Stabilizing Generative Models using Self-Supervision

Akshay Dharmavaram

CMU-RI-TR-22-18

April 27, 2022

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my family, friends and teachers.

Abstract

Generative Models have been shown to be adept in mimicking the behavior of an unknown distribution solely from bootstrapped data. However, deep learning models have been shown to overfit in either the minimization or maximization stage of the two player min-max game, resulting in unstable training dynamics. In this work, we explore the use self-supervision as a way to incorporate domain-knowledge to stabilize the training dynamics and avoid overfitting. In this work, we investigate the use self-supervision to stabilize the training of generative models in the single-step and multi-step domains.

Recently, there has been an increased interest in rewriting single-step generative models to output realistic images that are semantically similar to a handful of fixed, user-defined sketches. However, replicating images from complex poses or distinctive, minimalist art styles (e.g. "the Picasso horse") have been shown to be difficult. To rectify these failure cases, we propose a method that builds upon the GANSketching architecture by introducing a translation model that shifts the distribution of fake sketches to be more similar to that of the user-sketches while also retaining the essence of the originally generated image. Such a formulation avoids overfitting by the discriminator, thus reducing the discriminability and improving gradient propagation. We also illustrate how the choice in the direction of translation affects the number of training steps required as well as the overall performance of the generator.

The current landscape of multi-step apprenticeship learning is dominated by Adversarial Imitation Learning (AIL) based methods. In this work, we investigate the issues faced by these algorithms and introduce a novel self-supervised loss that encourages the discriminator to approximate a richer reward function. We also employ our method to train a graph-based multi-agent actor-critic architecture that learns a centralized policy, conditioned on a learned latent interaction graph. We show that our method outperforms prior state-of-the-art methods for both the single-agent and multi-agent domains. Furthermore, we prove that our new regularizer is part of the family of AIL methods by providing a theoretical connection to cost-regularized apprenticeship learning. Moreover, we leverage the self-supervised formulation to illustrate novel reward shaping capabilities as well as the introduction of a novel teacher forcing-based curriculum (Trajectory Forcing) that improves sample efficiency by progressively increasing the length of the generated trajectory.

Acknowledgments

I am extremely thankful to my advisor Prof. Katia Sycara for allowing me to explore the areas of my interest and supporting me through my master's journey. I learned how to research a problem in a methodical manner and present my work in a cogent and structured way. I am thankful to my thesis committee members Prof. Jean Oh and Tabitha Lee for the discussions and feedback on my work.

During my master's program, I have had the opportunity to work with and meet many awesome people. I am thankful to my collaborators Prof. Jun-Yan Zhu, Tejus Gupta, Sheng-Yu Wang, Mayank Mali, and Rohit Jena. I have had many long discussions with other lab-mates such as Swaminathan Gurumurthy, Siddharth Agrawal, Vidhi Jain, and Zongyue Zhao who helped me furnish my ideas. My lab-mates were brilliant and amicable people and I had a splendid time working with them. I am grateful to Dana Hughes for helping me out with the engineering aspects of my research whenever I needed help. I am thankful to Lynnetta Miller for handling the logistics in our lab and to Barbara Jean and Prof. George Kantor for smoothly conducting the MS in Robotics program. I am thankful to everyone I met at the Robotics Institute and at CMU as a whole.

I would also like to thank my family for constantly believing in me and providing me the much needed mental strength and support, without which this wouldn't have been possible.

Funding

This work was supported by the Defense Advanced Research Projects Agency (DARPA). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Generative models are a class of statistical models that can generate or mimic an unknown distribution, and reproduce data-instances that are indistinguishable from the target distribution. There has recently been an increased interest in these models, and one particular approach has become widely adopted, namely the Generative Adversarial Network (GANs) [13]. GANs have been extremely successful in modeling and replicating high dimensional distributions, such as generating realistic human portraits [41, 60, 26]. These models are trained in an adversarial approach, which involves a joint optimization scheme where a generative model and a discriminative model compete against each other. The training dynamics is formulated as a min-max game [13], and the generator and discriminator take turns optimizing competing loss functions. Intuitively, this approach would be expected to generate rich estimators for the target distribution; however, in practise it has been observed that training such models requires considerable engineering prowess [42].

Another recent trend in machine learning, specifically in unsupervised learning, involves employing supervised learning approaches along with pseudo-labels, generated from the data itself, to create expressive unsupervised models without the need for carefully constructed input and label pairs. These approaches fall under the broad umbrella of self-supervision [47, 2, 40]. Self-supervision can be broadly classified into two categories: Contrastive Self-Supervised Learning (CSS) and Non-Contrastive Self-Supervised Learning (NCSS). CSS based approaches employ the use of positive and negative samples to optimize a contrastive loss to learn in a latent space [38].

Combining Self-Supervision with recent adversarial approaches creates new possibilities with respect to stabilizing the training dynamics of these adversarial methods. Self-supervision helps the designer encode domain knowledge into the training paradigm.

In the multi-step generative frameworks, the discriminator is used as a way to help improve the RL agent's performance by artificially creating a competition. Some popular algorithms in this domain fall under the umbrella of Adversarial Imitation Learning (AIL). Some examples of popular algorithms in this domain are GAIL [20], AIRL [9] and f-MAX [10]. The primary objective in these algorithms is to optimally act in an environment in a manner that is indistinguishable from the expert. The algorithms are modeled as a 2-player competitive game between an agent and a discriminator. The discriminator tries to distinguish agent trajectories from expert trajectories. The agent tries to fool the adversary by generating trajectories that are similar to expert trajectories. At convergence, the agent trajectories closely mimic the expert trajectories and the adversary cannot distinguish between them. Other popular applications that use an adversary as a tool are those of Robust RL or Adversarial attacks in RL [25, 36, 51].

In the single-step generative category, the discriminator functions in a similar manner; however, the generator in this setting only takes a single step prior to evaluation. This setting can be thought of as a simplification of the multi-step setting, akin to the connection between Reinforcement Learning and contextual bandits [30, 59, 48].

In this work, we will explore how to go about incorporating self-supervision into common generative approaches, and how these methods improve the baseline performance from an empirical perspective. In particular, we look at the following scenarios:

1. **Self-Supervised Generative Adversarial Imitation Learning (SS-GAIL)**: The current landscape of imitation learning is broadly dominated by two families of algorithms - Behavioral Cloning (BC) and Adversarial Imitation Learning (AIL) [35]. BC approaches suffer from compounding errors, as they ignore the sequential decision-making nature of the trajectory generation problem [45]. Furthermore, they cannot effectively model multi-modal behaviors [7]. While AIL methods solve the issue of compounding errors and multi-modal policy

training, they are plagued with instability in their training dynamics [23]. In this work, we address this issue by introducing a novel self-supervised loss that encourages the discriminator to approximate a richer reward function. We show that our method (SS-GAIL) outperforms prior state-of-the-art methods on real-world prediction tasks, as well as on custom-designed synthetic experiments. We prove that SS-GAIL is part of the family of AIL methods by providing a theoretical connection to cost- regularized apprenticeship learning. Moreover, we leverage the self-supervised formulation to introduce a novel teacher forcing-based curriculum (Trajectory Forcing) that improves sample efficiency by progressively increasing the length of the generated trajectory. The SS-GAIL framework improves imitation capabilities by stabilizing the policy training, improving the reward shaping capabilities, as well as providing the ability for modeling multi-modal trajectories.

2. **TroGAN**: There has been an increased interest in employing sketches as a user-friendly representation to control the out- put of a generative model. Sketch Your Own GAN [52] focuses on rewriting a generative model to output realistic images that are semantically similar to a handful of fixed, user- defined sketches. However, there are failure cases – GANSketching [52] cannot replicate images from complex poses or distinctive, minimalist art styles (e.g. "the Picasso horse"). We hypothesize that this is occurring because the user-sketch distribution is considerably different from the generated sketch distribution, given the underlying dataset. To rectify these failure cases, we propose a method that builds upon the GAN Sketching architecture by introducing a translation model that shifts the distribution of fake sketches to be more similar to that of the user-sketches while also retaining the essence of the initially generated image. Such a formulation avoids overfitting by the discriminator, thus reducing the discriminability and increasing gradient propagation. We also experiment with translating the user-sketches instead and discuss how the choice in the direction of translation affects generator performance. Finally, we show that our method reduces the number of training steps required when compared with GAN Sketching.

Chapter 2

Background

Generative Models have been shown to be exceptionally proficient in mimicking the behavior of an unknown distribution solely from bootstrapped data. These models are usually trained to optimize a distinguishability loss that is estimated by a learned discriminative model. This approach is known as adversarial training, and it involves the joint optimization of a generator and a discriminator [13]. These two models compete against each other according to a min-max objective function defined by the adversarial training regiment [12]. However, with modern deep learning architectures, the exceptional function-approximating ability of these models can quickly destabilize the training dynamics by overfitting in either the minimization or maximization stage of the two player game [42]. In this work, we explore the use self-supervision as a way to stabilize the training dynamics and avoid overfitting.

2.1 Single-Step Generative Models

GANSketching focuses on rewriting a generative model to output realistic images that are semantically similar to a handful of fixed, user-defined sketches. However, there are failure cases – GANSketching cannot replicate images from complex poses or distinctive, minimalist art styles (e.g. "the Picasso horse") [52]. This is believed to occur because the user-sketch distribution is considerably different from the generated sketch distribution, given the underlying dataset, as illustrated in Fig. 2.1. Such a formulation does not avoid overfitting by the discriminator, as the sketch distributions and the distribution after the mapping network might differ considerably. We illustrate this issue in Fig. 2.1. This reduces the discriminability and reduces gradient propagation. This method would naturally require a large number of training steps required when compared with GANSketching.

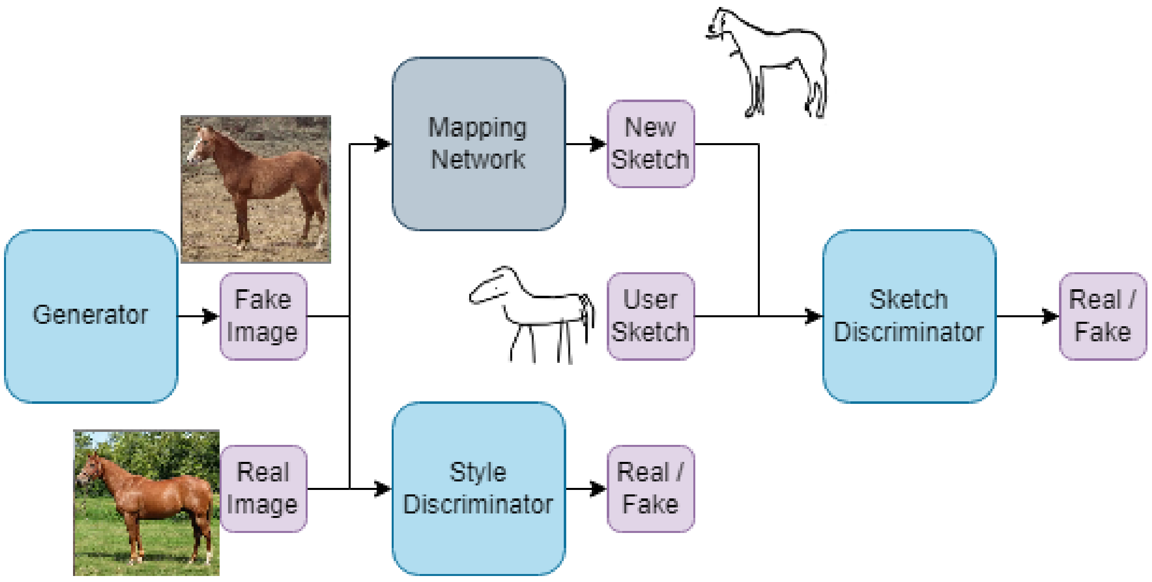


Figure 2.1: GAN Sketching Architecture: The main forward pass involves sampling an image from the generator, passing it through an encoding network or mapping network, followed by a sketch discriminator and a style discriminator. The learned networks are illustrated as Blue squares, while frozen differentiable networks are shown in grey. The purple boxes are the intermediate pictures after the corresponding networks, and we show a sample example of how it would look at a given stage in the pipeline.

