Improvements and Applications of Machine Learning Algorithms

By Thang Doan

Desautels Faculty of Management

McGill University

Montreal, Québec

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

Abstract

Due to availability of copious amount of data nowadays and its potential value, detecting patterns in data is a crucial task that allows one to use the insights gleaned to make better decisions. In this context, due to its power and scalability for better leveraging data structure and better understanding the processes that generate data, Machine Learning (ML) algorithms have become popular among academics and practitioners. For example, Generative Adversarial Networks (GANs) have shown impressive results in learning distributions and thereby generating realistic customers for new products [52] or real human faces [46]. Another popular ML branch is Reinforcement Learning (RL), which is not only successful in dealing with complexities of AlphaGo [70] game, but also provides a cogent framework for solving decision making problems. In this dissertation, we present three chapters that investigate various theoretical and practical issues related to the above two techniques. Two chapters are dedicated to GANs –one to design an effective algorithm to tackle the problem of mode collapse and the other one to show its applicability in a retail context. The third one tackles exploration issues in continuous problems in RL, which is a major challenge especially for deceptive reward environment domains.

The first essay tackles a well-known theoretical problem encountered in training GANs: mode collapse. Indeed, due to the min-max optimization required in GANs, it is common for the generator in this technique to generate only a subset of all modes, resulting in a lack of diversity in the generated data. To overcome this issue, we use multiple discriminators with increasing representational capacity. We learn a curriculum by adaptively learning the weight assigned to each discriminator through a multi-armed bandit algorithm [5]. Empirically, we show not only better performance in term of diversity of generated data but also faster convergence of our algorithm.

The second essay uses GANs to learn the distribution of transactional data of customers for a drugstore chain. One can learn the purchasing patterns of individual customers by learning their distributions, and hence can generate items that follow the same distribution. This can then be used to predict the items that a "similar" arriving customer is likely to buy during her/his shopping trips. This algorithm can be used for a variety of purposes including as a recommendation system or for personalized bundle pricing. Empirical results show that the patterns of items generated by our model overlap heavily with real data, i.e., our model has good prediction power.

The third essay deals with exploration problems in Reinforcement Learning (RL), focusing especially on environments with deceptive rewards. Deceptive reward domains are challenging because their reward landscape is multimodal. Additionally, large state space makes the problem even more difficult (e.g., controlling a humanoid agent with many degrees of freedom). A good exploration strategy should have a high coverage of the solution space. To handle this, we adopt a multi-agent framework and design an exploration strategy wherein the agents coordinate their exploration efforts. Specifically, we use an attraction-repulsion mechanism to preserve diversity throughout the search over the solution space. Empirical results show that our method not only performs better in humanoid and other high-dimensional tasks, but also converges faster.

Abrégé

Aujourd'hui, l'accès à de gros volumes de données ainsi que leurs valeurs potentielles étant facilités, détecter des tendances dans les données est devenue une tâche cruciale pour collecter des informations et ainsi prendre de meilleures décisions. Dans ce contexte, l'apprentissage machine (ML) est devenu populaire grâce à sa puissance et sa flexibilité pour apprendre la structure des données et mieux comprendre le processus ayant généré les données. Par exemple, les modèles génératifs antagonistes (GANs) ont montré d'impressionnants résultats pour apprendre des distributions et ainsi générer des données de consommateurs artificielles pour un nouveau produit [52] ou encore des photos réalistes d'êtres humains inexistants [46]. Une autre branche populaire de ML est l'apprentissage par renforcement (RL), qui peut non seulement résoudre des problèmes complexes comme le jeu AlphaGo [70], mais résoudre les problèmes de prise de décision. Dans cette dissertation, nous présentons trois chapitres qui étudient théoriquement et empiriquement les deux précédentes méthodes. Deux chapitres sont dédiés aux GANs- dont un qui s'attaque au problème d'effondrement des modes de la distribution et l'autre qui est une application dans le contexte du commerce de détail. Le troisième chapitre s'attèle au problème d'exploration des domaines continus en RL, qui est un challenge majeur, spécialement pour les environnements à récompenses fallacieuses.

Le premier essai étudie un problème théorique bien connu rencontré dans l'entraînement des GANs: l'effondrement du nombre de modes appris. En effet, du à l'opérateur min-max dans l'optimisation des GANs, il est très fréquent pour le modèle de ne générer qu'un sous-ensemble du nombre de modes total, entraînant une réduction de la diversité des données générées. Pour remédier à ce problème, nous utilisons un ensemble de discriminateurs à capacité croissante. Nous varions les poids assignés à chaque discriminateur à l'aide d'un apprentissage progressif en utilisant un algorithme de bandit [5]. Empiriquement, notre méthode démontre non seulement de meilleures

performances en termes de diversité des données générées, mais aussi une meilleure convergence.

Le deuxième essai est une application des GANs pour l'apprentissage de la distribution des données transactionnelles des consommateurs d'une chaîne de pharmacie. La tendance d'achat des consommateurs peut être apprise à travers leur distribution et ainsi des données artificielles suivant cette même distribution peuvent être générées. Il serait possible d'utiliser cette méthode pour prédire les futurs achats de consommateurs "similaires". Quelques applications de cet algorithme incluent les systèmes de recommandation ou la personnalisation des prix. Les résultats empiriques montrent que la tendance des produits artificiellement générés est très semblable aux produits réels achetés. Notre modèle a un grand pouvoir prédictif.

Le troisième essai s'attaque aux problèmes d'explorations rencontrés dans l'apprentissage par renforcement (RL), et spécialement dans les environnements à récompenses fallacieuses. Ces domaines sont difficiles, car la surface de la fonction de récompense est multimodale. De plus, les espaces d'état de grandes dimensions rendent le problème encore plus dur (ex: contrôler un agent humanoïde avec plusieurs degrés de liberté). Une bonne stratégie d'exploration doit permettre une couverture importante de l'espace des solutions. Pour remédier à cela, nous nous plaçons dans le cadre multi-agent et développons une stratégie d'exploration dans laquelle les agents coordonnent leur exploration. Précisément, nous utilisons un mécanisme d'attraction et de répulsion pour préserver une diversité dans l'espace des solutions au cours de l'entraînement. Nos résultats empiriques démontrent de meilleures performances dans les environnements avec un grand degré de liberté, ainsi qu'une meilleure convergence.

Acknowledgements

I would like to thank my advisors Professor Saibal Ray and Professor Shanling Li for all their support and supervision during my doctoral studies. I would also like to thank everyone at the McGill School of Computer Science. They welcomed me like as one of their own students, provided support and treated me as a colleague. I met some wonderful people and enjoyed productive collaborations with two chapters in the present dissertation as a result.

Special thanks to all the doctoral students in the department, who overlapped with me during my doctoral studies and especially my officemate with whom I faced difficult times as well as happy ones. I would also like to thank Desautels faculty members and staff who helped manoeuver administrative hurdles and shared helpful advice. Also, I would like to extend a heartfelt thank you to DDSS (Desautels Doctoral Student Society) for providing a social network that created nice work life balance. And I cannot forget the hazelnut coffee that helped me survive through the Montreal winter.

Finally, I would like to thank my family: my parents, my girlfriend and my friends without whom this would not have been possible. Their constant practical support and encouragement was invaluable in helping me deal with the challenges of the PhD life.

Contributions of Authors

The first essay was done in collaboration with Joao Monteiro, Isabelle Albuquerque, Bogdan Mazoure, Audrey Durand, R Devon Hjelm and Joelle Pineau. It was done under the supervision of the two latter people respectively Adjunct Professor and Associate Professor at Montreal Institute of Learning Algorithm (Mila). The algorithms and experiments has been done and run by the authors of this thesis.

The second essay "Generating Realistic Sequence of Customer-levelfor Transactions for Retail Datasets" is a work done while interning at Rubikloud Technologies during summer 2018. It was done in collaboration with Neil Viera, a Rubikloud intern and Brian Keng, the Chief Data Scientist at Rubikloud. The main algorithms and experiments has been done and run by the authors of this thesis.

The third essay is co-authored with Bogdan Mazoure (which allowed me to put it in the present thesis) and done in collaboration with Audrey Durand, Joelle Pineau and R Devon Hjelm. The work has been conducted under the supervision of the two latter mentioned people. The design of the algorithm and experiments have been done equally by both authors.

While the essay chapter has been presented at the DMS workshop of the International Conference on Data Mining (ICDM, 2018), the second essay has been presented at the 33rd Conference on Artificial Intelligence (AAAI, 2019). The last essay is submitted at the 8th International Conference on Learning Representations (ICLR, 2020).

Table of Contents

List of Figures

List of Tables

Introduction

Access to huge volume of data available nowadays from various sources (loyalty card transaction, cookies, social network), makes it possible to analyze them to glean valuable information and insights that can then be used for better decision-making. However, such analysis requires use of sophisticated algorithms with Machine Learning (ML) techniques being one of the most popular ones. ML techniques are able to learn data patterns and can be used as a tool (among other things) to infer customer preferences, detect trends and predict/model future behavior of customers. Common commercial applications of ML include recommendation systems [87] as well as targeted advertising [61]. Other applications of ML are mainly in computer vision (facial recognition [7]) and robotics [34] fields.

ML techniques can divided into three main groups: Supervised Learning, Unsupervised Learning and Reinforcement Learning. Supervised learning algorithms tackle prediction and classification tasks. Specifically, given a dataset and its true label, the goal is to learn a function that maps from a given input to its corresponding label. For example, these techniques can predict the future demand of items based on historical data in a retail context. Classification techniques in computer vision can be used by cameras to recognize people entering and leaving a sensitive area. In contrast, Unsupervised learning techniques leverage inherent data structures to detect trends/patterns, in absence of any labels on the dataset. For example, clustering methods such as k-means [45] are commonly used in the marketing literature. Proper segmenting of customers ensure they have relatively similar behaviour and might help in designing better strategies to target specific group of customers. Another recent application of unsupervised learning is Generative Adversarial Network (GANS [27]) that has shown tremendous potential in learning probability densities. SUch models can generate realistic human faces [46], customers, or

even new apparel designs [81] by learning distributions of relevant input data. Finally, Reinforcement Learning (RL) is a framework for optimizing decision making problems. Based on the Markov Decision setting [77], it gives a coherent and succinct framework to solve either continuous control problems [94] or discrete action space problems [12]. It has gained notable popularity recently with applications in the sphere of playing complicated games (e.g., AlphaGo).

In this thesis, we address both unsupervised learning and RL methods in three chapters. Specifically, two chapters are dedicated to GANs and one to RL. In Chapter 2, we first tackle a common theoretical issue met by GANs: mode collapse where the generator concentrates the density onto a subset of the total modes of the real data distribution. This can generate loss of information. For instance, if one wants to learn the distribution of customers that buy a specific type of item, mode collapse in GANs will result in generation of only one type of customers. This can have business consequences since other types of potential customers will not be taken into account. To deal with that pathology, we train a generator with a set of increasing capacity discriminators and use multi-armed bandit to regulate the learning process. By this we are able to successfully deal with the mode collapse problem as we demonstrate empirically. Chapter 3 is an application of GANs in the retail context for generating realistic customer items. One can take advantage of the high volume of data from loyalty cards to learn patterns in customer behavior. In collaboration with a retail analytics company, we use GANs to model and learn about the distribution of items users are buying during their shopping trips based on historical transactional data. Subsequently, we are able to generate shopping baskets for future customers containing items users could have bought that seem to closely mimic reality. This method can serve as a basis for recommendation and personalized pricing systems.

Chapter 4 of the thesis is focused on continuous control for reinforcement learning. A particular domain of interest are the deceptive reward environments [17]. Such environ-

ments are characterized by a multimodal reward landscape. To see this, imagine controlling a humanoid agent that learns to walk. Not only do you need to adapt the locomotion of the legs but also balance arm movements. No natural reward exist in such case; instead, handcrafted rewards are usually provided (like distance from starting point, alive bonus reward, velocity etc...). This can easily trap policy in suboptimal solutions (agents trying to fall the furthest when stuck in an unbalanced movement or learning non-viable gait) and eventually no stable walk is learnt. One direction is to design a smart exploration strategy. Because of the large state space (376 for humanoid agent in MuJoco [94]), we handle this problem under the multi-agent framework and hypothesize that coordinated exploration among our agents might be the solution for an efficient exploration mechanism. We show emprical results that our algorithm is successful in this context.

In summary, this thesis contributes theoretically by developing solutions for certain challenges faced by ML techniques and also practically by showing how ML techniques can be used in reality to make better decisions.

Chapter 1
On-line Adaptative Curriculum Learning for GANs

1.1 Introduction

Generative Adversarial Networks [27] have reshaped the state of machine learning in tasks that involve generating data. A GAN is an unsupervised method that consists of two neural networks, a generator and a discriminator, with opposing (or *adversarial*) objectives. The typical goal of the generator is to transform noise (e.g., drawn from a normal distribution) into samples whose statistical and structural characteristics match well those of an empirical target dataset (such as a collection of images). The discriminator, which acts as an *adversary* to the generator, needs to discriminate between (or *classify*) samples as coming from the real data or the generator.

While GANs can achieve impressive qualitative performance (most notably with image data, e.g., see) [46, 69, 82], the most successful methods depart from the original formulation to address various instabilities and other optimization difficulties [3, 4]. One such difficulty in training GANs occurs when the generator produces samples only from a small subset of the target distribution, a phenomenon known as *missing modes*(a.k.a., *mode-dropping*, e.g. see) [16]. Numerous works try to address the problem by modifying the original objective, such as unrolling [66], aggregating samples [56], stacked architectures [43, 46], mutual information / entopy maximization [11], multiple discriminators [44, 71], or multiple generators [37, 53, 95].

In our work, we follow the intuition that missing modes in GANs are due in part to mode-specific vanishing gradients. As a simple illustrative example which we explore in detail in our experiments below (Fig. 1.1), consider a discriminator that is well represent-

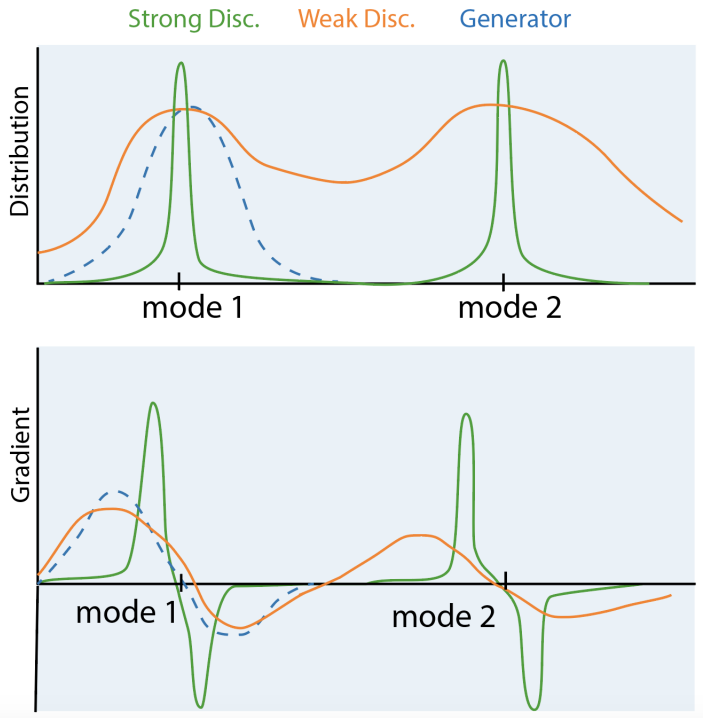


Figure 1.1: Recovering dropped modes via multiple discriminators. The weak discriminator provides feedback, allowing the generator to recover forgotten modes. The strong discriminator experiences vanishing gradient and cannot help the generator to recover modes.

ing the target distribution and a generator that is only generating a subset of the modes in the data. If any of the missing modes are *disjoint* from those represented in the generator (i.e., are composed of sets of features with low intersection), there is no way for the generator to receive gradient signal on missing modes from the discriminator. However, if the discriminator only represents the data approximately (in the sense that it also cannot fully distinguish between these modes), it may be possible to recover the missing mode gradient signal. If this can be achieved by using a low capacity[1] discriminator, it is ultimately undesirable given that the end goal is to generate samples that resemble well the target dataset. From now on, we will refer to such low capacity discriminators as *weak* and to high capacity discriminators as *strong*. In order to ensure both high quality and mode coverage, we consider multiple discriminators(as in [21]) with different strengths to train

[1]Throughout the paper, we refer to *capacity* as the architecture size of a given neural network in terms of number of parameters.

the generator. We propose to train the generator using a curriculum based on an on-line multi-armed bandit algorithm [28, 63], dynamically changing the weight/resources allocated to each discriminator, which we show is crucial for achieving good results. Our primary contributions are:

1. We provide important insights into the missing mode problem as demonstrated by the gradient signal available to the generator from the discriminator.
2. As a potential solution to the missing modes problem, we introduce a new framework based on adversarial bandits [5, 24, 58] resource allocation, where the generator gets its training signal from a set of teacher networks with increasing capacity.
3. We show that the proposed approach leads to a curriculum learning characterized by successive phases of the generator prioritizing different discriminators.

The remainder of this paper is organized as follows. Previous literature relevant to this work is briefly reviewed on Section 1.2. The proposed approach is formally introduced in Section 1.3, and an empirical analysis is reported in Section 1.4. Conclusions and future directions are finally presented in Section 1.5.

1.2 Related Work

Mode coverage and data / model augmentation

The intuition that missing modes are due to vanishing gradients resonates with some successful approaches on stabilizing and improving GAN training through data and model augmentation. Instance noise [3] has been shown to improve stability (see also [82]), which can be understood as smoothing the data modes in the pixel space. Progressively reducing the downsampling through training (either by copying parameters or feeding low resolution samples into a larger generator) have also been considered previously [43, 46] as solutions to increase mode overlap. This is akin to a hand-crafted cur-

riculum, progressively increasing the difficulty of the problem at a-priori chosen points in the complete training procedure.

Multiple discriminators and generators

Several works have also incorporated multiple generators or discriminators in order to improve learning. Multiple-generator methods [37,53,95] typically work by encouraging the generators to divide the task of generating by modes in the target dataset (without additional supervision). Using multiple discriminators [44,71], on the other hand, is known to provide a better learning signal for the generator if said discriminators compositionally represent well the target datasets. Closest to our work, [21] consider discriminators of different complexity to provide varied signal. We will show that wisely designing the reward allows to track the progress made by the generator and encourages a curriculum learning.

Multi-armed bandit as a curriculum learning method for GANs

Curriculum learning [14] phrases a given machine learning problem as a set of tasks of increasing difficulty. GANs can also be said to share aspects with curriculum learning: the discriminator defines an objective of progressive difficulty,

thus allowing the generator to gradually learn to more faithfully mimic the target distribution. However, there is no explicit mechanism to encourage a sensible curriculum for either model. For example, if the discriminator learns to represent disjoint modes faster than the generator learns to cover them, this can lead to the generator missing modes with no gradient signal to recover.

In this paper, we propose an algorithm which gives rise to a curriculum in a direct manner. Our approach borrows from curriculum learning in multi-armed bandit setting [28,63], where learning is typically done by measuring the change in a performance criterion of a given agent (i.e. a loss function, score or gradient norm can be used) that appears to affect the form of the optimal policy. In our method, given a set of discrimi-

nators, the goal is to weight the feedback received by the generator proportionally to the information contained in the gradients from each discriminator.

1.3 Adaptative Curriculum GAN

Here we formulate the problem and approach for training a single generator on a target dataset using a curriculum over multiple discriminators, which we call *Adaptative Curriculum GAN* (acGAN). First, define a generator function, $G : \mathcal{Z} \mapsto \mathcal{X}$, which maps noise from a domain $\mathcal{Z}$ to the domain of a target dataset, $\mathcal{X}$ (such as the space of images). Let $p(x)$ denote the target density [2], and let $p(z)$ denote the prior density defined on Z used to draw noise samples for input into the generator. We wish to train this generator function using N discriminators, $\mathcal{D} = \{D_i : \mathcal{X} \mapsto \mathbb{R}\}_{i=1}^{N}$, such that on each episode t, we select the mixture of discriminators that provides the best learning signal.

Note that, in theory, given enough capacity for the discriminator and generator, we can have some convergence proofs and indeed some exist for this min-max problem [27] under some mild assumptions. But in this work, we use highly non-linear functions such as Neural Networks. This makes it analytically difficult to develop theoretical guarantees about bounds or convergence.

1.3.1 Mixing discriminators

This mixture-of-experts problem, where each discriminator plays the role of a teacher, can be tackled under the full-information adversarial bandit setting [5, 24, 58]. On each episode t, a bandit player associates normalized weights $\Pi(t) = \{\pi_i(t)\}_{i=1}^{N}$ with discriminators $\{D_i\}_{i=1}^{N}$. The generator is then trained based on the mixture described by $\Pi(t)$, and a reward $\mathcal{R}_i(t)$ is observed for each discriminator D_i, characterizing the generator's improvement with respect to D_i. Let $\mathcal{R}(t) = \sum_{i=1}^{N} \pi_i \mathcal{R}_i(t)$ denote the total ob-

[2] Here, we assume for the sake of notation that the target data admits a density.

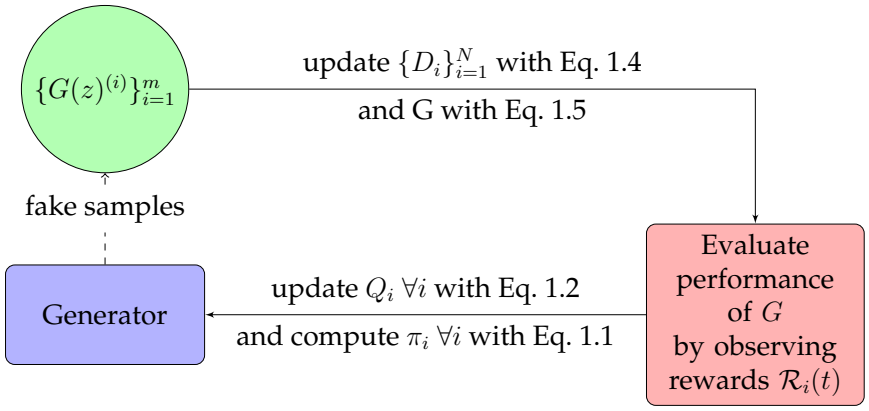


Figure 1.2: Proposed procedure for training the generator

served reward at time t. The goal of the player is to learn the optimal policy $\Pi^\star(t) := \operatorname{argmax}_{\Pi\in\Delta(N-1)} \mathbb{E}_{\Pi(t),p(z)}[\mathcal{R}(t)]$ that maximizes the expected total reward[3].

The Hedge algorithm [24], also known as Boltzmann or Gibbs distribution, addresses this full-information game by maintaining probabilities

$$\pi_i(t) = \frac{\exp \lambda Q_i(t)}{\sum_{j=1}^{N} \exp \lambda Q_j(t)}, \quad \lambda \geq 0, \tag{1.1}$$

for each discriminator D_i, where $Q_i(t)$ estimates the gain of D_i at episode t. In this case, λ is a parameter of the distribution: $\lambda = 0$ corresponds to a uniform distribution over all models. We found experimentally that using a moving average on previous rewards (which also featured in [63]) stabilizes the training:

$$Q_i(t) = \alpha \mathcal{R}_i(t) + (1-\alpha) Q_i(t-1), \tag{1.2}$$

where $\alpha \in (0, 1)$ is the smoothing parameter.

To demonstrate how this can be used to train GANs, consider the usual value function [27]:

$$V(D, G) = \mathbb{E}_{p(x)}[\log(D(x))] + \mathbb{E}_{p(z)}[\log(1 - D(G(z)))]. \tag{1.3}$$

[3] $\Delta(N-1)$ denotes the standard simplex on $\mathbb{R}^N$.

On each episode t, given the mixture of discriminators $\Pi(t)$, each discriminator is trained by taking a gradient step to increase the expected value function

$$\mathbb{E}_{\Pi(t)}[V(D_i, G)] = \sum_j \pi_j(t) V(D_j, G), \tag{1.4}$$

and the generator is trained by taking a gradient step to increase

$$\mathbb{E}_{\Pi(t)}[\mathbb{E}_{p(z)}[\log(D(G(z)))]]. \tag{1.5}$$

The latter corresponds to the non-saturated version of Eq. 1.4 for the generator. The intuition is that training the generator with all the discriminators *simultaneously* (as a mixture) should force the generator to fool all discriminators at the same time [21]. Since each discriminator has an increasing level view of the modes distribution, they should have a complementary role. While the weaker discriminator focuses on modes coverage, the stronger discriminator ensures samples quality (showed in Section 1.4.1). This should result into a better overall coverage of the modes in the input distribution.

Algorithm 1 describes our proposed acGAN procedure. We denote and parameterize this algorithm as acGAN$(\lambda, \alpha, \mathcal{R}_r)$ where $\lambda \geq 0, \alpha \in (0, 1)$.

Algorithm 1 Generic acGAN algorithm

Given: N: number of discriminators, T_{max}: time steps, T_{warmup}: warmup time, α: moving average coefficient, λ: Boltzmann constant
$Q_i(0) \leftarrow 0, \forall i = 1, \ldots, N$
for $t = 1, \ldots, T_{max}$ **do**
  Update all discriminators $\{D_i\}_{i=1}^N$ using Eq. 1.4
  Update the generator G using Eq. 1.5
  if $t \geq T_{warmup}$ **then**
    Evaluate the performance of G and observe a reward $\mathcal{R}_i(t)$ for each discriminator i
    Update all values $\{Q_i(t)\}_{i=1}^N$ according to Eq. 1.2
    $\pi_i(t) \leftarrow \exp^{\lambda Q_i(t)} / \sum_{j=1}^{N} \exp^{\lambda Q_j(t)} \quad \forall i = 1 \ldots N$
  end if
end for

Remark 1 *At the beginning of the training, we define a warm-up period T_{warmup}, prior to which we train D_i and G with a uniform probability, i.e $\pi_i = \frac{1}{N}, \forall i = 1, \ldots, N$. In other words, we consider $\lambda = 0, \forall t \leq T_{warmup}$. This guarantees that each discriminator is updated a minimum number of times (or provides feedback a minimum number of times to the generator) and prevents one D_j from dominating the others (i.e, $\pi_j \gg \pi_i, \forall i \neq j$) at the beginning of the training. Without this safeguard, the remaining weights $\pi_i, i \neq j$ would hardly recover a significant probability and the generator may never get informative gradient from the corresponding discriminator. Note that warm-ups are not uncommon either in bandits algorithm, e.g. for adding robustness to the tails of reward distributions [9].*

1.3.2 Reward shaping

In order to provide meaningful feedback for learning efficient mixtures of discriminators, we consider different reward functions to generate $\mathcal{R}_i(t)$. We argue that progress (i.e., the learning slope [28, 63]) of the generator is a more sensible way to evaluate our policy. Let $\theta(t)$ be the generator parameters at episode t. We define the two following quantities for measuring generator progress:

$$\mathcal{R}_i^{\mathcal{S}}(t) = \mathbb{E}_{p(z)}[D_i(G(z;\theta(t))) - D_i(G(z;\theta(t-1)))], \tag{1.6}$$

$$\mathcal{R}_i^{\mathcal{V}}(t) = \mathbb{E}_{p(z)}[V(D_i, G(z;\theta(t))) - V(D_i, G(z;\theta(t-1)))]. \tag{1.7}$$

The former measures the progress of the generator with respect to the discriminator i score $D_i(\cdot)$, while the latter assess the change in the loss function (Eq. 1.3). Since the change in the quality sample (Eq .1.6) led to better performance than the change in the loss function (Eq .1.7), all our experiments (see Section. 2.4) use Eq .1.6.

