eta^2}{2}\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] + (1 - \tfrac{q}{2n} + \frac{9q}{100n})\sigma\mathbb{E}[H_t] ,
\end{aligned}$$

where $\bar{\gamma}^2 := \frac{1}{n}\sum_{k=1}^n \frac{\gamma_i^2}{n\pi_i}$. We can thus conclude the proof using the strong convexity of F,

$$\mathscr{L}_{t+1} \leq \left(1 - \min\left\{\left(\frac{\eta\mu}{2} + \frac{9\eta^2\bar{\gamma}^2}{20}\right), \frac{2q}{5n}\right\}\right)\mathscr{L}_t .$$

□

C.3 Details on the SVRE–GAN Algorithm

C.3.1 Practical Aspect

Noise dataset. Variance reduction is usually performed on finite sum dataset. However, the noise dataset in GANs (sampling from the noise variable z for the generator G) is in practice considered as an infinite dataset. We considered several ways to cope with this:

- Infinitely taking new samples from a predefined latent distribution p_g. In this case, from a theoretical point of view, in terms of using finite sum formulation, there is no convergence guarantee for SVRE even in the strongly convex case. Moreover, the estimators (4.14) and (4.15) are biased estimator of the gradient (as $\boldsymbol{\mu}_D$ and $\boldsymbol{\mu}_G$ do not estimate the full expectation but a finite sum).
- Sampling a different noise dataset at each epoch, i.e. considering a different finite sum at each epoch. In that case, we are performing a variance reduction of this finite sum over the epoch.
- Fix a finite sum noise dataset for the entire training.

In practice, we did not notice any notable difference between the three alternatives.

Adaptive methods. Particular choices such as the optimization method (*e.g.* Adam [Kingma and Ba, 2015]), learning rates, and normalization, have been established in practice as almost *prerequisite* for convergence[1], in contrast to supervised classification problems where they have been shown to only provide a marginal value [Wilson et al., 2017]. To our knowledge, SVRE is the only method that works with a constant step size for GANs on non-trivial datasets. This combined with the fact that recent works empirically tune the first moment controlling hyperparameter to 0 (β_1, see below) and the variance reduction (VR) one (β_2, see below) to a non-zero value, sheds light on the reason behind the success of Adam on GANs.

However, combining SVRE with adaptive step size scheme on GANs remains an open problem. We first briefly describe the update rule of Adam, and then we propose a new adaptation of it that is more suitable for VR methods, which we refer to as variance reduced Adam (VRAd).

---

[1] For instance, Daskalakis et al. [2018], Gidel et al. [2019] plugged Adam into their principled method to get better results.

Adam. Adam stores an exponentially decaying average of both past gradients m_t and squared gradients v_t, for each parameter of the model:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{C.46}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{C.47}$$

where $\beta_1, \beta_2 \in [0,1]$, $m_0 = 0$, $v_0 = 0$, and $t = 1, \dots T$ denotes the iteration. m_t and v_t are respectively the estimates of the first and the second moments of the stochastic gradient. To compensate the bias toward 0 due to initialization, Kingma and Ba [2015] propose to use bias-corrected estimates of these first two moments:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{C.48}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{C.49}$$

The Adam update rule can be described as:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}. \tag{C.50}$$

Adam can be understood as an approximate gradient method with a diagonal step size of $\eta_{Adam} := \frac{\eta}{\sqrt{\boldsymbol{v}_t} + \epsilon}$. Since VR methods aim to provide a vanishing $\boldsymbol{v}_t$, they lead to a too large step-size η_{Adam} of $\frac{\eta}{\epsilon}$. This could indicate that the update rule of Adam may not be a well-suited method to combine with VR methods.

VRAd. This motivates the introduction of a new Adam-inspired variant of adaptive step sizes that maintain a reasonable size even when $\boldsymbol{v}_t$ vanishes,

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta \frac{|\hat{\boldsymbol{m}}_t|}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} \hat{\boldsymbol{m}}_t. \tag{VRAd}$$

This adaptive variant of Adam is motivated by the step size $\eta^* = \eta \frac{\boldsymbol{m}_t^2}{\boldsymbol{v}_t}$ derived by Schaul et al. [2013]. (VRAd) is simply the square-root of η^* in order to stick with Adam's scaling of $\boldsymbol{v}_t$.

C.4 Restarted SVRE

Alg. 6 describes the restarted version of SVRE presented in § 4.3.3. With a probability p (fixed) before the computation of $\boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}}$ and $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}}$, we decide whether to restart SVRE (by using the averaged iterate as the new starting point–Alg. 6, Line 6–$\bar{\boldsymbol{\omega}}_t$) or computing the batch snapshot at a point $\boldsymbol{\omega}_t$.

Algorithm 6 Pseudocode for Restarted SVRE.

```
Input: Stopping time $T$, learning rates $\eta_{\theta}, \eta_{\varphi}$, both players' losses $\mathcal{L}^G$ and $\mathcal{L}^D$, probability of restart $p$.
Initialize: $\varphi, \theta, t = 0$                                  ▷ $t$ is for the online average computation.
for $e = 0$ to $T-1$ do
    Draw restart $\sim \mathrm{B}(p)$.                                ▷ Check if we restart the algorithm.
    if restart and $e > 0$ then
        $\varphi \leftarrow \bar{\varphi}, \quad \theta \leftarrow \bar{\theta}$ and $t = 1$
    end if
    $\varphi^{\mathcal{S}} \leftarrow \varphi$ and $\mu_{\varphi}^{\mathcal{S}} \leftarrow \frac{1}{|\mathcal{Z}|}\sum_{i=1}^{n} \nabla_{\varphi} \mathcal{L}_i^D(\theta^{\mathcal{S}}, \varphi^{\mathcal{S}})$
    $\theta^{\mathcal{S}} \leftarrow \theta$ and $\mu_{\theta}^{\mathcal{S}} \leftarrow \frac{1}{|\varphi|}\sum_{i=1}^{n} \nabla_{\theta} \mathcal{L}_i^G(\theta^{\mathcal{S}}, \varphi^{\mathcal{S}})$
    $N \sim \mathrm{Geom}\big(1/n\big)$                                ▷ Length of the epoch.
    for $i = 0$ to $N-1$ do
        Sample $i_{\theta} \sim \pi_{\theta}$, $i_{\varphi} \sim \pi_{\varphi}$, do extrapolation:
        $\tilde{\varphi} \leftarrow \varphi - \eta_{\theta} d_{\varphi}(\theta, \varphi, \theta^{\mathcal{S}}, \varphi^{\mathcal{S}})$ , $\tilde{\theta} \leftarrow \theta - \eta_{\varphi} d_{\theta}(\theta, \varphi, \theta^{\mathcal{S}}, \varphi^{\mathcal{S}})$      ▷ (4.14) and (4.15)
        Sample $i_{\theta} \sim \pi_{\theta}$, $i_{\varphi} \sim \pi_{\varphi}$, do update:
        $\varphi \leftarrow \varphi - \eta_{\theta} d_{\varphi}(\tilde{\theta}, \tilde{\varphi}, \theta^{\mathcal{S}}, \varphi^{\mathcal{S}})$ , $\theta \leftarrow \theta - \eta_{\varphi} d_{\theta}(\tilde{\theta}, \tilde{\varphi}, \theta^{\mathcal{S}}, \varphi^{\mathcal{S}})$      ▷ (4.14) and (4.15)
        $\bar{\theta} \leftarrow \frac{t}{t+1}\bar{\theta} + \frac{1}{t+1}\theta$ and $\bar{\varphi} \leftarrow \frac{t}{t+1}\bar{\varphi} + \frac{1}{t+1}\varphi$      ▷ Online computation of the average.
        $t \leftarrow t + 1$                                ▷ Increment $t$ for the online average computation.
    end for
end for
Output: $\theta, \varphi$
```

C.5 Details on the implementation

For our experiments, we used the PyTorch[2] deep learning framework, whereas for computing the FID and IS metrics, we used the provided implementations in Tensorflow[3].

C.5.1 Metrics

We provide more details on the implementation of the metrics enumerated in § 4.5. Both FID and IS use: (i) the *Inception v3 network* [Szegedy et al., 2015] that has been trained on the ImageNet dataset consisting of ~1 million RGB images of 1000 classes, $C = 1000$. (ii) a sample of m generated images $x \sim p_g$, where usually $m = 50000$.

Inception Score

Note that the range of IS scores (see definition of IS in § 2.4.1) at convergence varies across datasets, as the Inception network is pretrained on the ImageNet classes. For example, we obtain low IS values on the SVHN dataset as a large fraction of classes are numbers, which

---

[2] https://pytorch.org/

[3] https://www.tensorflow.org/

typically do not appear in the ImageNet dataset. Since **MNIST** has greyscale images, we used a classifier trained on this dataset and used $m = 5000$. For the rest of the datasets, we used the original implementation[4] of IS in TensorFlow, and $m = 50000$.

Fréchet Inception Distance

The FID metric is defined in § 2.4.1 We used the original implementation of FID[5] in Tensorflow, along with the provided statistics of the datasets.