2.1.1 Rewriting Generative Models using Sketches

Being able to control the output of a generative model would enable engineers and scientists to create realistic images in a matter of seconds, which would have otherwise taken weeks or months to synthesize [11]. Moreover, there has recently been an increased interest in employing sketches as a user-friendly representation to control the output of a generative model [52]. Rewriting generative models [3, 53] is one such approach that enables fast adaptation of pre-trained models for customized image synthesis. The framework presented in this paper builds upon the GAN rewriting work by Wang [52].

The GAN sketching framework illustrated above involves initializing a generative model with a set of pre-trained weights, followed by a training regiment that encourages the model to match a set of user defined sketches. The architecture is illustrated in Fig 2.1. The type of object to mimic (e.g. horse, cat, etc.) is either known beforehand or identified by a human-drawn user-sketch. A generator and two discriminators are trained end-to-end. The framework employs two soft constraints that are defined by two adversarial losses – a style loss and a sketch loss. One discriminator provides a style loss of the generated image using a dataset of real images of the object, and the other one (the main focus of our paper) provides a loss of the sketch outlines of the generated image against one or a handful (single-sketch and multi-sketch paradigm, respectfully) human-drawn sketches. The sketch of the generated image are provided by an image to sketch translation network (a pre-trained pix2pix [22] network). Note that the network must train end-to-end for every new sketch(es) provided. In this paper, "sketches" or "fake sketches" will refer to the output of fake images under this translation network, "human/user-sketch" refers to the human-drawn sketch, and "contours" (not yet introduced) will refer to the sketch-translation of real images under some image translation network.

The flow of the algorithm involves first sampling a generated image $G(z)$ from the generator. Next, this image is passed into a mapping or encoding network represented by $\mathcal{F}$. In GAN Sketching [52], the mapping network is initialized using a pre-trained network similar to Photosketch[32] and identifies the contours in the image. The output at this stage $F(G(z))$ would look similar to a sketch drawn by a human, and is thus used as the fake images for training a sketch discriminator. The real user-sketches used to train the sketch discriminator would come from a subset of the QuickDraw dataset [5]. Finally, the original sampled image $G(z)$ is passed into a style discriminator along with other real images to ensure that the style remains intact.

2.2 Multi-Step Generative Models

In this work, we consider sequential decision making problems, and we model them using the framework of Markov decision processes (MDP). An MDP can be represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, T)$ with state-space $\mathcal{S}$, action-space $\mathcal{A}$, dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $r(s, a)$, initial state distribution ρ_0, and horizon T.

Imitation learning [37] methods aim to learn task policies directly from expert demonstrations. The family of imitation learning algorithms is broadly divided into two classes, behavioral cloning (BC) [1] and Adversarial Imitation Learning (AIL) [9]. BC directly regresses from the expert's states to its decisions; however, such a straightforward supervised learning approach ignores the sequential nature of the problem and policy errors cascade during execution.

Adversarial IL methods formulate the imitation learning problem as an adversarial game between the policy and the discriminator. The discriminator measures some divergence between the expert and policy's state-action distribution, and the policy aims to fool this discriminator. For instance, GaIL [20] minimizes the JS divergence between the expert and policy's state distribution. Their algorithm implements a min-max optimization procedure,

$$\min_{\pi \in \Pi} \max_{D \in (0,1)^{S \times \mathcal{A}}} -\lambda_H H(\pi) + \mathbb{E}_{\pi}[\log(D(s, a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s, a))] \tag{2.1}$$

AIL methods provide a compelling approach to imitation since they do not face the issue of compounding errors. However, such adversarial methods are known to be unstable to train. Our work aims to deal with this instability while retaining the advantages of AIL methods.

Previous work for multi-agent behavior prediction such as NRI [28] and DNRI [14] model the expert's behavior using a VAE [27] in which the encoder and the decoder networks are modeled as GNNs [29] and the latent code represents the intrinsic interaction graph. Since their network architecture is designed to represent and work on the interaction graph of the agents, they can be used to understand the relationships between the agents qualitatively. These models are trained using multi-step BC and, therefore, cannot model multi-modal behavior. This is a significant drawback since real-world behavior is often multi-modal. Our work builds upon their policy architecture by training them using an actor-critic algorithm in an AIL setting.

Chapter 3

TroGAN

3.1 Introduction

Prior to the advent of the camera, creating realistic pictures based on a particular stylistic penchant required artistic skill and meticulous attention to detail, which resulted in exorbitant costs and man-hours. Nowadays, with a smartphone in most people's pockets we have a plethora of realistic image generation methods. However, we still have not overcome the barrier of editing images to match a particular style, at scale. There are applications such as Photoshop [34] that can edit images; however, they require dozens of man-hours and cannot be scaled to millions of images. In this paper we intend on investigating a neural approach to editing realistic images and mass producing variations using relatively simple sketches as standard archetypes.

GANs have recently been shown to be powerful tools in creating generative models for image synthesis, and there has been interest in being able to control the generated outputs [18, 39, 52]. In the GANSketching model [52], a generator is trained to create fake images similar to a human-provided drawing (the "user-sketch"). The goal of their architecture is to allow users to create their own generative network from only a single drawing or a handful of sketches.

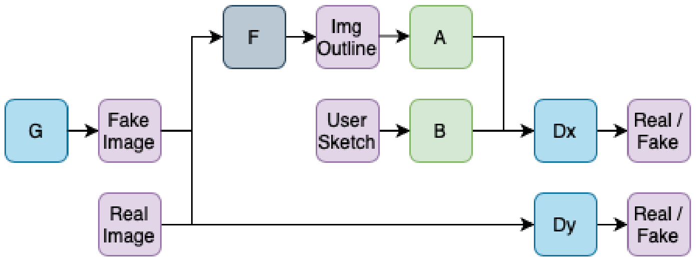
As is discussed later, our hypothesis is that the ability for the generator to learn involves fooling the discriminator by mimicking the semantics of the sketch, which becomes increasing difficult the farther apart the distributions between the user-sketch and the generated-contours are. If the distributions are considerably far apart, the discriminator's job becomes "easy" and the generator cannot learn as well due to low gradient magnitudes. This becomes the case for user-sketches that are far from the fake sketch distribution, e.g. for user-sketches of cats in difficult poses, or minimalist, over-simplified drawings with rare sketch styles.

Our goal therefore becomes to understand how to rectify failure cases (discussed in the previous chapter) by bringing these two distributions closer together. In a sense, we want to make it harder for the sketch discriminator to discern between the two distributions so that gradients can better help the generator replicate the user-sketch styles or poses. It is important to note; however, that making the discriminator's inputs indiscernible from the get-go also is not desired as the generator will get little to no training signal.

Failure Cases In GANSketching [52], the authors identify a few failure-cases for their architecture, one such case being user-sketches drawn with a distinctive style. They illustrate this issue with an example of a horse drawn to imitate the style of Picasso. This example demonstrates the sketch-space-mismatch issue with this framework, namely the problem arising from a distributional shift in the user-sketches and the generated sketches. We will build upon the issues arising from this mismatch shortly, while also presenting some empirical analysis in the Experiments 3.3 section.

The Picasso Horse [52] and other rare drawing styles example sheds light on the interesting problems surrounding sketch generation. This discussion motivates our choice to use an unpaired image-2-image translation model [38] as a method to bridge the gap arising from the distributional shifts between the sketch space of the model and the user-sketches.

Distributional Shifts The central objective behind the adversarial min-max problem [13] is to converge to a Nash equilibrium point which describes a state where the generator's outputs are indistinguishable from the ground-truth samples. However, if there is still a considerably distinct deviation between the two distributions when the model converges, then a discriminator might overfit to the classification objective, thereby resulting in vanishing gradients passing back to the generator. This can be fixed with approaches such as increasing the number of generator update steps; however, this doesn't solve the underlying issue of the current GANSketching [52]



(a) Generalized experimental template.

Variant	A	B
v1	SketchRNN	Identity
v2	Identity	SketchRNN
v3	Identity	U-Net
v4	Identity	CUT
v5	CUT	Identity

(b) A table of architectural variants (which networks go in A,B in 3.1a).

Figure 3.1: (Left) the general template for our experimental architectures. Rectifying the distribution between user-sketches and fake-sketches involve putting image-translation models (ResNet18) in slots A or B. (Right) a summary of experimental variants in terms of what networks go in slots A,B in 3.1a

framework on rare drawing styles. The issue with the current framework is that the fake sketches arising from the generated samples cannot match the user-sketches as they are from entirely different distributions even after the model has converged.

This leads us back to the issues discussed in the previous sub-section and motivates the design choices that we will present in the following sections. We aim to solve the distributional mismatch issue with an unpaired image-2-image translation model [38], thus resulting in the possibility for the generator to potentially converge to a Nash equilibrium. Furthermore, this avoids the overfitting issue that we would otherwise see when the discriminator optimizes over semantically inappropriate features, which would not result in meaningful weight updates for the generator.

3.2 Method

In order to rectify the distributional shift, we introduce a translation model that modifies one sketch distribution as described in Figure 3.1a. Our goal is to modify either sketch distributions in a way that closes the gap between the two distributions, while also retaining the original meaning of the user-sketch. We acknowledge that incorporating a translation model between the output of the mapping network F and the sketch discriminator could cause sparse gradients due to the depth of the pipeline.

(a) TroGAN S2C iteration=0

(b) TroGAN S2C iteration=10K

(c) TroGAN S2C iteration=20K

Figure 3.3: TroGAN S2C Samples Psi0.5

3.2.1 Architectural Variants

While we initially planned to use the translation model to modify ground-truth user-sketches, we also experiment with rectifying the distribution shift by transforming the user-sketches as shown in Figure 3.1. The general template of our modified GANSketching model is shown in Figure 3.1a and the variants listed in 3.1b. We also try experimenting with biasing the training dataset for GANSketching by identifying nearest-neighbor real images to the user-sketch (in the sketch domain). Note that we did not implement v1, v2, or v3, but discuss observations or anticipations of each variant below.

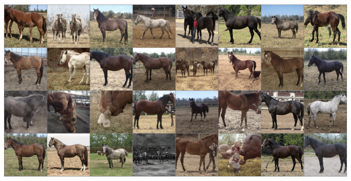
3.2.2 Differences between A and B

We have a choice of inserting an image translation model in either A or B, while leaving the other slot empty. If a model is inserted into A, then it is intended to serve as fake sketch to user-sketch image translation. The general idea is to pre-train a translation network to make fake sketches look more like user-sketch domain, and these variants are v1 & v5, as shown in 3.1b. On the other hand, if a model is inserted into B, it is intended to translate sketches from the user-sketch to the fake sketch distribution. Ideally a pre-trained model is used, and these variants are v2,v3, and v4, as shown in 3.1b.

