# OptiGAN: Generative Adversarial Networks for Goal Optimized Sequence Generation

Mahmoud Hossam*, Trung Le*, Viet Huynh*, Michael Papasimeon† and Dinh Phung*
*Faculty of Information Technology, Monash University, Australia
†School of Computing and Information Systems, The University of Melbourne, Australia

***Abstract*—One of the challenging problems in sequence generation tasks is the optimized generation of sequences with specific desired goals. Current sequential generative models mainly generate sequences to closely mimic the training data, without direct optimization of desired goals or properties specific to the task. We introduce OptiGAN, a generative model that incorporates both Generative Adversarial Networks (GAN) and Reinforcement Learning (RL) to optimize desired goal scores using policy gradients. We apply our model to text and real-valued sequence generation, where our model is able to achieve higher desired scores out-performing GAN and RL baselines, while not sacrificing output sample diversity.**

***Index Terms*—Sequential Data, Generative Adversarial Networks, Reinforcement Learning, Policy Gradients.**

## I. Introduction

Learning to generate realistic sequences from existing data is essential to many artificial intelligence applications, including text generation, drug design, robotics, and music synthesis. In these applications, a generative model learns to generate sequences of different data types according to each task. For instance, natural language and speech are sequences of words or utterances, in robot motion planning, a trajectory is an action sequence learned from experiences or sensory data. Recently, there has been a growing interest in deep models for sequence generation following the success of Generative Adversarial Networks (GANs) [1] and Variational Autoencoders (VAEs) [2] in image generation tasks [3], [4], [5], [6], [7].

However, realizing the full potential of these models in aforementioned applications has many challenges, and one of these key challenges is the absence of mechanisms to optimize the generated outputs according to certain metrics or useful properties. Most of current work on generative sequence models mainly learn to "resemble" the data, meaning to generate outputs that are close to the real distribution. However, in many applications, we are not only interested in generating data similar to the real ones, but we need them to have specific useful properties or attributes. For example, in drug design, useful properties include solubility and ease of synthesis [8], [9]. In music generation, we might want the music to have specific pitch or tempo, or in text applications, the user might be interested in generating sentences according certain sentiment or tense [10]. Therefore, the lack of optimization mechanisms in current models hinders their practical use in wide range of real world applications.

In this paper, we propose a new sequential generative framework, named OptiGAN[1], that can generate sequences resembling those in a given dataset and achieving high scores according to an optimized goal (e.g., solubility and ease of synthesis in drug design). Our proposed framework leverage GAN for mimicking real data and policy gradient reinforcement learning (RL) [11] for optimizing a score of interest. It is very well-known that although GANs can resemble real data, they face the mode collapsing problem [12], [13], [14], [15], [16], hence leading to generate less diverse examples. To tackle this issue in the context of sequence generation, we propose a principled combination of maximum likelihood and GANs (see Section IV-A) in which we prove that in the final optimization problem, Kullback-Leibler (KL) and Jensen-Shannon (JS) divergences between real data distribution and generated data distribution are simultaneously maximized, hence relieving the mode collapsing problem, concurring with [13]. We then leverage policy gradient RL into our model for optimizing a score of interest according to a desired goal (see Section IV-B). We observe that when incorporating policy gradient RL to our current framework -which is based on GAN and maximum likelihood- the variance in estimating gradient is very high, hence leading to unstable training. To resolve this issue, we resort the Monte Carlo rollout in [17] with a slight modification (see Section IV-C).

We demonstrate the capacity of our OptiGAN in two applications: text generation (discrete data) and air combat trajectory generation (real-valued data). For text generation task, we aim to generate sentences resembling real sentences in a given text corpus, while optimizing the BLEU [18] score for obtaining better sentences from human justification. For aircraft trajectory generation task, we aim to generate a trajectory plan for air-combat maneuver scenario between two aircrafts and optimize the McGrew score [19] which reflects the tactic quality of aircraft trajectories in an air combat [19]. In both applications, we show that we can generate high quality outputs and achieve higher scores than current related models aided by the RL component, while preserving the diversity of generated outputs using our hybrid maximum likelihood GAN.