1.3.3 Connection to existing methods

Interestingly, some existing methods in the GAN literature can be seen as a specific case of acGAN:

GMAN: The original GMAN [21] algorithm can be recovered by setting $\alpha = 1$ and taking the loss function to be the reward $\mathcal{R}_i(t) = V(D_i, G)$. Note how the authors of GMAN call their algorithm GMAN-λ, where λ is also the Boltzmann coefficient.

Uniform: The uniform case is defined by assigning a fixed uniform probability for each discriminator D_i:

$$\pi_i(t) = \frac{1}{N}, \quad \forall t \in \mathbb{N}.$$

This corresponds to Eq. 1.1 with $\lambda = 0$.

To support the results of our theoretical work, we conducted a set of experiments which we describe below.

1.4 Experiments

In this section, we first give an understanding of how each discriminator provides informative feedback to the generator. We then compare our proposed approach (acGAN) against existing methods from the literature.

1.4.1 Retaining mode information through weaker capacity discriminators and smoothness

We begin by analyzing the gradient norm of the discriminator networks and we show that weak capacity discriminators are *smoother* than strong discriminators. This property corresponds to a "coarse-grained" representation of the distribution, which allows the

generator to recover missing modes. We further show we can increase the smoothness of a weak discriminator by corrupting its inputs with white noise. This results in an increase of the discriminator's entropy (see Supplementary Material for more details) and hence smoother gradient signal.

Weak Discriminators: a way to retain modes

We now highlight the role of weaker capacity discriminators. To this extent, we performed the following experiments on the 8 Gaussian synthetic dataset:

- We pretrained the generator (with 3 dense layers of 400 units with ReLU activation layers except for the last layer) with one discriminator on only 2 of the original 8 modes.
- We trained a (vanilla) GAN on all 8 Gaussian components, initializing with the 2-mode generator above. The discriminator had 3 dense layers of 400 units (ReLU hidden activation layers).
- We trained acGAN with the generator initialized with the 2-mode generator (as with vanilla GAN). We considered 3 discriminators, with 1, 2 and 3 dense layers respectively (same activation scheme as previously applies here).

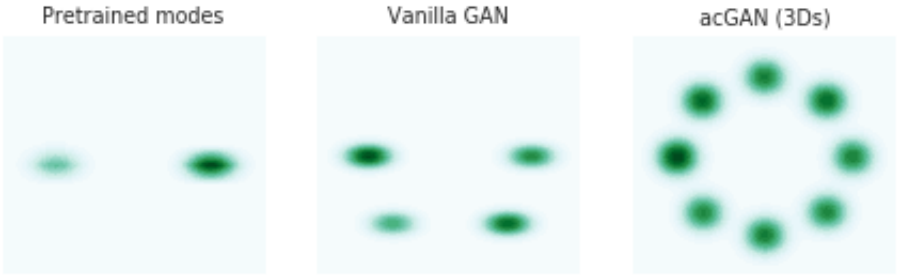


Figure 1.3: Modes used for pretraining the generator (left) and modes recovered by Vanilla GAN (middle) and acGAN (right). The more modes the better.

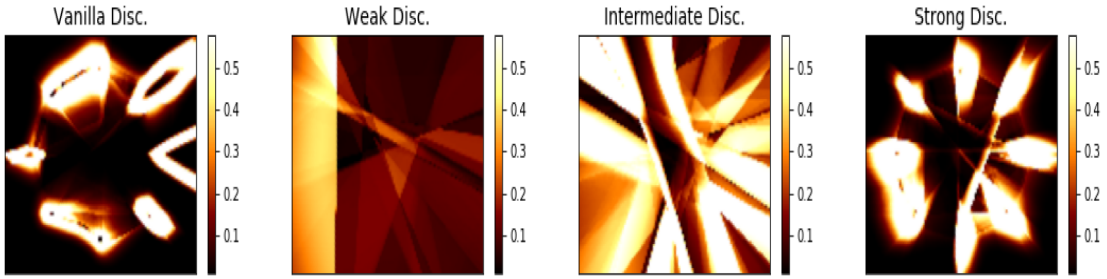


Figure 1.4: Gradient norm of each discriminator with respect to the input. We clipped the magnitude with respect to the weaker discriminator range. Since weaker discriminators are smoother by construction, they help the generator to recover missing modes. On the other hand, vanilla GAN can hardly recover modes due to its vanishing gradient.

Results (Fig. 1.3) show the Vanilla GAN could only retrieve 2 additional modes, while acGAN recovered all (8) modes. We examined the gradients provided by the discriminators using a density plot (Fig. 1.4) of the gradient norm for each discriminator with respect to the input, i.e., $||\nabla_X D(X)||_2$ for $X \in [-2, 2]^2$. Observe that there is a clear progression from a stronger discriminator with more distinct, higher gradients to the weaker discriminator smoother gradients. Additionally, note that the discriminator from the vanilla GAN, which has very high gradient norm values, has gradients for modes not present in the generator: the discriminator has information useful for learning about these missing modes, but the generator does not learn these modes due to vanishing gradients. Our results support both our original hypothesis that missing modes are due to vanishing gradients and that using a coarse-grain discriminator can be used to recover missing modes. To provide further insight, we show the evolution of the gradient norm of each discriminator at training time in the Supplementary Material. We also note that the discontinuities in the gradients is due to the ReLU activation partitioning the subspace through overlapping half-planes, which contrasts the smooth decay of hyperbolic tangent and sigmoid[4] nonlinearities, and we further explore the effect of different nonlinear activation layers on the gradient norm of the weak discriminator in the Supplementary Material.

[4] $\sigma(y) = 1/1 + e^{-y}$

	FD	Modes	Quality samples
Vanilla GAN	7.28	17	88%
Uniform (3D)	6.64	20	93.4%
acGAN (3D)	6.65	25	92.9%

Table 1.1: Results on the Gaussian mixture synthetic data. Our method acGAN could cover all 25 modes.

1.4.2 Performance of acGAN against existing baselines

In this section, we evaluate the performance of our proposed method (acGAN), on various datasets. All experiments consider the reward shown in Eq. 1.6. We first conducted a sanity check on 2 mode-dropping datasets: synthetic data consisting of a mixture of 25 Gaussians and Stacked-MNIST with 1000 modes. We then tested it on CIFAR10 and finally show generated samples on celebA dataset (see Supplementary Material). We aim to analyze specific properties such as diversity of generated samples and quality in terms of (FID [36]) score when available, along with convergence of the method (how fast it reaches its minimum FID score). Additionally, our results hint at the emergence of a curriculum during the training process.

All parameters used to obtain the results can be found in the Supplementary Material. We split the batch of inputs between discriminators. We abuse of language with the term *epoch*, which in the context of the current paper means that the generator has been trained on a number of iterations equivalent to an epoch. For example, CIFAR-10 has 50,000 training images and, assuming a batch size of 64, one epoch represents roughly 781 iterations for the generator.

Synthetic Gaussian mixture dataset

The synthetic dataset is composed of 25 bivariate Gaussian mixtures arranged in a two-dimensional grid. We launched a single run of 15 epochs for all methods with 3 discriminators. We report 3 measures in Table 1.1: the Fréchet Distance (FD), the number of recovered modes and the proportion of high quality samples (which is the proportion of

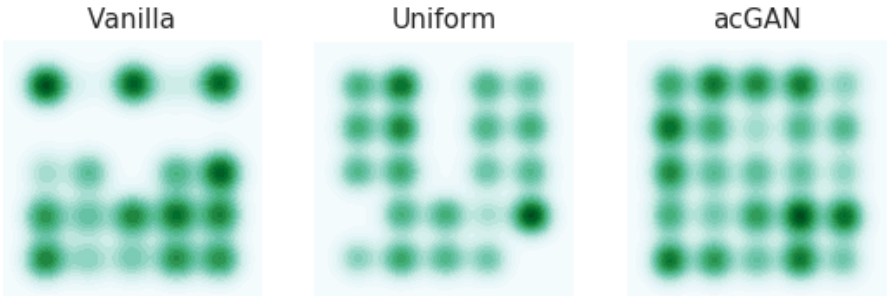


Figure 1.5: KDE plots of the modes recovered by each examined approach with 3 discriminators.

	Modes (max 1000)	KL
DCGAN [79]	99.0	3.40
ALI [20]	16.0	5.40
Unrolled GAN [66]	48.7	4.32
VEEGAN [89]	150.0	2.95
PacGAN [56]	1000.0 ± 0.00	$0.06 \pm 1.0e^{-2}$
GAN+MINE [11]	1000.0 ± 0.00	$0.05 \pm 6.3e^{-3}$
acGAN (3D)	1000.0 ± 0.00	$7.4e^{-2} \pm 0.0$
acGAN (5D)	1000.0 ± 0.00	$9.65e^{-2} \pm 0.0$

Table 1.2: Number of modes covered and Kullback-Leiber divergence between the real and generated distributions on Stacked-MNIST. acGAN could recover the 1000 modes.

samples covering a mode). More details on those metrics can be found in the Supplementary Material.

We compared the performance of our proposed methods to that of the Uniform algorithm and of the vanilla GAN [27]. Our proposed methods could cover the 25 modes. KDE plots for the 3 discriminators case are shown in Fig. 1.5.

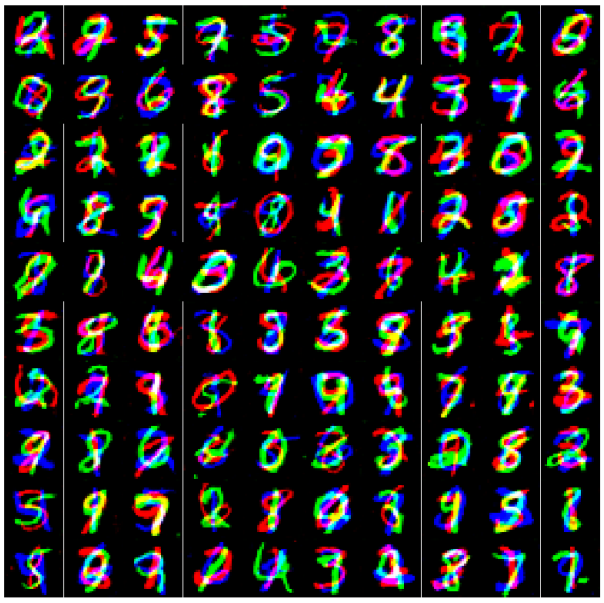


Figure 1.6: Stacked-MNIST generated samples for acGAN with 3 discriminators.

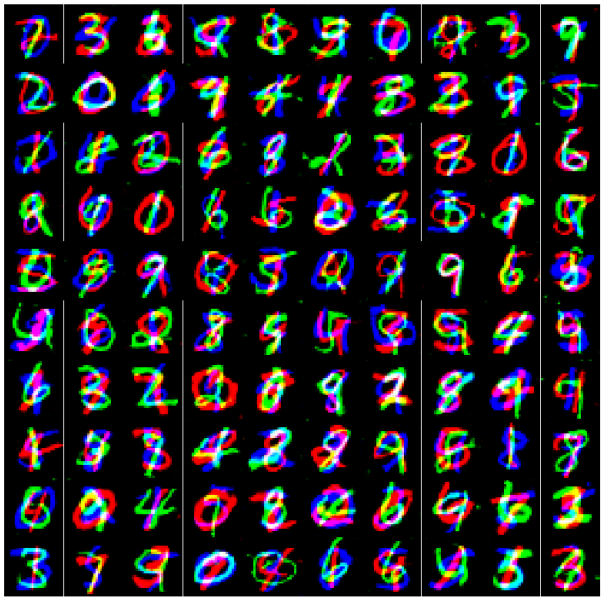


Figure 1.7: Stacked-MNIST generated samples for acGAN with 5 discriminators.

Stacked-MNIST

We use the Stacked-MNIST dataset [89] to measure the mode coverage of our proposed approach. The dataset is generated by stacking 3 randomly selected digits from the MNIST dataset: one on each RGB channel to produce a final $28 \times 28 \times 3$ RGB tensor. The dataset has 128,000 training images and is assumed to have 10^3 modes. Results of our experiments are shown in Table 1.2.

We report our results (averaged over 10 runs) in Table 1.2 and compare them with other existing baselines in the literature. Our method could recover all 1000 modes like PaCGAN [56] and MINE [11]; these two approaches either increase the dimensionality of the generator inputs either by packing multiple samples or by adding a latent code vector which helps overcoming mode collapse. Generated samples are shown in Fig. 1.6 and Fig. 1.7, our results further verify our hypothesis that acGAN is a sensible approach to ensuring good mode coverage and sample quality.

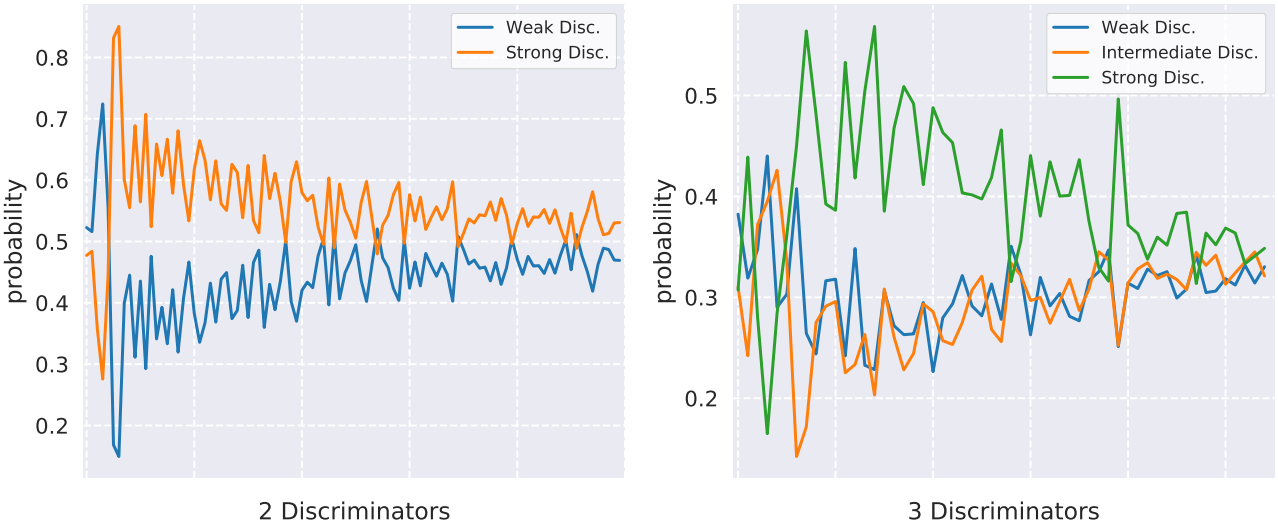


Figure 1.8: Weight π_i of each discriminator over the training epochs. We can see phase switching at the beginning where each discriminator's weight is dominating before eventually converging to a uniform distribution.

CIFAR-10

We conducted an in-depth study of acGAN's performance on CIFAR-10 by running experiments on 5 independent seeds for 50 epochs each.

We found a particular pattern in the acGAN's learning process: it consists of distinct regimes where one discriminator's weight π_i dominates over the others. To illustrate this, we averaged the sampling probability of each discriminator over every 200 iterations and plotted results in Fig. 1.8 for 2 and 3 discriminators, respectively. The reported curves suggest that, for $N = 2$ discriminators, the weakest discriminator network is often sampled at the beginning until the generator G learns enough from it, at which point it begins to use the stronger discriminator more often. Note how the strong discriminator is sampled more frequently than the weak one. In fact, because the generator needs to produce samples of higher quality to fool the strong discriminator, training with the latter might take longer as opposed to using weaker discriminators (which are more lenient). By the end of training, all discriminators are being used in equal proportions, meaning that every discriminator plays a complementary role from mode coverage to quality samples. A similar pattern is observed for the 3-discriminators case.

To assess the quality of produced results, we report the minimum Fréchet Inception Distance (FID [36]) (and corresponding epoch) reached in Table 1.3. The squared FID was computed every epoch with 1,000 held-out samples at training time. As in [22], a ResNet pre-trained on CIFAR-10 was employed to obtain representations for FID computation rather than Inception V3. Proceeding this way yields a more informative score, given that our classifier was trained on the same data as the generative models. Details on the FID score can be found in the Supplementary Material. We compared our results to [21]. Since the authors reported that GMAN-1 ($\lambda = 1$) had an overall better performance, we used this version in our experiments and refer to it as GMAN. Previously, we observed that the feedback provided to the generator is shared between all the discriminators. Especially, not all gradient comes from the strong discriminator (unlike for the Vanilla GAN). One might be concerned by a degradation of the quality samples. We

show that having more discriminators leads to better mode coverage and samples quality (see the FID curves for an increasing number of discriminators in the Supplementary Material). Overall, we noticed that acGAN achieved the best FID score when compared to the baseline as presented in Fig. 1.9 and 1.10 (plots are shown in a larger format in in the Supplementary Material). GMAN performed worse than expected and increasing the number of discriminators did not significantly improve its FID score. We suspect that the original loss function of the GAN (which is equivalent to the Jensen-Shannon divergence minimization) is not a good signal to assess the progress of G. Indeed, [4] argued and introduced a toy example showing that this version of adversarial nets is not informative when there is little overlap between the supports of the true and approximate distributions, as commonly seen at the beginning of the training process. Finally, not keeping a moving average via a Q-value can lead to high variance.

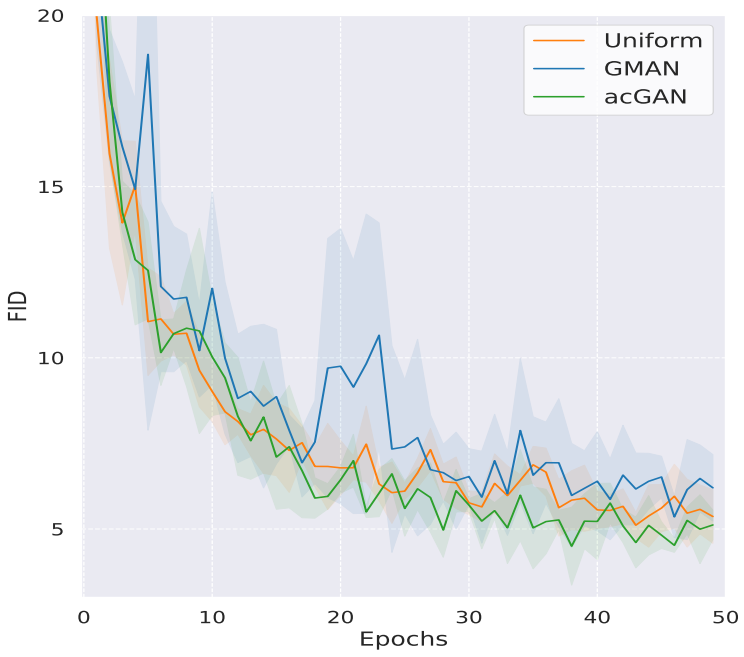


Figure 1.9: FID scores computed with 1,000 samples at the end of each epoch for different methods with 3 discriminators. acGAN outperforms the baselines Uniform and GMAN.

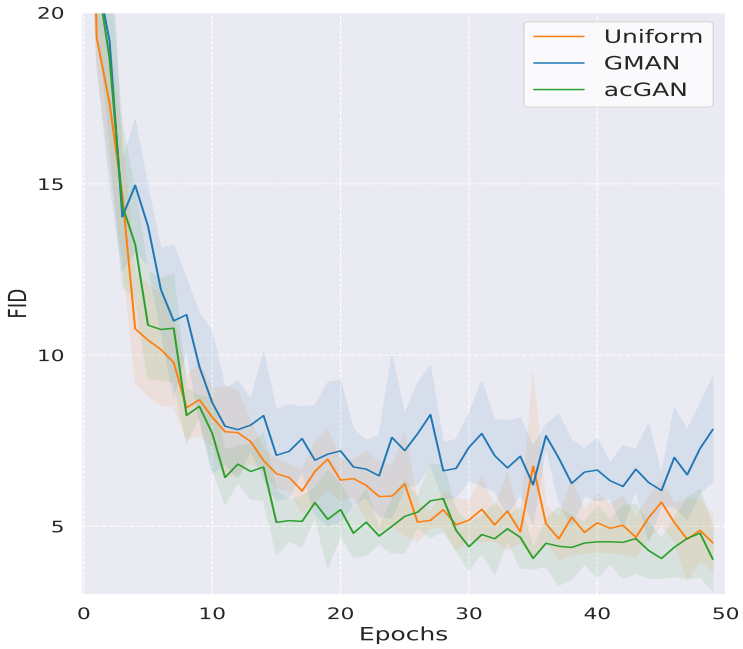


Figure 1.10: FID curves with 5 discriminators. acGAN presented earlier convergence and reached lower FID values.

		Best FID *(epoch)*	Mean Best FID
	Vanilla GAN	*5.02 (20) - 5.28 (27) -* ***4.27 (30)*** *- 4.80 (34) - 4.63 (41)*	*4.80*
	WGAN-GP[5]	*4.29 (43) - 4.24 (28) - 3.98 (47) - 3.99 (37) -* ***3.93 (50)***	*4.08*
3 Disc	Uniform	*4.18 (20) -* ***4.07 (39)*** *- 4.35 (45) - 5.07 (30) - 4.39 (47)*	*4.41*
3 Disc	GMAN	***3.87 (43)*** *- 4.05 (46) - 5.24 (42) - 5.71 (42) - 4.10 (22)*	*4.59*
3 Disc	acGAN	*3.93 (39) - 3.57 (38) - 4.25 (42) - 3.43 (40) -* ***3.11 (43)***	**3.66**
5 Disc	Uniform	***3.42 (47)*** *- 3.69 (49) - 4.37 (37) - 3.64 (37) - 3.47 (40)*	*3.72*
5 Disc	GMAN	*4.58 (44) - 4.40 (20) -* ***3.91 (47)*** *- 4.81 (25) - 4.42 (38)*	*4.42*
5 Disc	acGAN	*3.62 (35) -* ***2.62 (49)*** *- 4.14 (35) - 2.66 (42) - 3.67 (34)*	**3.34**

Table 1.3: Best FID scores on CIFAR-10 computed on 1,000 samples during training time (lower is better).

1.5 Conclusion

In this work, we model the training of the generator against discriminators of increasing complexity within a one-student/multiple-teachers paradigm. We address this mixture-

[5]We replaced the batch norm layer with instance norm

of-experts problem under the adversarial bandit setting with full-information, where we rely on the Hedge algorithm to learn the weights assigned to each discriminator in the mixture. Since designing a suitable reward function is a key ingredient to control the shape of the learned policy, we examined two sensible reward functions which relied on sample quality and the GAN loss function. We empirically found the high quality sample reward (Eq. 1.6) to yield the best results. Keeping a moving average on the rewards helped smoothing the weights put on discriminators and resulted in a more stable mixture.

Then, we demonstrated a complementary regulation mechanism between weak and strong discriminators. While weaker discriminators enjoy smoother properties and provide more informative feedback to the generator, stronger discriminators focus one finer grain detail to ensure sample quality.

Finally, we conducted a series of experiments to show the emergence of a curriculum during the training process. That is, lower-capacity discriminators have higher weights at the beginning but, as the training progresses, higher weights are allocated to higher-capacity discriminators. We showed how existing algorithms could be recovered from our model via the Q-value. The performed experiments showed that our proposed approach leads to an earlier convergence and a better FID score compared to existing baselines in the field, i.e. Uniform and GMAN.

As a direction for future investigation, approaches not relying on the adversarial framework could be investigated to model the non-stationarity of the reward distributions. For example, finding a meaningful representation for the state of the generator could allow the use of contextual bandits algorithms.

1.6 Supplementary Material

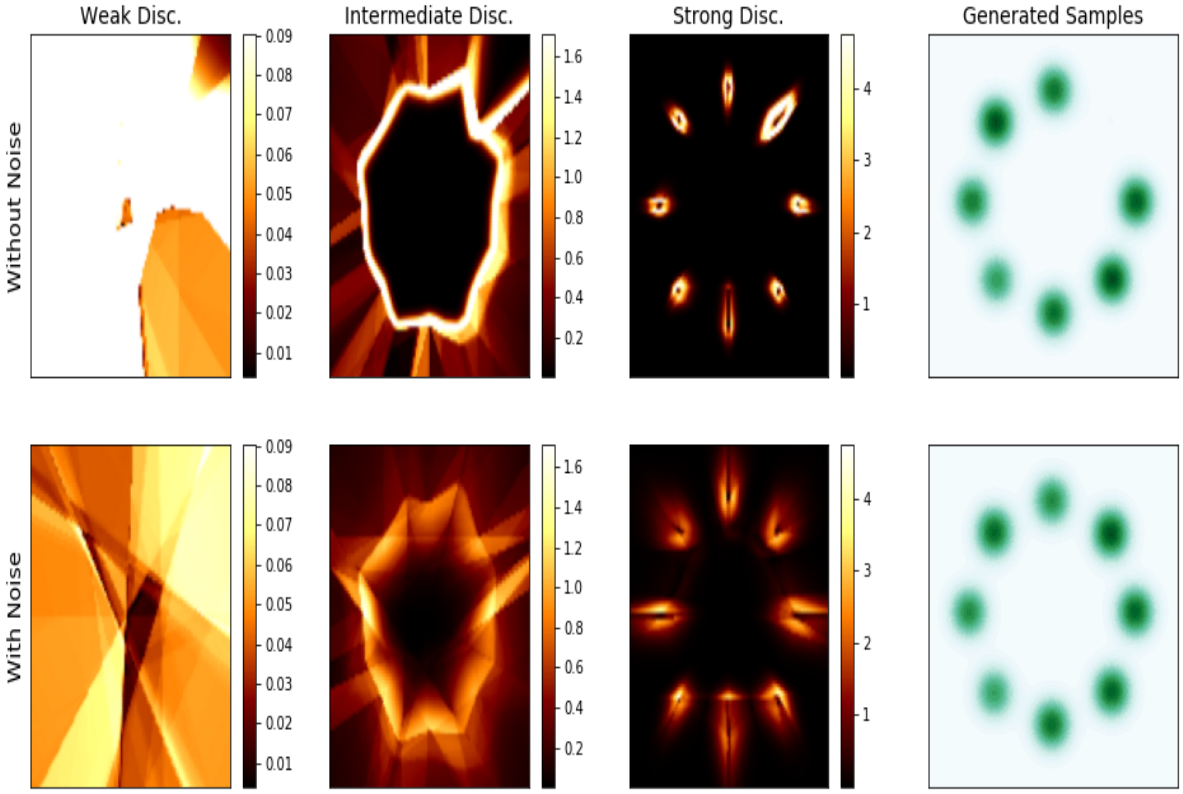


Figure 1.11: Adding noise (bottom row) reduces gradient norm magnitude of each discriminator. This increases their smoothness properties and helps recovering modes of the distribution. We clipped the gradient magnitude with respect to the corresponding discriminator corrupted with noise.