3.2.3 SketchRNN Variants (v1, v2)

Inspired by the observation that fake sketches usually have more numerous and smooth strokes than user-sketches, we hypothesized that a recurrent model such

(a) GANSketching iteration=0


(b) GANSketching iteration=10K

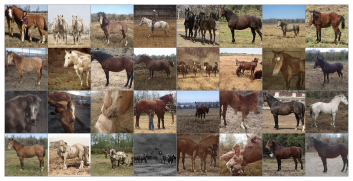

(c) GANSketching iteration=20K

Figure 3.4: GANSketching Baseline Samples Psi0.5

as SketchRNN [16] will sequentially add sketches to the user-sketch to look more like fake-sketches (v2). Conversely, we might imagine a scheme where strokes are sequentially removed from the fake-sketch to look more like the user-sketch (v1). Although recurrent networks would theoretically add/remove strokes one by one in this hypothesized manner, v1 inserts a RNN in an already deep pipeline and training may suffer from vanishing gradients. Both variants need to train with multi-step rollouts and the number of steps may add additional complexity as an unknown hyperparameter. Furthermore, pre-training these network requires training with image pairs, which requires human labeling or some semi-supervised paradigm with nearest neighbor search. In any case, we decided to employ non-recurrent image-to-image translation models.

3.2.4 U-NET Variants (v3)

U-nets are shown to be powerful tools for image-to-image translation [44], so our v3 variant proposes to insert a U-net model in slot B, where the network is pre-trained to translate user-sketches to fake sketches. The main difficulty with this model is that it still requires image pairs to translate. Unlike SketchRNN, U-net is not a recurrent model and requires no rollouts, but the image-pair creation problem still persists and we would like to proceed with unsupervised translation models.

3.2.5 CUT Variants (v4, v5)

The contrastive unpaired translation model (CUT) [38] is a good model candidate since it does not require supervised pair of images. It works by training an encoder-decoder generation model to translate image styles while learning to keep corresponding patches of the input and output semantically similar. The use of CUT will eliminate multi-step rollouts as in SketchRNN and supervised image pairing during pre-training. v4 introduces a CUT model in B, and v5 introduces a CUT model in A. For v4 since we are transforming user-sketch to look like fake sketches (which often have non-horse related strokes), the translated image may not be semantically correct, although having the style of these fake sketches. We implement v4 and v5 and identify some good results with v5 later, which still may need more samples or training.

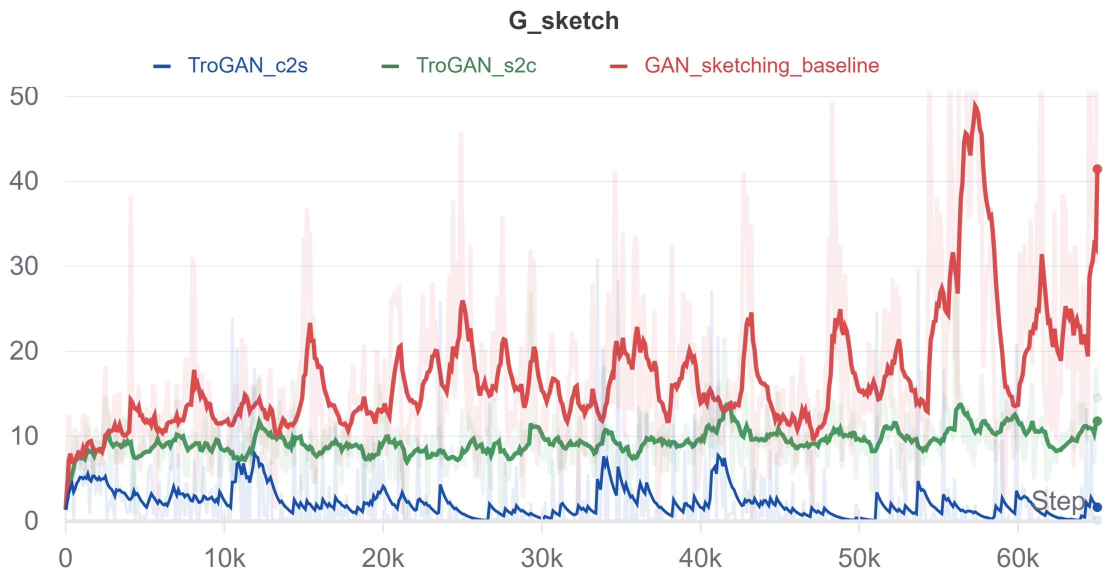


Figure 3.5: The Generator sketch loss defined as: softplus($-D(G(z))$). A lower loss indicates an increased predilection by the discriminator into believing the generated images are indeed real. We observe higher mean and variance in the baseline (red) compared to the two TroGAN architectures (blue and green).

3.3 Experiments

We hypothesize that introducing an image-2-image translation model to the GANSketching [52] model will rectify the distributional shift between the generated sketch and user-sketch, which will thereby result in improved performance – both qualitatively and quantitatively. In this section we present experiments that justify our claim, along with a few failure cases as well.

3.3.1 Evaluations

Datasets. To compare our model against the GANSketching baseline we needed to first construct an unpaired image-2-image translation dataset, consisting of horse contours and sketches. We collected 200 horse contours by sampling 400 pictures of horses from the LSUN Dataset [56], passing them through the QuickDraw model [5], and selecting the top 200 qualitatively appropriate horse contours.

Next, we sampled a subset of 25,000 horse images from the LSUN Dataset for evaluating our image loss. We have included instructions on how to download this dataset in the repository released along with this paper. Finally, we obtained a set of 200 sketches from the QuickDraw dataset [5] by obtaining JSON files with stroke information and rendering the vector images. Using this process, we qualitatively evaluate the sketches and select the best 200 for our task.

Performance Metrics. Apart from a qualitative evaluation of our models, we also evaluate the number of horses that were initially in an incorrect configuration, and the number that remained incorrect after 20,000 epochs.

We also plot the Generator Sketch Loss, which is described as: $\text{softplus}(-D(G(z)))$, where $G(z)$ is the generated sketch and D is the sketch discriminator. This loss quantitatively evaluates the degree to which the generator is able to fool the sketch discriminator. In other words, the generator sketch loss evaluates the number of false positives by this discriminator.

3.3.2 Generator Sketch Loss

In Fig. 3.5, we plot the generator sketch loss for the GANSketching Baseline, the TroGAN sketch-to-contour (s2c, v4) model, and the TroGAN contour-to-sketch (c2s, v5) model. The loss is defined as $\text{softplus}(-D(G(z)))$, where G is the generated sketch and D is the discriminator output given a sketch. This loss quantitatively evaluates the degree to which the generator is able to fool the sketch discriminator. This metric was chosen as a quantitative approach to validate our hypothesis, namely that introducing an image-2-image translation model would allow for better gradient propagation, and lower generator losses. We notice that the baseline has considerably greater mean and variance compared to the both TroGAN models. Though these results must be taken in conjunction with the qualitative analysis, we show that the introduction of a CUT [38] model improves training dynamics.

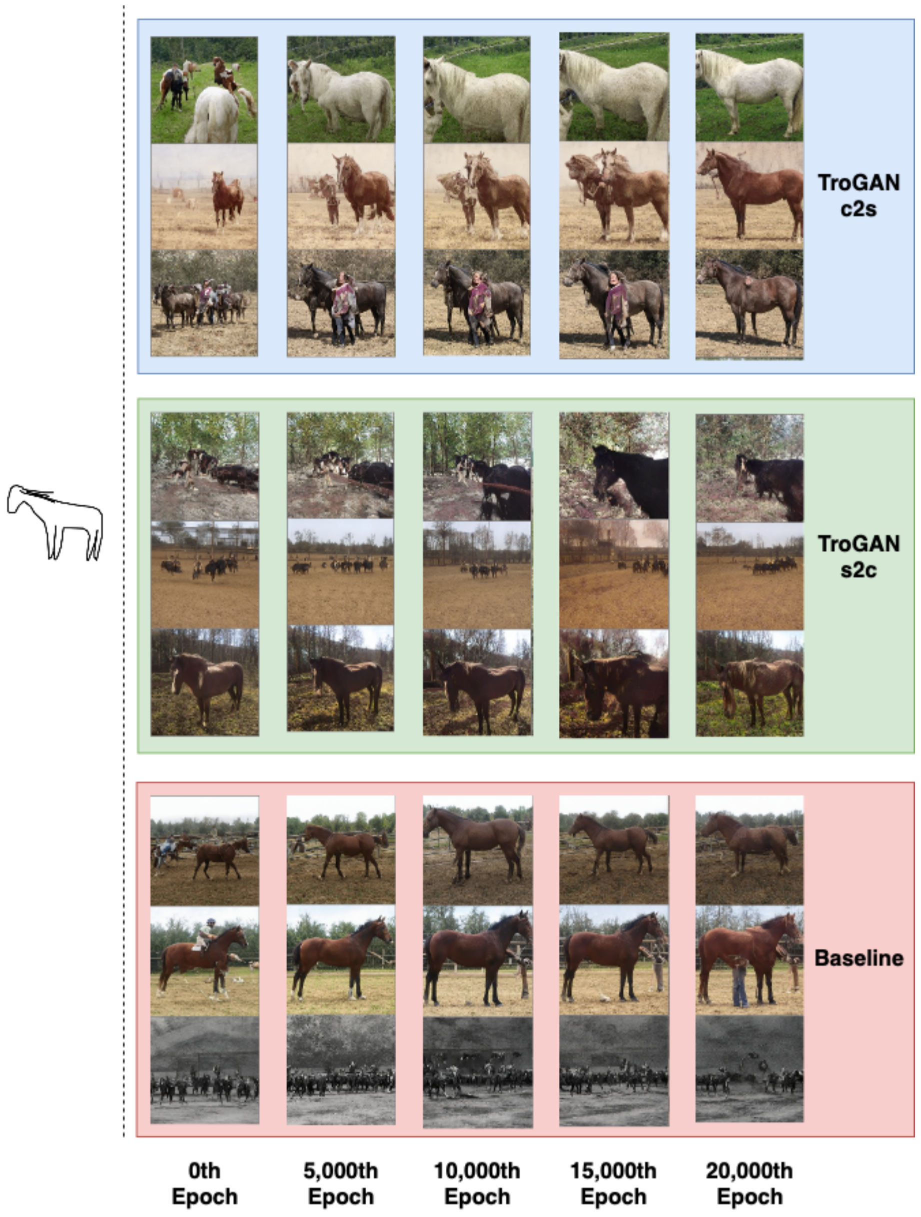


Figure 3.6: We illustrate randomly sampled outputs from the generator for time-steps at multiples of 5,000 for the baseline (red), the TroGAN s2c (green), and TroGAN c2s (blue). We show the sketch used for GAN rewriting to the left of the dotted line.