Second Moment Estimate

To evaluate SVRE effectively, we used the **second moment estimate** (SME, uncentered variance, see § C.3.1) of the gradient estimate throughout the iterations $t = 1 \dots T$ per parameter, computed as: $v_t = \gamma v_{t-1} + (1-\gamma) g_t^2$, where g_t denotes the gradient estimate for the parameter and iteration t, and $\gamma = 0.9$. For SVRE, g_t is $\boldsymbol{d_\varphi}$ and $\boldsymbol{d_\theta}$ (see Eq. 4.14 and 4.15) for G and D, respectively. We initialize $g_0 = 0$ and we use bias-corrected estimates: $\hat{v} = \frac{v_t}{1-\gamma^t}$. As the second moment estimate is computed per each parameter of the model, we depict the average of these values for the parameters of G and D separately.

In this work, as we aim at assessing if SVRE *effectively* reduces the variance of the gradient updates, we use SME in our analysis as it is computationally inexpensive and fast to compute.

Entropy & Total Variation on MNIST

For the experiments on **MNIST** illustrated in Fig. C.1a & 4.2b in § 4.5, we plot in § C.6 the **entropy** (E) of the generated samples' class distribution, as well as the **total variation** (TV) between the class distribution of the generated samples and a uniform one (both computed using a pretrained network that classifies its 10 classes).

C.5.2 Architectures & Hyperparameters

Description of the architectures. We describe the models we used in the empirical evaluation of SVRE by listing the layers they consist of, as adopted in GAN works, e.g. [Miyato et al., 2018]. With "conv." we denote a convolutional layer and "transposed conv" a transposed convolution layer [Radford et al., 2016]. The models use Batch Normalization [Ioffe and Szegedy, 2015] and Spectral Normalization layers [Miyato et al., 2018].

---

[4] https://github.com/openai/improved-gan/

[5] https://github.com/bioinf-jku/TTUR

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
transposed conv. (ker: 3×3, 128 → 512; stride: 1)
Batch Normalization
ReLU
transposed conv. (ker: 4×4, 512 → 256, stride: 2)
Batch Normalization
ReLU
transposed conv. (ker: 4×4, 256 → 128, stride: 2)
Batch Normalization
ReLU
transposed conv. (ker: 4×4, 128 → 1, stride: 2, pad: 1)
$Tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{1\times 28\times 28}$
conv. (ker: 4×4, 1 → 64; stride: 2; pad:1)
LeakyReLU (negative slope: 0.2)
conv. (ker: 4×4, 64 → 128; stride: 2; pad:1)
Batch Normalization
LeakyReLU (negative slope: 0.2)
conv. (ker: 4×4, 128 → 256; stride: 2; pad:1)
Batch Normalization
LeakyReLU (negative slope: 0.2)
conv. (ker: 3×3, 256 → 1; stride: 1)
$Sigmoid(\cdot)$

Table C.1 – Description of the DCGAN architectures [Radford et al., 2016] used for experiments on **MNIST**. We use *ker* and *pad* to denote *kernel* and *padding* for the (transposed) convolution layers, respectively. With $h \times w$ we denote the kernel size. With $c_{in} \rightarrow y_{out}$ we denote the number of channels of the input and output for (transposed) convolution layers.

Architectures for experiments on MNIST

For experiments on the **MNIST** dataset, we used the DCGAN architectures [Radford et al., 2016], listed in Table C.1, and the parameters of the models are initialized using PyTorch default initialization. We used mini-batch sizes of 50 samples, whereas for full dataset passes we used mini-batches of 500 samples as this reduces the wall-clock time for its computation. For experiments on this dataset, we used the *non saturating* GAN loss as proposed [Goodfellow et al., 2014]:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_d} \log(D(x)) + \mathbb{E}_{z\sim p_z} \log(D(G(z))) \tag{C.51}$$

$$\mathcal{L}_G = \mathbb{E}_{z\sim p_z} \log(D(G(z))), \tag{C.52}$$

where p_d and p_z denote the data and the latent distributions (the latter to be predefined).

For both the baseline and the SVRE variants we tried the following step sizes $\eta = [1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}]$. We observe that SVRE can be used with larger step sizes. In Table C.7, we used $\eta = 1 \times 10^{-4}$ and $\eta = 1 \times 10^{-2}$ for SE–A and SVRE(–VRAd), respectively.

Choice of architectures on real-world datasets

We replicate the experimental setup described for **CIFAR-10** and **SVHN** in [Miyato et al., 2018], described also below in § C.5.2. We observe that this experimental setup is highly sensitive to the choice of the hyperparameters (see our results in § C.6.3), making it more difficult to compare the optimization methods for a fixed hyperparameter choice. In particular, apart from the different combinations of learning rates for G and D, for the baseline this also included experimenting with: β_1 (see (C.46)), a multiplicative factor of exponential learning rate decay scheduling γ, as well as different ratio of updating G and D per iteration. These observations, combined with that we had limited computational resources, motivated us to use shallower architectures, which we describe below in § C.5.2, and which use an inductive bias of so-called Self–Attention layers [Zhang et al., 2018]. As a reference, our SAGAN and ResNet architectures for **CIFAR-10** have approximately 35 and 85 layers, respectively–in total for G and D, including the non linearity and the normalization layers. For clarity, although the deeper and the shallower architectures differ as they are based on ResNet and SAGAN, we refer these as *deep* (see § C.5.2) and *shallow* (see § C.5.2), respectively.

Shallower SAGAN architectures

We used the SAGAN architectures [Zhang et al., 2018], as the techniques of self-attention introduced in SAGAN were used to obtain the state-of-art GAN results on ImageNet [Brock et al., 2019]. In summary, these architectures: (i) allow for attention-driven, long-range dependency modeling, (ii) use spectral normalization [Miyato et al., 2018] on both G and D (efficiently computed with the *power iteration* method); and (iii) use different learning rates for G and D, as advocated in [Heusel et al., 2017]. The foremost is obtained by combining weights, or alternatively *attention vectors*, with the convolutions across layers, so as to allow modeling textures that are consistent globally–for the generator, or enforcing geometric constraints on the global image structure–for the discriminator.

We used the architectures listed in Table C.3 for **CIFAR-10** and **SVHN** datasets, and the architectures described in Table C.4 for the experiments on **ImageNet**. The models' parameters are initialized using the default initialization of PyTorch.

For experiments with SAGAN, we used the hinge version of the adversarial non-saturating loss [Lim and Ye, 2017, Zhang et al., 2018]:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_d} \max(0, 1 - D(x)) + \mathbb{E}_{z\sim p_z} \max(0, 1 + D(G(z))) \tag{C.53}$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z} D(G(z)).,\tag{C.54}$$

where consistent with the notation above, p_d and p_z denote the data and the latent distributions.

For the SE–A baseline we obtained best performances when $\eta_G = 1\times10^{-4}$ and $\eta_D = 4\times10^{-4}$, for G and D, respectively. Similarly as noted for **MNIST**, using SVRE allows for using larger order of the step size on the rest of the datasets, whereas SE–A with increased step size ($\eta_G = 1\times10^{-3}$ and $\eta_D = 4\times10^{-3}$ failed to converge. In Table 4.3, $\eta_G = 1\times10^{-3}$, $\eta_D = 4\times10^{-3}$, and $\eta_G = 5\times10^{-3}$, $\eta_D = 8\times10^{-3}$, $\beta_1 = 0.3$ for SVRE and SVRE–VRAd, respectively. We did not use momentum for the vanilla SVRE experiments.

Self–Attention Block (d **– input depth**)		
Input: $t \in \mathbb{R}^{d \times H \times W}$		
i: conv. (ker: 1×1, $d \rightarrow \lfloor d/8 \rfloor$)	*ii:* conv. (ker: 1×1, $d \rightarrow \lfloor d/8 \rfloor$)	*iii:* conv. (ker: 1×1, $d \rightarrow d$)
iv: softmax(*(i)* ⊗ *(ii)*)		
Output: $\gamma\big((iv) \otimes (iii)\big) + t$		

Table C.2 – Layers of the self–attention block used in the SAGAN architectures (see Tables C.3 and C.4), where ⊗ denotes matrix multiplication and γ is a scale parameter initialized with 0. The columns emphasize that the execution is in parallel, more precisely, that the block input t is input to the convolutional layers *(i)*–*(iii)*. The shown row ordering corresponds to consecutive layers' order, *e.g.* softmax is done on the product of the outputs of the *(i)* and *(ii)* convolutional layers. The 1×1 convolutional layers have stride of 1. For further details see [Zhang et al., 2018].

Deeper ResNet architectures

We experimented with ResNet [He et al., 2015] architectures on **CIFAR-10** and **SVHN**, using the architectures listed in Table C.6, that replicate the setup described in [Miyato et al., 2018] on **CIFAR-10**. For experiments with ResNet, we used the hinge version of the adversarial non-saturating loss, Eq. C.53 and C.54. For this architectures, we refer the reader to § C.6.3 for details on the hyperparameters, where we list the hyperparameters along with the obtained results.