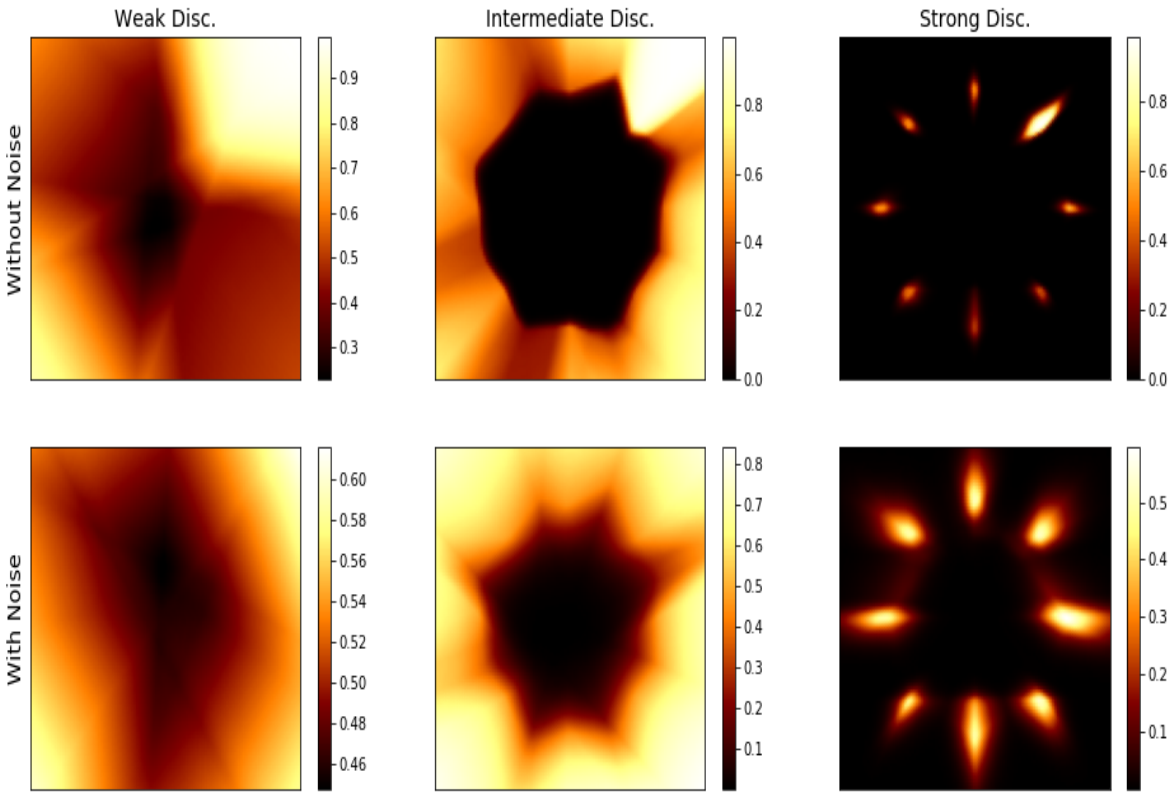


Figure 1.12: Probability $D(x)$ for each discriminator without (top) and with (bottom) white Gaussian noise. Noise tends to smooth their decision boundary and increase their entropy. That helps to provide more informative gradient to the generator.

Regularizing the discriminator through additive white noise

As was explored in [3], one way to stabilize GAN training is corrupting the input of the network with additive white Gaussian noise of the form $N(0, \sigma)$. Here, we explore smoothing the discriminator by using the noise, We ran the following experiment in order to illustrate the mechanism. We (once again) train G on the 8 Gaussian synthetic dataset with 3 discriminators (1,2 and 3 hidden layers of 256 units) both with and without adding independent Gaussian noise to the discriminator's input. A noticeable downside of feeding corrupted inputs to the network is the degradation of samples' quality: the so-called salt and pepper effect becomes more visible as the discriminators train. To solve this issue, we decay the noise at time step t by a multiplicative coefficient: $\exp \frac{t}{C}$, where $C > 0$ is a real constant controlling the noise reduction speed. Initial Gaussian noise was picked to be of the form $N(0, \sigma_i)$, with variances of $\sigma_1 = 0.06$, $\sigma_2 = 0.04$, $\sigma_3 = 0.02$, for $i = 1$ being the weakest discriminator and $i = 3$ the strongest. Adding white noise increases the entropy (read uncertainty) of the discriminator (a proof is shown in the Supplementary Material) and tends to smooth its decision boundary (see the probability and gradient norm values in Figs. 1.12 and 1.11). Fitting a discriminator to uncorrupted input is prone to faster overfitting as opposed to training on noisy data when fixing the number of parameters, a great illustration of which is provided by [29]. Empirical results are shown in Fig. 1.11. We see that by corrupting the real data we manage to cover all 8 modes and the sample quality is conserved by decaying the variance of the noise. The evolution of generated data points is shown in Fig. 1.15.

1.6.1 Effect of different nonlinear activation layer on the weak discriminator's smoothness

In this section, we aim to illustrate the effect of 3 nonlinear activation layers (Tanh, Leaky RelU and ReLU) on the gradient norm of the discriminator. We ran the training of the generator with 3 discriminator (using Soft-acGAN) on the 8 Gaussian dataset. For some performance issue, we just replace the activation layer of the weak discriminator with respectively the 3 above mentionned activation layers and let the other discriminator with ReLU. Fig. 1.13 shows the gradient norm of the weak discriminator on the whole space $[-2, 2]^2$. We see that Tanh has a very uniform gradient norm across the space while ReLU is the most discontinuous. Leaky ReLU has an intermediate pattern. Yet, Tanh seems to have flat behaviour (very small magnitude), this may be due to the Tanh function that has very low gradient signal at the extremity (indeed, we witness very poor performance with that activation layer). Leaky ReLU is less discontinuous although it also partitions the subspace in the same way as ReLU.

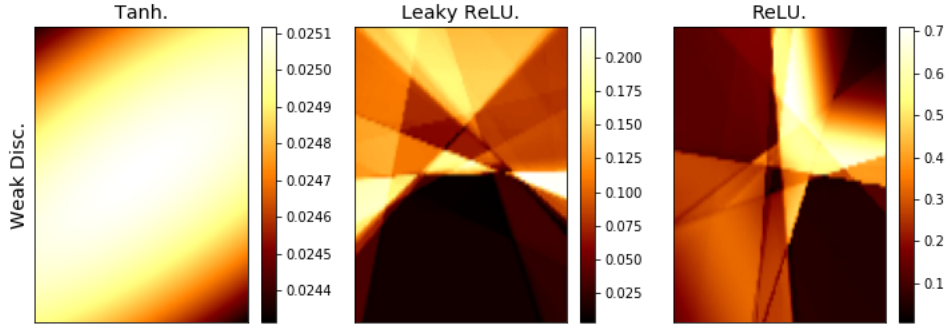


Figure 1.13: Although Tanh (left) presents smoother partition of the subspace than LeakyReLU (middle) and ReLU (right), it seems to have weak gradient signal (small gradient norm magnitude).

1.6.2 Evolution of the gradient norm during the training

In this section, we show the evolution of the gradient norm of each discriminator throughout the training process (results shown in Fig. 1.14). We see at the beginning of the training (first row) as the generator has just learned the top left modes, discriminator has flat behavior on the bottom right part of the subspace and has higher gradient norm on the top left part. A the training process, we see that missing modes has high gradient norm (second row third column). Finally, at the end when the generator has learned all the modes the weak discriminator seems to have more uniform gradient norm on the space while strong discriminator has equal gradient norm value at each modes locations.

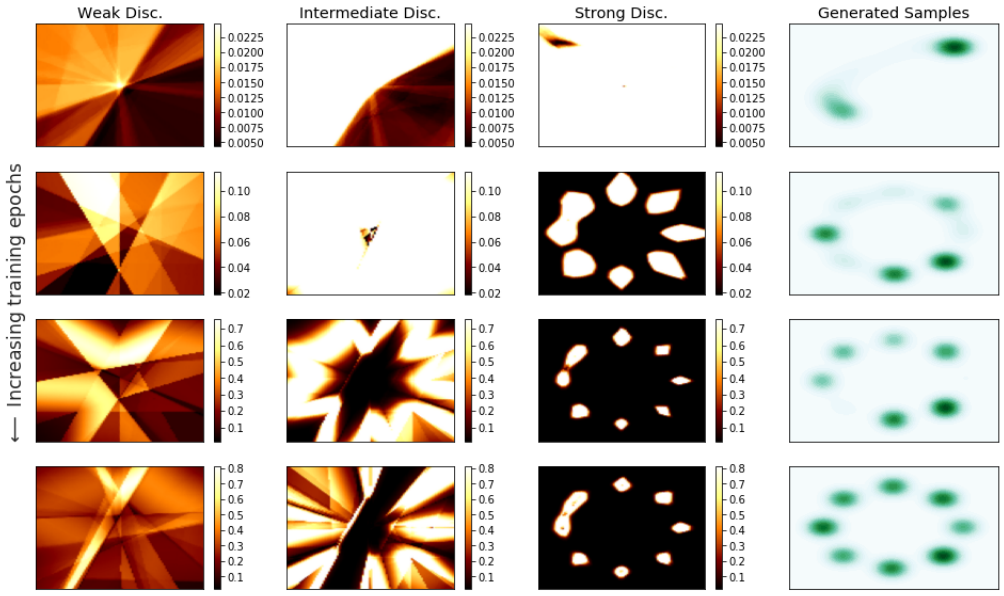


Figure 1.14: Evolution of the gradient norm for each discriminator and samples generated (last column). The generator recovers modes thanks to the gradients provided by the weak and intermediate discriminators. Each discriminator in turn evolves to learn its coarse to fine-grained representation of the data. Note also that the strong discriminator has a good representation of all the modes before the generator has learned them, indicating that mode dropping in this setting is not due to those modes being absent in the discriminator. We have clipped the gradient range with respect to the weak discriminator of the corresponding row.

1.6.3 Regularizing the discriminator through additive noise

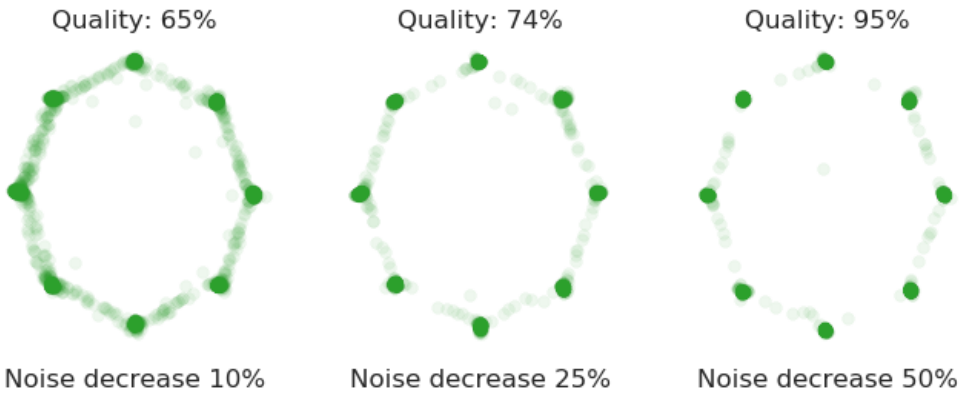


Figure 1.15: As we exponentially decay the noise, samples quality increase (2500 samples are plotted).

Increasing entropy with additive Gaussian noise. In order to have discriminator networks with varying degrees of strength, we first resorted to nested architectures: for instance, the stronger discriminator should have a more complex architecture than the weaker. Moreover, we proceeded to corrupt the inputs with additive Gaussian white noise. Formally, to the input matrix $X_i \in \mathcal{X}$ of the discriminator D_i we added $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$, thus creating new input $Y_i = X_i + \varepsilon_i$ which was then fed to the discriminator. For practical purposes, noise for image data should be on a bounded support $R(\varepsilon_i) \subseteq R(X_i)$ in order to obtain meaningful RGB values.

Letting the weaker discriminators train on inputs corrupted with a Gaussian noise with larger variance allows the network to learn a high-level representation of the dataset, while feeding uncorrupted inputs will let the corresponding D_i to specialize. This trade-off between sample space coverage and estimation accuracy is known as the spike-and-slab prior and is frequently used in Bayesian variable selection methods similar to the one proposed by [68].

Consider the following relation, known in information theory as the entropy power inequality (EPI). Let X be a continuous, real-valued and independent random vector on a

bounded support and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, both of dimension n:

$$e^{\frac{2}{n}h(X+\varepsilon)} \geq e^{\frac{2}{n}h(X)} + e^{\frac{2}{n}h(\varepsilon)} \,. \tag{1.8}$$

Applying logarithm on both sides and using $\log(a+b) = \log(a) + \log(1+\frac{b}{a})$, we get an expression for the entropy of the sum X and ε:

$$\begin{aligned}
\frac{2}{n}h(X+\varepsilon) &\geq \log(e^{\frac{2}{n}h(X)} + e^{\frac{2}{n}h(\varepsilon)}) \\
&= \frac{2}{n}h(X) + \log(1 + e^{\frac{2}{n}(h(\varepsilon)-h(X)}) \\
&\geq \frac{2}{n}h(X) + \log(1 + e^{\frac{2}{n}(\log(\sigma\sqrt{2\pi e})-\log(b-a)}) \\
&\geq \frac{2}{n}h(X) + \log\left(1 + \left[\frac{\sigma\sqrt{2\pi e}}{(b-a)}\right]^{\frac{2}{n}}\right) \\
&\geq \frac{2}{n}h(X),
\end{aligned} \tag{1.9}$$

from which it follows that adding i.i.d. Gaussian noise to the inputs increases the total entropy of the data. Here we used the fact that $\log(1+\exp(x)) \geq 0$ for all $x \in \mathbb{R}$ and that the uniform distribution U has the maximal entropy over all continuous random variables with bounded support $R(U) = (a, b)$. The quantity $\log \frac{\sigma\sqrt{2\pi e}}{(b-a)}$ controls the tightness of the bound. Because Eq. 1.9 is valid for all σ, it is necessary valid for $\sup_{\sigma>0} \log \frac{\sigma\sqrt{2\pi e}}{(b-a)}$. Maximizing the expression shows that picking $\sigma >> 0$ increases the overall entropy $h(X+\varepsilon)$ and approaches the uniform distribution.

Finally, recall that the entropy $h(X) = -D_{KL}(X||U) + c$ where $U \sim$ Uniform(a, b). That is, maximizing the entropy of a distribution is equivalent up to an additive constant to minimizing the Kullback-Leibler divergence between the distribution and a uniform random variable with identical support (provided adequate restrictions on the support).

Fitting a weak discriminator to the corrupted data should increase its capacity to generalize more than that of the stronger discriminator by acting as a regularization technique and preventing the network from overfitting.

Analogous mechanisms are widely used in conjunction with other learning algorithms,

such as support vector machines, where adding noise to the data is equivalent to increasing the classification margin as shown by [100].

As a final remark, it is important to select a proper noise distribution in order to avoid introducing bias and respect the original structure of the data.

1.6.4 Experimental parameters

Algorithm parameters	
	acGAN
α	0.01
λ	15
T_{warmup}	$15 \times N$
N	number of discriminators
Optimizer parameters	
Stacked-MNIST	RMSprop ($\alpha = 0.9, l_r = 0.0001$)
CIFAR-10	Adam ($\beta_1 = 0.5, \beta_2 = 0.999, l_r = 0.0002$)
Synthetic (25 Gaussians)	Adam ($\beta_1 = 0.5, \beta_2 = 0.999, l_r = 0.0002$)
Synthetic (8 Gaussians)	Adam ($\beta_1 = 0.5, \beta_2 = 0.999, l_r = 0.0001$)
CelebA	Adam ($\beta_1 = 0.5, \beta_2 = 0.999, l_r = 0.0002$)

Table 1.4: General experimental hyperparameters.

1.6.5 Synthetic data

We utilize the 2D-ring with 8 Gaussians and the 2D-grid with 25 Gaussians [30]. Three metrics were employed to evaluate the results:

1. $\%$ High Quality samples

2. Number of Covered modes

3. Fréchet Distance (FD)

The percentage of "High Quality" samples is defined as the proportion of generated samples $G(z)$ which are within 3 standard deviation of the closest mode. The next metric reported is the number of modes covered, i.e. the count of modes that has generated samples closes enough (3σ). The Fréchet Distance originally from [18] is defined as:

$$\text{FD} = ||m_d - m_g||^2 + \text{Tr}(C_d + C_g - 2(C_d C_g)^{\frac{1}{2}}), \tag{1.10}$$

where m_d, C_d and m_g, C_g are first and second order moments of the real data distributions and estimates from generated data, respectively.

Architecture. The generator network's architecture comprises 4 dense layers of 512 units each. We used 3 discriminators with respectively 2,3 and 4 dense layers of 512 units. ReLU activations were used in all layers, except for the last one, where a linear activation function was used for the generator and a sigmoid for the discriminator.

Stacked-MNIST

Architecture. We used DCGAN's architecture [79] to create lower capacity discriminators (in terms of feature representation power). For the 3Ds case, we used discriminators 3, 4, 5 (described in the following tables). For the 5Ds case, we used discriminators 1, 2, 3, 4, 5.

Layer	Outputs	Kernel size	Stride	BN	Activation
Input: $z \sim \mathcal{N}(0, I_{100})$					
Fully connected	2*2*512	4, 4	2, 2	Yes	ReLU
Transposed convolution	4*4*256	4, 4	2, 2	Yes	ReLU
Transposed convolution	8*8*128	4, 4	2, 2	Yes	ReLU
Transposed convolution	14*14*64	4, 4	2, 2	Yes	ReLU
Transposed convolution	28*28*3	4, 4	2, 2	No	Tanh

Table 1.5: Generator's architecture.

Layer	Outputs	Kernel size	Stride	BN	Activation
Input	28*28*3				
Convolution	14*14*64	4, 4	2, 2	No	LeakyReLU
Convolution	7*7*128	4, 4	2, 2	Yes	LeakyReLU
Convolution	4*4*256	4, 4	2, 2	Yes	LeakyReLU
Convolution	2*2*512	4, 4	2, 2	Yes	LeakyReLU
Convolution	1	4, 4	2, 2	No	Sigmoid

Table 1.6: Discriminator 5.

Layer	Outputs	Kernel size	Stride	BN	Activation
Input	28*28*3				
Convolution	14*14*64	4, 4	2, 2	No	LeakyReLU
Convolution	7*7*128	4, 4	2, 2	Yes	LeakyReLU
Convolution	4*4*256	4, 4	2, 2	Yes	LeakyReLU
Convolution	2*2*512	4, 4	2, 2	Yes	LeakyReLU
Fully connected	1	4, 4	2, 2	No	Sigmoid

Table 1.7: Discriminator 4.

Layer	Outputs	Kernel size	Stride	BN	Activation
Input	28*28*3				
Convolution	13*13*64	6, 6	2, 2	No	LeakyReLU
Convolution	6*6*128	6, 6	2, 2	Yes	LeakyReLU
Convolution	2*2*256	6, 6	2, 2	Yes	LeakyReLU
Convolution	1	6, 6	2, 2	No	Sigmoid

Table 1.8: Discriminator 3.

Layer	Outputs	Kernel size	Stride	BN	Activation
Input	28*28*3				
Convolution	13*13*64	6, 6	2, 2	No	LeakyReLU
Convolution	6*6*128	6, 6	2, 2	Yes	LeakyReLU
Convolution	2*2*256	6, 6	2, 2	Yes	LeakyReLU
Fully connected	1	6, 6	2, 2	No	Sigmoid

Table 1.9: Discriminator 2.

Layer	Outputs	Kernel size	Stride	BN	Activation
Input	28*28*3				
Convolution	12*12*64	8, 8	2, 2	No	LeakyReLU
Convolution	4*4*128	8, 8	2, 2	Yes	LeakyReLU
Convolution	1	8, 8	2, 2	No	Sigmoid

Table 1.10: Discriminator 1.

(a) acGAN - 3 disc. **(b)** acGAN - 5 disc.

Figure 1.16: Stacked-MNIST generated samples.

1.6.6 CIFAR-10

FID score. FID scores, as introduced in [36], were computed for CIFAR-10. It is defined as the squared Fréchet distance between the Gaussian having the first and second order statistics matching those obtained from image features. The late layers of a pretrained classifier are used as low dimensional representation of images for statistics estimation.

Architecture. For our strongest discriminator we use the DCGAN architecture but with halved the number of filter, i.e. $\{64, 128, 256, 512\}$. For the 3D case, we introduced two extra discriminators with kernel sizes of 6 and 8. For the 5D case, we add two discriminators with kernel sizes 4 and 6 respectively to the set of 3D discriminator networks. In both 3D and 5D, we replaced the last layer from the DCGAN model with a fully connected dense layer. The generator network was taken from the original DCGAN architecture but with halved filter sizes too, i.e. $\{512, 256, 128, 64\}$. ReLU activation units were used for the generator network while LeakyRelu is used for the discriminators with a coefficient of 0.2.

Influence of the number of discriminators. An important assumption in the current paper is that increasing the number of discriminator networks helps the model converge faster. To assess that, we conducted experiments with the acGAN algorithm while varying the number of discriminators for $N \in \{1, 3, 5\}$ ($N = 1$ being the Vanilla GAN) and averaging results over 5 seeds. According to Fig. 1.17, we see that a higher number of discriminators indeed leads to earlier convergence of the FID score curve.

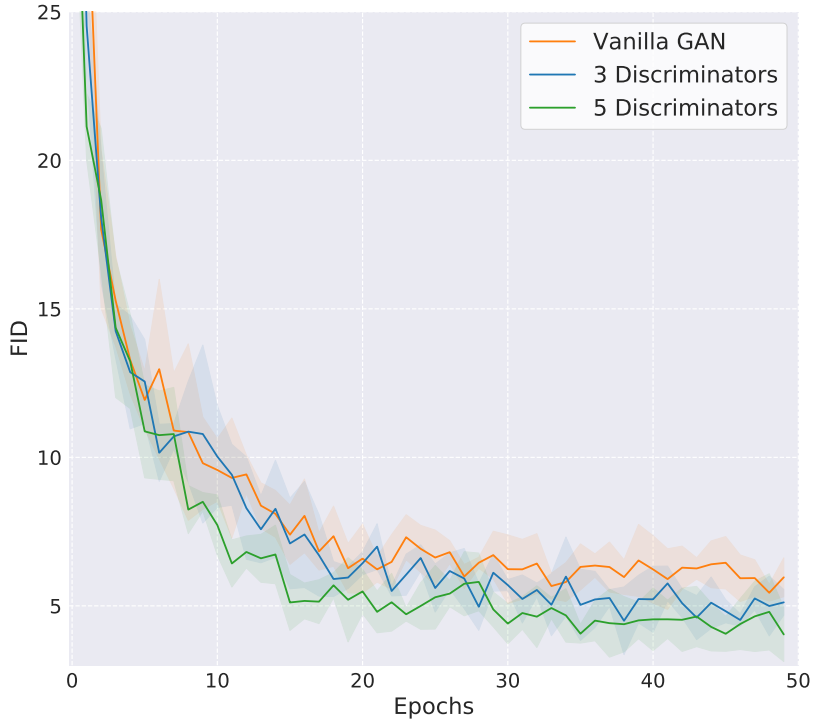
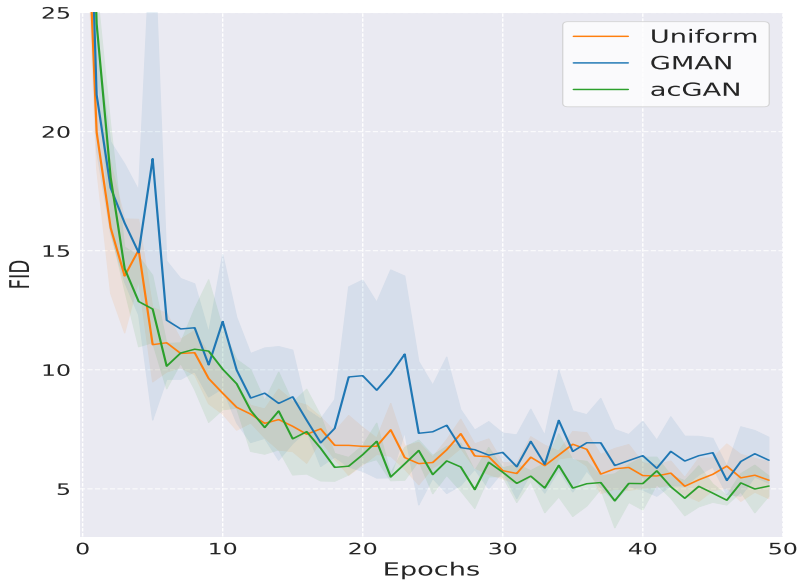
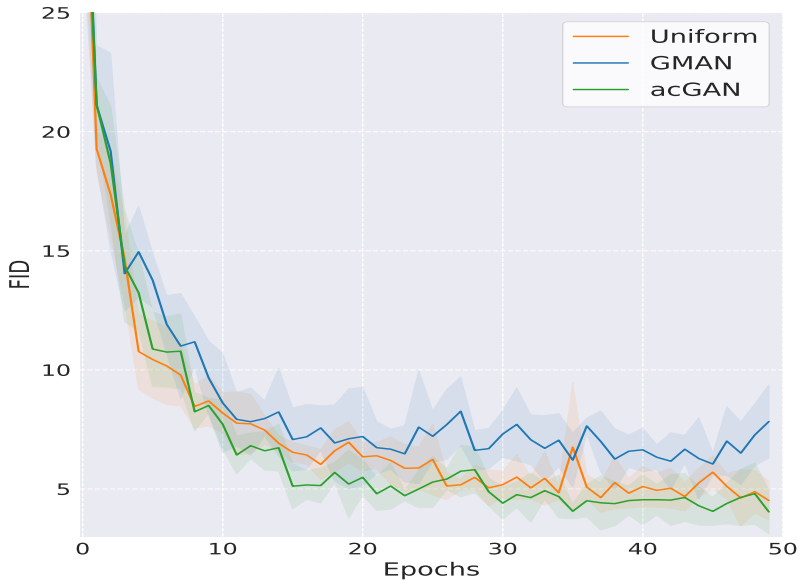


Figure 1.17: Increasing the number of discriminators induces an earlier convergence of FID. Moreover, lower FID values are reached.



(a) 3 Discriminators



(b) 5 Discriminators

Figure 1.18: Average FID score of each method for different number of discriminators. In both plots, the acGAN algorithm presented faster convergence compared to the other methods.