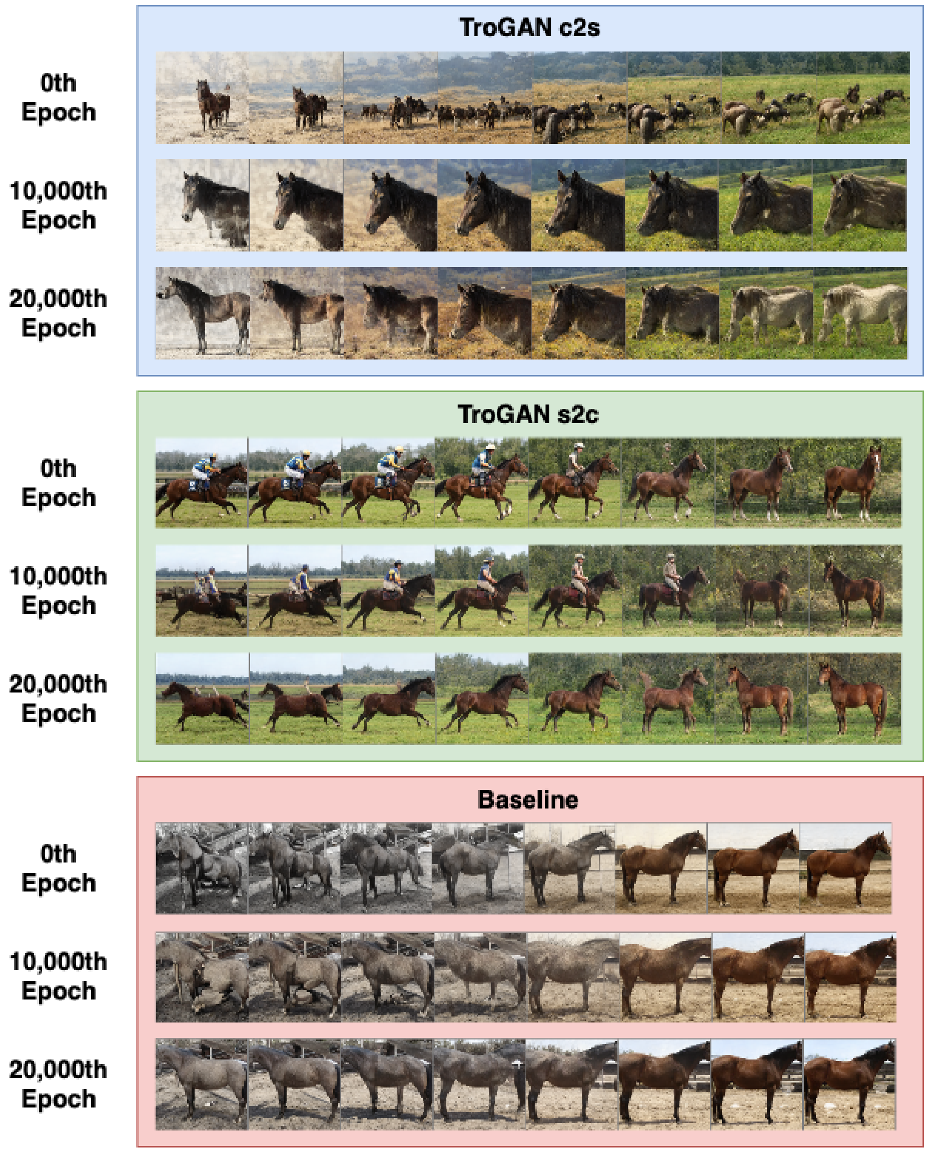
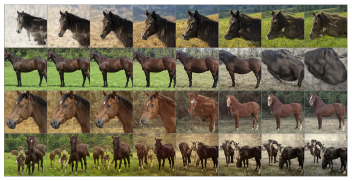


Figure 3.7: We illustrate randomly sampled interpolations between outputs from the generator for time-steps at multiples of 10,000 for the baseline (red), the TroGAN s2c (green), and TroGAN c2s (blue). The sketch used for GAN rewriting is the same as in Fig 3.6.

3.3.3 Faster Convergence

As described in Fig. 3.5, the TroGAN architectures have richer gradients, thereby resulting in faster convergence. We can see the results of this in Fig. 3.6. We

(a) TroGAN C2S iteration=0


(b) TroGAN C2S iteration=10K

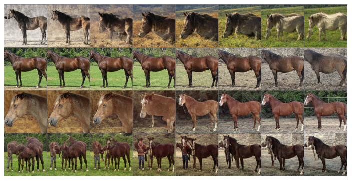
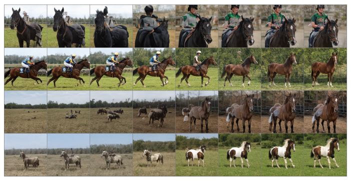
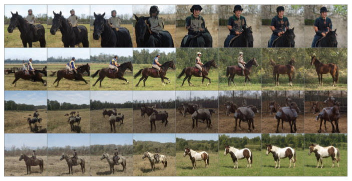

(c) TroGAN C2S iteration=20K

Figure 3.8: TroGAN C2S Interp Psi0.5


(a) TroGAN S2C iteration=0


(b) TroGAN S2C iteration=10K

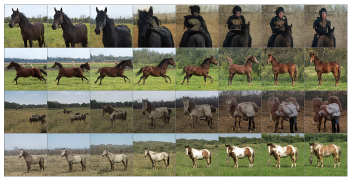

(c) TroGAN S2C iteration=20K

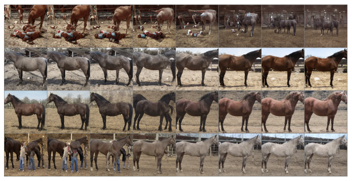
Figure 3.9: TroGAN S2C Interp Psi0.5

notice that the baseline and TroGAN s2c have still not converged even after 20,000 epochs. However, the TroGAN c2s model has indeed converged. We have added more samples in the Appendix for further reference. We measure convergence based on the qualitative likeliness to the sketch – illustrated to the left of the dashed line in Fig. 3.6.

3.3.4 Interpolations

In this experiment, we show some interpolated examples between the horse samples. As expected, the TroGAN architectures fare better than the baseline. However, we can

(a) GANSketching iteration=0


(b) GANSketching iteration=10K

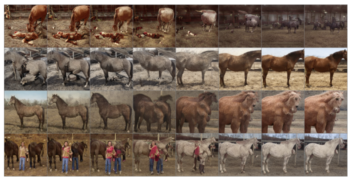

(c) GANSketching iteration=20K

Figure 3.10: GANSketching Baseline Interp Psi0.5

see that not all interpolations are semantically similar to the left-facing horse sketch. There could be various explanations, and one is overfitting by the architecture over exclusively the sampled images. This could mean that the bulk of the architecture is in fact the same, while only portions of the state space have been modified to look like the sketch.

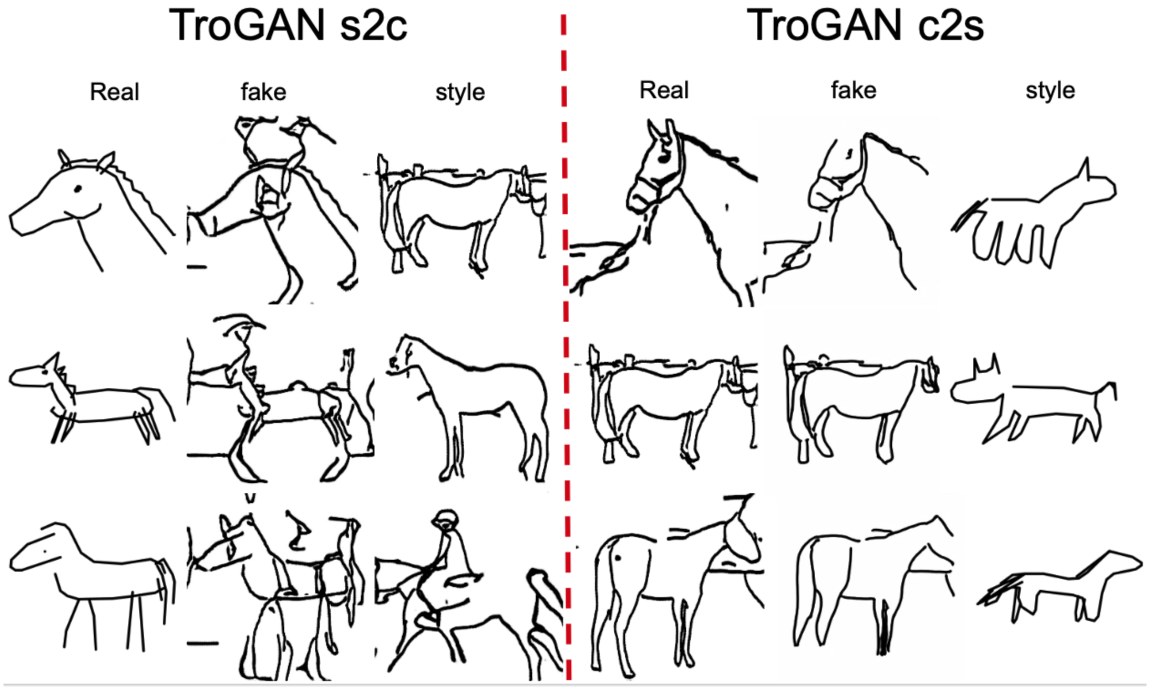


Figure 3.11: We show samples of contour and sketch translations after training, along with the style the model is trying to imitate.

3.3.5 Semantically Incorrect Lines

We can train the translation model to transform sketches to contours as well contours to sketches. However, we observe that the TroGAN c2s model vastly outperforms TroGAN s2c. This can be attributed to the fact that the translation model from sketches-to-contours adds spurious details to the sketches, which need not be semantically correct. We show a few examples in Fig. 3.11. The addition of these lines that may not be semantically appropriate would be picked up as causal features by the Discriminator, and would be used for classification. However, these features were just stylistic renditions introduced by the CUT model to mimic the style of a contour.

This significantly hampers the performance of the overall model, as seen in the results present previously. Thus, translating from contours-to-sketches is the optimal process, as the spurious features are removed, thus resulting in only the important features being taken into account by the Discriminator. This conclusion is bolstered by the improved results seen by TroGAN c2s over its s2c counterpart.

3.3.6 Nearest Neighbors

Another idea we implemented was biasing the training dataset for our v4 model (CUT translates user-sketch to fake sketches). The hypothesis is that if the training images look more like the user-sketch semantically, then there will be less generation diversity, and the fake distribution will be semantically closer to the desired target in the user-sketch domain.

We start with a pre-trained CUT model that translates user-sketches to fake sketches and keeps the translated user-sketch as a template. Then we translate the entire image dataset to the contour domain by using the F translation network used in [52]. Finally, we find the contours that have the smallest Chamfer Distance to the template in the sketch domain. Note that for simplicity, we find the contours of the sketches using OpenCV and compare these points to find the closest corresponding images. Using 1/5 nearest neighbors of the original $\tilde{2}$5K dataset to the template, training v4 using the picasso horse did not give results better than v4 on the original dataset (TroGAN_s2c in 3.5).

3.4 Conclusion

In general, we outline five methods 3.1 to rectify the distributional shifts between the sketch discriminator inputs, in order to assist in the generation of images that are semantically similar to hard/rare user-sketches. We implement and test two methods involving using CUT [38] to translate either distribution. Our results show that for the user-sketches that we tested upon, the output of the TroGAN c2s generator converges to the desired pose faster than the GANSketching model [52]. However, the picasso-horse sketch still proves to be difficult to emulate, and persists as a failure case in our experiments. This can be attributed to the issue of identifying *why* a failure case is a failure case, which is rather difficult. Our modified architecture seems to converge faster on normal input but does not verify our hypothesis on user-sketches with rare or distinctive styles. Without further testing, it is unclear whether modifying the sketch discriminator inputs in this way converges at different speeds for user-sketches closer or farther from distribution. We leave this to future work.