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
transposed conv. (ker: 4×4, 128 → 256; stride: 1) Spectral Normalization Batch Normalization ReLU
transposed conv. (ker: 4×4, 256 → 128, stride: 2, pad: 1) Spectral Normalization Batch Normalization ReLU
Self–Attention Block (128)
transposed conv. (ker: 4×4, 128 → 64, stride: 2, pad: 1) Spectral Normalization Batch Normalization ReLU
Self–Attention Block (64)
transposed conv. (ker: 4×4, 64 → 3, stride: 2, pad: 1) $Tanh(\cdot)$

(a) Generator architecture

Discriminator
Input: $x \in \mathbb{R}^{3\times 32\times 32}$
conv. (ker: 4×4, 3 → 64; stride: 2; pad: 1) Spectral Normalization LeakyReLU (negative slope: 0.1)
conv. (ker: 4×4, 64 → 128; stride: 2; pad: 1) Spectral Normalization LeakyReLU (negative slope: 0.1)
conv. (ker: 4×4, 128 → 256; stride: 2; pad: 1) Spectral Normalization LeakyReLU (negative slope: 0.1)
Self–Attention Block (256)
conv. (ker: 4×4, 256 → 1; stride: 1)

(b) Discriminator architecture

Table C.3 – *Shallow* SAGAN architectures for experiments on **SVHN** and **CIFAR-10**, for the Generator (Tab. a) and the Discriminator (Tab. b). The self-attention block is described in Table C.2. We use the default PyTorch hyperparameters for the Batch Normalization layer.

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
transposed conv. (ker: 4×4, 128 → 512; stride: 1)
Spectral Normalization
Batch Normalization
ReLU
transposed conv. (ker: 4×4, 512 → 256, stride: 2, pad: 1)
Spectral Normalization
Batch Normalization
ReLU
transposed conv. (ker: 4×4, 256 → 128, stride: 2, pad: 1)
Spectral Normalization
Batch Normalization
ReLU
Self–Attention Block (128)
transposed conv. (ker: 4×4, 128 → 64, stride: 2, pad: 1)
Spectral Normalization
Batch Normalization
ReLU
Self–Attention Block (64)
transposed conv. (ker: 4×4, 64 → 3, stride: 2, pad: 1)
$Tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{3\times64\times64}$
conv. (ker: 4×4, 3 → 64; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
conv. (ker: 4×4, 64 → 128; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
conv. (ker: 4×4, 128 → 256; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
Self–Attention Block (256)
conv. (ker: 4×4, 256 → 512; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
Self–Attention Block (512)
conv. (ker: 4×4, 512 → 1; stride: 1)

Table C.4 – *Shallow* SAGAN architectures for experiments on **ImageNet**, for the Generator (left) and the Discriminator (right). The self–attention block is described in Table C.2. Relative to the architectures used for **SVHN** and **CIFAR-10** (see Table C.3), the generator has one additional “common” block (conv.–norm.–ReLU), whereas the discriminator has additional “common” block as well as self–attention block (both of more parameters).

G–ResBlock
Bypass:
Upsample(×2)
Feedforward:
Batch Normalization
ReLU
Upsample(×2)
conv. (ker: 3×3, 256 → 256; stride: 1; pad: 1)
Batch Normalization
ReLU
conv. (ker: 3×3, 256 → 256; stride: 1; pad: 1)

D–ResBlock (ℓ–th block)
Bypass:
[AvgPool (ker:2×2)], if $\ell = 1$
conv. (ker: 1×1, $3_{\ell=1}/128_{\ell\neq1}$ → 128; stride: 1)
Spectral Normalization
[AvgPool (ker:2×2, stride:2)], if $\ell \neq 1$
Feedforward:
[ReLU], if $\ell \neq 1$
conv. (ker: 3×3, $3_{\ell=1}/128_{\ell\neq1}$ → 128; stride: 1; pad: 1)
Spectral Normalization
ReLU
conv. (ker: 3×3, 128 → 128; stride: 1; pad: 1)
Spectral Normalization
AvgPool (ker:2×2)

Table C.5 – ResNet blocks used for the ResNet architectures (see Table C.6), for the Generator (left) and the Discriminator (right). Each ResNet block contains skip connection (bypass), and a sequence of convolutional layers, normalization, and the ReLU non–linearity. The skip connection of the ResNet blocks for the Generator (left) upsamples the input using a factor of 2 (we use the default PyTorch upsampling algorithm–nearest neighbor), whose output is then added to the one obtained from the ResNet block listed above. For clarity we list the layers sequentially, however, note that the bypass layers operate in parallel with the layers denoted as "feedforward" [He et al., 2015]. The ResNet block for the Discriminator (right) differs if it is the first block in the network (following the input to the Discriminator), $\ell = 1$, or a subsequent one, $\ell > 1$, so as to avoid performing the ReLU non–linearity immediate on the input.

Generator	**Discriminator**
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$	*Input:* $x \in \mathbb{R}^{3\times32\times32}$
Linear(128 → 4096)	D–ResBlock
G–ResBlock	D–ResBlock
G–ResBlock	D–ResBlock
G–ResBlock	D–ResBlock
Batch Normalization	ReLU
ReLU	AvgPool (ker:8×8)
conv. (ker: 3×3, 256 → 3; stride: 1; pad:1)	Linear(128 → 1)
$Tanh(\cdot)$	Spectral Normalization

Table C.6 – *Deep* ResNet architectures used for experiments on **SVHN** and **CIFAR-10**, where G–ResBlock and D–ResBlock for the Generator (left) and the Discriminator (right), respectively, are described in Table C.5. The models' parameters are initialized using the Xavier initialization [Glorot and Bengio, 2010].

	IS			FID		
	SE–A	SVRE	SVRE–VRAd	SE–A	SVRE	SVRE–VRAd
MNIST	8.62	8.58	8.56	0.17	0.15	0.18
CIFAR-10	6.61	6.50	**6.67**	37.20	39.20	38.88
SVHN	2.83	3.01	**3.04**	39.95	24.01	**19.40**
ImageNet	7.22	**8.08**	7.50	89.40	**75.60**	81.24

Table C.7 – Best obtained IS and FID scores for the different optimization methods, using *shallow* architectures, for a fixed number of iterations (see § C.5). The architectures for each dataset are described in: **MNIST**–Table C.1, **SVHN** and **CIFAR-10**–Table C.3, and **ImageNet**–Table C.4. The standard deviation of the Inception scores is around 0.1 and is omitted. Although the IS metric gives relatively close values on **SVHN** due to the dataset properties (see § C.5.1), we include it for completeness.

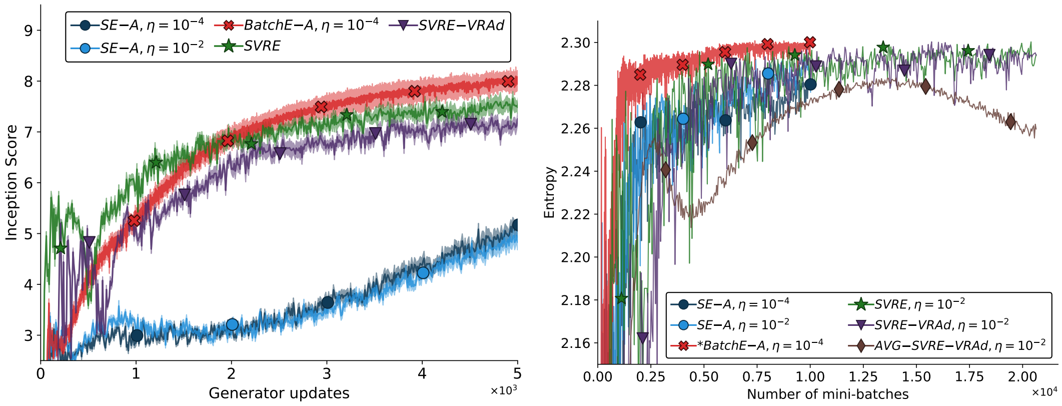
C.6 Additional Experiments

C.6.1 Results on MNIST

The results in Table 4.3 on **MNIST** are obtained using 5 runs with different seeds, and the shown performances are the averaged values. Each experiment was run for $100K$ iterations. The corresponding scores with the standard deviations are as follows: (i) IS: 8.62±.02, 8.58±.08, 8.56±.11; (ii) FID: 0.17±.03, 0.15±.01, 0.18±.02; for SE–A, SVRE, and SVRE–VRAd, respectively. On this dataset, we obtain similar final performances if run for many iterations, however SVRE converges faster (see Fig. 4.2). Fig. C.1 illustrates additional metrics of the experiments shown in Fig. 4.2.

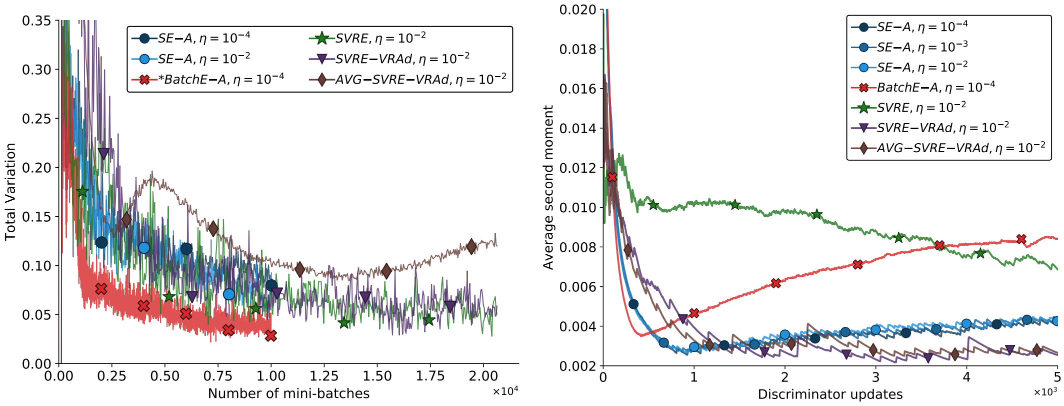
C.6.2 Results with shallow architectures

Fig. C.2 depicts the results on **ImageNet** using the *shallow* architectures described in Table C.4, § C.5.2. Table C.7 summarizes the results obtained on **SVHN**, **CIFAR-10** and **ImageNet** with these architectures. Fig. C.3 depicts the SME metric (see § C.5.1) for the the SE–A baseline and SVRE shown in Fig. 4.2c, on **SVHN**.