The main contributions of our paper include:

- We propose OptiGAN which has the following advantages: (i) *an end-to-end generative framework with incorporated goal optimization mechanism,* (ii) *general*

[1]Our code is available here : https://github.com/mahossam/OptiGAN

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

*formulation that can be used for wide variety of different goals and models*, and (iii) *optimizing for desired goals without sacrificing output sample diversity*.

- We investigate the problems of interest comprehensively and our findings would advance the understanding of the behavior when incorporating a generative model and a RL component. Specifically, we empirically found that if we apply pure RL component to maximize a score of interest, we might obtain generated examples with high scores but poor diversity. For instance, in the case of text generation, the model somehow cheats the BLEU score by generating sentences in which a few words repeated all the times. In contrast, if we apply only a generative model to resemble real data, we cannot achieve higher values for the score of interest. Our solution is to leverage both generative model and RL to simultaneously obtain realistic diverse outputs with good scores. In addition, to obtain diverse outputs, the applied generative model is able to avoid the mode collapsing problem, aided by our proposed maximum likelihood GAN.

## II. Background and related work

### A. Background

*Generative Adversarial Networks (GAN):* Generative Adversarial Networks (GAN) [1] use adversarial training between two players to learn the density function of input data. The goal of the first player, the generator $G$, is to get good at generating data that is close to the real data distribution $p_d(x)$. The goal of the second player, the discriminator $D$, is to distinguish real data from fake data generated by the generator. The standard GAN objective to optimize is the minimax game between $D$ and $G$ is :

$$\min_G \max_D \left(\mathbb{E}_{x\sim p_d} \log D(x) + \mathbb{E}_{z\sim p_z} \log(1 - D(G(z)))\right), \tag{1}$$

where $z$ is the random noise input to $G$ and $p_z$ is the prior distribution of the $z$. After the training is finished, the generator is used to generate data from any random input $z$.

*Reinforcement Learning using Policy Gradients:* Reinforcement learning [20] is a general learning framework in which an agent learns how to take actions to maximize cumulative future rewards in the environment. Rewards received by the agent during learning encourage it to learn a policy that maximizes cumulative rewards (the returns).

Policy gradients [20] are group of methods in reinforcement learning that enable optimizing future returns by direct optimization of the policy. The objective is to maximize the return rewards over an episode of $T$ time steps $J(\theta) = \mathbb{E}_\pi \left[U_t\right]$ , where $\pi$ is the "policy" and $U_t$ specifies the cumulative reward of an episode which is defined as follows:

$$U_t \doteq R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots + \gamma^{T-t} R_T,$$

where $\gamma$ is a discount factor, and $R_t$ is the reward received from the environment.

A policy could be parameterized by some parameters $\theta$ and be directly optimized through taking the gradient of $J(\theta)$ with respect to $\theta$ . This method is called policy gradients. A well-known policy gradients algorithm is REINFORCE [11], a Monte Carlo algorithm to find the optimal policy $\pi$. The model is updated via gradient ascent with:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[U_t \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi \left(a_t \mid s_t, \theta\right)\right] \quad , \tag{2}$$

where $a_t$ is an action chosen at time step $t$ by the agent's policy $\pi$ given the current state of the environment $s_t$.

### B. Related work

In general, sequential deep generative models are either based on the variational approximation of maximum likelihood (like Variational Autoencoders VAE [2]) or on GANs [1]. Models based on variational approximation [21], [22], [23], [24] are mainly based on autoregressive models like Long Short-Term Memory (LSTM) [25], incorporated into VAE training framework. These models were applied to many sequence generating tasks including handwriting and music generation. However, training VAE based models with autoregressive networks suffers from the problem of "*posterior collapse*", where the latent variables are often ignored, especially when trained for discrete data like text [26].