Chapter 4

SS-GAIL

4.1 Introduction

Training an agent to imitate an expert is a promising approach to learning intelligent behavior and can be used in applications such as autonomous driving and robotic manipulation. More specifically, the ability of the agent to robustly learn optimal policies in real-world scenarios is a current challenge facing the field. The most promising approaches for imitation learning are Behavioral Cloning (BC) and Adversarial Imitation Learning (AIL). BC methods have been shown to produce compounding errors [45], which makes it unsuitable for complex applications. Adversarial Imitation learning methods, such as GAIL [20], iteratively train a discriminator, and use it as a proxy reward function for updating a policy. We show that the proxy reward function learned by GAIL fails to provide dense supervision for policy updates and leads to inefficient and unstable training.

Let's consider the example task of training two agents to draw the planar letters "ML" on a piece of paper, with the expert trajectories shown in red, and the initial policy shown in blue in Fig. 1(a). Training an RL agent to imitate the expert policy would ideally require a rich reward landscape with a clear gradient starting from the current policies, in the lower half of the image, and terminating at the expert sketches, as depicted in Fig. 1(b). However, the reward function learned by GAIL is almost constant throughout the state space, while abruptly changing at the decision boundary, as depicted in Fig. 1(c). In theory, GAIL would be able to learn the

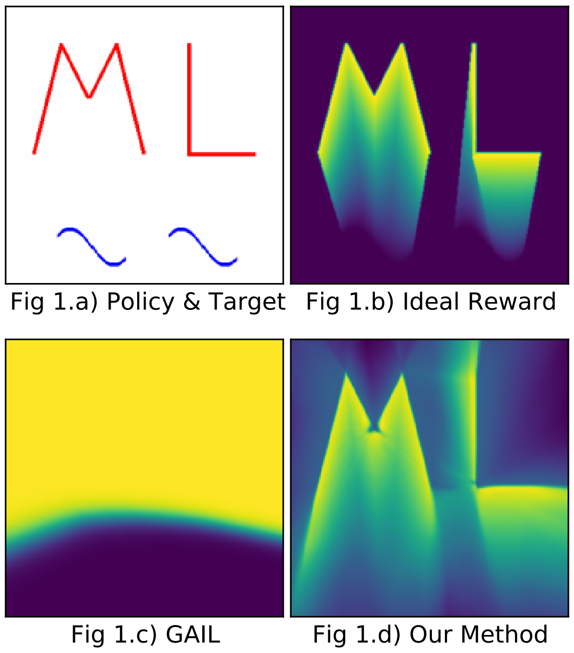


Figure 4.1: (a) An example task that starts with an initial policy shown in blue, and aims to imitate the expert trajectories shown in red. (b)-(d) show the reward landscapes of the ideal reward function, the learned discriminator in GAIL, and our method (SS-MAIL), respectively.

optimal policy with sufficient exploration; however, in practice, the training dynamics resulting from the sparse rewards in the local neighborhood of the current policy lead to sub-optimal policies. We see that apart from a minuscule sliver of the state space, the agent is left in the dark when it comes to a prospective policy update. This leads us to wonder how the agents will ever learn to imitate the expert if they are left to explore in the dark. There have been attempts such as WAIL [54] to address this issue; however, it is also plagued with unstable critic-training, which we will elaborate upon in the upcoming sections. To briefly motivate the rest of the paper, we tease the results of our methods (SS-MAIL and SS-GAIL) in Fig. 1(d), and we will explain in the upcoming sections the details of the framework that helped us get these results.

The problem illustrated in the toy example above would be exacerbated in larger state spaces. We see that despite the recent progress in Adversarial Imitation Learning (AIL), the application of methods such as GAIL in scenarios with larger state spaces may be impeded by the exponential cost of exploring in ever-increasing hyper-spaces.

The exploding cost of exploration would make tasks such as self-driving and robotic manipulation infeasible. These shortcomings are addressed in the SS-MAIL method introduced in this work.

In this work, we address these issues by introducing a novel self-supervised loss that encourages the discriminator to approximate a richer reward function. We employ our method to train a graph-based multi-agent actor-critic architecture that learns a centralized policy, conditioned on a learned latent interaction graph. We show that our methods (SS-MAIL and SS-GAIL) outperform prior state-of-the-art methods on real-world prediction tasks, as well as on custom-designed synthetic experiments. We prove that SS-MAIL is part of the family of AIL methods by providing a theoretical connection to cost-regularized apprenticeship learning. Moreover, we leverage the self-supervised formulation to introduce a novel teacher forcing-based curriculum (Trajectory Forcing) that improves sample efficiency by progressively increasing the length of the generated trajectory. The SS-MAIL and SS-GAIL frameworks improves expert imitation capabilities by stabilizing the policy training, improving the reward shaping capabilities, as well as providing the ability for modeling multi-modal trajectories.

4.2 Method

In this section we will introduce SS-MAIL, and its components, namely: **1) SS-GAIL:** a self-supervised AIL method to train the Discriminator, **2) MAIL:** an off-policy graph convolutional actor-critic architecture that imitates the expert using supervision from the Discriminator, **3) Trajectory Forcing:** a teacher forcing based curriculum generator for stabilized training dynamics.

4.2.1 SS-GAIL: Self-Supervised Generative Adversarial Imitation Learning

From Apprenticeship Learning to SS-GAIL

[20] show that GAIL is a form of apprenticeship learning with a specific cost regularization, which optimizes the min-max objective:

$$\psi_{\mathrm{GA}}^{*}\left(\rho_{\pi}-\rho_{\pi_{E}}\right)=\max_{D \in(0,1)^{\mathcal{S}} \times \mathcal{A}} \mathbb{E}_{\pi}[\log (D(s, a))]+\mathbb{E}_{\pi_{E}}[\log (1-D(s, a))] \tag{4.1}$$

In practise, the rewards from the discriminator do not provide rich policy-gradients, which result in sub-optimal policies. We address this issue by introducing a self-supervised loss function for our discriminator that encourages it to provide an enriched reward signal, not only in the local neighborhood of the current trajectory, but globally as well.

Our self-supervised loss is presented in Eq. 3, below. We sample $\alpha \in [-1, 1]$ and take the weighted average of the trajectories resulting from the policy and the expert, specifically $\tau_G \sim \pi$ and $\tau_E \sim \pi_E$. This new trajectory can be mathematically formulated as $\tau_\alpha = \alpha\tau_G + (1-\alpha)\tau_E$. Our intuition behind creating the self-supervised loss is that the state action pairs of τ_α should regress smoothly based on their relative distance to τ_G, or in other words $\mathbb{E}_{\tau_\alpha}[D_\theta(s, a)] = \alpha$. Thus, we convert the Binary Cross Entropy loss to a Mean Squared Error (MSE) loss and use the sampled α values as self-supervised labels.

$$\psi_{\mathrm{SS}}^{*}\left(\rho_{\pi}-\rho_{\pi_{E}}\right)=\mathbb{E}_{\pi, \pi_{E}, \alpha}\left[\underbrace{\left(0-D\left(s_{G}, a_{G}\right)\right)^{2}}_{\text{Generated MSE}}+\underbrace{\left(1-D\left(s_{E}, a_{E}\right)\right)^{2}}_{\text{Expert MSE}}+\underbrace{\left(\alpha-D\left(s_{\alpha}, a_{\alpha}\right)\right)^{2}}_{\text{Self-Supervised MSE}}\right] \tag{4.2}$$

An interesting consequence of this formulation is that the third term in the self-supervised optimization objective, the self-supervised MSE term, serves as a natural from of exploration during the policy optimization stage.

Theorem 1. *SS-GAIL is an instantiantion of cost-regularized apprenticeship learning, i.e. the policy at the saddle point* (π, D) *of the min-max problem described above is a solution to* $\min_\pi \max_c(-\psi(c) + E_\pi[c(s,a)] - E_{\pi_E}[c(s,a)])$ *for some specific* ψ*.*

Proof. Ho and Ermon show that the cost-regularized apprenticeship learning problem

$$\mathrm{AL}_\psi(\pi_E) = \min_\pi \max_c -\psi(c) + E_\pi[c(s,a)] - E_{\pi_E}[c(s,a)] \tag{4.3}$$

the above is equivalent to min-max problem

$$\min_\pi \psi^*(\rho_\pi, \rho_E) \tag{4.4}$$

where ψ^* is the convex conjugate of the cost regularizer ψ.

For SS-GAIL,

$$\psi^*_{\mathrm{SS}}(\rho_\pi - \rho_{\pi_E}) = \mathbb{E}_{\pi,\pi_E,\alpha}\left[\underbrace{(0 - D(s_G, a_G))^2}_{\text{Generated MSE}} + \underbrace{(1 - D(s_E, a_E))^2}_{\text{Expert MSE}} + \underbrace{(\alpha - D(s_\alpha, a_\alpha))^2}_{\text{Self-Supervised MSE}}\right] \tag{4.5}$$

The SS-GAIL algorithm aims to solve the min-max optimization problem

$$\max_\pi \min_D \mathbb{E}_{\pi,\pi_E,\alpha}[(0 - D(s_G, a_G))^2 + (1 - D(s_E, a_E))^2 + (\alpha - D(s_\alpha, a_\alpha))^2]$$

using the gradient descent-ascent algorithm. The inner discriminator optimization is a supervised learning problem with the self-supervised loss. The outer policy optimization is an off-policy policy improvement problem, and we use the soft actor-critic algorithm.

Therefore, SS-GAIL is an instantiation of cost-regularized apprenticeship learning with the cost regularizer ψ_{SS}. □

Algorithm 1 SS-GAIL

```
Input: Expert Trajectories $\tau_E \sim \pi_E$, Initial Policy $\pi_\phi$, Initial Discriminator $D_\theta$
Initialize: Policy $\pi_\phi$, Discriminator $D_\theta$
while Policy Improves do
    Sample Trajectories $\tau_G \sim \pi_\phi$
    Sample $\alpha$ and Compute: $\tau_\alpha = \alpha\tau_G + (1-\alpha)\tau_E$
    Update $D_\theta$ using gradient:
        $\mathbb{E}_{\tau_G}\left[\nabla_\theta \left(D_\theta(s,a)\right)^2\right] + \mathbb{E}_{\tau_E}\left[\nabla_\theta \left(1 - D_\theta(s,a)\right)^2\right] + \mathbb{E}_{\tau_\alpha,\alpha}\left[\nabla_\theta \left(\alpha - D_\theta(s,a)\right)^2\right]$
    Update $\phi$ with SAC to increase the following objective: $\mathbb{E}_{\tau_i \sim \{\tau_G, \tau_\alpha\}}\left[D_\theta(s,a)\right]$
```

Algorithm

In the SS-GAIL algorithm we start by sampling a trajectory from the current policy. Next, we compute the weighted average between the current trajectory and a sampled expert trajectory. We denote this trajectory as τ_α. According to our Discriminator update rule, derived previously, we train our discriminator to learn a smooth reward function by reconstructing the self-supervised labels, α. Next, we need to update our policy function.

In GAIL, [20] use the on-policy TRPO update rule. However, according to the min-max function derived above, our policy should be optimized on not only the current trajectory, but also τ_α. This serves as an inherent exploration term for our policy training, as we are exploring states that are disparate from the current trajectory. Thus, we use the SAC update policies to accommodate the off-policy updates from τ_α.