(a) IS (higher is better)

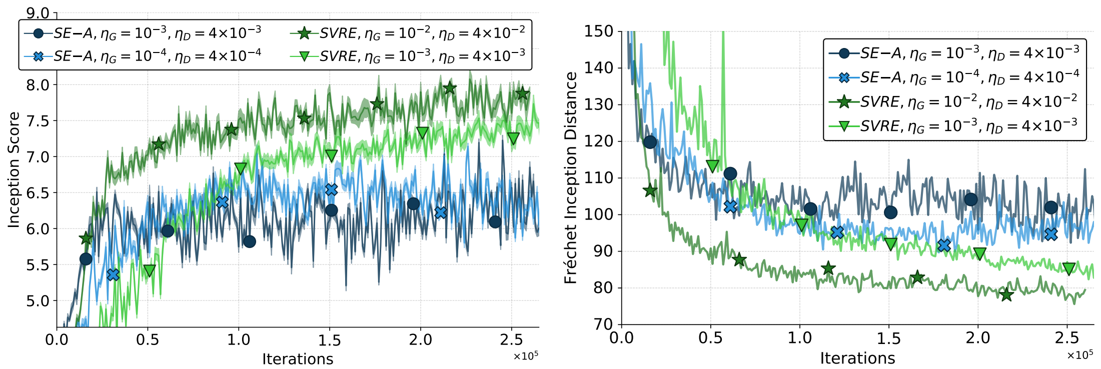
(b) Entropy (higher is better)



(c) Total variation (lower is better)

(d) Discriminator

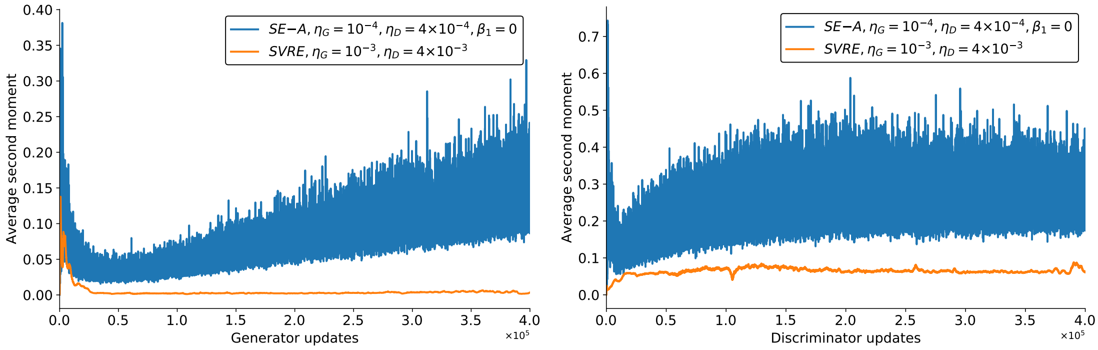
Figure C.1 – Stochastic, full-batch and variance reduced versions of the extragradient method ran on **MNIST**, see § 4.5.1. **BatchE–A* emphasizes that this method is **not** scaled with the number of passes (x-axis). The input space is 1×28×28, see § C.5.2 for details on the implementation.



(a) IS (higher is better)

(b) FID (lower is better)

Figure C.2 – Comparison between *SVRE* and the *SE–A* baseline on **Imagenet**, using the *shallow* architectures described in Table C.4. See § C.5.1 for details on the used IS and FID metrics.



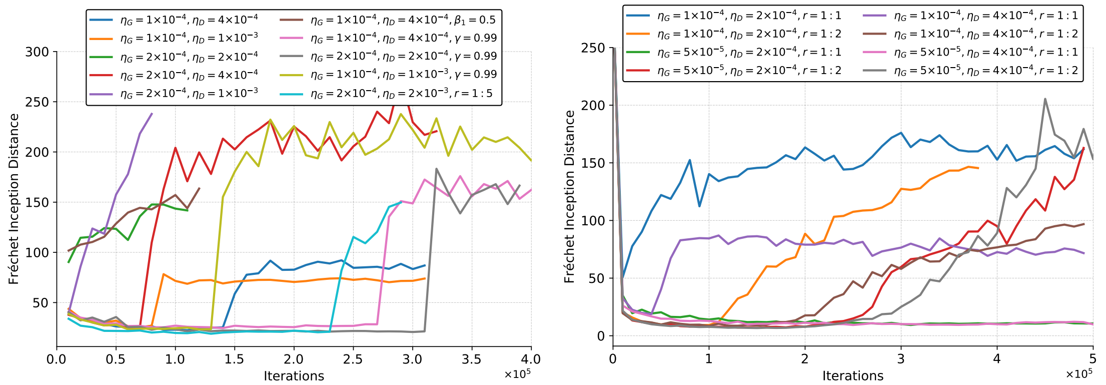
(a) Generator (b) Discriminator

Figure C.3 – Average second moment estimate (see § C.5.1) on **SVHN** for the Generator (left) and the Discriminator (right), using the *shallow* architectures described in Table C.3. The corresponding FID scores for these experiments are shown in Fig. 4.2c.

C.6.3 Results with deeper architectures

We observe that GAN training is more challenging when using *deeper* architectures and some empirical observations differ in the two settings. For example, our stochastic baseline is drastically more unstable and often does not start to converge, whereas SVRE is notably *stable*, but slower compared to when using shallower architectures. In this section, all our discussions focus on *deep* architectures (see § C.5.2).

Stability: convergence of the GAN training. For our stochastic baselines, *irrespective whether we use the extragradient or gradient method*, we observe that the convergence is notably more *unstable* (see Fig. C.4) when using the *deep* architectures described in § C.5.2. More precisely, either the training fails to converge or it diverges at later iterations. When updating G and D equal number of times *i.e.* using 1 : 1 update ratio, using SE–A on **CIFAR-10** we obtained best FID score of 24.91 using $\eta_G = 2 \times 10^{-4}$, $\eta_D = 4 \times 10^{-4}$, $\beta_1 = 0$, while experimenting with several combinations of η_G, η_D, β_1. Using exponential learning rate decay with a multiplicative factor of 0.99, improved the best FID score to 20.70, obtained for the experiment with $\eta_G = 2 \times 10^{-4}$, $\eta_D = 2 \times 10^{-4}$, $\beta_1 = 0$. Finally, using 1 : 5 update ratio, with $\eta_G = 2 \times 10^{-4}$, $\eta_D = 2 \times 10^{-4}$, $\beta_1 = 0$ provided best FID of 18.65 for the baseline. Figures C.4a and C.4b depict the hyper-parameter sensitivity of SE–A and SG–A, respectively. The latter denotes the alternating GAN training with Adam, that is most commonly used for GAN training.



(a) SE–A, **CIFAR-10**

(b) SG–A, **SVHN**

Figure C.4 – FID scores (lower is better) with different hyperparameters for the SE–A baseline on **CIFAR10** (left) and the SG–A baseline on **SVHN** (right), using the *deep* architectures described in Table C.6, § C.5.2. SG–A denotes the standard stochastic *alternating* GAN training, with the Adam optimization method. Where omitted, $\beta_1 = 0$, see (C.46) where this hyperparameter is defined. With r we denote the update ratio of generator versus discriminator: in particular 1 : 5 denotes that D is updated 5 times for each update of G. γ denotes a multiplicative factor of exponential learning rate decay scheduling. In Fig. C.4b, $\gamma = 0.99$ for all the experiments. We observed in all our experiments that training diverged in later iterations for the stochastic baseline, when using *deep* architectures.

We observe that SVRE is more stable in terms of hyperparameter selection, as it always starts

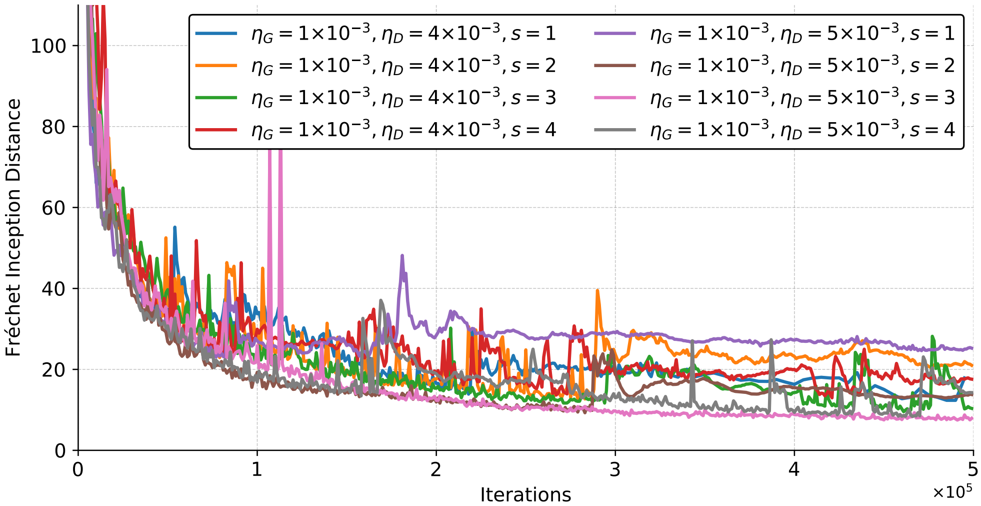


Figure C.5 – Obtained FID (lower is better) scores for SVRE, using the *deep* architectures (see § C.5.2) on **SVHN**. With s we denote the fixed random seed. The update ratio for all the experiments is 1 : 1. We illustrate our results on the same plot (besides the reduced clarity) so as to summarize our observation that, contrary to the SE–A baseline for these architectures, SVRE *always converges*, and does not diverge.