The other group of models based on GANs are mainly focused on discrete data like text. There are two main approaches for these models; they either use reinforcement learning [17], [27], or a fully differentiable GAN [28], [29], [30], [31]. The first approach uses policy gradients in an adversarial training framework. The other approach however, employs a fully differentiable GAN network, where they use Gumbel-Softmax trick [32], [33] or distance measure on feature space [29] to overcome the non-differentiability problem for discrete data.

Some recent work tried to address the goal optimized generation problem. In [34], authors used convolutional GAN or autoregressive VAE to generate music with specific pitch and timbre. For text generation, [10] uses semi-supervised VAE approach to generate text based on sentiment and tense. Interest is growing as well in the biological sequences and drug design applications [35], where VAE latent space or GAN based model with reinforcement learning are used for molecular design. However, these models either optimize using a RL objective or employ feature learning in the generative VAE or GAN model. There is not much work on using RL to guide GAN learning for goal optimization, combining benefits of GAN unsupervised learning with goal optimization. MolGAN [8], is a recent work in that direction, that uses both GAN and RL for optimized graph generation. However, MolGAN uses a different RL technique to ours, that needs more network parameters to learn the rewards.

## III. Problems of interest

We demonstrate the capacity of our proposed framework in two applications of interest: *text generation* and *air combat trajectory generation*. For each application, our task is to generate sequences that achieve two concurrent goals: i) mimicking those in a given dataset and ii) obtaining high scores specified by an optimized goal which might be varied for specific tasks.

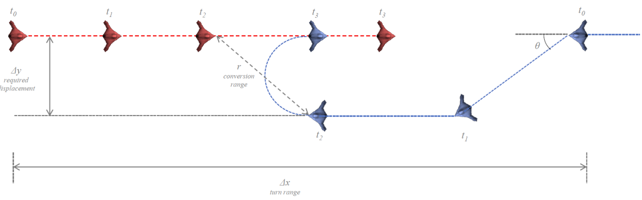

Fig. 1. “Stern Conversion” Flight Maneuver.

### A. *Text generation*

We need to generate sentences that are similar to real sentences in a given text corpus and have high quality from human justification. A well-known score used to measure the quality of generated sentences is BLEU score [18]. Specifically, the BLEU score for each sentence computes the ratio of n-grams generated from the model that matches with a true ground truth, called *reference* sentences and is defined as follows:

$$\text{BLEU(N)} = \sum_{n=1}^{N} \frac{Count(\text{ Model generated n-grams} \cap X_{test-ngrams})}{Count(\text{ Model generated n-grams})}$$

In our proposed model, beside generating realistic sentences, we also aim to maximize the BLEU score of generated sentences. As shown later, we utilize the BLEU score as reward function in our RL inspired framework.

### B. *Air combat trajectory generation*

For air combat missions, pilots are trained to conduct certain maneuvers according the combat situation they face. There are well known maneuvers that the pilots are trained on, either defensive, offensive, or neutral. We consider a specific air combat maneuver between two fighters called “**Stern Conversion**” maneuver [36]. In this maneuver, the opponent (the red aircraft) flies in a straight and level line, and does not detect the blue aircraft, while the blue aircraft, on the other hand, tries to get behind the opponent aircraft, in order to increase the chance to engage it (see Fig. 1).

In this specific task, in addition to generating realistic trajectories, we also need to maximize the McGrew score [19], which measures the score of how well an aircraft was doing relative to another aircraft in an attempt to get behind the other aircraft (refer to [19] for more detail). Due to security restrictions, we cannot access the real trajectories sensory data. Instead, we use *ACE-Zero [37]* air combat flight simulator to generate the training data. This *simulator* was developed by domain experts to imitate the real aircraft trajectories. As demonstrated in the experiments section, our model can generate novel trajectories with high McGrew score close to the average scores for ACE-Zero trajectories.