We formulate our update rules such that the discriminator converges faster than the actor, similar to the two-timescale approach in the actor-critic setting. Thus, we no longer require balancing the discriminator, as seen in GAIL. Also, the change in our Discriminator values converges to zero, as opposed to WAIL, which constantly updates the fluid surface of the Discriminator output, as there is no grounding of the outputs to any specific value.

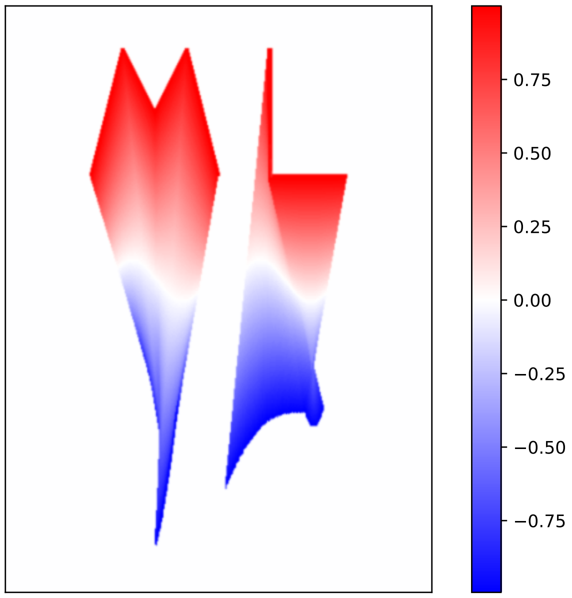


Figure 4.2: Vector field induced by negative α.

Self-Supervision as a form of Reward-Shaping

Our novel self-supervised loss gives us considerable flexibility in the sampling approaches used to obtain α. As stated previously in our discussion on the discriminator approximating a vector field, we noticed that only taking α values from [0,1] would be insufficient for training as during exploration, the agent would explore states that would naturally correspond to negative α values. As the discriminator was not trained on these states, generalization isn't guaranteed. Thus, to complete the neighborhood around the current policy trajectory, we also consider negative α values. We show an empirical analysis in the next section. The negative α values assist in inducing a complete vector field around the neighborhood of our trajectory, as shown in Fig 3.

We can custom design the sampling approach and the loss functions to suit our need, which will open up greater flexibility and control for the community to experiment with.

In 4.2, we observe that sampling $\alpha \in [-1, 1]$ was not sufficient for reproducing a drop in the reward function after the expert trajectories. Thus, we increased our sampling range to $[-1, 1.5]$ and set the reward function to 0 for values of $\alpha > 1$. This helped our discriminator learn a reward function that does not monotonically increase as we get farther from the current policy.

We can always split our sampling procedure to get different results. We can make use of the non-linearity of the neural-network architectures to even approximate piece-wise distributions. Another extension could be to have steeper slopes farther away from the current policy and smoother slopes closer to the policy state-distribution.

4.2.2 MAIL: Multi-Agent Imitation Learning

In this section we introduce the architecture of the policy network, which is divided into two parts - the graph encoder and the graph soft actor-critic (G-SAC). In our architectures, we use the message passing standard established by [28], and model the node-to-edge and edge-to-node message passing operations as follows:

$$\mathbf{h}^l_{(i,j)} = f^l_e\left(\left[\mathbf{h}^l_i, \mathbf{h}^l_j, \mathbf{s}_{(i,j)}\right]\right); \qquad \mathbf{h}^{l+1}_j = f^l_v\left(\left[\sum_{i\in\mathcal{N}_j} \mathbf{h}^l_{(i,j)}, \mathbf{s}_j\right]\right) \tag{4.6}$$

where the subscripts of (i, j) represent edge related features, while subscripts denoted with single letters, such as i and j represent node related features. h represents the corresponding edge or node embedding, and s represents states. Finally, f represents a neural network function approximator, and $\mathcal{N}_j$ corresponds to indices of the neighboring nodes connected to the node indexed by the subscript, which in this example is j.

Graph Encoder

The goal of the Graph Encoder is to infer the underlying interaction graph of the various agents, conditioned upon the observed history. This latent interaction graph will be used as weights in the graph convolution step in G-SAC. Our Graph Encoder is based on the encoder network [14]. We employ a fully connected graph convolutional network to output a distribution over edge weights. However, unlike in DNRI, we do not include prior networks that train for precognition of the future evolution of the graph based on the observed history. Our model takes in as input the cumulative observable state space, and employs an LSTM to keep track of the history. We model this as follows:

$$\mathbf{h}^{t+1}_{(i,j),\text{enc}} = \text{LSTM}_{\text{enc}}\left(\mathbf{h}^{t}_{(i,j)}, \mathbf{h}^{t}_{(i,j),\text{enc}}\right), \tag{4.7}$$

$$\underbrace{q_\phi\left(\mathbf{z}^{t}_{(i,j)} \mid \mathbf{x}\right)}_{\text{Graph Encoder Output}} = \text{softmax}\left(f_{\text{enc}}\left(\mathbf{h}^{t}_{(i,j\rangle,\text{enc}}\right)\right) \tag{4.8}$$

Graph Soft Actor-Critic (G-SAC)

G-SAC samples a dynamically computed interaction graph from the Graph Encoder and uses the graph as the activations for the respective edges among the nodes of the graphs. It then computes a graph convolutional to obtain the mean and standard deviation values, similar to the output of SAC [17]. We also include another head which functions as a critic. The critic head approximates the Q-function of the policy and is used to train the network. As in SAC, we also keep a target critic, which is updated based on polyak averaging.

$$\mathbf{h}^{t}_{(i,j\rangle} = \sum_k z_{ij,k} f^k_e\left(\left[\mathbf{x}^t_i, \mathbf{x}^t_j\right]\right)$$

$$\boldsymbol{\mu}^{t+1}_j = f_\mu\left(\sum_{i\neq j}\mathbf{h}^{t}_{(i,j)}\right); \qquad \boldsymbol{\sigma}^{t+1}_j = f_\sigma\left(\sum_{i\neq j}\mathbf{h}^{t}_{(i,j)}\right);$$

$$\underbrace{p\left(\mu^{t+1}_j, \sigma^{t+1}_j \mid \mathbf{x}^t, \mathbf{z}\right)}_{\text{G-SAC Output}} = \mathcal{N}\left(\boldsymbol{\mu}^{t+1}_j, \sigma^2\mathbf{I}\right)$$

4.2.3 Trajectory Forcing

Teacher Forcing based curriculums are an important tool for training sequence models [43]. In a sequence generation setting, we expect the learned models to use the outputs of the previous timesteps as inputs. However, training such generative models was shown to be sensitive to weight intitialization [58], in the absence of pedagogical intervention, such as teacher forcing. Scheduled Sampling [4] is a curriculum based on teacher forcing that attempts to gradually transition from using the ground-truths as inputs, to using the model's previous outputs. NRI and DNRI use teacher forcing in their approaches; however, such an approach could result in compounding errors during test time, due to the distributional shift. The use of such curriculums is not

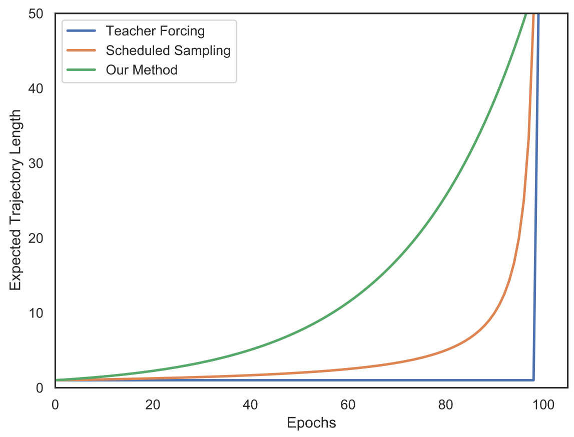


Figure 4.3: We plot the expected length of the generated trajectory during training in between pedagogical intervention for teacher forcing, scheduled sampling and trajectory forcing. We show that our curriculum provides a gradual increase in expected trajectory lengths for better policy training.

prevalent in AIL, as the frameworks accommodate only for on-policy updates. There are methods that try to work around this by pre-training using BC [23]. However, in SS-GAIL, the off-policy functionality is built into the formulation. Thus, we can leverage α values close to 1 and use them as a proxy for teacher forcing. Our goal is to take advantage of the exploration functionality of our loss function to gradually reduce the teacher forcing frequency to zero. In other words, our curriculum progressively increases the intervals of pedagogical intervention. We show this in Section 4.5.

We mathematically model the frequency of interventions as $1.5^{-\text{epoch}/\beta}$, where beta is a hyperparameter that assists in generating a progressively increasing sequence. Our intuition behind modeling the length of the generated trajectory as an exponential with respect to the epoch, is that we intend on doubling the size of the generated trajectory every β epochs. A linear model would result in reduced sample efficiency during the later stages of training, as the generalizability of the model would outpace the curriculum. The exponential increase ensures efficient data utilization.

4.3 Results

In this section, we quantitatively evaluate the strengths of our framework using two custom-built environments and a real-world dataset. These experiments illustrate the advantages of the SS-MAIL framework, namely: i) increased stability of its training dynamics, ii) multi-modal trajectory modeling capabilities, iii) increased sample-efficiency of Trajectory Forcing, iv) enhanced versatility in reward-shaping using self-supervision, and v) robustness to compounding errors on real-world datasets.

4.3.1 Experimental Setup

To evaluate the capabilities of our methods against prior SOTA baselines, we evaluate our framework on one custom-built environment (Y-Junction), and one real world prediction task (Noisy Mocap). To minimize the prevalence of confounding variables, the experiment design of Y-Junction was simplified considerably to ensure that any promising improvement in performance is warranted by the inherent attributes of the frameworks, and not any other external confounding factors, such as variable overflow [8]. The Y-Junction environment is a trajectory forecasting environment that tests the multi-modal capabilities of AIL and BC based frameworks by simulating a 3-way Y-Junction scenario. The environment has three agents that follow one another on a one-way street until the beginning of a fork or Y-Junction. Then the lane splits into two, thus simulating two potential modes. Our goal is to pick one of the two lanes or modes. We provide more details in the appendix. The Noisy Mocap prediction task involves training a multi-agent policy on the CMU Motion Capture Dataset [6] for subject #35, and observing the zero-shot generalization in the presence of noisy inputs.

4.3.2 Stability in Training Dynamics

Robustness of the training algorithm to initialization, stochasticity in the environment, and training data is very important to assure predictability in the training dynamics. The Y Junction experiment, upon which we evaluate the robustness, provides a simple multi-agent testing scenario to illustrate the differences between various AIL approaches. To have a uniform evaluating strategy, we use the same architecture for all methods, and solely swap-out the corresponding AIL loss functions. We elaborate further in the Appendix.