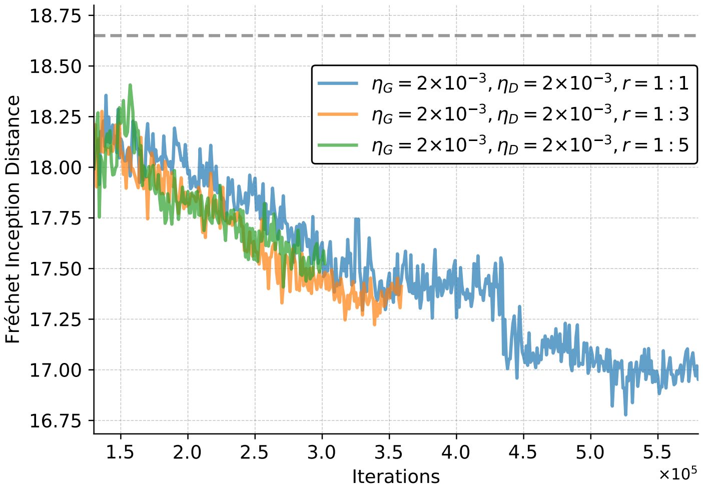
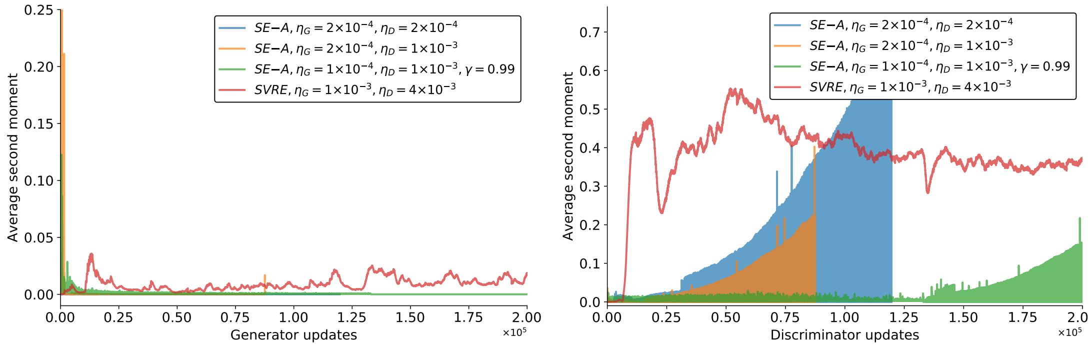


Figure C.6 – Obtained FID (lower is better) scores for WS–SVRE, using the *deep* architectures (see § C.5.2) on **CIFAR-10**, where the seed is fixed to 1 for all the experiments. With r we denote the update ratio of generator versus discriminator: in particular 1 : 5 denotes that D is updated 5 times for each update of G. We start from the best obtained FID score for the stochastic baseline, i.e. FID of 18.65 (see Table 4.3)–shown with dashed line, and we continue to train with SVRE.

to converge and *does not diverge* at later iterations. Relative to experiments with shallower architectures, we observe that with deeper architectures SVRE takes longer to converge than its baseline for this architecture. With constant step size of $\eta_G = 1 \times 10^{-3}$, $\eta_D = 4 \times 10^{-3}$ we obtain FID score of 23.56 on **CIFAR-10**. Note that this result outperforms the baseline when using no additional tricks (which themselves require additional hyperparameter tuning). Fig. C.5 depicts the FID scores obtained when training with SVRE on the **SVHN** dataset, for two different hyperparameter settings, using four different seeds for each. From this set of experiments, we observe that contrary to the baseline that either did not converge or diverged in all our experiments, SVRE always converges. However, we observe different performances for different seeds. This suggests that more exhaustive empirical hyperparameter search that aims to find an empirical setup that works *best* for SVRE or further combining SVRE with adaptive step size techniques are both promising research directions (see our discussion below). Fig. C.6 depicts our WS–SVRE experiment, where we start from a stored checkpoint for which we obtained best FID score for the SE–A baseline, and we continue the training with SVRE. It is interesting that besides that the baseline diverged after the stored checkpoint, SVRE further reduced the FID score. Moreover, we observe that using different update ratios does not impact much the performance, what on the other hand was necessary to make the baseline algorithm converge.



(a) Generator

(b) Discriminator

Figure C.7 – Average second moment estimate (SME, see § C.5.1) on **CIFAR-10** for the Generator (left) and the Discriminator (right), using the *deep* architectures described in Table C.6. The obtained FID scores for these experiments are shown in Fig. C.4a, where we omit some of the experiments for clarity. All of the baseline SE–A experiments diverge at some point, what correlates with the iterations at which large oscillations of SME appear for the Discriminator. Note that the SE–A experiments were stopped after the algorithm diverges, hence the plotted SME is up to a particular iteration for two of the experiments (shown in blue and orange). The SE–A experiment with $\gamma = 0.99$ diverged at later iteration relative to the experiments without learning rate decay, and has lower SME.

Second moment estimate (SME). Fig. C.7 depicts the second moment estimate (see § C.5.1) for the experiments with *deep* architectures. We observe that: (i) the estimated SME quantity is more bounded and changes more smoothly for SVRE (as we do not observe large oscillations of it as it is the case for SE–A); as well as that (ii) divergence of the SE–A baseline *correlates* with large oscillations of SME, in this case, observed for the Discriminator. Regarding the latter, there exist larger in magnitude oscillations of SME (note that the exponential moving average hyperparameter for computing SME is $\gamma = 0.9$, see § C.5.1).

D Appendix for Chapter 6

Foremost, we elaborate in more detail the annotation procedure in Section D.1. We then list details regarding the provided annotations in Section D.2. Details on our implementation of the camera calibration are given in Section D.3. Section D.4 provides further details of the statistics summarized in Section 3.4. In Section D.5 we discuss recommended training and testing splits of the WILDTRACK dataset. Finally, we provide additional tracking results in Section D.6.

D.1 Annotation process

Annotation tool. As an area of interest we consider a $12 \times 36m$ ground plane of the $3D$ space lying in the intersection of the fields of view of the seven cameras. We discretize this ground surface as a grid of 480×1440 points, what corresponds to an offset of $2.5cm$ in both directions. Given such a regular high-density grid of, at each location we construct a cylinder volume whose height and width correspond to the humans' average height and width. Each such cylinder projects into the separate 2D views as a rectangle. The position of these rectangles in all of the views is then calculated in pixel coordinates using the camera calibration. Finally, we use these pre-calculated projections to integrate them into our annotation tool.

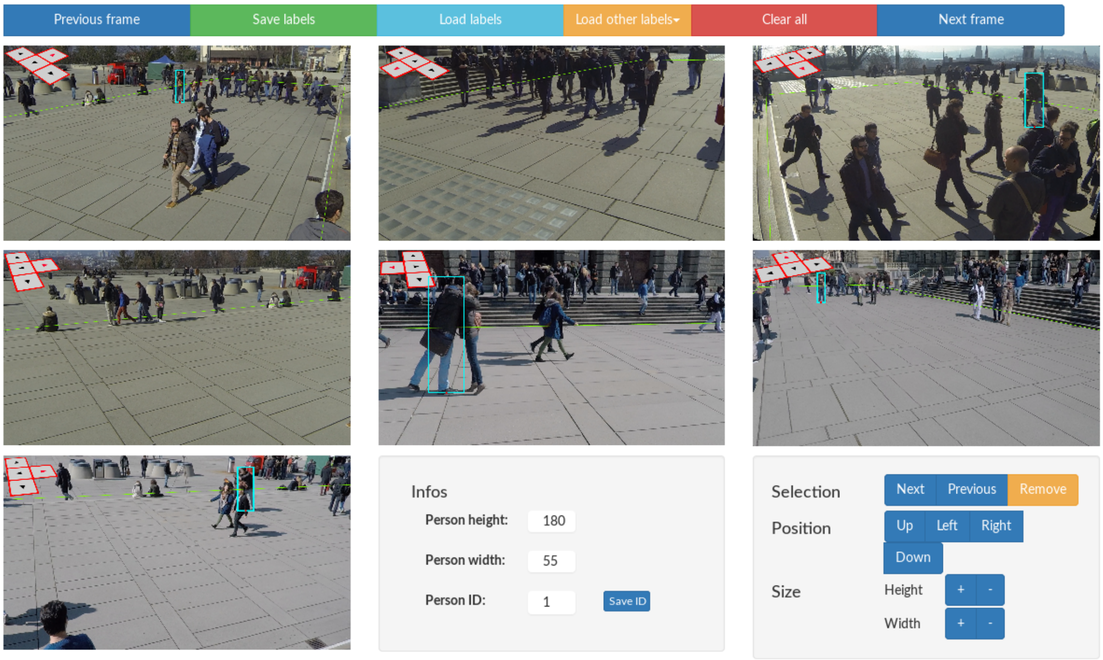
The labelling tool is a Python-based web application. It is built with a very responsive design, and its graphical user interface (GUI) is illustrated in Fig. D.1. Our annotation tool is hosted on a website[1], which was created and managed using Django. The source-code is also available for download[2].