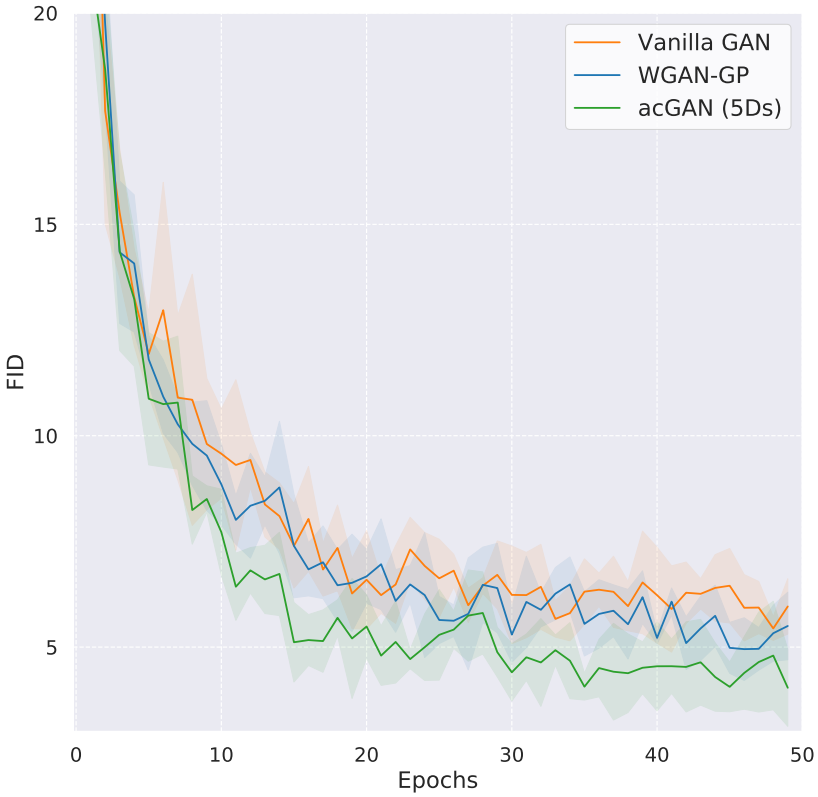
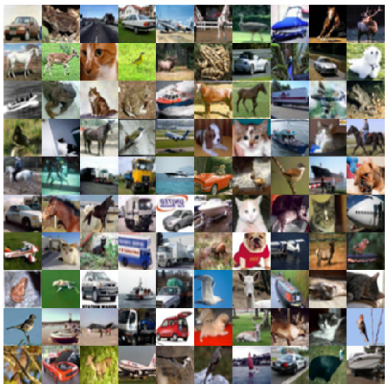
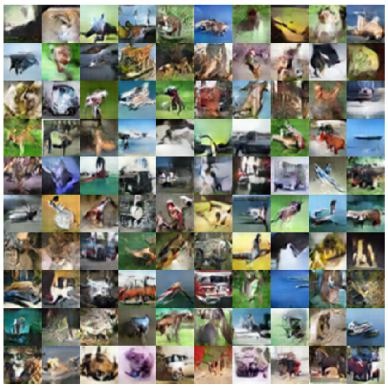


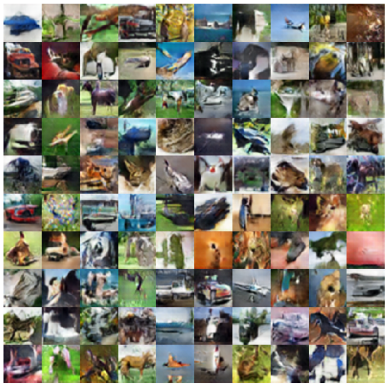
Figure 1.19: acGAN with 5 discriminators shows earler convergence and better performance than Vanilla GAN (1 Disc) and WGAN-GP.



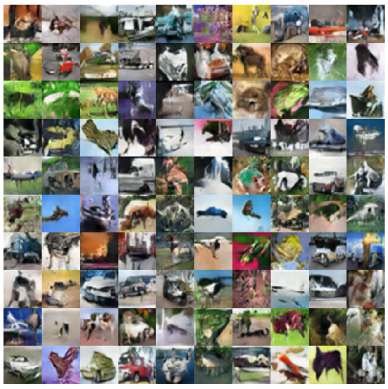
(a) Real Images



(b) Vanilla GAN



(c) Uniform - 3 disc.

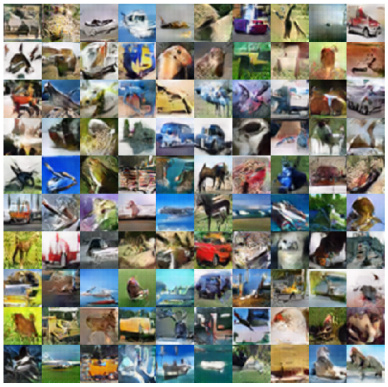


(d) Uniform - 5 disc.

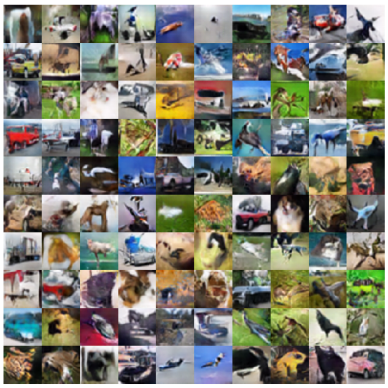
(e) GMAN - 3 disc.

(f) GMAN - 5 disc.

Figure 1.20: CIFAR-10 generated samples (1).



(a) acGAN - 3 disc.



(b) acGAN - 5 disc.

Figure 1.21: CIFAR-10 generated samples (2).

1.6.7 CelebA

For both the CelebA [60] datasets, we conducted single-run experiments of 50,000 iterations each counted in generator steps (≈ 15 epochs). We downscaled the original images to 64×64 pixels out of practical concerns.

Results. Similarly to CIFAR-10, we observe the emergence of a curriculum in Fig. 1.24. In particular, we note the presence of alternating phases during which a specific discriminator is dominating in the 3D and 5D cases. In the end, all discriminator probabilities converge to a stationary (*i.e.* long term) uniform distribution just like for previously mentioned datasets.

Architecture. We used the same architecture as for the CIFAR-10 experiments except that the original numbers of filters were set to: $\{128, 256, 512, 1024\}$ for the discriminators and $\{1024, 512, 256, 128\}$ for the generator.

(a) Real Images

(b) Vanilla GAN

(c) Uniform - 3 disc.

(d) Uniform - 5 disc.

Figure 1.22: CelebA generated samples (1).

(a) GMAN - 3 disc.

(b) GMAN - 5 disc.

(c) acGAN - 3 disc.

(d) acGAN - 5 disc.

Figure 1.23: CelebA generated samples (2).

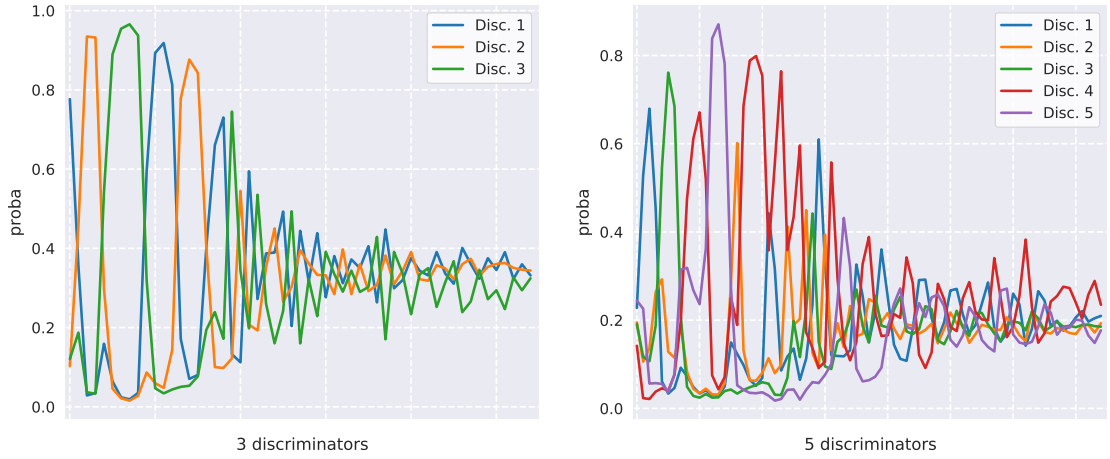
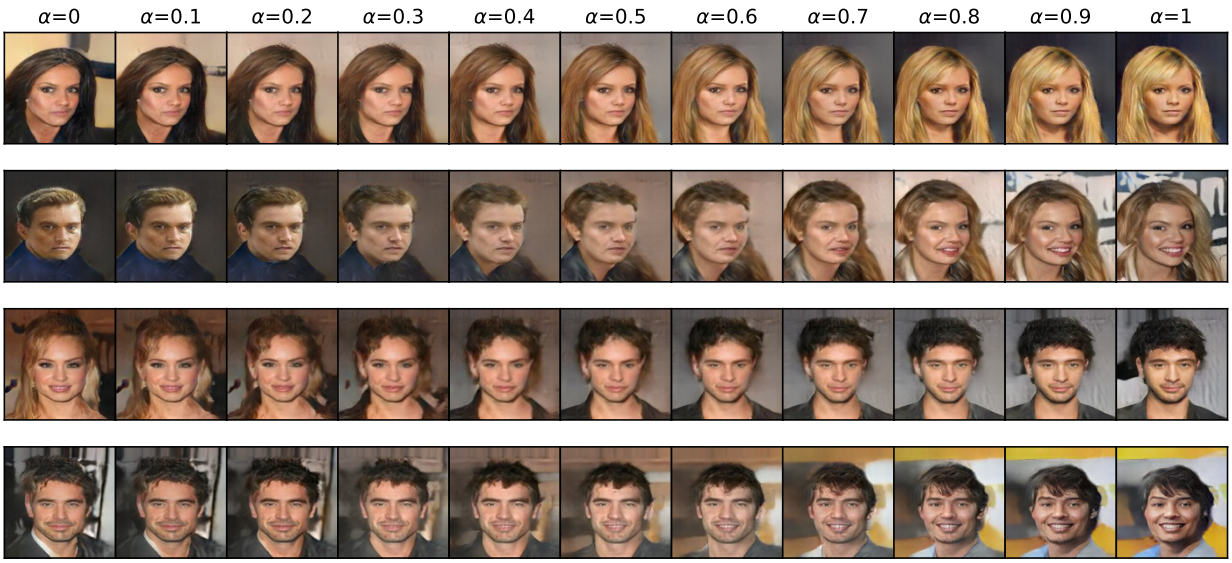
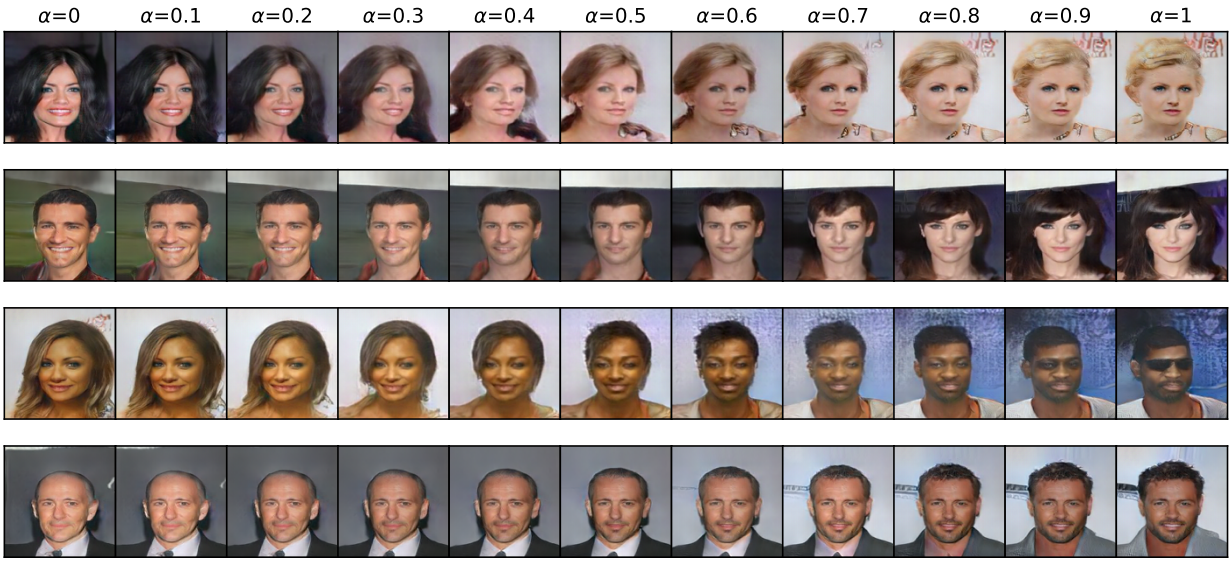


Figure 1.24: Weight π_i of each discriminator over the training epochs. We could see switching phase, where one discriminator's weight π_i is dominant with respect to the rest. After some epochs, all weights π_i converge to a uniform regime.

Generating 128x128 images. In this experiment, we generated high resolution images with 3 and 5 discriminators. A convolutional layer with 2048 feature maps was added to both generator and discriminators architectures. The 3 discriminators settings used a kernel size of 4,6 and 8. For the 5 discriminators case, we added a discriminator of kernel size 4 and 6 but replaced the last layer with dense layers. The same parameters was employed as for CelebA 64x64.



(a) acGAN - 3 discriminators

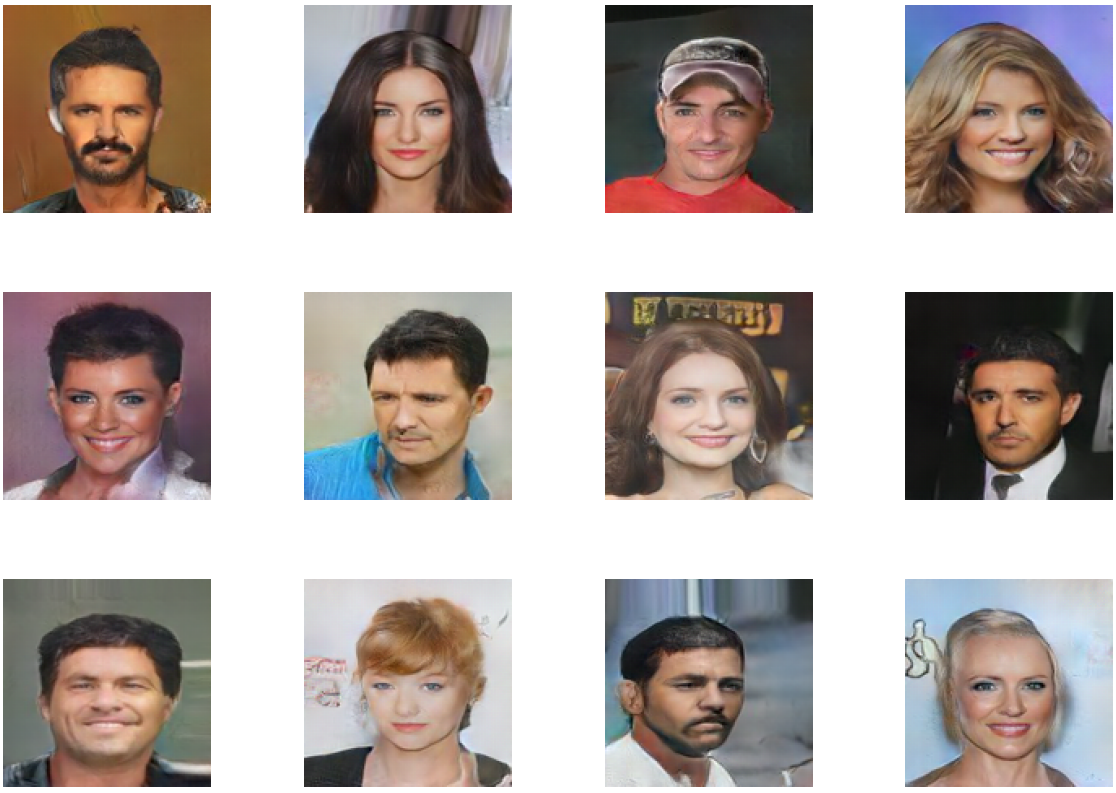


(b) acGAN - 5 discriminators

Figure 1.25: Interpolating in latent space with 3 and 5 Discriminators.

(a) acGAN - 3 discriminators

Figure 1.26: 128x128 CelebA samples for acGAN trained for 50 epochs with 3 discriminators.



(a) acGAN - 5 discriminators

Figure 1.27: 128x128 CelebA samples for acGAN trained for 50 epochs with 5 discriminators.

Chapter 2
Generating Realistic Sequence of Customer-level for Transactions for Retail Datasets

2.1 Introduction

Modern retailers collect, store and utilize massive amounts of consumer behaviour data through their customer loyalty programs. Sources such as customer-level transactional data, customer profiles, and product attributes allow the retailer to better service their customers by utilizing data mining techniques for customer relationship management (CRM) and direct marketing systems [57]. Better data mining techniques for CRM databases can allow retailers to understand their customers more effectively, leading to increased loyalty, better service and ultimately increased sales.

Modelling customers is a complex problem with many facets. First, a retailer's loyalty data provides a censored view into a customer's behaviour because it only shows the transactions for a single retailer, leading to noisy observations. Second, the sequential nature of consumer purchases adds additional complexity as changes in behaviour and long term dependencies need to be taken into account. Finally, the large number of customers (100M+) multiplied by the catalog of products (100K+) results in a vast amount of transactional data, but it is simultaneously very sparse at the level of individual customers. These complexities make modelling customers a difficult problem even with modern techniques.

Indirect approaches to modelling customer behaviour for specific tasks have been widely studied. Techniques that utilize customer-level transactional data such as customer lifetime value [15], recommendations [47, 101], and incremental sales [78], formulate these tasks as supervised learning problems. More direct approaches to modelling

customers have been through simulators. There are a wide variety of applications for customer marketing simulators from aiding in decision support [86] to understanding how behavioural phenomena affect consumer decisions [102]. Another notable application of customer simulators is in the context of direct marketing activities [1, 73, 88, 93]. These methods use the customer simulator to understand individual-level interactions between a customers and a marketing agent. Typically, the goal is to find an ideal marketing policy to maximize a pre-defined reward over time. However, the primary focus of this work has been on techniques for generating an optimal marketing policy with less focus on generating realistic simulations of customer transactional data.

Generative modelling methods [27] have proven to be very successful in learning realistic distributions from the data in many different contexts. A relevant recent work by [52] presents a technique to generate realistic orders from an e-commerce dataset. They provide a method to effectively learn the complex relationships between customer and product to generate realistic simulations of customer orders, but do not take into account how customer behaviour changes over time in their method.

In this work, we present a novel method to generate realistic sequences of customer-level baskets of products over time using a customer-level retail transactions dataset. Our technique is able to generate samples of both customers and traces of their transaction baskets over time. This general formulation of the customer modelling problem allows one to essentially generate new customer-level transactional datasets that retain most of the distributional properties of the original data. This opens up possibilities for new applications such as generating a distribution of likely products to be purchased by an individual customer in the future to derive insights for better service, or by providing external researchers with access to generated data for a source dataset that otherwise would be restricted due to privacy concerns.

The proposed method uses a multi-step approach to generating customer-level transactional data using using a combination of Generative Adversarial Networks (GAN) [27] and Recurrent Neural Networks (RNN) [39]. First, we train a RNN to generate a cus-

tomer embedding by using a multi-task learning approach. The inputs to the RNN are product embeddings derived from their textual descriptions. This allows one to describe the customer state given their previous transactions. Next, to determine the number of products in the next basket, we extract a sample based on historical basket sizes of similar customers. A GAN trained by conditioning on a customer embedding at the current time is used to predict the next product in a basket for a given customer. This is repeated until all products in the basket are filled. This provides a single customer-level transaction basket. Finally, the generated products are fed back into the RNN to generate the next state of the customer and the process repeats.

Evaluation of GANs and generative models are difficult in general [92, 103] especially for non-visual domains. We demonstrate the effectiveness of the technique via several qualitative and quantitative metrics. We first show that the generator can reproduce the relative frequencies of various product features including types, brands, and prices to within a 5% difference. We further show that the generated data retains most of the strongest sequential patterns between products in the real data set. Finally, we show that most of the real and generated baskets are indistinguishable, with a classifier trained to separate the two being able to achieve an accuracy of only 63% at the category level.

2.2 Background and Related Work

2.2.1 Transaction-Based Item and Customer Embeddings

Learning a representation of customers from their transactional data is a common problem in retail data mining. Borrowing inspiration from Natural Language Processing (NLP), different methods try to embed customers into a common vector space based of their transaction sequences. For instance, [72] and [10] learn the embeddings by adapting the Paragraph Vector-Distributed Bag-of-Words or the n-skip-gram models from [67]. The underlying idea behind these methods is that by solving an intermediate task such as predicting the next word in a sentence or the next item a customer will purchase, one

can learn general–purpose features that are meaningful and have good predictive power for a wide variety of tasks.

For example, [15] examines the lifetime value of a customer in the context of an e-commerce website. Towards that end, they also use an n-skip-gram model to learn customer embeddings and track its evolution over time as purchases are made. [8] uses a stacked denoising autoencoder to learn customer embeddings for improving their campaign decisions or clustering clients into classes.

2.2.2 Item Prediction and Recommendation Systems

Various techniques from recommendation systems such as collaborative filtering [47, 83] have long been used to predict a customer's preference for items, although usually they are not directly predicting a customer's next purchase.

More recent advancements in deep learning have shown to be quite practical in modelling a customer's next purchase over time. Techniques such as [99] mimic a recurrent neural network (RNN) by feeding historical transaction data as input to a neural network which predicts the next item. [101] and [6] both use a RNN to predict the next basket of items to great effect.

2.2.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [27] are a class of generative models aimed at learning a distribution. The method is founded on the game theoretical concept of two-player zero-sum games, wherein two players each try to maximize their own utility at the expense of the other player's utility. By formulating the distribution learning problem as such a game, a GAN can be trained to learn good strategies for each player. A generator G aims to produce realistic samples from this distribution while a discriminator D tries to differentiate fake samples from real samples. By alternating optimization steps between the two components, the generator ultimately learns the distribution of the real data.

In detail, the generator network $G : Z \to X$ is a mapping from a high-dimensional noise space $Z = \mathbb{R}^{d_z}$ to the input space X on which a target distribution f_X is defined. The generator's task consists of fitting the underlying distribution of observed data f_X as closely as possible. The discriminator network $D : X \to \mathbb{R} \cap [0, 1]$ scores each input with the probability that it belongs to the real data distribution f_X rather than the generator G.

The classical GAN optimization algorithm minimizes the Jensen-Shannon divergence (JS) between the real and generated distributions. However, [4] suggests replacing the JS metric by the Wasserstein-1 or Earth-Mover divergence. We make use of an improved version of this algorithm, the Wasserstein GAN (WGAN) with Gradient Penalty [30]. Its objective is given below:

$$\min_G \max_D \mathop{\mathbb{E}}_{x \sim f_X(x)} [D(x)] + \mathop{\mathbb{E}}_{x \sim G(z)} [-D(x)] + p(\lambda), \tag{2.1}$$

where $p(\lambda) = \lambda(||\nabla_{\tilde{x}} D(\tilde{x})|| - 1)^2,\ \tilde{x} = \varepsilon x + (1 - \varepsilon)G(Z)$, $\varepsilon \sim \text{Uniform}(0, 1)$, and $Z \sim f_Z(z)$. Setting $\lambda = 0$ recovers the original WGAN objective.

2.2.4 Simulating Customer Behaviour

A customer's state with respect to a given retailer (*i.e.* the types of products they are interested in and the amount they are willing to spend) evolves over time, and there exist a wide variety of techniques used to model this state. In marketing research, agent-based approaches such as [86, 102] have aided in building simple simulations of how customers interact and make decisions.

Data mining and machine learning approaches to model a customer's state in the context of direct marketing activities have also been widely studied [93]. Techniques such as [1, 73, 88] model the problem in the reinforcement learning framework by attempting to learn the optimal marketing policy to maximize rewards over time. As part of their work, they use various techniques to represent and simulate the customer's state over

time. However, the method does not use the customer's state to generate its future orders, but rather consider it more narrowly in the context of the defined reward.

More recently, [52] was the first to generate plausible customer e-commerce orders for a given product using a Generative Adversarial Network (GAN). Given a product embedding, [52] generates a tuple containing a product embedding, customer embedding, price, and date of purchases, which summarizes a typical order. The e-commerce GAN has applications in providing insights into product demand, customer preferences, price estimation and seasonal variations by simulating what are likely potential orders. However, it only generates realistic orders and does not directly model customer behaviour over time.

2.3 Methodology

In this section, we present a novel methodology for generating realistic sequences of future transactions for a given customer. The proposed pipeline involves a GAN module and an LSTM module intertwined in a sequence of product generation and customer state updating steps. The GAN is trained to generate a basket of products conditioned on a time-sensitive customer representation, while the LSTM models the sequential nature of the customer's state as it interacts with products. Each of these components uses semantic embeddings of customers and products for representational purposes, which are defined in the first two subsections, while the training of the GAN and generation of customer transactions are presented in the latter two subsections.

2.3.1 Product Representations

To capture the semantic relationships between products that exist independently of customer interactions, we learn product representations based on their associated textual descriptions. Specifically, a corpus is created wherein a sentence is defined for each product as the concatenation of the product name and description. Preprocessing is applied

to remove stopwords and other irrelevant tokens. The resulting corpus contains $11,443$ products and a vocabulary size of $21,894$ words. The word2vec skipgram model [67] is then trained on this corpus using a context window size of 5 and an embedding dimensionality of 128. Finally, each product representation is defined as the arithmetic mean of the word embeddings in the product's name and description. This is similar to the common practice of representing a sentence by the mean of word vectors within the sentence [54], and is motivated by the observation that sums of word vectors produce semantically meaningful vectors.

2.3.2 Customer Representations

To characterize customers by their purchasing habits we learn customer embedding representations from their transactional data. Inspired by [52], this is accomplished using a Long-Short Term Memory (LSTM) [38] module. The LSTM takes as input a sequence of transaction baskets for a given customer, where each transaction basket is defined by a set of product embeddings for a week of purchase. Products within the same basket are ordered randomly during training. The LSTM is trained to learn the customer's sequential patterns via a multi-task optimization procedure. Specifically, the LSTM output is fed as inputs for the following three prediction tasks:

1. Predict whether or not a product is the last product in the basket.
2. Predict the category of the next product.
3. Predict the price of the next product.

The LSTM is trained to maximize the performance of all three subtasks by randomly and uniformly sampling a single task in each step and optimizing for this task. After convergence, the hidden state of the LSTM is used to characterize a customer's purchasing habits, and thus a customer's state. As a result, customers with similar behaviour will be closer together in the resulting embedding space. Figure 2.1 illustrates the process of learning this embedding.

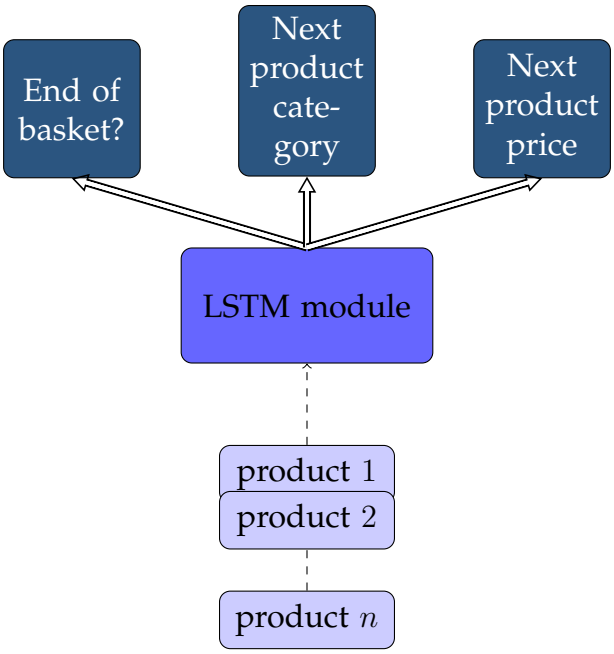


Figure 2.1: Embedding customers via multi-task Learning with an LSTMs. The input is the sequence of products a customer has purchased throughout their transactional history. After convergence, the hidden state of the LSTM will characterize a customer's state.