## IV. Proposed framework

In what follows, we present in details our proposed framework. We employ a neural autoregressive model $G$ (e.g., Bi-RNN or RNN) as a generator to map from a noise $z \sim p_z$ to a sequence that can mimic those in a given dataset and achieve high score corresponding to the desired goal. In terms of modeling, we start from the maximum likelihood (ML) principle and then propose to incorporate adversarial learning to the learning process in a principled way. The coupling of ML and adversarial learning principles helps us to generate realistic and diverse sequences to imitate those in a given dataset. Moreover, to reach high scores according to a given desired goal, we propose to incorporate policy gradient RL that allows us to train our model end-to-end. Finally, to stabilize the training process, we apply variance reduction technique when training with policy gradients. The final model is named OptiGAN whose overview architecture is shown in Fig. 2.

### A. *Maximum likelihood and adversarial training*

A sample $X$ in our setting is defined as a sequence of $T$ tokens denoted by $X = [x_1, x_2, ..., x_T]$, where we assume that all samples have length $T$. For our autoregressive model with model parameters $\theta$, the log-likelihood can be written as:

$$\log\, p_G(X \mid \theta) = \sum_{i=2}^{T} \log\, p_G(x_i \mid h_{i-1}, \theta) + \log\, p_G(x_1 \mid \theta),$$

This is the default neural autoregressive model formulation. Now we start introducing an adversarial learning framework for this model by introducing a latent variable $z$ to the autoregressive model, where we rewrite $\log\, p(x_1 \mid \theta)$ as marginalization over the $z$:

$$\begin{aligned} \log\, p_G(x_1 \mid \theta) = \log \sum_z p_G(x_1, z \mid \theta) \geqslant \qquad (3) \\ -I_{KL}(q(z \mid x_1, \phi) \,||\, p(z)) + \mathbb{E}_{q(z|x_1,\phi)}[\log\, p_G(x_1 \mid z, \theta)], \end{aligned}$$

where $I_{KL}$ is Kullback–Leibler divergence, $q(z \mid x_1, \phi)$ is an approximation of the posterior $p\,(z \mid x_1, \theta)$ and $p(z)$ is a prior distribution to $z$. The right hand side of Eq. (3) is a lower bound for $\log\, p_G(x_1 \mid \theta)$. We can then write $\log\, p_G(X \mid \theta)$ in terms of a lower bound as:

$$\begin{aligned} \log\, p_G(X \mid \theta) \geqslant \sum_{i=2}^{T} \log\, p_G(x_i \mid h_{i-1}, \theta) \qquad (4) \\ - I_{KL}(q(z \mid x_1, \phi) \,\|\, p(z)) + \mathbb{E}_{q(z|x_1,\phi)}[\log\, p_G(x_1 \mid z, \theta)]. \end{aligned}$$

We propose to incorporate adversarial learning to autoregressive sequential model in a principled way. One generator $G\,(z)$ and one discriminator $D\,(X)$ are employed to create a game like in GAN while the task of the discriminator is to discriminate true data and fake data and the task of the generator is to generate fake data that maximally make the discriminator confused. In addition, the generator $G$ is already available which departs from a noise $z \sim p_z$, uses the conditional distribution $p\,(x_1 \mid z, \theta)$ to generate $x_1$, and follows the autoregressive model to consecutively generate $x_{2:T}$. We come with the following minimax problem:

$$\begin{aligned} \max_G \min_D \Big[ \mathbb{E}_{X \sim p_d} [\log\, p_G\,(X \mid \theta)] - \mathbb{E}_{X \sim p_d} [\log\, D\,(X)] \\ - \mathbb{E}_{z \sim p_z} [\log\, [1 - D\,(G\,(z))]] \Big], \qquad (5) \end{aligned}$$

where the generator $G$ consists of the decoder $p\,(x_1 \mid z, \theta)$, the autoregressive model, hence $G$ is parameterized by $(\theta, \phi)$, and $\log\, p_G\,(X \mid \theta)$ is substituted by its lower bound in Eq. (4).We can theoretically prove that the minimax problem in Eq. (5)