For the selected frame to be annotated, the tool displays the seven corresponding images at the same time (see Fig. D.1). In order to provide a multi-view annotation, the user of the tool first has to mark the placement of the bounding boxes. This is achieved by a *single click*, whose location should be at the feet of the person to be annotated, in either of the views where it is

---

[1] https://pedestriantag.epfl.ch/

[2] https://github.com/cvlab-epfl/multicam-gt



Infos

Person height: 180

Person width: 55

Person ID: 1

Figure D.1 – Graphical User Interface (GUI) of our multi-view labeller.

visible. Instantaneously, the boxes automatically appear in the views in which the person is visible. To complete the multi-view annotation, the user shall next adjust the position of the bounding boxes.

For this purpose, the keyboard arrows shall be used. More precisely, the *left*, *right*, *up* and *down* keys should be used in order to shift the $3D$ *imaginary* cylinder on the ground plane. To help annotators, the correspondence “key-direction” for each of the views is also depicted in the tool and optionally visible while annotating. In addition, a *zooming feature* can be used, that once a multi-view annotation is selected, allows for zooming-in the corresponding bounding boxes. This was implemented in order to make it easier for the annotators to obtain more precise locations of the annotations. The arrow key presses that translate into a movement of the $3D$ cylinder, are instantly visible in all of the views that capture the person currently being annotated. After getting used to the annotation process, annotators become more and more precise on the first step: placing the bounding boxes, which significantly reduces the time required to annotate as less adjustments are required.

Once the frame has been fully labelled and the user has moved to the next frame, optionally (s)he is able to reload the annotations from the previous frame, traverse each of the annotations, and refine their positions. Additional features such as keyboard short-cuts are also supported for these utilities.

Finally, a more elaborate version of these instructions is provided in the annotation tool, accompanied with numerous illustrations.

Annotating on Mechanical Turk. We used Amazon Mechanical Turk [Buhrmester et al., 2011] to obtain our annotations. Due to the risk of the annotators prioritizing profit over quality of the annotations, we were highly involved in the process.

In our experience of annotating frames of our dataset, pre-loading annotations from the previous frame, traversing these, and adjusting each, often proves less time-consuming then starting to annotate each multi-view frame from scratch. This motivated providing the feature of *pre-loading annotations* explained above. Hence, to help accelerate the annotation process the recruited annotators were assigned frames in batches of size 10. To ensure that this feature is not negatively utilised by the annotators, we also store flags indicating if these “imported” annotations have been adjusted or not.

As explained, annotators were found via Mechanical Turk. However, since the dataset is quite challenging, annotating locations in 3D for crowded scenes may require substantial attention and dedication. Despite all our efforts to make the tool easy to use, it turned out that most MT workers were reluctant to provide this level of effort and they were almost never achieving the required quality. We therefore had to select few workers to whom we personally explained the level of detail needed. They were then able to annotate with higher accuracy.

On average, annotating one frame takes ~10 minutes for a trained annotator, and approximately half of that when importing the annotations from the previous frame.

D.2 Annotations

D.2.1 File formats

The annotations are provided in a separate file per each multi-view frame. Each annotation file is provided in the JavaScript Object Notation (JSON) open-standard file format. This format is human readable and programming language independent. Many programming languages integrate libraries that offer support for working with these files, including Python.

Each multi-view annotation contains the following information:

- **Person ID**: A unique identifier of a person appearing in the sequence.
- **3D location**: (X, Y) location of the target in meters on the ground plane with respect to the origin.
- **pixel coordinates in each of the views**: For each of the seven cameras, the detection location in pixel coordinates for that view are given: $\{(x^c_{min}, y^c_{min}, x^c_{max}, y^c_{max})\}$, $c = 1, \ldots, 7$.

D.2.2 Memory size

Images. We refer as a *frame* a set of 7 images, synchronized with the same time stamp. The extracted and pre-processed frames with removed distortions contain 36000 × 7 images, while each image is of size ~2.9 MB. This corresponds to 10 frames per second for 1h and 7 cameras. Currently there are 400 annotated frames, at 2fps (see Section 3.4).

Videos. Each of the 7 videos is approximately 1:50h long, and of size ~25GB.

D.3 Camera calibration

Intrinsic. The intrinsic calibration was obtained for each camera separately. For this purpose we used the *OpenCV* function *calibrateCamera* which provides also the distortion coefficients. Precisely, we used 3 radial distortion coefficients. In particular, we used the asymmetric circle grid provided by *OpenCV* with sizes of 4 × 11, and 20 frames to obtain each camera's intrinsic matrix. To obtain higher accuracy, we made sure that the target (the grid of circles), is captured in as many parts of the field of view of the camera as possible.

Extrinsic. In our implementation, for each of the seven views we used 23, 26, 15, 19, 21, 28 and 19 pairs of points, respectively. We used the *OpenCV*'s module *solvePnP* [Bradski, 2000], which given the intrinsics provides the rotation and the translation vector. The 3D measurements and the annotated corresponding points will also be made available, so as camera calibration methods could make use of these.

Bundle adjustment. In our implementation, we used the open source C++ library *Ceres* [Agarwal et al.], which offers extensive support for bundle adjustment problems. We used linear optimisation which in Ceres is referred to as *Iterative Schur*.

D.4 Additional statistics

Fig. D.2 depicts the number frames in which a person appears. In particular, we consider a frame rate of 2 fps, a total number of frames of 400, and 313 different identities. On average, each person appears in 30.41(47.87) frames, and the mode is 22 frames.

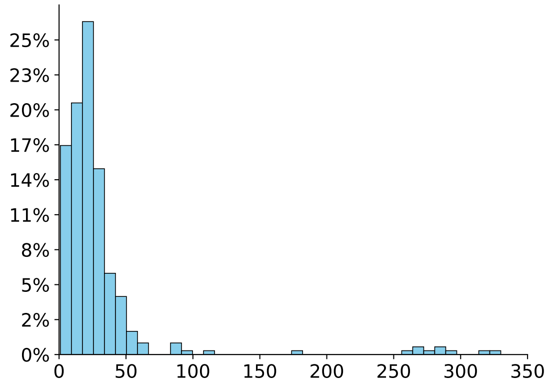


Figure D.2 – Histogram of the number of frames in which one person appears: the normalized number of different identites (y-axis) that appear within a range of number of frames (x-axis).

D.5 Recommended splits of the WILDTRACK dataset

We regard two use-cases of the WILDTRACK dataset, and we discuss recommended partitions for each. Please consider visiting the website for downloading the dataset[3], for up to date details.

Scenario A: Supervised methods. We recommend that the last 10% of the annotated frames at 2 fps are used for testing. This amounts to a total of 40 frames at 2 fps. For training one shall use the remaining portion of the dataset, with optional sampling frame rate.

Scenario B: Unsupervised methods. In this case, we recommend that the entire annotated portion at a fixed frame rate of 2 fps is used for benchmarking unsupervised methods. The remaining portion for which annotations are not provided can be used for training, using an optional sampling rate.

D.6 Additional tracking benchmarks

Table D.1 shows additional tracking results, where we use the same notation for the methods as in Chapter 6.

---

[3] https://cvlab.epfl.ch/data/wildtrack

Table D.1 – Additional tracking results on the WILDTRACK dataset.

Method	IDF1	IDP	IDR	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP
ResNet-DeepMCD+KSP	62.5	84.9	49.5	11	11	19	154	2081	35	30	50.9	75.1
ResNet-DeepMCD+KSP+ptrack	64.2	93.1	49.0	10	9	22	49	2239	5	5	50.4	75.9
ResNet-View 1+KSP	28.5	18.7	60.7	34	4	0	10050	257	162	58	-140.9	59.0
ResNet-View 1+KSP+ptrack	30.2	20.1	60.6	30	8	0	9173	410	131	51	-123.5	58.0
ResNet-View 2+KSP	29.1	19.4	58.8	28	6	1	8428	265	172	47	-121.3	50.7
ResNet-View 2+KSP+ptrack	31.4	21.2	60.6	28	6	1	7698	249	128	31	-101.6	50.2
ResNet-View 3+KSP	25.8	17.0	53.7	35	4	1	9874	286	177	52	-133.5	51.2
ResNet-View 3+KSP+ptrack	27.2	18.1	54.2	33	5	2	9208	402	150	46	-120.5	49.1
ResNet-View 4+KSP	20.5	12.6	54.4	14	5	1	4362	137	42	16	-255.3	60.5
ResNet-View 4+KSP+ptrack	22.1	13.9	54.4	13	2	5	3904	180	32	11	-222.1	60.3
ResNet-View 5+KSP	39.7	32.6	50.9	20	13	3	2560	598	117	79	5.8	54.2
ResNet-View 5+KSP+ptrack	41.7	35.0	51.7	18	12	6	2334	672	94	54	10.9	55.2
ResNet-View 6+KSP	26.6	17.5	55.4	34	4	1	10200	375	172	77	-136.1	52.2
ResNet-View 6+KSP+ptrack	29.4	19.9	56.4	30	8	1	8860	498	127	51	-108.4	52.8
ResNet-View 7+KSP	38.6	27.1	67.0	22	3	0	4488	171	72	28	-61.1	65.1
ResNet-View 7+KSP+ptrack	41.7	30.3	66.8	19	3	3	3791	253	49	21	-39.4	64.9

Bibliography

[1] EPFL-RLC data-set. http://cvlab.epfl.ch/data/rlc.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[3] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org.