2.3.3 Learning Product Distributions with a Conditional GAN

To learn the product distributions, we use a conditional Wasserstein GAN [4]. In the optimization process the discriminator and generator are involved in a min-max game. In this game the discriminator aims to maximize the following loss function:

$$\max_{D} \mathop{\mathbb{E}}_{x\sim f_X(x)}[D(x|(h,w)] + \mathop{\mathbb{E}}_{x\sim G(z|(h,w))}[-D(x|(h,w)] \\ +\lambda(||\nabla_{\tilde{x}} D(\tilde{x}|(h,w)|| - 1)^2, \tag{2.2}$$

where λ is a penalty coefficient, $\tilde{x} = \varepsilon x + (1-\varepsilon)G(z|(h,w))$, and $\varepsilon \sim$ Uniform$(0,1)$. The first term is the expected score (which can be thought of as likelihood) of seeing an product x being purchased by the given customer and week (h,w). The second term is the score of seeing product z being purchased by that same customer and week, (h,w). Taken together, these first two terms encourage the discriminator to maximize the ex-

pected score of the real products $x \sim f_X(x)$ given the context (h, w) and minimize the score of the generated products $x \sim G(z|(h, w))$. The third term in Eq. 2.2 is a regularization penalty to ensure that D satisfies the 1-Lipschitz conditions.

The generator is trained to minimize the following loss function:

$$\max_{G} \mathop{\mathbb{E}}_{x \sim G(z|(h,w))} [D(x|(h, w)] \tag{2.3}$$

Intuitively, this objective aims to maximize the likelihood that the generated product $x \sim G(z|(h, w))$ is plausible given the context (h, w), where plausibility is determined by the discriminator $D(x|(h, w))$. With successive steps of optimization we obtain a G which will generate samples that are more similar to the real data distribution.

While the generator learned from Eq. 2.3 can yield realistic product embeddings, in practice one may wish to obtain specific instances from a database $P = \{p_i\}_{i=1}^{n}$ of known products. This can be useful, for instance, to obtain a product recommendation for customer h at week w. Given a generated product embedding $G(z|(h, w))$, this can be accomplished by computing the closest product from the database according to the L_2 distance metric: $p = \text{argmin}_{p_i \in P} ||G(z|(h, w)) - p_i||_2^2$. Note that other distance metrics such as cosine distance could also be used for this purpose.

2.3.4 Generating Sequences of Products

In this subsection we develop a pipeline to generate a sequence of baskets of products that a customer will likely purchase over several consecutive weeks. The pipeline incorporates the product generator G to produce individual products in the basket as well as the LSTM module to model the evolution of a customer's state over a sequence of baskets.

The procedure works as follows. Given a new customer with a transaction history $B_1, B_2, \ldots, B_i$, where each B_i denotes a basket for week w_i and $i \geq 1$, we wish to generate a basket B_{i+1} for the following week. We extract the customer embedding at week w_i, denoted h_i, by passing the transaction sequence through the LSTM module and extracting

the hidden state. We then find the k most similar customers from the database of known customers by L_2 distance from h_i. This is similar to the process of retrieving known products from a database as described in the previous section. We then determine the number of products to generate for him/her in week w_i. To accomplish this we uniformly sample from the basket sizes of the k most similar customers' baskets to retrieve the number of products to generate, n_i. The generator network is then used to generate n_i products via our generator, $G(h_i, w_i)$.

This procedure can be extended to generate additional baskets by feeding B_{i+1} back into the LSTM, whose hidden state is updated as if the customer had purchased B_{i+1}. The updated customer representation h_{i+1} is once again used to estimate the basket size n_{i+1} and fed into the generator $G(h_{i+1}, w_{i+1})$ which yields a basket of products for the week w_{i+1}. This cycle can be iterated multiple times to generate basket sequences of arbitrary length, or alternatively generate multiple sequences of baskets for the same customer. The procedure is described in detail by the pseudo-code in Algorithm 2 and illustrated in Figure 2.2. Note that all values in Algorithm 2 are also indexed by the customer index c which has been omitted in this discussion for the brevity. To simplify the notation, we also use the symbol B_0^c in Algorithm 2 to denote the entire history of customer c.

In this manner we can effectively augment a new customer's transaction data by predicting their actions for an arbitrary amount of time. The intuition behind the approach is that a customer's embedding representation evolves as they purchase products, and therefore might share some common properties with other customers through their purchase experience. One can derive insights from this generated data by learning a better characterization of their distribution of likely purchase sequences into the future.

2.4 Experimental Results

In this section we empirically demonstrate the effectiveness of the proposed methodology by comparing the generated basket data against real customer data. Evaluation is first

Algorithm 2 Sequence of basket generation

Input: LSTM L, generator G, set of historical basket sequences for each customer $\{B_0^c\}_{c=1}^C$, hyperparameter k, number of weeks W
for $c = 1, \ldots, C$ **do**
  Compute initial customer embedding h_0^c via $L(B_0^c)$
  for $w = 1, \ldots, W$ **do**
    Sample n_w^c via k-nearest customers of h_w^c
    Generate basket B_w^c of n_w^c products from $G(h_w^c, w)$
    Update the customer embedding with the LSTM: $h_{w+1}^c = L(B_w^c, h_w^c)$.
  end for
end for

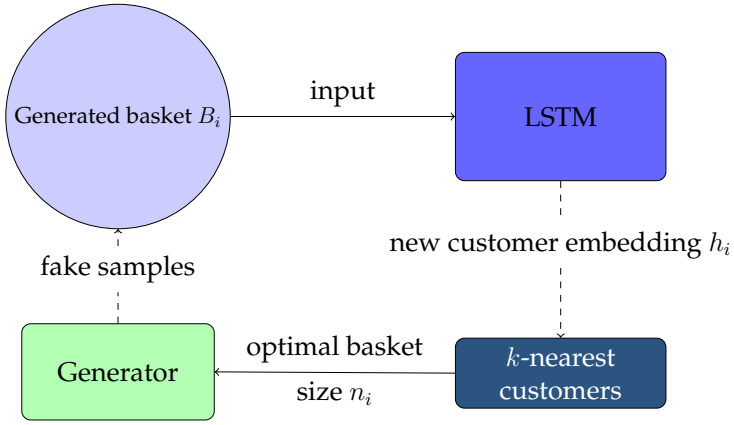


Figure 2.2: Basket sequence generation process using the LSTM and Generator modules.

performed with respect to the distributions of key metrics aggregated over the entire data sets, including product categories, brands, prices, and basket sizes. Next we compare sequential patterns that exist between products in both data sets, and finally we examine the separability between the real and generated baskets with multiple different basket representations.

2.4.1 Experimental Setup

The basket generation methodology is evaluated using a data set from an industrial partner which consists of 742,686 transactions over a period of 5 weeks during the summer of 2016. This data is composed of 174,301 customer baskets with an average size of 4.08

products and price of $\$12.2$. A total of 7,722 distinct products and 66,000 distinct customers exist across all baskets.

Figure 2.3 shows the product embedding representations extracted from textual descriptions as described in Section 2.3.1 projected into a 2-dimensional space using the t-SNE algorithm [97]. Products are classified into functional categories such as Hair Styling, Eye Care, etc, each of which corresponds to a different color in Figure 2.3. We observe that products from the same category tend to be clustered close together, which reflects the semantic relationships between such products. At a higher level we observe that similar product categories also occur in close proximity to one another; for example the categories of Hair Mass, Hair Styling and Hair Designer are mapped to adjacent clusters, as are the categories of Female Fine Frag and Male Fine Frag. This property is critical to the basket generation scheme which directly generates only product embeddings, while instances of specific products are obtained based on their proximity to other products in the embedding space.

The LSTM is trained on this data set with a multi-task optimization procedure as described in section 2.3.2 (see Figure 2.1) for 25 epochs. For each customer, we obtain an embedding from the LSTM hidden state after passing through all of their transactions. These embeddings are then used to train the conditional GAN. The GAN was trained for 100 epochs using the Adam [50] optimizer with the hyperparameters values of $\alpha = 0.5$ and $\beta = 0.9$. The discriminator is composed of two hidden layers of 256 units each with ReLU activation functions, with the exception of the last layer which is free of activation functions. The generator uses the same architecture except for the last layer which has a tanh activation function. During training the discriminator is prioritized by applying five update steps for each update step to the generator. This helps the discriminator converge faster so as to better guide the generator.

Once the LSTM and GAN are trained we run our basket sequence generation. For each customer, we generate 5 weeks of baskets following the procedure in Algorithm 2.

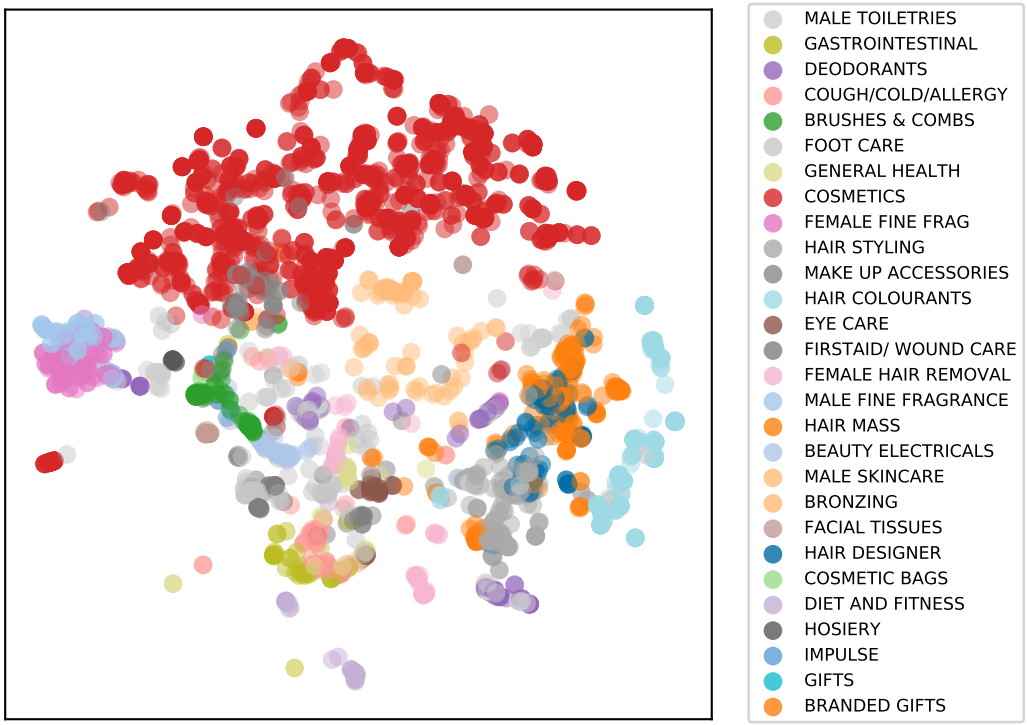


Figure 2.3: Visualization of product embeddings in a 2D space (mapped using t-SNE)

2.4.2 Feature Distributions

Figures 2.4, 2.5, 2.6, and 2.7 compare the frequency distributions of the categories, brand, prices, and basket sizes, respectively, between the generated and real baskets. Additional metrics are provided in Tables 2.1 and 2.2. Note that for the brand, we restrict the histogram plots to include only the top 30 most frequent brands. We observe that in general our generative model can reasonably replicate the ground-truth distribution. This is further evidenced by Table 2.2 which indicates that the highest absolute difference in frequency of generated brands is 5.6%. The lowest discrepancy occurs for the category feature, where the maximum deviation is 3.2% in the generated products. In addition, the generated basket size averages 3.85 products versus 4.08 for the real data which is

a difference of approximately 5%. The generated product prices are an average of $\$3.1$ versus $\$3.4$ for the real data (a 10% difference). This demonstrates that the generation methodology can mimic the aggregated statistics of the real data to a reasonable degree. Note that we should not expect the two distributions to match exactly because we are projecting each customer's purchases into the future, which won't necessarily have the same distributive properties.

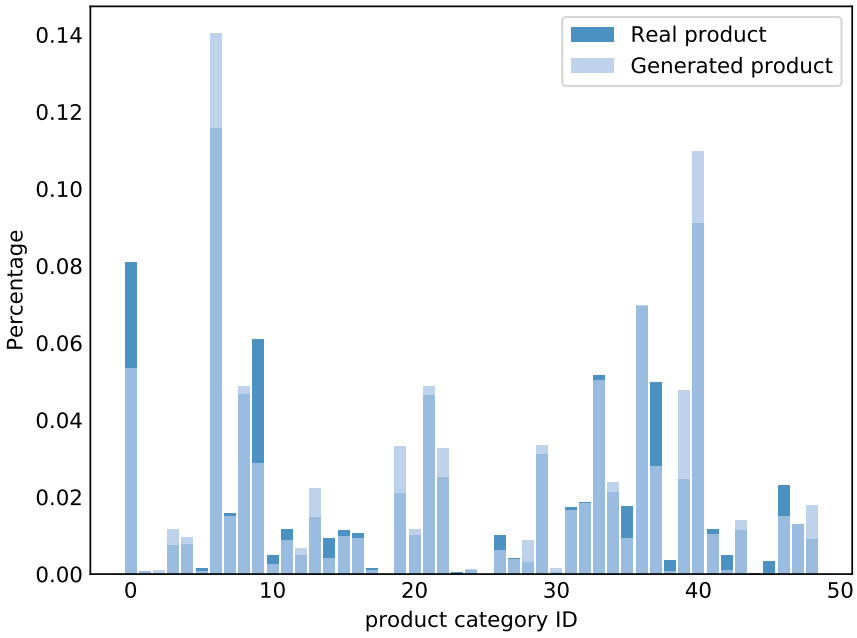


Figure 2.4: Product category distributions of real and generated data

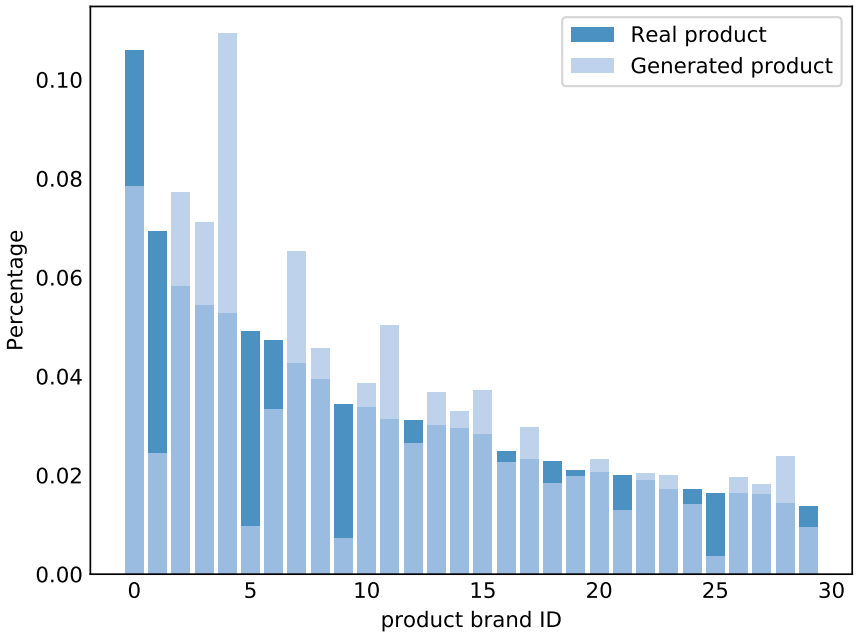


Figure 2.5: Product brand distributions of real and generated data

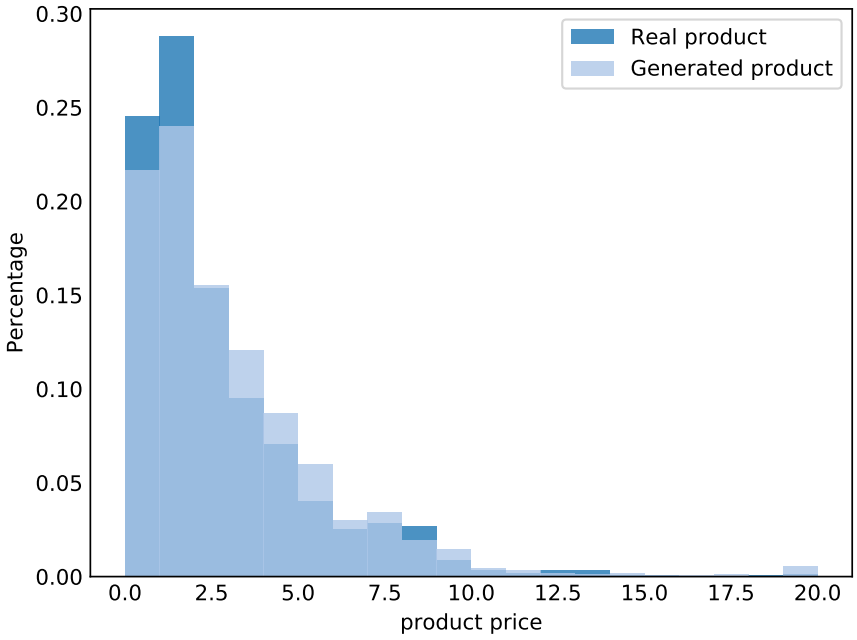


Figure 2.6: Product price distributions of real and generated data

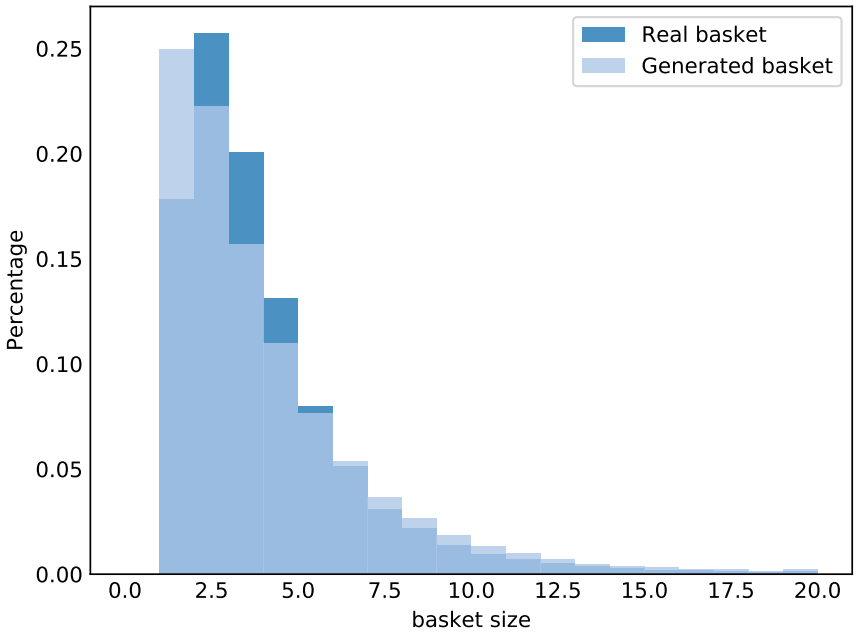


Figure 2.7: Basket size distributions of real and generated data

	Real Transactions	Generated Transactions
Average basket size	4.08	3.85
Average basket price	$3.1	$3.4

Table 2.1: Statistics

Table 2.2: Discrepancies between real and generated data

Criterion	Max absolute deviation (in %)
Category	3.2%
Brand	5.6%
Price	5.2%
Basket size*	4.1%

* this metric only applies for basket size ≤ 20

2.4.3 Sequential Pattern Mining

Sequential pattern mining [26] (SPM) is a technique to discover statistically relevant subsequences from a sequence of sets ordered by time. One frequent application of SPM is

Sequence	Real support	Generated support
Hemorrhoid relief, Skin treatment & dressings	0.045	0.098
Skin treatment & dressings, Female fine frag	0.029	0.100
Facial moisturizers, Skin treatment & dressings	0.028	0.075
Shower products, Female fine frag	0.028	0.056
Hemorrhoid relief, Female fine frag	0.028	0.093
Skin treatment & dressings, Facial moisturizers	0.027	0.076
Skin treatment & dressings, Preg test & ovulation	0.027	0.082
Shower products, Skin treatment & dressings	0.026	0.056
Hemorrhoid relief, Preg test & ovulation	0.026	0.075
Female fine frag, Preg test & ovulation	0.025	0.081
Facial moisturizers, Hemorrhoid relief	0.025	0.069
Skin treatment & dressings, Skin treatment & dressings, Hemorrhoid relief	0.007	0.014

Table 2.3: Sequential patterns comparison between real and generated transaction data

in retail transactions where we wish to determine subsequences of items across baskets customers have bought over time. For example, given an set of baskets a customer has purchased ordered by time: {*milk, bread*}, {*cereal, cheese*}, {*bread, oatmeal, butter*}, one sequential pattern we can derive is: {*milk*}, {*bread, butter*} because {*milk*} in the first basket comes before {*bread, butter*} in the last basket. A pattern is typically measured by its *support*, which is defined as the number of customers containing the pattern as a subsequence. We refer the reader to [26] for further details.

For this set of experiments, sequential pattern mining is performed on the real and generated datasets via the SPFM [23] library using a minimum support of 1% of the total number of customers. Figure 2.8 plots the percentage of the top-k most common real sequential patterns that are also found in the generated data as k varies from 1 to 1000. Here items are defined at either the category or subcategory level, so that two products are considered equivalent if they belong to the same functional grouping. We see that for the category-level, we can recover 98% of the top-100 patterns, while at the subcategory-level, we can recover 63%. This demonstrates that our method is generating plausible sequences of baskets for customers because most of the real sequential patterns show up in the generated data. Not all patterns are found however, which might imply that the generated data might have some drift in the sequences due to the method of projecting customer's purchases into the future.

Table 2.3 shows examples of the top sequential patterns of length 2 and 3 from the real data at the subcategory level that also appeared in the generated transactional data. The two right columns show the support for both the real and generated datasets, which is normalized by dividing by the total number of customers. We can see that the generated data has higher support for the patterns from generated data, indicating that it may have an easier time replicating common patterns.

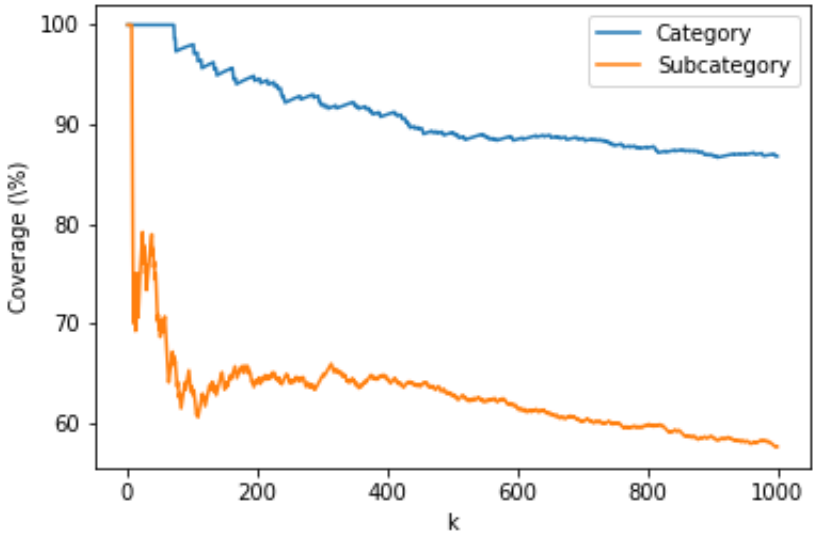


Figure 2.8: Generated pattern coverage of the top-k most common real patterns

2.4.4 Basket Distributions

In this section we directly compare the generated and real baskets based on the products they contain. For each basket of products $B_i = \{p_{i,j}\}_{j=1}^{|B_i|}$ we a define a vector representation v_i using a bag-of-products scheme. Let P denote the set of all known products. Then v_i is a $|P|$-dimensional vector with $v_i^{(j)} = 1$ if $p_j \in B_i$ or $v_i^{(j)} = 0$ otherwise. P can be defined at various levels of precision such as the product serial number, the brand, or the category levels. At the category level, for instance, two products would be considered equivalent and correspond to the same index j if they belong to the same category.

The resulting vectors are then projected into two dimensions using t-SNE for visualization purposes. The distributions of the real and generated data are plotted in Figures 2.9. For an alternative viewpoint Figure 2.10 plots the vectors projected using Principal Component Analysis (PCA). These plots qualitatively indicate that the distributions match quite closely.

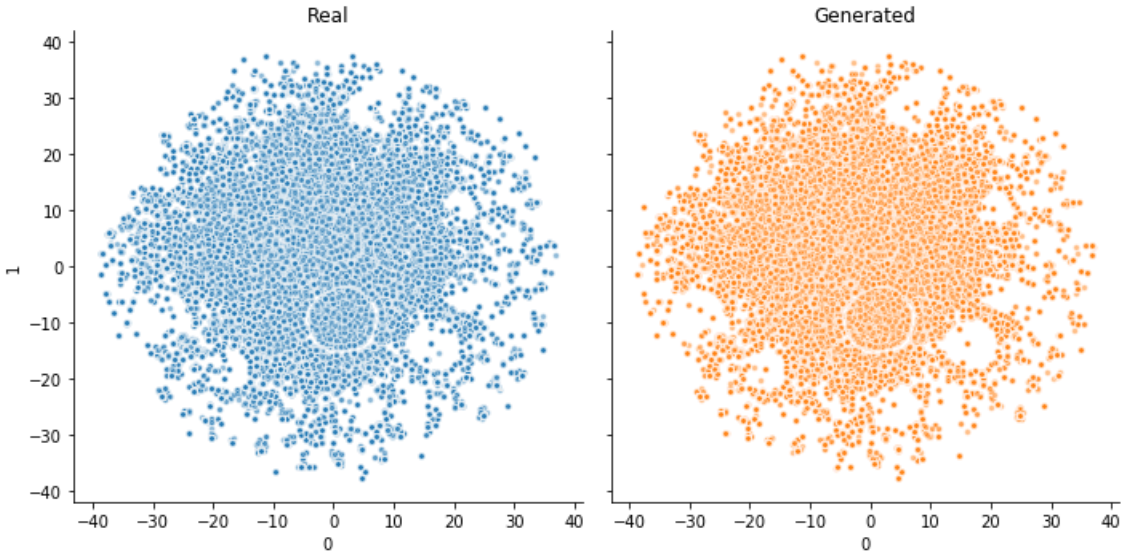


Figure 2.9: Basket representations as bags-of-products vectors at the category level, projected using t-SNE.

This observation can be further analyzed quantitatively by training a classifier to distinguish between points from the two distributions. By measuring the prediction accuracy of this classifier we obtain an estimate of the degree of separability between the data sets. For our experiments we randomly sample a subset of the generated points such that the number of real and generated points are equal. This way a perfectly indistinguishable generated data set should yield a classification accuracy of 50%. We note that this classification task is fundamentally unlike that which is performed by the discriminator during the GAN training, as the latter operates on the embedding representation of a single product while the former operates on the bag-of-items representation of a basket.