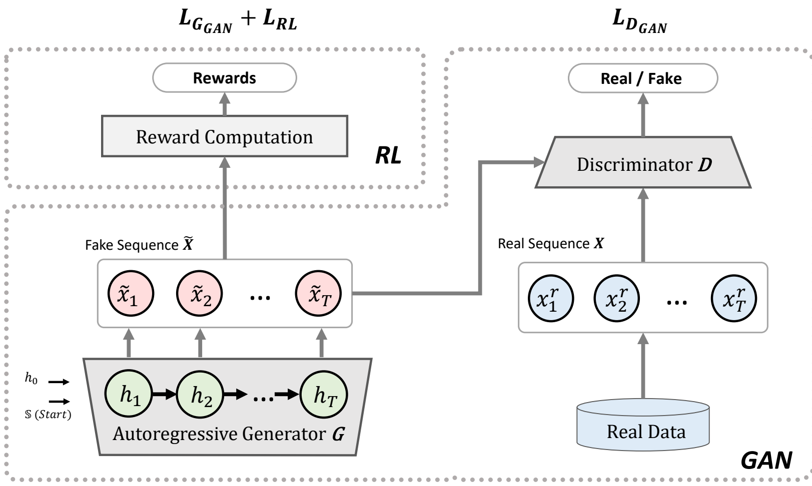


Fig. 2. Overview of OptiGAN framework. The Reinforcement Learning (RL) component is incorporated with sequence GAN model. The generator $G$ is trained by combining two losses, the GAN loss and the RL loss, $L_{G_{GAN}}$ and $L_{RL}$.

is equivalent to the following optimization problem (see the proof in Appendix A):

$$\min_G I_{KL}\left(P_d \,||\, P_G\right) + I_{JS}\left(P_d \,||\, P_G\right), \tag{6}$$

where $I_{JS}$ is Jenshen-Shannon divergence and $P_G$ is the generative distribution. The optimization problem in Eq. (6) reveals that at the Nash equilibrium point the generative distribution $P_G$ is exactly the data distribution $P_d$, thus overcoming the mode-collapse issue caused by original GAN formulation [13].

To train our model, we alternatively update $G$ and $D$ with relevant terms. We note that in the optimization for updating $G$ regarding $\log\, p_G\left(X \mid \theta\right)$, we maximize its lower bound in Eq. (4) instead of the likelihood function.

**Training procedure.** To train our model, we alternatively update the discriminator and generator:

- Update $D$:
$$\max_D \mathbb{E}_{X\sim p_d}\left[\log\, D\left(X\right)\right] + \mathbb{E}_{z\sim p_z}\left[\log\left[1 - D\left(G\left(z\right)\right)\right]\right].$$
- Update $G$:
$$\begin{aligned}&\max_G \mathbb{E}_{X\sim p_d}\left[\log\, p\left(X \mid \theta\right)\right] - \mathbb{E}_{z\sim p_z}\left[\log\left[1 - D\left(G\left(z\right)\right)\right]\right] \\ &= \max_G \mathbb{E}_{X\sim p_d}\left[\log\, p\left(X \mid \theta\right)\right] + \mathbb{E}_{z\sim p_z}\left[\log\, D\left(G\left(z\right)\right)\right].\end{aligned} \tag{7}$$

It is worth noting that for discrete data (e.g. text), we define the likelihood $p(x_i \mid h_i) = \text{softmax}(W_o h_i)$ where $W_o$ is the output weight matrix. In addition, to allow end to end training, we apply Gumbel softmax [32], [33] trick for the discrete case, and fix start token to $p(x_1|z) = 0$, as we depend on Gumbel Softmax for random output sampling. For real-valued data (e.g., air combat trajectory), we employ $p(x_i \mid h_i) = \mathcal{N}\left(W_o h_i, \sigma^2\right)$ where $\sigma$ is the standard deviation parameter.

### *B. Optimizing score corresponding to a goal with reinforcement learning*

To incorporate the ability to model the data to maximize rewards from the environment, we use policy gradient to learn a policy that maximizes the total rewards from environment. Following [20], the learning objective to maximize the return rewards over an episode from $t = [0, 1, ..., T-1, T]$ is:

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[U_t \log \pi\left(A_t|S_t, \boldsymbol{\theta}\right)\right] \quad ,$$

where $\pi$ is the "policy", or the probability distribution of actions given states of environment, $A_t$ and $S_t$ are the action and state at time $t$, $\theta$ are the parameters of $\pi$ and $U_t$ is "the return rewards" at time $t$.