[4] A. Alahi, L. Jacques, Y. Boursier, and P. Vandergheynst. Sparsity driven people localization with a heterogeneous network of cameras. *Journal of Mathematical Imaging and Vision*, 41(1-2):39–58, 2011.

[5] X. Alameda-Pineda, J. Staiano, R. Subramanian, L. Batrinca, E. Ricci, B. Lepri, O. Lanz, and N. Sebe. Salsa: A novel dataset for multimodal group behavior analysis. *TPAMI*, 2016.

[6] Z. Allen-Zhu and E. Hazan. Variance reduction for faster non-convex optimization. In *ICML*, 2016.

[7] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, 2008.

[8] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

[9] W. Azizian, I. Mitliagkas, S. Lacoste-Julien, and G. Gidel. A tight and unified analysis of extragradient for a whole spectrum of differentiable games. *arXiv:1906.05945*, 2019.

[10] P. Baque, F. Fleuret, and P. Fua. Deep Occlusion Reasoning for Multi-Camera Multi-People tracking. In *ICCV*, 2017.

[11] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV*, 2015.

[12] J. Berclaz, F. Fleuret, and P. Fua. Multiple object tracking using flow linear programming. In *Winter-PETS*, 2009.

[13] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, (9), 2011.

[14] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, 2008.

[15] A. Botev, I. Higgins, A. Jaegle, S. Racaniere, D. J. Rezende, and P. Toth. Hamiltonian generative networks. In *ICLR*, 2020.

[16] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.

[17] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[18] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

[19] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[20] M. Buhrmester, T. Kwang, and S. D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 2011.

[21] Z. Cai, M. J. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. *arXiv:1507.05348*, 2015.

[22] E. J. Candès, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, Dec 2008.

[23] T. Chavdarova and F. Fleuret. Deep multi-camera people detection. In *ICMLA*, pages 848–853, 2017.

[24] T. Chavdarova and F. Fleuret. SGAN: An alternative training of generative adversarial networks. In *CVPR*, 2018.

[25] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret. WILDTRACK: A multi-camera HD dataset for dense unscripted pedestrian detection. In *CVPR*, pages 5030–5039, 2018.

[26] T. Chavdarova, S. Stich, M. Jaggi, and F. Fleuret. Stochastic variance reduced gradient optimization of generative adversarial networks. 2018.

[27] T. Chavdarova, G. Gidel, F. Fleuret, and S. Lacoste-Julien. Reducing noise in gan training with variance reduced extragradient. In *NeurIPS*, 2019.

[28] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[29] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[30] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[31] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[32] C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng. Training GANs with optimism. In *ICLR*, 2018.

[33] Y. N. Dauphin and S. Schoenholz. Metainit: Initializing learning by learning to initialize. In *NeurIPS*. 2019.

[34] D. Davis. Smart: The stochastic monotone aggregated root-finding algorithm. *arXiv:1601.00698*, 2016.

[35] C. De Vleeschouwer, F. Chen, D. Delannay, C. Parisot, C. Chaudy, E. Martrou, A. Cavallaro, et al. Distributed video acquisition and annotation for sport-event summarization. In *NEM summit 2008:: Towards Future Media Internet*, 2008.

[36] A. Defazio and L. Bottou. On the ineffectiveness of variance reduced optimization for deep learning. *arXiv:1812.04529*, 2018.

[37] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, 2014.

[38] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *CVPR*, 2009.

[39] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *TPAMI*, 2012.

[40] X. Du, M. El-Khamy, J. Lee, and L. S. Davis. Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection. *WACV*, 2017.

[41] I. P. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. In *ICLR*, 2017.

[42] M. Enzweiler and D. Gavrila. Monocular pedestrian detection: Survey and experiments. *TPAMI*, 2009.

[43] A. Ess, B. Leibe, and L. V. Gool. Depth and appearance for mobile scene analysis. In *ICCV*, 2007.

[44] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. A mobile vision system for robust multi-person tracking. In *CVPR*, 2008.

[45] F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems Vol I*. Springer Series in Operations Research and Financial Engineering, Finite-Dimensional Variational Inequalities and Complementarity Problems. Springer-Verlag, 2003.

[46] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *Winter-PETS*, 2009.

[47] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-Camera People Tracking with a Probabilistic Occupancy Map. *TPAMI*, pages 267–282, 2008.

[48] B. J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge, MA, USA, 1998. ISBN 026206202X.

[49] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[50] W. Ge and R. T. Collins. Crowd detection with a multiview sampler. 2010.

[51] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[52] A. Ghosh, V. Kulharia, and V. P. Namboodiri. Message passing multi-agent gans. *arXiv:1612.01294*, 2016.

[53] A. Ghosh, V. Kulharia, V. P. Namboodiri, P. H. S. Torr, and P. K. Dokania. Multi-agent diverse generative adversarial networks. *arXiv:1704.02906*, 2017.

[54] G. Gidel, H. Berard, P. Vincent, and S. Lacoste-Julien. A variational inequality perspective on generative adversarial nets. In *ICLR*, 2019.

[55] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524*, 2013.

[56] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[57] M. Golbabaee, A. Alahi, and P. Vandergheynst. Scoop: A real-time sparsity driven people localization algorithm. *Journal of Mathematical Imaging and Vision*, 48(1):160–175, 2014. ISSN 1573-7683. doi: 10.1007/s10851-012-0405-4. URL http://dx.doi.org/10.1007/s10851-012-0405-4.

[58] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv:1701.00160*, 2016.

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[60] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618.

[61] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *NeurIPS*. 2019.

[62] A. Grover and S. Ermon. Boosted generative models. *arXiv:1702.08484*, 2017.

[63] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein GANs. In *NIPS*, 2017.

[64] P. T. Harker and J.-S. Pang. Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. *Mathematical programming*, 1990.

[65] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[66] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[67] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[68] Q. Hoang, T. Dinh Nguyen, T. Le, and D. Phung. Multi-Generator Generative Adversarial Nets. *arXiv:1708.02556*, 2017.

[69] T. Hofmann, A. Lucchi, S. Lacoste-Julien, and B. McWilliams. Variance reduced stochastic gradient descent with neighbors. In *NIPS*, 2015.

[70] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. ISSN 0027-8424. doi: 10.1073/pnas.79.8.2554.

[71] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, Mar. 1991. ISSN 0893-6080. doi: 10.1016/0893-6080(91)90009-T.

[72] J. Hosang, M. O., R. Benenson, and B. Schiele. Taking a deeper look at pedestrians. In *CVPR*, 2015.

[73] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[74] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[75] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[76] A. Iusem, A. Jofré, R. I. Oliveira, and P. Thompson. Extragradient method with variance reduction for stochastic variational inequalities. *SIAM Journal on Optimization*, 2017.

[77] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013.

[78] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv:1602.02410*, 2016.

[79] A. Juditsky, A. Nemirovski, and C. Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 2011.

[80] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, M. Boonstra, V. Korzhova, and J. Zhang. Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol. *PAMI*, 2009.

[81] C. Keller, D. Llorca, and D. Gavrila. Dense stereo-based roi generation for pedestrian detection. In *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*. 2009. ISBN 978-3-642-03797-9.

[82] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[83] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[84] T. Klinger, F. Rottensteiner, and C. Heipke. Probabilistic multi-person localisation and tracking in image sequences. *ISPRS Journal of Photogrammetry and Remote Sensing*, 127:73–88, 2017.

[85] N. Kodali, J. D. Abernethy, J. Hays, and Z. Kira. How to train your DRAGAN. *arXiv:1705.07215*, 2017.

[86] G. Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 1976.

[87] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

[88] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[89] T. Kurutach, A. Tamar, G. Yang, S. J. Russell, and P. Abbeel. Learning plannable representations with causal infogan. In *NeurIPS*, 2018.

[90] W.-S. Lai, J.-B. Huang, and M.-H. Yang. Semi-supervised learning for optical flow with generative adversarial networks. In *NIPS*. 2017.

[91] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In *Computer Vision Workshops (ICCV Workshops)*, 2011.

[92] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942*, 2015.

[93] R. Leblond, F. Pederegosa, and S. Lacoste-Julien. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *JMLR*, 19(81):1–68, 2018.

[94] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. 1998. URL http://yann.lecun.com/exdb/mnist/.

[95] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, page 319, 1999. ISBN 3540667229.

[96] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv:1609.04802*, 2016.

[97] J. H. Lim and J. C. Ye. Geometric GAN. *arXiv:1705.02894*, 2017.

[98] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[99] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[100] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs Created Equal? A Large-Scale Study. *arXiv:1711.10337*, 2017.