The results are given in Table 2.4 using a logistic regression classifier. Each row corresponds to a different level of granularity in the definition of the bag-of-products represen-

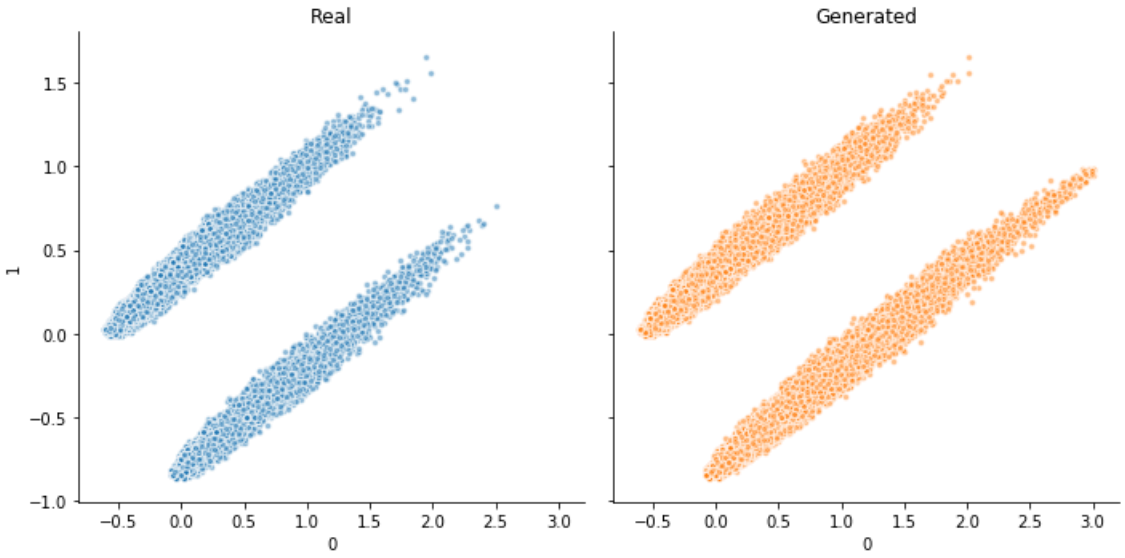


Figure 2.10: Basket representations as bags-of-products vectors at the category level, projected using PCA.

tation, with category being the most coarse–grained and sku being the most fine–grained. We see that the classifier performs quite poorly at the category levels, meaning that the generated baskets of categories are quite plausible.

However, note that the bag-of-products representation does not preserve the semantic similarity between products in that any two products with different skus are perfectly separable even if they have very similar functions and descriptions. Therefore, we instead define the sku level basket representation as the mean of embeddings of the products in the basket. This is given in the last row of Table 2.4. Note that these representations come from the embeddings of the nearest neighbor product rather than the output of the generator. As expected, the classification accuracy is still quite low considering the fine-grained level at which the prediction occurs.

Basket Representation	Classification Accuracy
Bag-of-products category	0.634
Bag-of-products subcategory	0.663
Basket embedding sku-level	0.704

Table 2.4: Separability between real and generated baskets.

2.5 Conclusion

In this paper, we propose a novel method of generating sequences of realistic customer baskets for customer-level transactional data. After learning customer embeddings with an LSTM, we generate a product basket conditioned on the customer embedding using the generator from the GAN. The generated basket of products is fed back into the LSTM to generate a new customer embedding, and the process repeats. We show that the proposed methods can replicate to a reasonable degree the statistics of the real data distribution (category, brand, price and basket size). As additional experiments, we verified that common sequential patterns exist between products in the generated and real data, and that the generated orders are difficult to distinguish from the real orders.

Chapter 3
Attraction-Repulsion Actor-Critic for Continuous Control Reinforcement Learning

3.1 Introduction

Many reinforcement learning (RL) tasks such as robots and self-driving cars pose a major challenge due to large action and state spaces [55]. In particular, environments with large non-convex continuous action spaces are prone to *deceptive* rewards, i.e. local optima [17]. Applying traditional policy optimization algorithms to these domains often leads to locally optimal, yet globally sub-optimal policies. This implies that learning should involve some form of exploration.

Not all RL domains that require exploration are suitable for understanding how to train agents that are robust to deceptive rewards. For example, Montezuma Revenge, a game in the Atari Learning Environment [12], has *sparse rewards*; algorithms that perform the best on this task encourage exploration by providing learning signal to the agent [90].

On the other hand, continuous control problems, such as MuJoCo [94], already provide the agent with a dense reward signal. Yet, the high-dimensional action space induces a multimodal (potentially deceptive) reward landscape. Such domain properties can lead the agent to sub-optimal policies. For example, in the biped environments, coordinating both arms and legs is crucial for performing well on even simple tasks such as forward motion. However, simply learning to maximize the reward can be detrimental in the long run: agents will tend to run and fall further away from the start point instead of discovering a stable walking motion. Exploration in this setting serves to provide a more reliable learning signal for the agent by covering more different types of actions during learning.

One way to maximize action space coverage is the maximum entropy RL framework [104], which prevents variance collapse by adding a policy entropy auxiliary objective. One such prominent algorithm, Soft Actor-Critic (SAC, [33]), has been shown to excel in large continuous action spaces. To further improve on exploration properties of SAC, one can maintain a population of agents that cover non-identical sections of the policy space. To prevent premature convergence, a diversity-preserving mechanism is typically put in place; balancing the objective and the diversity term becomes key to converging to a global optimum [40]. This paper studies a particular family of population-based exploration methods, which conduct coordinated local search in the policy space. Prior work on population-based strategies improves performance on continuous control domains through stochastic perturbation on a single actor's parameter [76] or a set of actor's parameters [17,49,59]. We hypothesize that exploring directly in the policy space is more important than perturbing the parameters of the policy, as the latter does not guarantee diversity (i.e., different neural network parameterizations can approximately represent the same function).

Given a population of RL agents, we enforce local exploration using an Attraction-Repulsion (AR) mechanism. The later consists in adding an auxiliary loss to encourage pairwise attraction or repulsion between members of a population, as measured by a divergence term. We make use of the Kullback-Leibler (KL) divergence because of its desirable statistical properties and its easiness of computation. However, naively maximizing the KL term between two Gaussian policies can be detrimental (e.g. drives both means apart). As a result, we parametrize the policy with a general family of distributions called Normalizing Flows (NF) [80]; this modification allows to improve upon AR+Gaussian (see Appendix Figure 3.6). NFs are shown to improve the expressivity of the policies using invertible mappings while maintaining entropy guarantees [65,91]. The AR objective blends particularly well with SAC, since computing the KL requires stochastic policies with tractable densities for each agent.

3.2 Preliminaries

We first formalize the RL setting in a Markov decision process (MDP). A discrete-time, finite-horizon, MDP [13,77] is described by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}^+$, and a reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$.[1] On each round t, an agent interacting with this MDP observes the current state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, and observes a reward $r(s_t, a_t) \in \mathbb{R}$ upon transitioning to a new state $s_{t+1} \sim \mathcal{P}(s_t, a_t)$. Let $\gamma \in [0, 1]$ be a discount factor. The goal of an agent evolving in a discounted MDP is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ such as taking action $a_t \sim \pi(\cdot|s_t)$ would maximize the expected sum of discounted returns,

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s\right].$$

In the following we use ρ_{π} to denote the trajectory distribution induced by following policy π. If $\mathcal{S}$ or $\mathcal{A}$ are vector spaces, action and space vectors are respectively denoted by $\vec{a}$ and $\vec{s}$.

3.2.1 Discovering new solutions through population-based Attraction-Repulsion

Consider evolving a population of M agents, also called *individuals*, $\{\pi_{\theta_m}\}_{m=1}^{M}$, each agent corresponding to a policy with its own parameters. In order to discover new solutions, we aim to generate agents that can mimic some target policy while following a path different from those of other policies.

Let $\mathcal{G}$ denote an archive of policies encountered in previous generations of the population. A natural way of enforcing π to be different from or similar to the policies contained

[1] $\mathcal{A}$ and $\mathcal{S}$ can be either discrete or continuous.

in $\mathcal{G}$ is by augmenting the loss of the agent with an Attraction-Repulsion (AR) term:

$$\mathcal{L}_{\text{AR}} = -\mathop{\mathbb{E}}_{\pi' \sim \mathcal{G}} \left[\beta_{\pi'} \mathrm{D}_{\text{KL}}[\pi || \pi'] \right], \tag{3.1}$$

where π' is an archived policy and $\beta_{\pi'}$ is a coefficient weighting the relative importance of the Kullback-Leibler (KL) divergence between π and π' which we will choose to be a function of the average reward (see Sec. 3.3.2 below). Intuitively, Eq. 3.1 adds to the agent objective the average distance between the current and the archived policies. For $\beta_{\pi'} \geq 0$, the agent tends to move away from the archived policy's behavior (i.e. *repulsion*, see Figure 3.1) a). On the other hand, $\beta_{\pi'} < 0$ encourages the agent π to imitate π' (i.e. *attraction*).

Requirements for AR In order for agents within a population to be trained using the proposed AR-based loss (Eq. 3.1), we have the following requirements:

1. Their policies should be stochastic, so that the KL-divergence between two policies is well-defined.

2. Their policies should have tractable distributions, so that the KL-divergence can be computed easily, either with closed-form solution or Monte Carlo estimation.

Several RL algorithms enjoy such properties [33, 84, 85]. In particular, the soft actor-critic [33] is a straightforward choice, as it currently outperforms other candidates and is off-policy, thus maintains a single critic shared among all agents (instead of one critic per agent), which reduces computation costs.

3.2.2 Soft actor-critic

SAC [33] is an off-policy learning algorithm which finds the information projection of the Boltzmann Q-function onto the set of diagonal Gaussian policies Π:

$$\pi = \underset{\pi' \in \Pi}{\operatorname{argmin}} \, \mathrm{D}_{\mathrm{KL}} \left(\pi'(.|\vec{s}_t) \middle\| \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\vec{s}_t, .)\right)}{Z^{\pi_{\text{old}}}(\vec{s}_t)} \right),$$

where $\alpha \in (0, 1)$ controls the temperature, i.e. the peakedness of the distribution. The policy π, critic Q, and value function V are optimized according to the following loss functions:

$$\mathcal{L}_{\pi,\text{SAC}} = \mathbb{E}_{\vec{s}_t \sim \mathcal{B}}[\mathbb{E}_{\vec{a}_t \sim \pi}[\alpha \log \pi(\vec{a}_t|\vec{s}_t) - Q(\vec{s}_t, \vec{a}_t)]] \tag{3.2}$$

$$\mathcal{L}_Q = \mathop{\mathbb{E}}_{(s,a,r,s') \sim \mathcal{B}} \left[\{Q(s,a) - (r + \gamma V_\nu^\pi(s'))\}^2 \right] \tag{3.3}$$

$$\mathcal{L}_V = \mathbb{E}_{\vec{s}_t \sim \mathcal{D}} \left[\frac{1}{2} \left\{ V_\nu^\pi(\vec{s}_t) - \mathbb{E}_{\vec{a}_t \sim \pi}[Q(\vec{s}_t, \vec{a}_t) - \alpha \log \pi(\vec{a}_t|\vec{s}_t)] \right\}^2 \right], \tag{3.4}$$

where $\mathcal{B}$ is the replay buffer. The policy used in SAC as introduced in [33] is Gaussian, which is both stochastic and tractable, thus compatible with our AR loss function in Eq. 3.1. Together with the AR loss in Eq. 3.1, the final policy loss becomes:

$$\mathcal{L}_\pi = \mathcal{L}_{\pi,\text{SAC}} + \mathcal{L}_{\text{AR}} \tag{3.5}$$

However, Gaussian policies are arguably of limited expressibility; we can improve on the family of policy distributions without sacrificing qualities necessary for AR or SAC by using Normalizing Flows [80].

3.2.3 Normalizing flows

NFs [80] were introduced as a means of transforming simple distributions into more complex distributions using learnable and invertible functions. Given a random variable $\vec{z}_0$

with density q_0, they define a set of differentiable and invertible functions, $\{f_i\}_{i=1}^N$, which generate a sequence of d-dimensional random variables, $\{\vec{z}_i\}_{i=1}^N$.

Because SAC uses explicit, yet simple parametric policies, NFs can be used to transform the SAC policy into a richer one (e.g., multimodal) without risk loss of information. For example, [65] enhanced SAC using a family of radial contractions around a point $\vec{z}_0 \in \mathbb{R}^d$,

$$f(\vec{z}) = \vec{z} + \frac{\beta}{\alpha + ||\vec{z} - \vec{z}_0||_2}(\vec{z} - \vec{z}_0) \tag{3.6}$$

for $\alpha \in \mathbb{R}^+$ and $\beta \in \mathbb{R}$. This results in a rich set of policies comprised of an initial noise sample $\vec{a}_0$, a state-noise embedding $h_\theta(\vec{a}_0, \vec{s}_t)$, and a flow $\{f_{\phi_i}\}_{i=1}^N$ of arbitrary length N, parameterized by $\phi = \{\vec{\phi}_i\}_{i=1}^N$. Sampling from the policy $\pi_{\phi,\theta}(\vec{a}_t|\vec{s}_t)$ can be described by the following set of equations:

$$\begin{aligned} \vec{a}_0 &\sim \mathcal{N}(0, \mathbf{I}); \\ \vec{z} &= h_\theta(\vec{a}_0, \vec{s}_t); \\ \vec{a}_t &= f_{\phi_N} \circ f_{\phi_{N-1}} \circ ... \circ f_{\phi_1}(\vec{z}), \end{aligned} \tag{3.7}$$

where $h_\theta = \vec{a}_0 \vec{\sigma} \mathbf{I} + \mu(\vec{s}_t)$ depends on the state and the noise variance $\sigma > 0$. Different SAC policies can thus be crafted by parameterizing their NFs layers.

3.3 ARAC: Attraction-Repulsion Actor-Critic

We now detail the general procedure for training a population of agents using the proposed diversity-seeking AR mechanism. More specifically, we consider here SAC agents enhanced with NFs [65]. Figure 3.1 displays the general flow of the procedure. Algorithm 3 (Appendix.) provides the pseudo-code of the proposed ARAC strategy, where sub-procedures for rollout and archive update can be found in the Appendix.

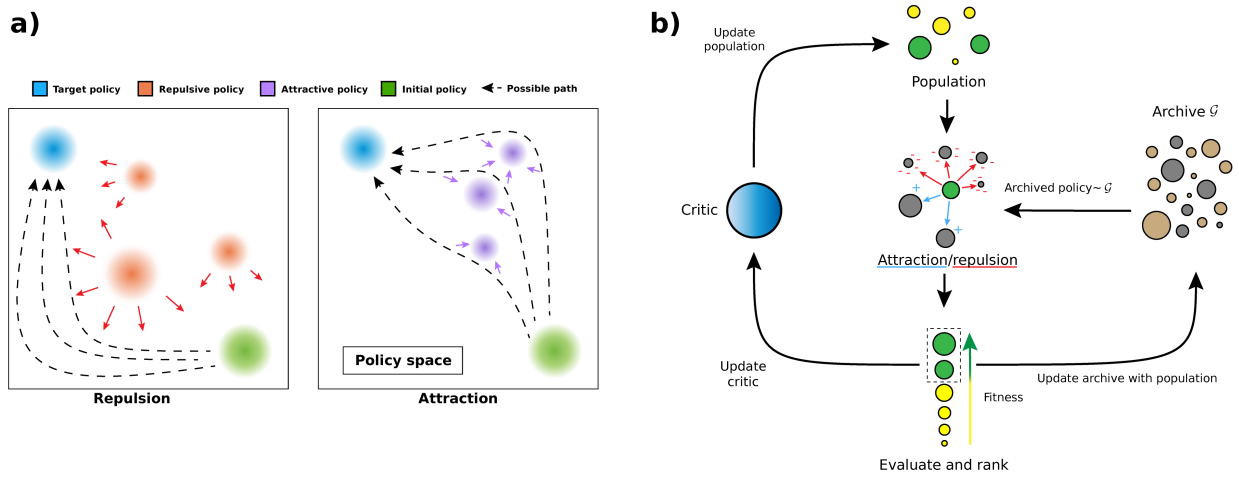


Figure 3.1: a) Augmenting the loss function with AR constraints allows an agent to reach a target policy by following different paths. Attractive and Repulsive policies represent any other agent's policy. **b)** General flow of the proposed ARAC strategy.

Overview ARAC works by evolving a population of M SAC agents $\{\pi^m_{\phi,\theta}\}^M_{m=1}$ with radial NFs policies (Eq. 3.7) and shared critic Q_ω, and by maintaining an archive of policies encountered in previous generations of the population. After performing T steps per agent on the environment (Alg. 3 L8-12), individuals are evaluated by performing R rollouts[2] on the environment (Alg. 3 L26-28). This allows to identify the top-K best agents (Alg. 3 L29), also called *elites*, which will be used to update the critic as they provide the most meaningful feedback (Alg. 3 L13-17). The archive is finally updated in a diversity-seeking fashion using the current population (Alg. 3 L30).

The core component of the proposed approach lies within the update of the agents (Alg. 3 L18-25). During this phase, elite individuals are updated using AR operations w.r.t. policies sampled from the archive (Eq. 3.5), whereas non-elites are updated regularly (Eq. 3.2).

3.3.1 Enhancing diversity in the archive

Throughout the training process, we maintain an archive $\mathcal{G}$ of maximum capacity G, which contains some previously encountered policies. The process goes as follow: un-

[2]These steps can be performed in parallel.

til reaching full capacity, the archive saves a copy of the parameters of every individual in the population after the evaluation step. However, by naively adding all individuals as if the archive were just a heap, the archive could end up filled with policies leading to similar rewards, which would result in a loss of diversity [64]. We mitigate this issue by keeping track of two fitness clusters (low and high) using the partition formed by running a k-means algorithm on the fitness value. Hence, when $|\mathcal{G}| = G$ is reached and a new individual is added to the archive, it randomly replaces an archived policy from its respective cluster. This approach, also known as *niching*, has proved itself effective at maintaining high diversity levels [32, 62].

3.3.2 Discovering new policies through Attraction-Repulsion

The crux of this work lies in the explicit search for diversity in the policy space achieved using the AR mechanism. Since the KL between two base policies (i.e. input of the first flow layer) can be trivially maximized by driving their means apart, we apply attraction-repulsion only on the flow layers, while holding the mean of the base policy constant. This ensures that the KL term doesn't depend on the difference in means and hence controls the magnitude of the AR mechanism. Every time the AR operator is applied (Alg. 3 L20-21), n policies are sampled from the archive and are used for estimating the AR loss (Eq. 3.1). As in [40], we consider two possible strategies to dictate the value of $\beta_{\pi'}$ coefficients for policies $\pi' \sim \mathcal{G}$:

$$\beta_{\pi'} = -\left[2\left(\frac{f(\pi') - f_{min}}{f_{max} - f_{min}} - 1\right)\right] \qquad \text{(proactive)} \tag{3.8}$$

$$\beta_{\pi'} = 1 - \frac{f(\pi') - f_{min}}{f_{max} - f_{min}} \qquad \text{(reactive)} \tag{3.9}$$

where $f(\pi)$[3] represents the fitness function of policy π (average reward in our case), and f_{min} and f_{max} are estimated based on the n sampled archived policies. The *proactive* strategy aims to mimic high reward archived policies, while the *reactive* strategy is more cau-

[3]We overload our notation f for both the normalizing flow and the fitness depending on the context

tious, only repulsing away the current policy from low fitness archived policies. Using this approach, the current agent policy will be attracted to some sampled policies ($\beta_{\pi'} < 0$) and will be repulsed from others ($\beta_{\pi'} \geq 0$) in a more or less aggressive way, depending on the strategy.

Unlike [40] who applied proactive and reactive strategies on policies up to 5 timesteps back, we maintain an archive consisting of two clusters seen so far: policies with low and high fitness, respectively. Having this cluster allows to attract/repulse from a set of diverse agents, replacing high-reward policies by policies with similar performance. Indeed, without this process, elements of the archive would collapse on the most frequent policy, from which all agents would attract/repulse. To avoid performing AR against a single "average policy" , we separate low-reward and high-reward agents via clustering.

3.4 Related Work

The challenges of exploration are well studied in the RL literature. Previously proposed approaches for overcoming hard exploration domains tend to either increase the capacity of the state-action value function [25, 35] or the policy expressivity [65, 91, 96]. This work rather tackles exploration from a diverse multi-agent perspective. Unlike prior population-based approaches for exploration [17, 49, 76], which seek diversity through the parameters space, we directly promote diversity in the policy space.

The current work was inspired by [40], who relied on the KL divergence to attract and repulse from a set of previous policies to discover new solutions. However, in their work, the archive is time-based (they restrict themselves to the 5 most recent policies), while our archive is built following a diversity-seeking strategy (i.e., niching and policies come from multiple agents). Notably, ARAC is different of previously discussed works in that it explores the action space in multiple regions simultaneously, a property enforced through the AR mechanism.

The proposed approach bears some resemblance with [59], who took advantage of a multi-agent framework in order to perform repulsion operations among agents using of similarity kernels between parameters of the agents. The AR mechanism gives rise to exploration through structured policy rather than randomized policy. This strategy has also been employed in multi-task learning [31], where experience on previous tasks was used to explore on new tasks.

3.5 Experiments

3.5.1 Didactic example

Consider a 2-dimensional multi-armed bandit problem where the actions lie in the real square $[-6, 6]^2$. We illustrate the example of using a proactive strategy where a SAC agent with radial flows policy imitates a desirable (expert) policy while simultaneously repelling from a less desirable policy. The task consists in matching the expert's policy (blue density) while avoiding taking actions from a repulsive policy π' (red). We illustrate the properties of radial flows in Figure 3.2 by increasing the number of flows (where 0 flow corresponds to a Gaussian distribution).

We observe that increasing the number of flows (bottom to top) leads to more complex policy's shapes and multimodality unlike the Gaussian policy which has its variance shrinked (indeed, the KL divergence is proportional to the ratio of the two variances, hence maximizing it can lead to a reduction in the variance which can be detrimental for exploration purpose). Details are provided in Supplementary Material.

3.5.2 MuJoCo locomotion benchmarks

We now compare ARAC against the CEM-TD3 [76], ERL [49] and CERL [48] baselines on seven continuous control tasks from the MuJoco suite [19]: `Ant-v2`, `HalfCheetah-v2`, `Humanoid-v2`, `HumanoidStandup-v2`, `Hopper-v2`, `Walker2d-v2` and

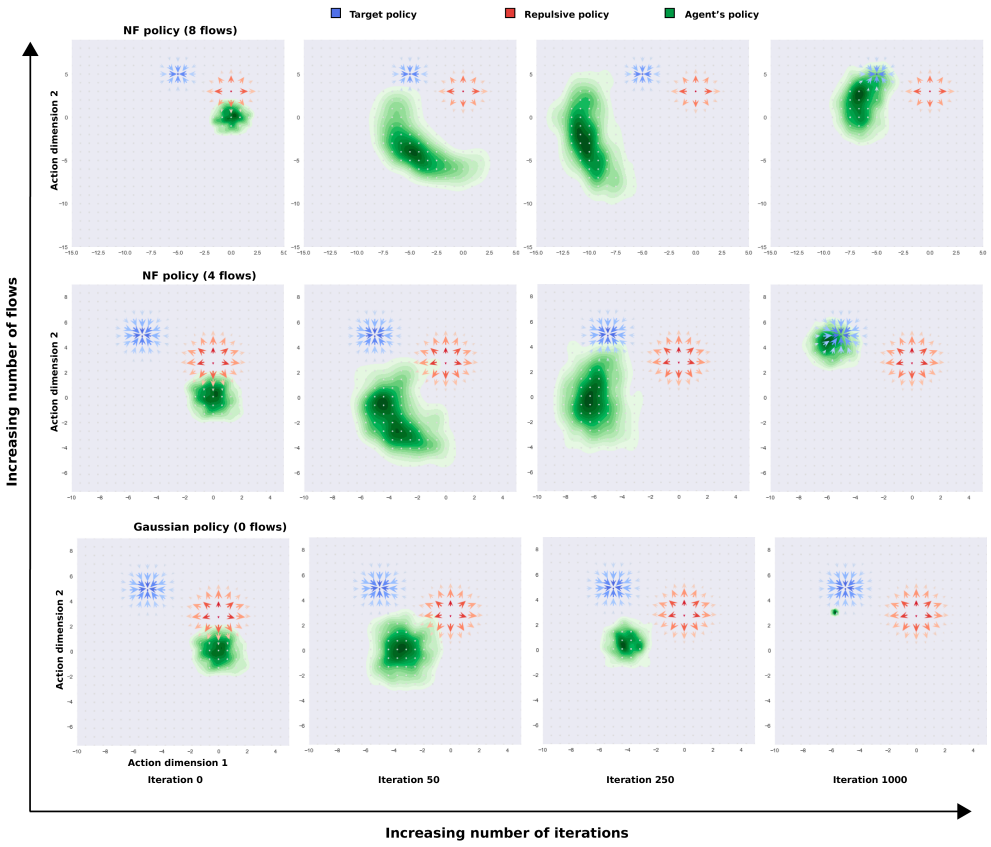


Figure 3.2: Agent trained to imitate a target while avoiding a repulsive policy using a proactive strategy. Increasing the number of flows leads to more complex policy's shape.

`Humanoid (rllab)`. We also designed a sparse reward environment `SparseHumanoid-v2`. All algorithms are run over 1M time steps on each environment, except `Humanoid (rllab)` which gets 2M time steps and `SparseHumanoid-v2` on 0.6M time steps.

ARAC performs $R = 10$ rollouts for evaluation steps every $10,000$ interaction steps with the environment. We consider a small population of $N = 5$ individuals with $K = 2$ as elites. Every SAC agent has one feedforward hidden layer of 256 units acting as state embedding, followed by a radial flow of length $\in \{3, 4\}$. A temperature of $\alpha = 0.05$ or 0.2 is used across all the environments (See appendix for more details). AR operations are carried out by sampling uniformly $n = 5$ archived policies from $\mathcal{G}$. Parameters details are provided in the Appendix (Table 3.4). All networks are trained with Adam optimizer [50]

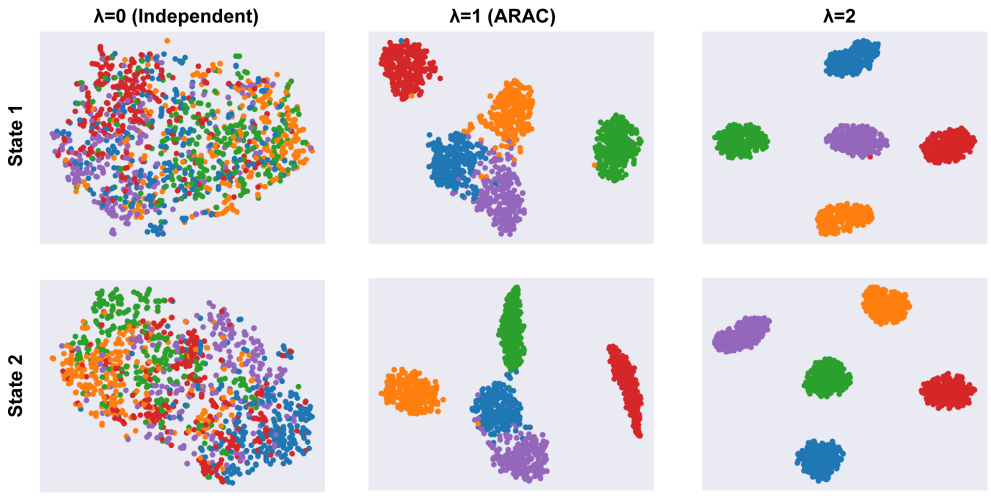


Figure 3.3: Mapping in two-dimensional space (t-SNE) of agents' actions for two arbitrary states. Each color represents a different agent.

using a learning rate of $3E^{-4}$. Baselines CEM-TD3[4], ERL[5], CERL[6] use the code contained in their respective repositories.