In our model, the policy is the generator $G$, and the state at time $t$ is the hidden state of the generator $h_t$. Thus the objective becomes:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{X\sim P_d}\left[U_t \log p\left(X \mid h_t, \boldsymbol{\theta}\right)\right] \quad .$$

We use the REINFORCE [11] to find the optimum parameters for policy $G$ by gradient ascent of the gradient of $J$ as

$$\boldsymbol{\nabla} J(\boldsymbol{\theta}) = \mathbb{E}_{X\sim P_d}\left[U_t \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p\left(X \mid h_t, \boldsymbol{\theta}\right)\right] \quad ,$$

where $U_t$ is computed as

$$\begin{aligned}U_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{(T-t-1)} R_T \\ &= R_{t+1} + \gamma U_{t+1},\end{aligned}$$

where $R_t$ is the reward value from the environment at time $t$, $\gamma \in [0, 1.0]$ is the "discount factor" of future rewards. Note that for the text generation task, we use the BLEU score as reward value and for the air combat trajectory generation task, we use the McGrew score as reward value.

**Total loss.** The final total loss to train the generator $G$ with adversarial training and policy gradients is:

$$\begin{aligned}\max_G \mathbb{E}_{X\sim p_d}\Bigg[&\log p\left(X \mid \theta\right) + \lambda \mathbb{E}_{z\sim p_z}\left[\log D\left(G\left(z\right)\right)\right] \\ &+ \alpha \mathbb{E}_{X\sim P_d}\left[U_t \log p\left(X \mid h_t, \boldsymbol{\theta}\right)\right]\Bigg],\end{aligned}$$

where $\lambda$ and $\alpha$ are hyper-parameters that control how much the effect of adversarial training and policy gradients are on the total loss.

### C. Reducing policy gradients variance

In order to reduce the variance of policy gradients, we use an algorithm similar to the Monte Carlo rollout in [17] with a slight modification. We generate few complete sentences at each time step onward, and take their average as $U_t$ at that time step. Instead of getting the reward value from a discriminator as in [17], we directly compute the reward according to the chosen score.

In addition, to further reduce the variance of the policy gradients and to help policy gradients converge faster toward optimal solution, we use policy gradients with baseline [20], where policy gradient is defined as:

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_\pi\left[\left(U_t - b\left(S_t\right)\right) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t, \theta\right)\right] \\ &= \mathbb{E}_\pi\left[\left(U_t - b\left(h_t\right)\right) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p\left(X \mid h_t, \boldsymbol{\theta}\right)\right],\end{aligned}$$

where $b\left(s_t\right)$ is a baseline, a function that can be estimated or learned during training. The use of a baseline does not change the gradient expected value, but in practice, reduces its variance. In our experiments, $b(s_t)$ is a fixed value equivalent to the average of computed rewards over training time.

## V. Experiments

### A. Baselines

We evaluate our proposed model for both discrete (in our case, text generation) and real-valued data (air-craft trajectory generation), summarized in Table I.

TABLE I
COMPARISON BASELINES

| | Discrete Data (Text) | Real-valued Data (Trajectories) |
|---|---|---|
| SeqGAN* | ✓ | – |
| LSTM | – | ✓ |
| OptiGAN-OnlyRL | ✓ | – |
| OptiGAN-OnlyGAN | ✓ | ✓ |
| OptiGAN | ✓ | ✓ |

*SeqGAN works only with discrete data, not real-valued data

For text generation, we compare with three baselines:

1) **SeqGAN** [17]: is a well-known baseline for sequential generative models that uses a discriminator as a reward signal for training the generator in reinforcement learning framework.
2) **OptiGAN-OnlyRL:** This model is the vanilla reinforcement learning using policy gradients. For fairness, we implement it by using our own model with GAN component canceled, by zeroing out the GAN loss part.
3) **OptiGAN-OnlyGAN**: The sequence GAN with LSTM and discrete relaxation nodes, without any policy gradient component. We implement it using our model with RL component canceled, by zeroing out the policy gradient loss.