In Fig. 4, we observe that the training loss of SS-GAIL reduces to zero, implying that the algorithm is able to successfully imitate the expert during training. SS-GAIL

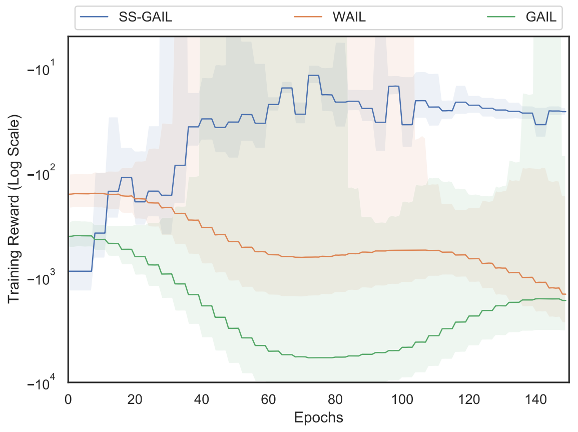


Figure 4.4: Training error over time in the Y-Junction environment shows that SS-GAIL successfully imitates the expert. The low standard deviation of error for SS-GAIL despite unfavorable initialization demonstrates the increased robustness of its training dynamics.

quickly converges to the expert policy for all initializations; however, GAIL and WAIL are unable to converge despite good policy initializations. We see that GAIL and WAIL start-out with better performances, due to their policy initializations; however, are unable to consistently improve and fluctuate considerably. As the Y-Junction experiment is unbounded in its state-space, there is no boundary that constrains diverging policies, thus resulting in the observed exploding losses. In contrast, we see that the performance of the policy trained with SS-GAIL consistently improves, and converges to zero. These results can be attributed to the richer family of reward functions being approximated by the Discriminator. The Binary Cross Entropy loss of GAIL [20] results in sparse reward signals for the policy training, thus resulting in unstable training. Furthermore, the inablility of the WAIL Discriminator to converge upon a designated output surface results in a non-stationary value-function, thereby destabilizing the training of the critic. This instability becomes more pronounced as the policy and expert approach each other in the space of trajectories, as small variations in the surface of the discriminator output would lead to prominent shifts in the value-function.

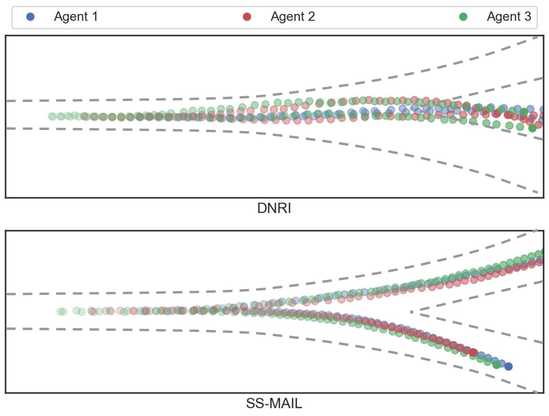


Figure 4.5: A visualization of the multi-modal trajectories learned by SS-GAIL and DNRI on the Y-Junction task. We observe that DNRI averages over the different modes, while SS-GAIL successfully differentiates between them. The inability to distinguish between multi-modal expert trajectories can prove disastrous in continuous state settings, as this may lead to visiting states that differ considerably from the expert state-distribution.

4.3.3 Multi-Modal Capabilities

Differentiating between the various modes in the expert-trajectory dataset is essential for training multi-modal policies. The gradient of the policy training loss should be in the direction of a specific mode, instead of the weighted average over all the modes. The Y Junction example provides a simple multi-agent testing scenario that demonstrates how BC based methods learn sub-optimal policies in the presence of multiple modes. To evaluate SS-GAIL and DNRI on their multi-modal modeling capabilities, we ensure both algorithms have similar policy architectures and differ solely in their BC and AIL based policy training steps.

We plot the trajectories in Fig 5, and illustrate issue of modal averaging, observed in BC methods. We observe that SS-GAIL converges to the different expert modes, implying that the Discriminator successfully approximates a reward function that has a discernible preference among prospective modalities. We do not observe this property in BC based DNRI algorithm, as shown in Fig. 5. The self-supervised loss function has an inherent positive feedback loop that progressively increases the gradients of the rewards for imitating the closest expert modality. This property is a direct consequence of the self-supervised loss, as the slope of the reward is inversely

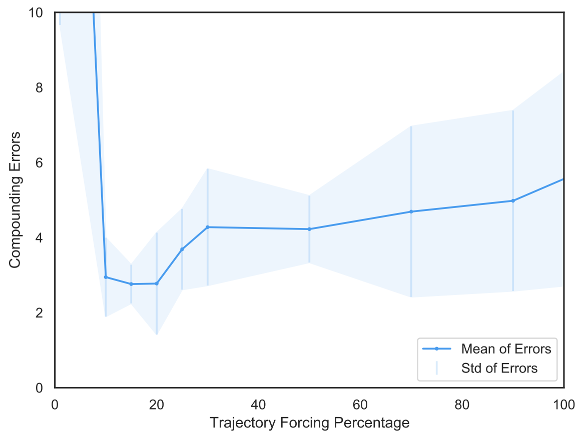


Figure 4.6: Running an ablation, over β, for the mean and standard deviation of testing error in the Y-Junction environment illustrates the existence of β values that result in both low mean and standard deviation. This highlights Trajectory Forcing's ability to improve the robustness to unfavorable weight initializations, increase the sample efficiency, as well as eliminate the issue of compounding errors.

proportional to the distance between the policy and expert trajectories.

4.3.4 Sample Efficiency and Curriculums

Teacher Forcing alleviates the issues of unfavorable initializations and low sample efficiency during the initial stages of training; however, it results in compounding errors during testing. The Y Junction experiment is a simple multi-agent scenario that demonstrates how Trajectory Forcing alleviates unfavorable initializations, increases sample efficiency during the initial stages of training, and eliminates the issue of compounding errors. To evaluate the effectiveness of the Trajectory Forcing curriculum, we train SS-GAIL with different values of β to show an ablation over β, which ranges from no Teacher Forcing (0%), seen in AIL approaches, to complete Teacher Forcing (100%), seen in BC approaches. In Figure 6, we focus on two aspects of the plot, the mean and standard deviation. A smaller mean loss, averaged over multiple random seeds, implies robustness in policy training to different model initializations. Moreover, a low standard deviation implies superior generalization during testing, and thereby lower compounding errors. we see that without teacher forcing the mean and standard deviation of the testing loss is considerably high, which is common for AIL methods. Teacher Forcing reduces the mean loss considerably; however, the

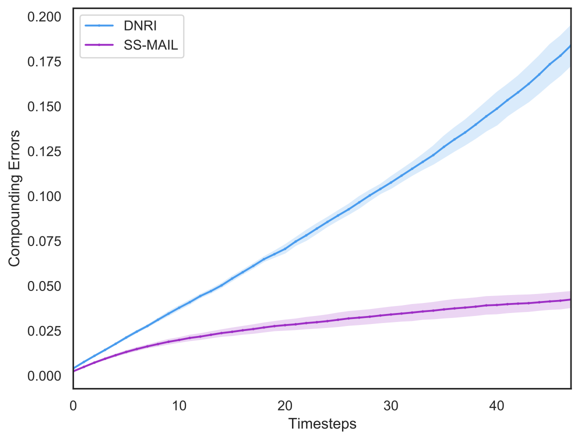


Figure 4.7: We plot the Compounding Errors over time for the Noisy Mocap environment shows that SS-GAIL successfully compensates for noisy inputs during test time, zero-shot. The low standard deviation and the decrease in slope for SS-GAIL illustrate its generalization capabilities.

standard deviation is high due to the compounding errors. We observe that for a β value of around 15%, the trajectory forcing curriculum considerably reduces the mean loss as well as the standard deviation during testing. This demonstrates the robustness to weight initializations, improvement in sample efficiency, and reduction in compounding errors.

4.3.5 Zero-Shot Generalization on Noisy Real-World Data

Compounding errors occur when the model is unable to correct for deviations from the training distribution of states, and fails to compensate for the observed deviations. This can be attributed to poor generalization, and is commonly caused by noise. The Noisy Mocap experiment is a real-world multi-agent prediction dataset that demonstrates how negligible amounts of Gaussian noise, with standard deviation of 0.05, can accumulate and permanently derail the generation of trajectories. To evaluate the issue of compounding errors in the presence of noise during test time, we evaluate SS-GAIL and DNRI using similar policy architectures.

In this experiment, we observe that DNRI progressively deviates further from the expert trajectories as the trajectory length increases. We also observe a positive correlation between the standard deviation and the trajectory length. However, in

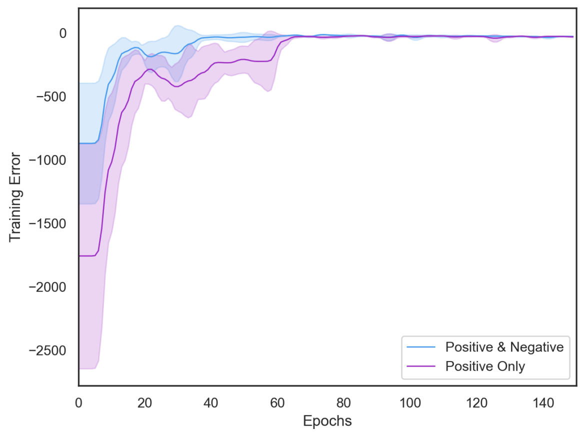


Figure 4.8: We plot the training loss over epochs for the Noisy Mocap environment and show that sampling negative α values improves the final loss and speeds up training. This can be attributed to the richer reward gradient in the local neighborhood of the current trajectory.

the case of MAIL we do not observe a similar linear increase in the compounding errors. Moreover, the slope decreases with the increase in trajectory length. This observed deviation can be attributed to the loss functions used to train the respective algorithms. BC approaches are trained to precisely match the expert. Despite having a lower validation loss, generalizing zero-shot to noisy environments is not guaranteed, as seen in Fig 7. SS-GAIL, on the other hand, learns a policy that maximizes the cumulative discriminator return. Thus, the trained policy is akin to a vector field surrounding the expert state-distribution, as it learns to optimize for reaching rewarding states, thereby resulting in robust recovery even in noisy environments.

4.3.6 Reward Shaping using Self-Supervision

Reward-Shaping provides engineers and scientists with a way to meticulously control the reward function, resulting in more efficient policy training and reduction in training time. In our method, the self-supervised sampling allows for precise control over the reward function the discriminator is approximating. In this experiment, we investigate one such customization of the reward function that results in enhanced training dynamics. The Noisy Mocap experiment provides us with a real-world dataset to investigate the ramifications of different sampling procedures on the training dynamics of SS-GAIL. To evaluate the effect of the self-supervised sampling, we keep the overall SS-GAIL algorithm constant and solely change the sampling distributions of α.

We see that sampling only positive values of α results in training dynamics that take longer to converge. We observe that with the introduction of negative α values, the training dynamics substantially improves. This demonstrates the importance of reward shaping using self-supervision, which opens up the possibilities for researchers to design more intricate reward function approximators. The improved performance observed upon the introduction of negative α values can be attributed to the resulting completion of the reward function in the local neighborhood of the current trajectory. As the current trajectory corresponds to an α value of zero, then the absence of negative α values during self-supervised sampling would result in an incomplete reward function in the local neighborhood of the policy's state-distribution.

The inclusion of negative α values is one such example of reward-shaping for enhanced performance and stable training dynamics. Based on the application, engineers and scientists can design custom rewards for the enhancing the policy training, based on the application constraints.

4.4 Related Work

In this section we provide a brief overview of the various research contributions that share similarities to the approaches described in our work. We start by discussing the recent work on graph-structured data, and then delve into multi-agent trajectory forecasting, and finally, imitation learning.