	ARAC	CEM-TD3	CERL	ERL	SAC - NF	SAC	TD3
Ant	**6044**	4239	1639	1442	4912	4370	4372
HC	10 264	10 659	5703	6746	8429	**11 900**	9543
Hopper	**3587**	**3655**	2970	1149	3538	2794	3564
Hu	**5965**	212	4756	551	5506	5504	71
Standup	**175 000**	29 000	117 000	12 900	116 000	149 000	54 000
Hu (rllab)	**14 230**	1334	3340	57	5531	1963	286
Walker2d	4704	4710	4386	1107	**5196**	3783	4682
Hu (Sparse)	**816**	0	1.32	8.65	547	88	0

Table 3.1: Maximum average return after 1M (2M for `Humanoid (rllab)` and 600k for `SparseHumanoid-v2`) time steps 5 random seeds. Bold: best methods when the gap is less than 100 units. See appendix for average return with standard deviation. Environment short names: HC: `HalfCheetah-v2`, Hu: `Humanoid-v2`, Standup: `HumanoidStandup-v2`

Figure 3.4 displays the performance of all algorithms on three environments over time steps (see Appendix Figure 3.7 for all environments). Results are averaged over 5 random seeds. Table 3.1 reports the best observed reward for each method.

[4] https://github.com/apourchot/CEM-RL
[5] https://github.com/ShawK91/erl_paper_nips18
[6] https://github.com/IntelAI/cerl

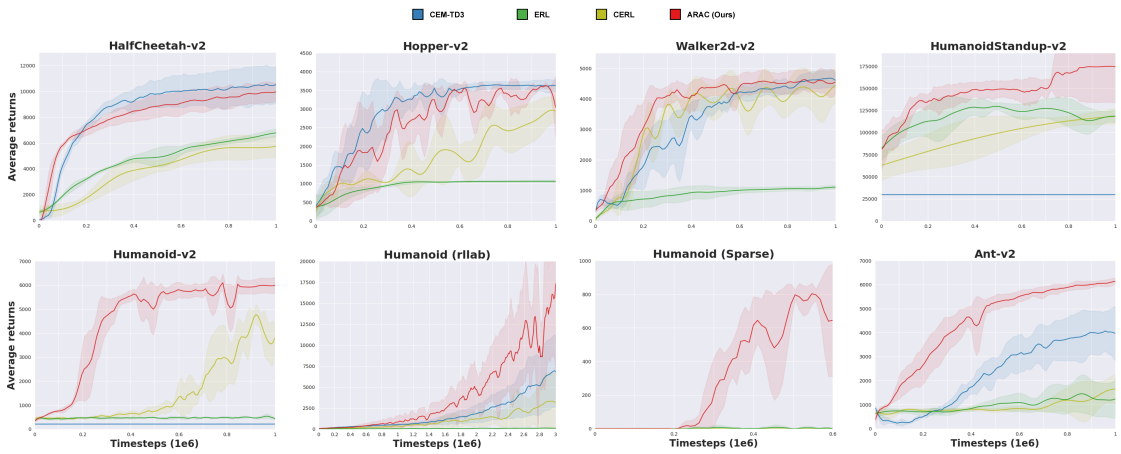


Figure 3.4: Average return and one standard deviation on 5 random seeds across 8 MuJoCo tasks. Curves are smoothed using Savitzky-Golay filtering with window size of 7.

Small state space environments `HalfCheetah-v2`, `Hopper-v2`, and `Walker2d-v2` are low-dimensional state space environments ($d \leq 17$). Except for `HalfCheetah-v2`, the proposed approach shows comparable results with its concurrent. Those results meet the findings of [75] that some environments with well-structured dynamics require little exploration. Full learning curves can be found in the Supplementary Material.

Deceptive reward and Large state space environments `Humanoid-v2`, `HumanoidStandup-v2` and `Humanoid (rllab)` belong to bipedal environments with high-dimensional state space ($d = 376$ and $d = 147$), and are known to trap algorithms into suboptimal solutions. Indeed, in addition to the legs, the agent also needs to control the arms, which may influence the walking way and hence induce deceptive rewards [17]. Figure 3.4 shows the learning curves on MuJoCo tasks. We observe that ARAC beats both baselines in performance as well as in convergence rate.

`Ant-v2` is another high-dimensional state space environment ($d \geq 100$). In an unstable setup, a naive algorithm implementing an unbalanced fast walk could still generate high reward, the reward taking into account the distance from start, instead of learning to stand, stabilize, and walk (as expected).

Sparse reward environment To test ARAC in a sparse reward environment, we created `SparseHumanoid-v2`. The dynamic is the same as Humanoid-v2 but rewards of +1 is granted only given is the center of mass of the agent is above a threshold (set to 0.6 unit in our case). The challenge not only lies in the sparse reward property but also on the complex body dynamic that can make the agent falling down and terminating the episode. As shown in Figure 3.4, ARAC is the only method that can achieve non zero performance. A comparison against single agent methods in the Appendix also shows better performance for ARAC.

Sample efficiency compared with single agent methods Figure 3.8 (in Supplementary Material) also shows that the sample efficiency of the population-based ARAC compares to a single SAC agent (with and without NFs) and other baselines methods (SAC, TD3). Indeed on `Humanoid-v2` and `Ant-v2` ARAC converges faster, reaching the 6k (4k, respectively) milestone performance after only 1M steps, while a single SAC agent requires 4M (3M, respectively) steps according to [33]. In general, ARAC achieves competitive results (no flat curves) and makes the most difference (faster convergence and better performance) in the biped environments.

Attraction-repulsion ablation study To illustrate the impact of repulsive forces, we introduce a hyperparameter λ in the overall loss (Eq. 3.5):

$$\mathcal{L}_{\theta,\phi,\lambda} = \mathcal{L}_{\theta,\phi,\text{SAC}} + \lambda \mathcal{L}_{\phi,\text{AR}} \tag{3.10}$$

We ran an ablation analysis on `Humanoid-v2` by varying that coefficient. For two random states, we sampled 500 actions from all agents and mapped these actions onto a two-dimensional space (via t-SNE). Figure 3.3 shows that without repulsion ($\lambda = 0$), actions from all agents are entangled, while repulsion ($\lambda > 0$) forces agents to behave differently and hence explore different regions of the action space.

The second ablation study is dedicated to highlight the differences between a Gaussian (like in [40] and a NF policy under AR operators. As one can observe in Figure 3.6, using a Gaussian policy deteriorates the solution as the repulsive KL term drives apart the means of agents and blows up/ shrinks the variance of the Gaussian policy. On the other side, applying the AR term on the NF layers maximizes the KL conditioned on the mean and variance of both base policies, resulting in a solution which allows sufficient exploration. More details are provided in the Appendix.

Also, through a toy example, under a repulsive term, we characterize the policy's shape when increasing the number of our radial flow policy in Figure 3.2 (Also show in Appendix). Unlike the diagonal Gaussian policy (SAC) that has symmetry constraint, increasing the number of flows allow radial policy to adopt more complex shape (from bottom to top).

3.6 Conclusion

In this paper, we introduced a population-based approach for structured exploration leveraging distributional properties of normalizing flows. Our method performs local search by means of Attraction-Repulsion strategies. Performing these operations with NF policies allowed a better handle over local solutions. The AR is done with respect to diverse ancestors across all training steps which are sampled from a bi-modal archive.

Empirical results on the MuJoCo suite demonstrate high performance of the proposed method in most settings. Moreover, in biped environments that are known to trap algorithms into suboptimal solutions, ARAC enjoys higher sample efficiency and better performance compare to its competitors. As future steps, borrowing from multi-objective optimization literature methods could allow one to combine other diversity metrics with the performance objective, to in turn improve the coverage of the solution space among the individuals by working with the corresponding Pareto front [41].

Appendix

Reproducibility Checklist

We follow the reproducibility checklist [74] and point to relevant sections explaining them here.

For all algorithms presented, check if you include:

- **A clear description of the algorithm, see main paper and included codebase.** The proposed approach is completely described by Alg. 3 (main paper), 4 (Appendix), and 5 (Appendix). The proposed population-based method uses attraction-repulsion operators in order to enforce a better policy space coverage by different agents.
- **An analysis of the complexity (time, space, sample size) of the algorithm.** See Appendix Figure 3.7 and 3.8. Experimentally, we demonstrate improvement in sample complexity as discussed in our main paper. In term of computation time, the proposed method scales linearly with the population size if agents are evaluated sequentially (as presented in Alg. 3 for clarity). However, this as mentioned in the paper, can be parallelized. All our results are obtained using M small network architectures with 1×256-units hidden layer followed by f layers of $|A| + 2$ units each (f being the number of radial flows and $|A|$ being the action space dimension).
- **A link to a downloadable source code, including all dependencies.** The code is included with Supplemental Material as a zip file; all dependencies can be installed using Python's package manager. Upon publication, the code would be available on Github.

For all figures and tables that present empirical results, check if you include:

- **A complete description of the data collection process, including sample size.** We use standard benchmarks provided in OpenAI Gym (Brockman et al., 2016).
- **A link to downloadable version of the dataset or simulation environment.** See: https://github.com/
- **An explanation of how samples were allocated for training / validation / testing.** We do not use a training-validation-test split, but instead report the mean performance (and one standard deviation) of the policy at evaluation time, openai/gym for OpenAI Gym benchmarks and https://www.roboti.us/index.html for MuJoCo suite. obtained with 5 random seeds.
- **An explanation of any data that were excluded.** We did not compare on easy environments (e.g. `Reacher-v2`) because all existing methods perform well on them. In that case, the improvement of our method upon baselines is incremental and not worth mentioning.
- **The exact number of evaluation runs.** 5 seeds for MuJoCo experiments, 1M, 2M or 3M environment steps depending on the domain.
- **A description of how experiments were run.** See Section 3.5 in the main paper and didactic example details in Appendix.
- **A clear definition of the specific measure or statistics used to report results.** Undiscounted returns across the whole episode are reported, and in turn averaged across 5 seeds.
- **Clearly defined error bars.** Confidence intervals and table values are always mean$\pm$ 1 standard deviation over 5 seeds.
- **A description of results with central tendency (e.g. mean) and variation (e.g. stddev)**. All results use the mean and standard deviation.

- **A description of the computing infrastructure used.** All runs used 1 CPU for all experiments (toy and MuJoCo) with 8Gb of memory.

Impact of repulsive force

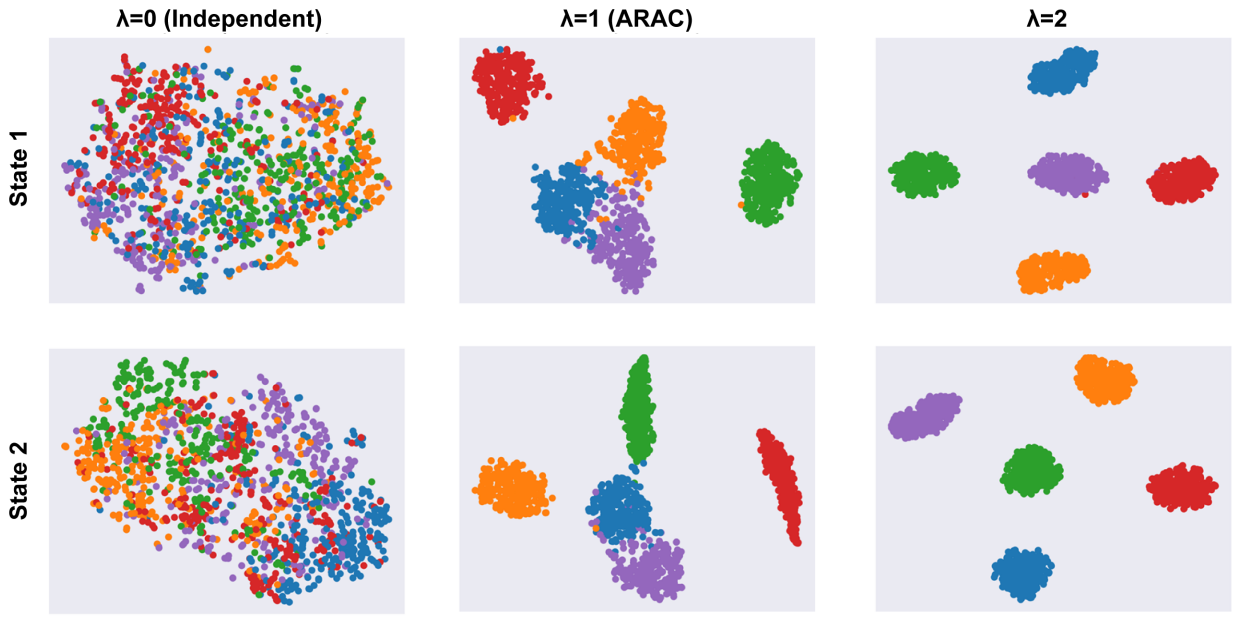


Figure 3.5: Mapping in two-dimension space (t-SNE) of agents' actions for two arbitrary states. Each color represents a different agent.

To illustrate the impact of the repulsive force coefficient λ, we ran an ablation analysis by varying that coefficient (recall that the overall loss function is $\mathcal{L}_\pi = \mathcal{L}_{\pi,\text{SAC}} + \lambda\mathcal{L}_{\text{AR}}$ where $\lambda = 1$ in our experiment).

For two random states, we sampled 500 actions from all agents and mapped theses actions in a common 2-dimensional space (t-SNE).

As shown in the Figure 3.5, policies trained without AR ($\lambda = 0$) result in entangled actions, while increasing the repulsive coefficient λ forces agents to have different actions and hence explore different regions of the policy space. Note that due to the specific nature of t-SNE , the policies are shown as Gaussians in a lower-dimensional embedding, while it is not necessarily the case in the true space.

Stabilizing Attraction-Repulsion with Normalizing Flow

In this section, we illustrate the consequence of the AR operators with a Gaussian policy (as in [40]) and our Normalizing flow policy for `Ant-v2`, `Humanoid-v2` and `HalfCheetah-v2`. As shown in the figure below, AR with Gaussian policies yield worse results. One reason is that the KL term drives apart the mean and variance of the Gaussian policy which deteriorates the main objective of maximizing the reward. On the other side, our method applies the AR only on the NF layers allows enough exploration by deviating sufficiently from the main objective function.

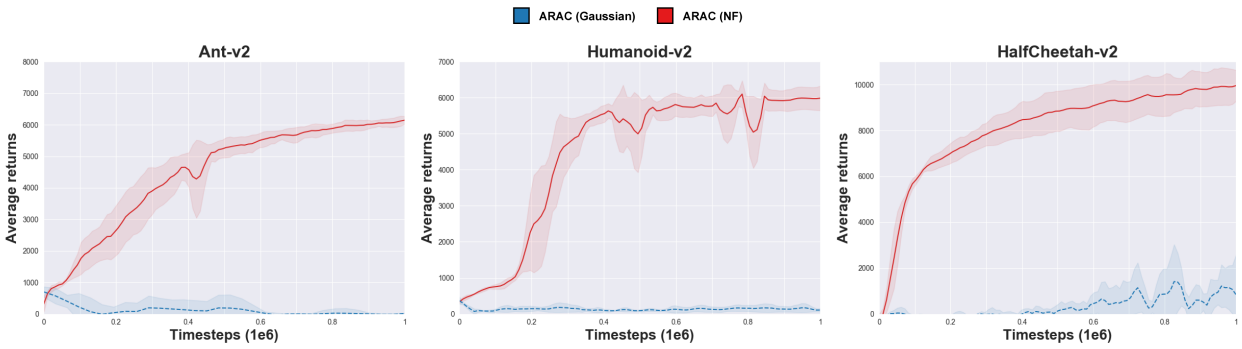


Figure 3.6: Comparison of ARAC agents using (1) AR with radial flows, (2) AR with only the base (Gaussian) policy and (3) no AR with radial flows.

Comparing ARAC against baselines on Mujoco tasks

Figure 3.7 shows the performance of ARAC and baselines (CEM-TD3, CERL and ERL) over time steps. Learning curves are averaged over 5 random seeds and displayed with one standard deviation. Evaluation is done every $10,000$ environment steps using 10 rollouts per agent. Overall, ARAC has reasonable performance on all tasks (no flat curves) and demonstrates high performance, especially in humanoid tasks.

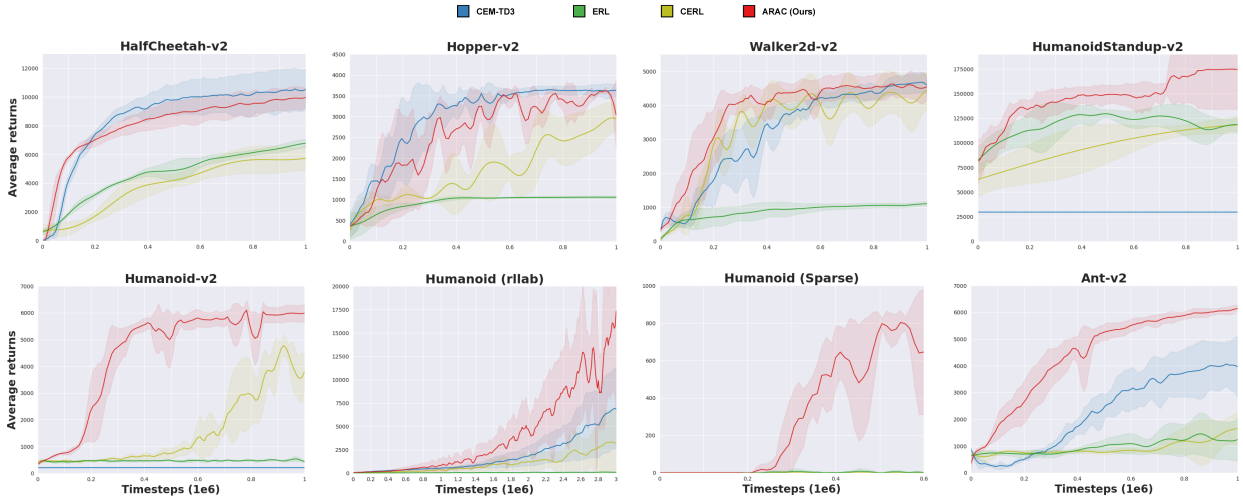


Figure 3.7: Average return and one standard deviation on 5 random seeds across 7 MuJoCo tasks for ARAC against baselines. Curves are smoothed using Savitzky-Golay filtering with window size of 7.

Benefits of population-based strategies: ARAC against single agents

In this section, we highlight the benefits of the proposed population-based strategy by comparing with single agents. Figure 3.8 shows the performance of ARAC against a single SAC agent (with and without normalizing flows). Learning curves are averaged over 5 random seeds and displayed with one standard deviation. Evaluation is done every $10,000$ environment steps using 10 rollouts per agent. We observe a high beneficial impact on the convergence rate as well as on the performance. ARAC outperforms single agents in almost all tasks (except for `HalfCheetah-v2` and `Walker-v2`) with large improvement. Note the high sample efficiency on humanoid environments (`Humanoid-v2` and `Humanoid (rllab)`), where it shows faster convergence in addition to better performance. Indeed, on `Humanoid (rllab)` a single SAC agent reaches the 4k milestone after 4M steps [33] while ARAC achieves this performance in less than 2M steps. Also, in `SparseHumanoid-v2`, due to its better coordinated exploration, ARAC could find a good solution faster than SAC-NF.

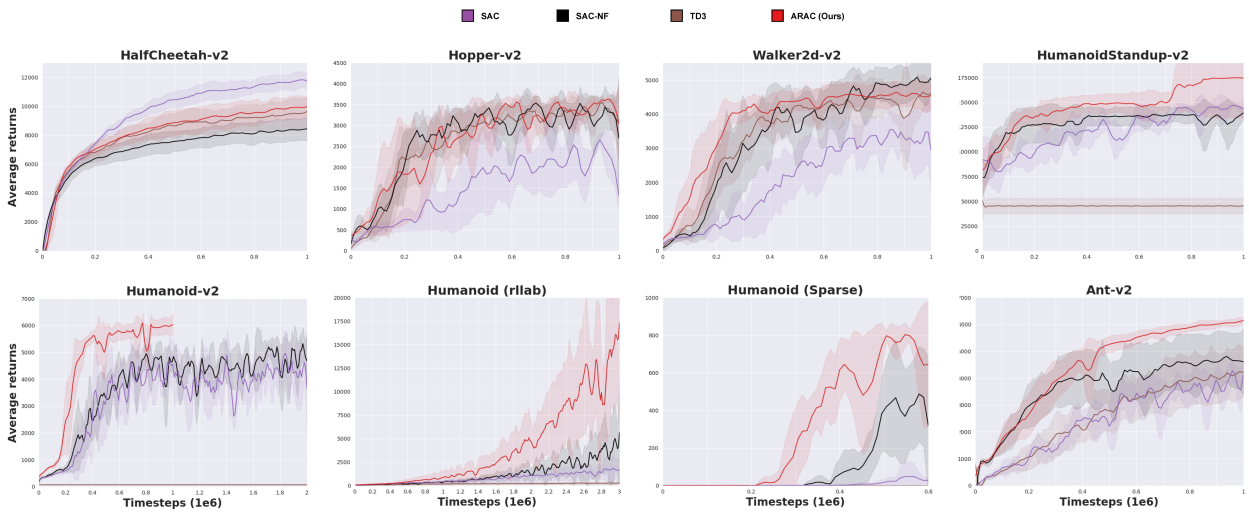


Figure 3.8: Average return and one standard deviation on 5 random seeds across 7 MuJoCo tasks for ARAC against single SAC agents (with and without NFs). Curves are smoothed using Savitzky-Golay filtering with window size of 7.

Overall performances on Mujoco tasks

	ARAC	CEM-TD3	CERL	ERL
Ant	**6,044 ± 216**	$4,239 \pm 1,048$	$1,639 \pm 564$	$1,442 \pm 819$
HC	$10,264 \pm 271$	10,659 ± 1,473	$5,703 \pm 831$	$6,746 \pm 295$
Hopper	**3,587 ± 65**	**3,655 ± 82**	$2,970 \pm 341$	$1,149 \pm 3$
Hu	**5,965 ± 51**	212 ± 1	$4,756 \pm 454$	551 ± 60
Standup	**175k ± 38k**	$29k \pm 4k$	$117k \pm 8k$	$129k \pm 4k$
Hu (rllab)	**14,234 ± 7251**	$1,334 \pm 551$	$3,340 \pm 3,340$	57 ± 17
Walker2d	4,704 ± 261	4,710 ± 320	4,3860 ± 615	$1,107 \pm 60$
SparseHu	**816 ± 20**	0 ± 0	1.32 ± 2.64	8.65 ± 15.90

	SAC - NF	SAC	TD3
Ant	$4,912 \pm 954$	$4,370 \pm 173$	$4,372 \pm 900$
HC	$8,429 \pm 818$	**11,896 ± 574**	$9,543 \pm 978$
Hopper	$3,538 \pm 108$	$2,794 \pm 729$	$3,564 \pm 114$
Hu	$5,506 \pm 147$	$5,504 \pm 116$	71 ± 10
Standup	$116k \pm 9k$	$149k \pm 7k$	$54k \pm 24k$
Hu (rllab)	$5,531 \pm 4,435$	$1,963 \pm 1,384$	286 ± 151
Walker2d	**5,196 ± 527**	3,783 ± 366	4,682 ± 539
SparseHu	547 ± 268	88 ± 159	0 ± 0

Table 3.2: Maximum average return after 1M (2M for `Humanoid (rllab)` and 600k for `SparseHumanoid-v2`) time steps ± one standard deviation on 5 random seeds. Bold: best methods when the gap is less than 100 units.Environment short names: HC: `HalfCheetah-v2`, Hu: `Humanoid-v2`, Standup: `HumanoidStandup-v2`

	CLEAR	TRPO	PPO	Trust-PCL	[75]	[96]	[40]
HalfCheetah-v2	$\mathbf{10,264}$	-15	$2,600$	$2,200$	$5,000$	$7,700$	$4,200$
Walker-v2	$\mathbf{4,764}$	$2,400$	$4,050$	400	850	500	N/A
Hopper-v2	$\mathbf{3,588}$	600	$3,150$	280	$2,500$	400	N/A
Ant-v2	$\mathbf{6,044}$	-76	$1,000$	$1,500$	N/A	N/A	N/A
Humanoid-v2	$\mathbf{5,939}$	400	400	N/A	N/A	N/A	$1,250$
HumanoidStandup-v2	$\mathbf{163,884}$	$80,000$	N/A	N/A	N/A	N/A	N/A
Humanoid (rllab)	$\mathbf{4,117}$	23	200	N/A	N/A	N/A	N/A

Table 3.3: Performance after 1M (except for rllab which is 2M) timesteps on 5 seeds. Values taken from their corresponding papers. N/A means the values were not available in the original paper.

Experimental parameters

Table 3.4 provides the hyperparameters of ARAC used to obtain results in the MuJoCo domains. The noise input for normalizing flows in SAC policies (see Sec. 3.2.3) is sampled from $\mathcal{N}(0, \sigma)$, where the variance σ is a function of the state (either fixed at a given value or learned).

ARAC parameters						
	# flows	σ	G	p	alpha	strategy
Ant-v2	3	0.2	10	1	0.2	proactive
HalfCheetah-v2	4	0.4	20	2	0.2	proactive
Hopper-v2	4	0.8	20	1	0.05	proactive
Walker2d-v2	4	0.6	10	3	0.05	proactive
Humanoid-v2	3	0.6	10	1	0.05	reactive
HumanoidStandup-v2	3	σ	20	1	0.2	reactive
Humanoid (rllab)	3	σ	10	1	0.05	proactive
SparseHumanoid-v2	2	0.6	20	1	0.2	proactive
Adam Optimizer parameters						
α_γ	3.10^{-4}					
α_ω	3.10^{-4}					
α_θ	3.10^{-4}					
α_ϕ	3.10^{-4}					
Algorithm parameters						
Batch size m	256					
Buffer size $\mathcal{B}$	10^6					
Archive sample size n	5					

Table 3.4: ARAC parameters.

Impact of number of flows on the policy shape

We used a single SAC agent with different radial flows numbers and randomly initialized weights, starting with actions centered at $(0, 0)$. All flow parameters are ℓ_1 regularized with hyperparameter 2. The agent is trained with the classical evidence lower bound (ELBO) objective augmented with the AR loss (Eq. 3.1), where the coefficient of the repulsive policy π' is given by $\beta_t = \frac{10}{t+1}$. Fig. 3.9 shows how both the NF and learned variance Gaussian policies manage to recover the target policy. We see that NF takes advantage of its flexible parametrization to adjust its density and can show asymmetric properties unlike the Gaussian distribution. This indeed can have advantage in some non symmetric environment where the Gaussian policy would be trapped into a suboptimal behavior. Finally, increasing the number of flows (from bottom to top) can lead to more complex policy's shape.