[101] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

[102] A. Maksai, X. Wang, F. Fleuret, and P. Fua. Non-markovian globally consistent multi-object tracking. In *ICCV*, 2017.

[103] S. Marsland. *Machine Learning: An Algorithmic Perspective.* Chapman & Hall/CRC, 1st edition, 2009. ISBN 1420067184, 9781420067187.

[104] L. Mescheder, S. Nowozin, and A. Geiger. The numerics of GANs. In *NIPS*, 2017.

[105] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017.

[106] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *AAAI*, 2017.

[107] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[108] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. URL http://ufldl.stanford.edu/housenumbers/.

[109] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv:1612.00005*, 2016.

[110] M. A. Nielsen. *Neural Networks and Deep Learning*. 2015.

[111] S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. 2016.

[112] N. Oliver, B. Rosario, and A. Pentland. A Bayesian Computer Vision System for Modeling Human Interactions. 22(8):831–843, 2000.

[113] D. P. Piotr's Computer Vision Matlab Toolbox (PMT). https://github.com/pdollar/toolbox.

[114] B. Palaniappan and F. Bach. Stochastic variance reduction methods for saddle-point problems. In *NIPS*, 2016.

[115] A. Paszke, S. Gross, S. Chintala, and G. Chanan. PyTorch. https://github.com/pytorch/pytorch, 2017.

[116] P. Peng, Y. Tian, Y. Wang, J. Li, and T. Huang. Robust multiple cameras pedestrian detection with multi-view bayesian network. *Pattern Recognition*, 48(5), 2015.

[117] D. Pfau and O. Vinyals. Connecting generative adversarial networks and actor-critic methods. *arXiv:1610.01945*, 2016.

[118] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[119] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv:1710.05941*, 2017.

[120] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for nonconvex optimization. In *ICML*, 2016.

[121] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2015.

[122] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[123] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[124] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV*, 2016.

[125] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

[126] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[127] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 27(3):832–837, 09 1956. doi: 10.1214/aoms/1177728190.

[128] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[129] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[130] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *ICCV*, 2017.

[131] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016.

[132] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *ICML*, 2013.

[133] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2017.

[134] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl. Measuring the effects of data parallelism on neural network training. *arXiv:1811.03600*, 2018.

[135] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* 2018. ISBN 0262039249.

[136] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[137] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv:1512.00567*, 2015.

[138] Y. Tian, P. Luo, X. Wang, and X. Tang. Deep learning strong parts for pedestrian detection. pages 1904–1912, 2015.

[139] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[140] I. O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf. AdaGAN: Boosting generative models. In *NIPS*. 2017.

[141] P. Tseng. On linear convergence of iterative methods for the variational inequality problem. *Journal of Computational and Applied Mathematics*, 1995.

[142] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[143] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv:1606.05328*, 2016.

[144] C. Villani. *Optimal Transport: Old and New*. Springer, 2009 edition, Sept. 2008. ISBN 3540710493.

[145] Y. Wang, L. Zhang, and J. van de Weijer. Ensembles of generative adversarial networks. *arXiv:1612.00991*, 2016.

[146] Wikipedia. Bundle adjustment — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Bundle_adjustment&oldid=770262831. [Online; accessed 13-July-2017].

[147] Wikipedia. Pinhole camera model — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Pinhole_camera_model&oldid=782167557. [Online; accessed 12-July-2017].

[148] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *NIPS*, 2017.

[149] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *CVPR*, 2009.

[150] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

[151] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[152] Y. Xu, X. Liu, Y. Liu, and S. Zhu. Multi-View People Tracking via Hierarchical Trajectory Composition. In *CVPR*, 2016.

[153] Y. Xu, X. Liu, Y. Liu, and S. C. Zhu. Multi-view people tracking via hierarchical trajectory composition. In *CVPR*, pages 4256–4265, 2016. doi: 10.1109/CVPR.2016.461.

[154] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv:1506.03365*, 2015.

[155] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv:1212.5701*, 2012.

[156] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[157] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-Attention Generative Adversarial Networks. *arXiv:1805.08318*, 2018.

[158] L. Zhang, L. Lin, X. Liang, and K. He. Is faster r-cnn doing well for pedestrian detection? In *ECCV*, 2016.

[159] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. How far are we from solving pedestrian detection? In *CVPR*, 2016.

[160] F. Ziliani and A. Cavallaro. Image Analysis for Video Surveillance Based on Spatial Regularization of a Statistical Model-Based Change Detection. 1999.

Tatjana Chavdarova

Curriculum Vitae

Education

2015–2020	**Ph.D. student, Electrical Engineering Doctoral program (EDEE)**, *École Polytechnique Fédérale de Lausanne – EPFL*, Lausanne, Switzerland.
2013–2014	**Master of Sciences in Electrical Engineering and Information Technologies, Dedicated Computer Systems study program**, *Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University*, Skopje, Republic of Macedonia. First Class Honours (*GPA 10.0*)
2009–2013	**Bachelor of Sciences in Electrical Engineering and Information Technologies, Informatics and computer engineering study program**, *Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University*, Skopje, Republic of Macedonia. First Class Honours (*GPA 9.44*)

Experience

Dec.2014–Present	**Research Assistant**, IDIAP RESEACH INSTITUTE, Martigny, Switzerland. Machine Learning group, supervisor: François Fleuret ○ Teaching Assistant: Deep Learning Course (EE-559) at EPFL, for MSc students
Aug.2019–Jan.2020	**Research intern**, DEEPMIND, GOOGLE, London, England. supervisor: Irina Higgins
Nov.2018–July 2019	**Visiting Researcher/Intern**, UNIVERSITÉ DE MONTRÉAL, Montreal, Canada. Montreal Institute for Learning Algorithms – MILA, supervisors: Yoshua Bengio & Simon Lacoste–Julien
Apr.2013–July 2014	**Research and Teaching Assistant**, FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGIES, Skopje, Republic of Macedonia. Computer Science and Computer Engineering Department ○ Teaching assistance & practical sessions ○ Research assistant: Building optimized baseband modules using VHDL, VISION project, CONTRACT N.240555

✉ *tatjana.chavdarova@epfl.ch* • *github.com/chavdarova*

Papers

2020	T. Chavdarova, M. Pagliardini, Martin Jaggi and F. Fleuret. Taming GANs with Lookahead. ArXiv preprint.
2019	T. Chavdarova, G. Gidel, F. Fleuret and S. Lacoste-Julien. Reducing Noise in GAN Training with Variance Reduced Extragradient. In Proceedings of the 33^{rd} Conference on Neural Information Processing Systems (NeurIPS).
2018	T. Chavdarova, S. Stich, M. Jaggi and F. Fleuret. Stochastic Variance Reduced Gradient Optimization of Generative Adversarial Networks. In Proceedings of the Theoretical Foundations and Applications of Deep Generative Models Workshop, 35^{th} International Conference on Machine Learning (ICML–workshop).
2018	T. Chavdarova and F. Fleuret. SGAN: An Alternative Training of Generative Adversarial Networks. In Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR).
2018	T. Chavdarova et al. WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection. In Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR).
2017	T. Chavdarova and F. Fleuret. Deep Multi-Camera People Detection. In Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA).
2014	S. Gievska, K. Koroveshovski, and T. Chavdarova. A Hybrid Approach for Emotion Detection in Support of Affective Interaction. In Proceedings of the IEEE International Conference on Data Mining (ICDM).
2014	T. Chavdarova, A. Tentov, and M. Kalendar. Parallel Architecture Prototype for 60 GHz High Data Rate Wireless Single Carrier Receiver. In Proceedings of the International Conference on Applied Innovations in IT (ICAIIT).
2013	T. Chavdarova et al. Analysis and Implementation of Frequency Domain Equalizer For Single Carrier System in The 60 GHz Band. New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering. Lecture Notes in Electrical Engineering, vol 312. Springer, Cham.
2013	T. Chavdarova, N. Srezoski, D. Cvetkovski and A. Tentov. A Concept and Kinematic Model Design For an Intelligent 4-DOF Robotic Arm. In Proceedings of the XI ETAI International Conference.

PhD Thesis

Title	*Deep Generative Models and Applications*
Supervisor	François Fleuret
Jury members	Pascal Frossard, Martin Jaggi, Ludovic Denoyer, Simon Lacoste–Julien
Date	19 May 2020

Masters Thesis

Title	*Design and Implementation of Parallel Architecture of 60 GHz Multi-Gigabit Data Rate Wireless Single-Carrier System in FPGA*

Awards

2013–2014	Scholarship for Master studies, provided by HI-TECH CORP
2013	Top Student Award – Bachelor studies, awarded by the Dean of the faculty
2012	Hardware & Software – International Student Competition, first prize (as team)
2009-2013	Scholarship for Bachelor studies in informatics, provided by the Ministry for Education

Workshops

2020	WiML (unworkshop format), Program Chair, to be held at ICML 2020

Coding training

May–Aug.2018 Google GodeU program – for selected computer science students from all over EMEA. Code: /tatjanach. The program included:

- Regular code reviews with a Google engineer & peer to peer code reviews;
- Learning industry best practices such as testing and debugging;
- A final algorithmic challenge on which we won the first prize (as a group).

Programming Languages

Expert	PyTorch, JAX, Python
Prior Experience	Tensorflow, Torch, C++, C, Lua, Matlab, VHDL, Scripting Languages

tatjana.chavdarova@epfl.ch • github.com/chavdarova