Graph structured data does not possess a fixed structure, and thus can not be used with traditional neural networks. To address these issues, there have been approaches such as [33], [46], and [29] that introduce the notion of message passing to accommodate for the irregular, yet structured nature of graphical data. In this paper, we use Graph Convolutional Networks [29] for our internal message passing; however, there are other methods proposed, such as Graph Attention Networks [49], and Graph Recurrent Networks (GRN) [19] as prospective forms of message passing. GRN also provides the notion of recurrence, which can be used to model time-series data. The Graph Convolutional RL [24] work is a relevant bridge between graph structured data and the sequential decision making setting, which employs message passing to deal with multi-agent reinforcement learning. Finally, we move on to the multi-agent trajectory prediction task. This setting provides is an interesting application of message passing, as the nature of the interactions among the agents may or may not be known a priori. Some recent work are: Evolve Graph [31], Social Attention [50], SocialGAN [15], and STGAT [21].

Adversarial imitation learning methods train a discriminator network to estimate the divergence between the expert and the policy's state distribution, and use this discriminator for policy improvement. [10] provide an unified perspective on this family of algorithms and show that they can be used to minimize any f-divergence between the expert and policy state distributions. [9] extend this line of work, and present an inverse reinforcement learning algorithm based on this adversarial setup. [35] move away from this adversarial setup, and present an IRL algorithm that directly optimizes the reward function to minimize any f-divergence between the expert and policy state distributions. Another line of work focuses on utilizing meta-learning methods to enable few-shot imitation or reward inference [57, 55].

4.5 Implementation Details

We begin by delineating all the experimental details that are common to each of the experiments. Our code is built on Pytorch version 1.2, and we use Adam as our optimizer of choice. We follow the teacher forcing training regiment provided in DNRI [14] to train our baseline reference models for comparison.

We save the best models after every epoch, and use the models with the best validation loss for testing. Similar to [14], we normalize our inputs between -1 to 1 using the maximum and the minimum state norms. For all the experimental evalutaions, we average the results over 5 seeds.

4.6 Reproducibility

This section describes our efforts towards ensuring reproducibility of our experiments. First, we provide code (https://github.com/Aks-Dmv/circles.git) for reproducing all of our experimental results. The code contains documentation for how to generate and plot each experimental result in this paper. In addition, we also provide details about our experimental setup, training hyperparameters, and datasets used in the appendix, which should be sufficient for an independent implementation. All of our experiments are done on at least five seeds to ensure reproducibility. We use author-provided implementations of DNRI and SAC to ensure consistency. Our experiments were done using a single Quadro RTX 6000 GPU, and therefore, are reasonably easy to reproduce within a small computational budget.

4.7 Conclusion

We introduced SS-GAIL and SS-MAIL to effectively model multi-agent experts and improve the stability of their training dynamics compared to previous AIL methods. Our method comprises of:

1. SS-GAIL: a self-supervised cost-regularized apprenticeship learning framework that has considerably more stable training dynamics compared to previous AIL methods, while also providing the flexibility for reward-design.
2. MAIL: A graph convolutional multi-agent actor-critic framework, which aids in multi-modal trajectory generation and alleviates compounding errors.
3. Trajectory Forcing: a teacher forcing based curriculum that takes advantage of our SS-GAIL formulation to stabilize the training dynamics and alleviates the issue of domain shift between training and testing.

The improved stability of the training dynamics and the increased sample efficiency of SS-GAIL and SS-MAIL allows us to train policies that are robust to weight initializations, which would enable efficient training of multi-agent interactions. Moreover, the ability to handle compounding errors would ensure enhanced generalization during test time, in the presence of noisy inputs. Finally, learning from expert imitation provides a useful framework to train control policies in the absence of the ground truth reward function, and SS-MAIL can be applied to such multi-agent control setting, such as autonomous driving and robotic interactions to learn proficient policies that achieve human-level performance.

Chapter 5

Conclusions

In this work we explored the use of self-supervision for stabilizing the training of generative models. We explored the single-step and multi-step domains, and explored the issues and potential self-supervised solutions.

In the single-step domain, we outline five methods 3.1 to rectify the distributional shifts between the sketch discriminator inputs, in order to assist in the generation of images that are semantically similar to hard/rare user-sketches. We implement and test two methods involving using CUT [38] to translate either distribution. Our results show that for the user-sketches that we tested upon, the output of the TroGAN c2s generator converges to the desired pose faster than the GAN Sketching model [52]. However, the picasso-horse sketch still proves to be difficult to emulate, and persists as a failure case in our experiments. This can be attributed to the issue of identifying *why* a failure case is a failure case, which is rather difficult. Our modified architecture seems to converge faster on normal input but does not verify our hypothesis on user-sketches with rare or distinctive styles. Without further testing, it is unclear whether modifying the sketch discriminator inputs in this way converges at different speeds for user-sketches closer or farther from distribution. We leave this to future work.

For the multi-step domain, we introduce SS-GAIL and SS-MAIL to effectively model multi-agent experts and improve the stability of their training dynamics compared to previous AIL methods. Our method comprises of:

1. SS-GAIL: a self-supervised cost-regularized apprenticeship learning framework

that has considerably more stable training dynamics compared to previous AIL methods, while also providing the flexibility for reward-design.

2. MAIL: A graph convolutional multi-agent actor-critic framework, which aids in multi-modal trajectory generation and alleviates compounding errors.
3. Trajectory Forcing: a teacher forcing based curriculum that takes advantage of our SS-GAIL formulation to stabilize the training dynamics and alleviates the issue of domain shift between training and testing.

Bibliography

[1] Michael Bain and Claude Sammut. “A Framework for Behavioural Cloning.” In: *Machine Intelligence 15*. 1995, pp. 103–129.

[2] Joshua Batson and Loic Royer. “Noise2self: Blind denoising by self-supervision”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 524–533.

[3] David Bau et al. “Rewriting a deep generative model”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 351–369.

[4] Samy Bengio et al. “Scheduled sampling for sequence prediction with recurrent neural networks”. In: *arXiv preprint arXiv:1506.03099* (2015).

[5] Salman Cheema, Sumit Gulwani, and Joseph LaViola. “QuickDraw: improving drawing experience for geometric diagrams”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012, pp. 1037–1064.

[6] Graphics Lab Motion Capture Database CMU. “CMU Graphics Lab Motion Capture Database”. In: ().

[7] Nachiket Deo and Mohan M Trivedi. “Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1179–1184.

[8] Will Dietz et al. “Understanding integer overflow in C/C++”. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 25.1 (2015), pp. 1–29.

[9] Justin Fu, Katie Luo, and Sergey Levine. “Learning robust rewards with adversarial inverse reinforcement learning”. In: *arXiv preprint arXiv:1710.11248* (2017).

[10] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. “A divergence minimization perspective on imitation learning methods”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 1259–1277.

[11] Andrew S Glassner. *Principles of digital image synthesis*. Elsevier, 2014.

[12] Ian Goodfellow. “Nips 2016 tutorial: Generative adversarial networks”. In: *arXiv preprint arXiv:1701.00160* (2016).

[13] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).

[14] Colin Graber and Alexander G Schwing. “Dynamic neural relational inference”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8513–8522.

[15] Agrim Gupta et al. “Social gan: Socially acceptable trajectories with generative adversarial networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2255–2264.

[16] David Ha and Douglas Eck. “A neural representation of sketch drawings”. In: *arXiv preprint arXiv:1704.03477* (2017).

[17] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1861–1870.

[18] Danijar Hafner et al. “Dream to Control: Learning Behaviors by Latent Imagination”. In: *International Conference on Learning Representations*. 2019.

[19] Ehsan Hajiramezanali et al. “Variational graph recurrent neural networks”. In: *arXiv preprint arXiv:1908.09710* (2019).

[20] Jonathan Ho and Stefano Ermon. “Generative adversarial imitation learning”. In: *arXiv preprint arXiv:1606.03476* (2016).

[21] Yingfan Huang et al. “Stgat: Modeling spatial-temporal interactions for human trajectory prediction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6272–6281.

[22] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[23] Rohit Jena, Changliu Liu, and Katia Sycara. “Augmenting GAIL with BC for sample efficient imitation learning”. In: *arXiv preprint arXiv:2001.07798* (2020).

[24] Jiechuan Jiang et al. “Graph convolutional reinforcement learning”. In: *arXiv preprint arXiv:1810.09202* (2018).

[25] Parameswaran Kamalaruban et al. “Robust reinforcement learning via adversarial training with langevin dynamics”. In: *arXiv preprint arXiv:2002.06063* (2020).

[26] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410.

[27] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[28] Thomas Kipf et al. “Neural relational inference for interacting systems”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2688–2697.

[29] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[30] Akshay Krishnamurthy, Alekh Agarwal, and John Langford. “PAC reinforcement learning with rich observations”. In: *Advances in Neural Information Processing Systems* 29 (2016).

[31] Jiachen Li et al. “Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning”. In: *arXiv preprint arXiv:2003.13924* (2020).

[32] Mengtian Li et al. “Photo-sketching: Inferring contour drawings from images”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 1403–1412.

[33] Yujia Li et al. “Gated graph sequence neural networks”. In: *arXiv preprint arXiv:1511.05493* (2015).

[34] Lev Manovich. “Inside photoshop”. In: *Computational Culture* (2011).

[35] Tianwei Ni et al. “F-irl: Inverse reinforcement learning via state marginal matching”. In: *arXiv preprint arXiv:2011.04709* (2020).

[36] Tuomas Oikarinen, Tsui-Wei Weng, and Luca Daniel. “Robust deep reinforcement learning through adversarial loss”. In: *arXiv preprint arXiv:2008.01976* (2020).

[37] Takayuki Osa et al. “An algorithmic perspective on imitation learning”. In: *arXiv preprint arXiv:1811.06711* (2018).

[38] Taesung Park et al. “Contrastive learning for unpaired image-to-image translation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 319–345.

[39] Taesung Park et al. “Swapping autoencoder for deep image manipulation”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7198–7211.

[40] Lerrel Pinto and Abhinav Gupta. “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours”. In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 3406–3413.

[41] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[42] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[43] Marc’Aurelio Ranzato et al. “Sequence level training with recurrent neural networks”. In: *arXiv preprint arXiv:1511.06732* (2015).

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on*

Medical image computing and computer-assisted intervention. Springer. 2015, pp. 234–241.

[45] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[46] Franco Scarselli et al. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[47] Yu Sun et al. "Unsupervised domain adaptation through self-supervision". In: *arXiv preprint arXiv:1909.11825* (2019).

[48] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[49] Petar Veličković et al. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[50] Anirudh Vemula, Katharina Muelling, and Jean Oh. "Social attention: Modeling attention in human crowds". In: *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 4601–4607.

[51] Eugene Vinitsky et al. "Robust reinforcement learning using adversarial populations". In: *arXiv preprint arXiv:2008.01825* (2020).

[52] Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. "Sketch your own gan". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14050–14060.

[53] Yaxing Wang et al. "Transferring gans: generating images from limited data". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 218–234.

[54] Huang Xiao et al. "Wasserstein adversarial imitation learning". In: *arXiv preprint arXiv:1906.08113* (2019).

[55] Kelvin Xu et al. "Learning a prior over intent via meta-inverse reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6952–6962.

[56] Fisher Yu et al. "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop". In: *arXiv preprint arXiv:1506.03365* (2015).

[57] Lantao Yu et al. "Meta-inverse reinforcement learning with probabilistic context variables". In: *arXiv preprint arXiv:1909.09314* (2019).

[58] Lantao Yu et al. "Seqgan: Sequence generative adversarial nets with policy gradient". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.

[59] Zihan Zhang, Xiangyang Ji, and Simon Du. “Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon”. In: *Conference on Learning Theory*. PMLR. 2021, pp. 4528–4531.

[60] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.