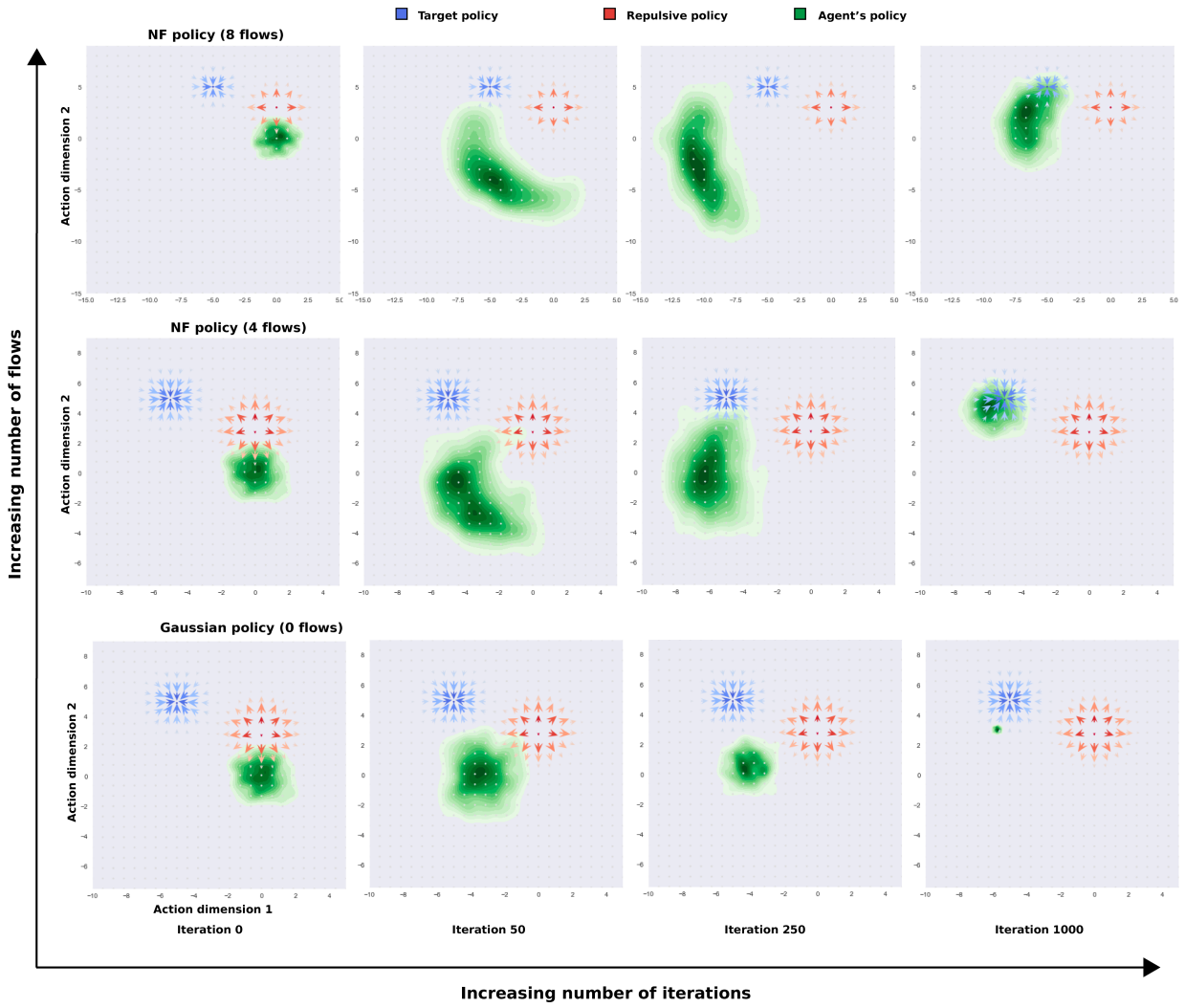


Figure 3.9: Single state didactic illustration of attraction-repulsion operators. Comparing behavior of NF policy against Gaussian policy with learned variance under a repulsive constraint.

3.6.1 Pseudo-code for ARAC

Algorithm 3 ARAC: Attraction-Repulsion Actor-Critic

Input: population size M; number of elites K; maximum archive capacity G; archive sample size n; number of evaluation rollouts R; actor coefficient p; strategy (either proactive or reactive).

Initialize value function network V_ν and critic network Q_ω
Initialize population of policy networks $\{\pi^m_{\phi,\theta}\}_{m=1}^M$
Initialize empty archive $\mathcal{G}$ and randomly assign K individuals to top-K

total_step $\leftarrow 0$
while total_step $\leq$ max_step **do**
  step $\leftarrow 0$

  for agent $m = 1 \dots M$ **do**
    $(\_, \text{step } s) \leftarrow$ `rollout`$(\pi^m, \text{with noise}, \text{over } 1 \text{ episode})$
    step $\leftarrow$ step $+ s$
    total_step $\leftarrow$ total_step $+ s$
  end for

} *Collect samples*

  $C = \text{step}/K$
  for policy π^e in top-K **do**
    Update critic with π^e for C mini-batches (Eq. 3.3)
    Update value function (Eq. 3.4)
  end for

} *Update critic*

  for agent $m = 1 \dots M$ **do**
    if policy π^m is in top-K **then**
      Sample n archived policies uniformly from $\mathcal{G}$
      Update actor π^m for $\frac{\text{step}}{M}.p$ mini-batches (Eq. 3.5 and 3.8 or 3.9)
    else
      Update actor π^m for $\frac{\text{step}}{M}.p$ mini-batches (Eq. 3.2)
    end if
  end for

} *Update actors*

  for agent $m = 1 \dots M$ **do**
    $(\text{Fitness}_m, \_) \leftarrow$ `rollout`$(\pi^m, \text{without noise}, \text{over } R \text{ episodes})$
  end for
  Rank population $\{\pi^m_{\phi,\theta}\}_{m=1}^M$ and identify top-K
  `update_archive`$(\mathcal{G}, \{\pi^m_{\phi,\theta}\}_{m=1}^M, G)$
end while

} *Evaluate actors*

Complementary pseudo-code for ARAC

Algorithms 4 and 5 respectively provide the pseudo-code of functions `rollout` and `update_archive` used in Algorithm 3.

Algorithm 4 `rollout`

Input: actor π; noise status; number of episodes E; replay buffer $\mathcal{B}$;

Fitness $\leftarrow 0$

for episode $= 1, \ldots, E$ **do**

$\vec{s} \leftarrow$ Initial state $\vec{s}_0$ from the environment

for step $t = 0 \ldots$ termination **do**

if with noise **then**

Sample noise z

else

Set $z \leftarrow 0$

end if

$\vec{a}_t \sim \pi(.|\vec{s}_t, z)$

Observe $\vec{s}_{t+1} \sim P(\cdot|\vec{s}_t, \vec{a}_t)$ and obtain reward r_t

Fitness $\leftarrow$ Fitness $+ r_t$

Store transition ($\vec{s}_t$,$\vec{a}_t$,r_t,$\vec{s}_{t+1}$) in $\mathcal{B}$

end for

end for

Fitness $\leftarrow$ Fitness$/E$

return Average fitness per episode and number of steps performed

Algorithm 5 `update_archive`

Input: archive $\mathcal{G}$; population of size M; maximal archive capacity G.

if $|\mathcal{G}| < G$ **then**

Add all agents of current population to $\mathcal{G}$

else

$c_1, c_2 \leftarrow$ 2-means(fitness of individuals in $\mathcal{G}$)

for agent $m = 1, \ldots, M$ **do**

Assign agent m to closest cluster $c \in \{c_1, c_2\}$ based on its fitness

Sample an archived agent $j \sim$ Uniform(c)

Replace archived individual j by m

end for

end if

return Updated archive $\mathcal{G}$

Chapter 4
Conclusion

This thesis deals with three theoretical and applied problems faced by machine learning algorithms – two of them related to unsupervised learning, specifically GANs, and one to reinforcement learning. Our first essay in Chapter 2 tackles a pathology GANs [27] often suffer from: mode collapse. This results in the GAN learning only a subset of modes of the original data distribution. This results in a loss of information if one wants to learn certain particular distribution, e.g., demand distribution for a given product in a retail context. To handle this problem, we designed a set of discriminators with increasing capacity that can be used to train the generator, with each discriminator assigned a different weight. We then used a multi-armed bandit algorithm to adaptatively learn those weights, resulting in a curriculum learning. While weaker discriminators have smoother gradients and hence provide a better learning signal to help recover missing modes, stronger discriminators focus on finer-grain detail to ensure sample quality. This process eventually creates a trade-off between mode coverage and sample quality. Empirical results on synthetic data showed that our model achieves better FID scores, indicating the generation of more realistic data.

The second essay in Chapter 3 is dedicated to an application of GANs. We learn the distribution of items that a customer buys during a shopping trip based on the transactional data from a drugstore chain. Preliminary work consisted of learning an embedding of customer's purchases (hidden layer of an LSTM) and of items via their description (using word2vec [67] embedding). Equipped with the latter information, we use a GAN to generate coordinates in the item embedding space. These are then mapped to real items using Euclidian distance. Results showed that our model yielded good results, generating baskets of items with similar price and number of items as the real data, as well as

similar proportions of items from each category. Moreover, using a classifier to discriminate items generated from the GAN and from real data, showed good accuracy assuming relative importance of noise contained in the real data.

The last essay in Chapter 4 deals with the exploration problem in continuous control for reinforcement learning. When attempting to solve the complex biped/humanoid agent task, reinforcement learning algorithms often get trapped in sub-optimal solutions because the task's reward landscape is multimodal. We tackle this problem by performing exploration under a multi-agent framework. We designed an algorithm that performs search over local solutions, and employs an Attraction-Repulsion (AR) mechanism (with respect to past solutions, stored in an archive) to maintain diversity of solutions throughout the search. The AR mechanism is enhanced by the use of normalizing flows which allows efficient Attraction-Repulsion compared to the usual diagonal Gaussian distribution because of the properties of multimodality and flexibility. Empirical results showed that our methods enjoys faster convergence and better performance, especially on the high-dimensional humanoid tasks.

Chapter 5
Future directions

As for future opportunities of research, an in-depth investigation of other normalizing flow properties (such as Inverse Autoregressive Flow [51] or Neural Autoregressive Flows [42]) can help tackling exploration problems in reinforcement learning domains. Leveraging this knowledge provides one with a better understanding to design well-suited algorithms that can handle environments-specific traps. Not only is the use of normalizing flows a simple (not costly computation-wise) addition of invertible layers, but it also allows one to benefit from a highly expressive distribution (multimodality). For this purpose, we believe that it can be a good add-on for vanilla algorithms such as Trust-Region Policy Optimization [84], Proximal Policy Optimisation [85] and can be used as a base for future methods.

Another promising direction would be to take advantage of invertible and monotonic mappings of normalizing flows to handle the non-linear Bellman equation [98]. If one can guarantee Lipschitz conditions, the latter trick can have nice properties such as reducing overestimation bias and variance, both of which affect Deep Q-learning [2, 70]. As a result, the accumulated overestimation error can be reduced and one can safely rely on the critic to take actions.

Bibliography

[1] ABE, N., VERMA, N., APTE, C., AND SCHROKO, R. Cross channel optimized marketing by reinforcement learning. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04* (2004), 767.

[2] ANSCHEL, O., BARAM, N., AND SHIMKIN, N. Deep reinforcement learning with averaged target DQN. *CoRR abs/1611.01929* (2016).

[3] ARJOVSKY, M., AND BOTTOU, L. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations* (2017).

[4] ARJOVSKY, M., CHINTALA, S., AND BOTTOU, L. Wasserstein GAN. *ArXiv e-prints*, arXiv:1701.07875 (Jan. 2017).

[5] AUER, P., CESA-BIANCHI, N., FREUND, Y., AND SCHAPIRE, R. E. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)* (1995), p. 322.

[6] BAI, T., NIE, J.-Y., ZHAO, W. X., ZHU, Y., DU, P., AND WEN, J.-R. An attribute-aware neural attentive model for next basket recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (New York, NY, USA, 2018), SIGIR '18, ACM, pp. 1201–1204.

[7] BALABAN, S. Deep learning and face recognition: the state of the art. *CoRR abs/1902.03524* (2019).

[8] BALDASSINI, L., AND RODRÍGUEZ SERRANO, J. A. client2vec: Towards Systematic Baselines for Banking Applications. *ArXiv e-prints*, arXiv:1802.04198 (Feb. 2018).

[9] BARANSI, A., MAILLARD, O.-A., AND MANNOR, S. Sub-sampling for multi-armed bandits. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2014), Springer, pp. 115–131.

[10] BARKAN, O., AND KOENIGSTEIN, N. Item2vec: Neural item embedding for collaborative filtering. *CoRR abs/1603.04259* (2016).

[11] BELGHAZI, I., RAJESWAR, S., BARATIN, A., HJELM, R. D., AND COURVILLE, A. C. MINE: mutual information neural estimation. *CoRR abs/1801.04062* (2018).

[12] BELLEMARE, M. G., NADDAF, Y., VENESS, J., AND BOWLING, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research 47* (2013), 253–279.

[13] BELLMAN, R. A markovian decision process. *Journal of Mathematics and Mechanics* (1957), 679–684.

[14] BENGIO, Y., LOURADOUR, J., COLLOBERT, R., AND WESTON, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, 2009), ICML '09, ACM, pp. 41–48.

[15] CHAMBERLAIN, B. P., CARDOSO, Â., LIU, C. H. B., PAGLIARI, R., AND DEISENROTH, M. P. Customer life time value prediction using embeddings. *CoRR abs/1703.02596* (2017).

[16] CHE, T., LI, Y., JACOB, A. P., BENGIO, Y., AND LI, W. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136* (2016).

[17] CONTI, E., MADHAVAN, V., SUCH, F. P., LEHMAN, J., STANLEY, K., AND CLUNE, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems (NeurIPS)* (2018), pp. 5027–5038.

[18] DOWSON, D. C., AND LANDAU, B. V. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis 12*, 3 (1982), 450–455.

[19] DUAN, Y., CHEN, X., HOUTHOOFT, R., SCHULMAN, J., AND ABBEEL, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning (ICML)* (2016), pp. 1329–1338.

[20] DUMOULIN, V., BELGHAZI, I., POOLE, B., MASTROPIETRO, O., LAMB, A., ARJOVSKY, M., AND COURVILLE, A. Adversarially Learned Inference. *ArXiv e-prints* (June 2016).

[21] DURUGKAR, I. P., GEMP, I., AND MAHADEVAN, S. Generative multi-adversarial networks. *CoRR abs/1611.01673* (2016).

[22] FEDUS, W., ROSCA, M., LAKSHMINARAYANAN, B., DAI, A. M., MOHAMED, S., AND GOODFELLOW, I. Many paths to equilibrium: Gans do not need to decrease adivergence at every step. *arXiv preprint arXiv:1710.08446* (2017).

[23] FOURNIER-VIGER, P., LIN, J. C.-W., GOMARIZ, A., GUENICHE, T., SOLTANI, A., DENG, Z., AND LAM, H. T. The spmf open-source data mining library version 2. In *Machine Learning and Knowledge Discovery in Databases* (Cham, 2016), B. Berendt, B. Bringmann, É. Fromont, G. Garriga, P. Miettinen, N. Tatti, and V. Tresp, Eds., Springer International Publishing, pp. 36–40.

[24] FREUND, Y., AND SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55*, 1 (1997), 119–139.

[25] GAL, Y., AND GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning (ICML)* (2016), pp. 1050–1059.

[26] GAN, W., LIN, J. C., FOURNIER-VIGER, P., CHAO, H., AND YU, P. S. A survey of parallel sequential pattern mining. *CoRR abs/1805.10515* (2018).

[27] GOODFELLOW, I. J., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative Adversarial Networks. *ArXiv e-prints*, arXiv:1406.2661 (June 2014).

[28] GRAVES, A., BELLEMARE, M. G., MENICK, J., MUNOS, R., AND KAVUKCUOGLU, K. Automated curriculum learning for neural networks. *CoRR abs/1704.03003* (2017).

[29] GU, C. Smoothing noisy data via regularization: statistical perspectives. *Inverse Problems 24*, 3 (2008), 034002.

[30] GULRAJANI, I., AHMED, F., ARJOVSKY, M., DUMOULIN, V., AND COURVILLE, A. C. Improved training of wasserstein gans. *CoRR abs/1704.00028* (2017).

[31] GUPTA, A., MENDONCA, R., LIU, Y., ABBEEL, P., AND LEVINE, S. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems (NeurIPS)* (2018), pp. 5302–5311.

[32] GUPTA, D., AND GHAFIR, S. An overview of methods maintaining diversity in genetic algorithms. *International journal of emerging technology and advanced engineering 2*, 5 (2012), 56–60.

[33] HAARNOJA, T., ZHOU, A., ABBEEL, P., AND LEVINE, S. Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)* (2018), pp. 1856–1865.

[34] HAARNOJA, T., ZHOU, A., HARTIKAINEN, K., TUCKER, G., HA, S., TAN, J., KUMAR, V., ZHU, H., GUPTA, A., ABBEEL, P., AND LEVINE, S. Soft actor-critic algorithms and applications. *CoRR abs/1812.05905* (2018).

[35] HENDERSON, P., DOAN, T., ISLAM, R., AND MEGER, D. Bayesian policy gradients via alpha divergence dropout inference. *NIPS Bayesian Deep Learning Workshop* (2017).

[36] HEUSEL, M., RAMSAUER, H., UNTERTHINER, T., NESSLER, B., KLAMBAUER, G., AND HOCHREITER, S. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR abs/1706.08500* (2017).

[37] HOANG, Q., NGUYEN, T. D., LE, T., AND PHUNG, D. Q. Multi-generator generative adversarial nets. *CoRR abs/1708.02556* (2017).

[38] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural Comput. 9*, 8 (Nov. 1997), 1735–1780.

[39] HOCHREITER, S., AND URGEN SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation 9*, 8 (1997), 1735–1780.

[40] HONG, Z.-W., SHANN, T.-Y., SU, S.-Y., CHANG, Y.-H., FU, T.-J., AND LEE, C.-Y. Diversity-driven exploration strategy for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)* (2018), pp. 10489–10500.

[41] HORN, J., NAFPLIOTIS, N., AND GOLDBERG, D. E. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence* (1994), pp. 82–87.

[42] HUANG, C.-W., KRUEGER, D., LACOSTE, A., AND COURVILLE, A. Neural autoregressive flows. In *International Conference on Machine Learning* (2018), pp. 2083–2092.

[43] HUANG, X., LI, Y., POURSAEED, O., HOPCROFT, J. E., AND BELONGIE, S. J. Stacked generative adversarial networks. *CoRR abs/1612.04357* (2016).

[44] JUEFEI-XU, F., BODDETI, V. N., AND SAVVIDES, M. Gang of gans: Generative adversarial networks with maximum margin ranking. *CoRR abs/1704.04865* (2017).

[45] KANUNGO, T., MOUNT, D. M., NETANYAHU, N. S., PIATKO, C., SILVERMAN, R., AND WU, A. Y. An efficient k-means clustering algorithm: Analysis and implementation, 2000.

[46] KARRAS, T., AILA, T., LAINE, S., AND LEHTINEN, J. Progressive growing of gans for improved quality, stability, and variation. *CoRR abs/1710.10196* (2017).

[47] KAWASAKI, M., AND HASUIKE, T. A recommendation system by collaborative filtering including information and characteristics on users and items. In *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings* (2 2018), vol. 2018-January, Institute of Electrical and Electronics Engineers Inc., pp. 1–8.

[48] KHADKA, S., MAJUMDAR, S., NASSAR, T., DWIEL, Z., TUMER, E., MIRET, S., LIU, Y., AND TUMER, K. Collaborative evolutionary reinforcement learning. *CoRR abs/1905.00976* (2019).

[49] KHADKA, S., AND TUMER, K. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)* (2018), pp. 1188–1200.

[50] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014).

[51] KINGMA, D. P., SALIMANS, T., JOZEFOWICZ, R., CHEN, X., SUTSKEVER, I., AND WELLING, M. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems (NeurIPS)* (2016), pp. 4743–4751.

[52] KUMAR, A., BISWAS, A., AND SANYAL, S. ecommercegan : A generative adversarial network for e-commerce. *CoRR abs/1801.03244* (2018).

[53] KWAK, H., AND ZHANG, B. Generating images part by part with composite generative adversarial networks. *CoRR abs/1607.05387* (2016).

[54] LE, Q., AND MIKOLOV, T. Distributed representations of sentences and documents. In *International Conference on Machine Learning* (2014), pp. 1188–1196.

[55] LEE, K., KIM, S.-A., CHOI, J., AND LEE, S.-W. Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In *International Conference on Machine Learning* (2018), pp. 2943–2952.

[56] LIN, Z., KHETAN, A., FANTI, G. C., AND OH, S. Pacgan: The power of two samples in generative adversarial networks. *CoRR abs/1712.04086* (2017).

[57] LINOFF, G. S., AND BERRY, M. J. A. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*, 3rd ed. Wiley Publishing, 2011.

[58] LITTLESTONE, N., AND WARMUTH, M. K. The weighted majority algorithm. *Information and computation 108*, 2 (1994), 212–261.

[59] LIU, Y., RAMACHANDRAN, P., LIU, Q., AND PENG, J. Stein variational policy gradient. In *Conference on Uncertainty in Artificla Intelligence (UAI)* (2017).

[60] LIU, Z., LUO, P., WANG, X., AND TANG, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)* (12 2015).

[61] LU, T., PÁL, D., AND PÁL, M. Showing relevant ads via lipschitz context multi-armed bandits. In *Thirteenth International Conference on Artificial Intelligence and Statistics* (2010).

[62] MAHFOUD, S. W. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign Champaign, USA, 1995.

[63] MATIISEN, T., OLIVER, A., COHEN, T., AND SCHULMAN, J. Teacher-student curriculum learning. *CoRR abs/1707.00183* (2017).

[64] MAULDIN, M. L. Maintaining diversity in genetic search. In *AAAI Conference on Artificial Intelligence (AAAI)* (1984), pp. 247–250.

[65] MAZOURE, B., DOAN, T., DURAND, A., HJELM, R. D., AND PINEAU, J. Leveraging exploration in off-policy algorithms via normalizing flows. *Proceedings of the 3rd Conference on Robot Learning (CoRL 2019)* (2019).

[66] METZ, L., POOLE, B., PFAU, D., AND SOHL-DICKSTEIN, J. Unrolled generative adversarial networks. *CoRR abs/1611.02163* (2016).

[67] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., AND DEAN, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[68] MITCHELL, T. J., AND BEAUCHAMP, J. J. Bayesian variable selection in linear regression. *Journal of the American Statistical Association 83*, 404 (1988), 1023–1032.

[69] MIYATO, T., KATAOKA, T., KOYAMA, M., AND YOSHIDA, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).

[70] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. A. Playing atari with deep reinforcement learning. *CoRR abs/1312.5602* (2013).

[71] NEYSHABUR, B., BHOJANAPALLI, S., AND CHAKRABARTI, A. Stabilizing GAN training with multiple random projections. *CoRR abs/1705.07831* (2017).

[72] NGUYEN, D., NGUYEN, T. D., LUO, W., AND VENKATESH, S. Trans2vec: Learning transaction embedding via items and frequent itemsets. In *PAKDD* (2018).

[73] PEDNAULT, E., ABE, N., AND ZADROZNY, B. Sequential cost-sensitive decision making with reinforcement learning. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02* (2002), 259.

[74] PINEAU, J. The machine learning reproducibility checklist.

[75] PLAPPERT, M., HOUTHOOFT, R., DHARIWAL, P., SIDOR, S., CHEN, R. Y., CHEN, X., ASFOUR, T., ABBEEL, P., AND ANDRYCHOWICZ, M. Parameter space noise for exploration. In *International Conference on Learning Representations (ICLR)* (2018).

[76] POURCHOT, A., AND SIGAUD, O. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *International Conference on Learning Representations (ICLR)* (2019).

[77] PUTERMAN, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[78] RADCLIFFE, N., AND SURRY, P. Real-world uplift modelling with significance-based uplift trees. *White Paper TR-2011-1, Stochastic . . .*, section 6 (2011), 1–33.

[79] RADFORD, A., METZ, L., AND CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR abs/1511.06434* (2015).

[80] REZENDE, D. J., AND MOHAMED, S. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)* (2015), pp. 1530–1538.

[81] ROSTAMZADEH, N., HOSSEINI, S., BOQUET, T., STOKOWIEC, W., ZHANG, Y., JAUVIN, C., AND PAL, C. Fashion-gen: The generative fashion dataset and challenge. *arXiv preprint arXiv:1806.08317* (2018).

[82] ROTH, K., LUCCHI, A., NOWOZIN, S., AND HOFMANN, T. Stabilizing training of generative adversarial networks through regularization. *CoRR abs/1705.09367* (2017).

[83] SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (New York, NY, USA, 2001), WWW '01, ACM, pp. 285–295.

[84] SCHULMAN, J., LEVINE, S., ABBEEL, P., JORDAN, M., AND MORITZ, P. Trust region policy optimization. In *International Conference on Machine Learning (ICML)* (2015), pp. 1889–1897.

[85] SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O. Proximal policy optimization algorithms. *arXiv preprint: 1707.06347* (2017).

[86] SCHWAIGER, A., AND STAHMER, B. Simmarket: Multiagent-based customer simulation and decision support for category management. *Multiagent System Technologies* (2003).

[87] SHANI, G., AND GUNAWARDANA, A. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 2011, pp. 257–297.

[88] SILVER, D., NEWNHAM, L., BARKER, D., WELLER, S., AND MCFALL, J. Concurrent Reinforcement Learning from Customer Interactions. *Icml (3)* (2013), 924–932.

[89] SRIVASTAVA, A., VALKOV, L., RUSSELL, C., GUTMANN, M. U., AND SUTTON, C. Veegan: Reducing mode collapse in gans using implicit variational learning. *ArXiv e-prints* (May 2017).

[90] TANG, H., HOUTHOOFT, R., FOOTE, D., STOOKE, A., CHEN, O. X., DUAN, Y., SCHULMAN, J., DETURCK, F., AND ABBEEL, P. # Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems (NeurIPS)* (2017), pp. 2753–2762.

[91] TANG, Y., AND AGRAWAL, S. Boosting trust region policy optimization by normalizing flows policy. *arXiv preprint: 1809.10326* (2018).

[92] THEIS, L., VAN DEN OORD, A., AND BETHGE, M. A note on the evaluation of generative models. *ArXiv e-prints*, arXiv1511.01844 (Nov. 2015).

[93] TKACHENKO, Y., KOCHENDERFER, M. J., AND KLUZA, K. Customer simulation for direct marketing experiments. *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016* (2016), 478–487.

[94] TODOROV, E., EREZ, T., AND TASSA, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2012), IEEE, pp. 5026–5033.

[95] TOLSTIKHIN, I. O., GELLY, S., BOUSQUET, O., SIMON-GABRIEL, C.-J., AND SCHÖLKOPF, B. Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems* (2017), pp. 5424–5433.

[96] TOUATI, A., SATIJA, H., ROMOFF, J., PINEAU, J., AND VINCENT, P. Randomized value functions via multiplicative normalizing flows. *arXiv preprint: 1806.02315* (2018).

[97] VAN DER MAATEN, L., AND HINTON, G. Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605.

[98] VAN HASSELT, H., QUAN, J., HESSEL, M., XU, Z., BORSA, D., AND BARRETO, A. General non-linear bellman equations. *CoRR abs/1907.03687* (2019).

[99] WAN, S., LAN, Y., WANG, P., GUO, J., XU, J., AND CHENG, X. Next basket recommendation with neural networks. In *Poster Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16, 2015.* (2015), P. Castells, Ed., vol. 1441 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[100] XU, H., CARAMANIS, C., AND MANNOR, S. Robustness and regularization of support vector machines. *Journal of Machine Learning Research 10*, Jul (2009), 1485–1510.

[101] YU, F., LIU, Q., WU, S., WANG, L., AND TAN, T. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR*

Conference on Research and Development in Information Retrieval (New York, NY, USA, 2016), SIGIR '16, ACM, pp. 729–732.

[102] ZHANG, T., AND ZHANG, D. Agent-based simulation of consumer purchase decision-making and the decoy effect. *Journal of Business Research 60*, 8 (2007), 912–922.

[103] ZHAO, J., MATHIEU, M., AND LECUN, Y. A Quantitative Measure of Generative Adversarial Network Distributions. *International Conference on Learning Representations* (2016).

[104] ZIEBART, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, figshare, 2010.