The GAN network implementation of our model is based on RELGAN with same hyperparamters and temperature scheduling, but using LSTM unit instead of relational memory.

For trajectory generation, we implement two different models to compare with; **LSTM** (The LSTM component of our model without adversarial training) and **OptiGAN-OnlyGAN** (our model without RL component), and we conduct ablation study for the effect of the RL component.

### B. Text generation

*1) Evaluation Metrics:* We use both BLEU score and negative log-likehood (NLL) mentioned below to evaluate the quality of our model.

*BLEU Score:* As discussed in Section III-A, BLEU score [18] is well-known text quality score in machine translation and text generation tasks.

The higher the BLEU score is, the more the number of matching n-grams with the test set. In practice, and as discussed later, the BLEU score can be easily cheated by repeating few matching n-grams in one sentence, or by generating only one or few high quality sentences from the model after training. This situation implies low output quality or diversity from the model.

*Negative Log-Likelihood (NLL):* We use the negative log-likelihood of the generator [28] to measure diversity, defined as:

$$\mathrm{NLL}_{\mathrm{gen}} = -\mathbb{E}_{x_{1:T} \sim P_d} \log P_{G_\theta}\left(x_1, \cdots, x_T\right)$$

where $P_d$ and $P_{G_\theta}$ are the real data and generated data distributions, respectively. The lower the value, the closer the model distribution is to the empirical data distribution.

*2) Datasets:* Two text datasets were used in our experiments for text generation are

- **The MS-COCO image captions dataset** [38] includes 4,682 unique words with the maximum sentence length 37. Both the training and test data contain 10,000 text sentences.
- **The EMNLP2017 WMT News dataset** [39] consists of 5,119 unique words with the maximum sentence length 49 after using first 10,000 sentences from [28]. Both the training and test data contain 10,000 sentences.

*3) Experimental settings and results:* For MS-COCO dataset, we use policy gradient baseline value $b\left(s_t\right) = 2.5$ and $\alpha = 2.0$ for both Vanilla-RL and our model. The number of Mone Carlo samples we use during training is 3. For EMNLP News, we use $b\left(s_t\right) = 2$ and 5 Monte Carlo samples.

In all experiments, we use gradient clipping value of 10.0 for the generator. In Tables II and III we report the means and standard deviations of test BLEU scores and training negative likelihoods values of our model compared to other baselines.

**Quality and diversity discussion**

Tables II and III show that, except for the OptiGAN-OnlyRL special case, our model outperforms the baselines in BLEU scores on MS-COCO dataset and all but BLEU-2 for EMNLP News dataset. Our model also achieves a competitive NLL value with the best model, OptiGAN-OnlyGAN. This means that our model does not sacrifice the diversity of generated output when optimizing for the given score. We find that SeqGAN suffers the worst NLL score, even when compared to OptiGAN-OnlyRL. Since SeqGAN modified generator objective does not encourage matching the model distribution to data distribution, it can be susceptible to diversity loss. On the other hand, GANs that use Gumbel-Softmax to keep the standard generator objective, like ours, are more able to match the model to data distribution.

TABLE II
BLEU SCORES AND NLL VALUES ON MS-COCO DATASET

| | BLEU-2 ↑ | BLEU-3 | BLEU-4 | BLEU-5 | NLL ↓ |
|---|---|---|---|---|---|
| SeqGAN | $75.09 \pm 0.84$ | $51.58 \pm 1.06$ | $32.06 \pm 0.98$ | $20.03 \pm 0.68$ | $0.830 \pm 0.176$ |
| OptiGAN-OnlyRL | $79.23 \pm 3.76$ | $59.23 \pm 6.21$ | $40.65 \pm 7.15$ | $27.11 \pm 6.36$ | $0.803 \pm 0.106$ |
| OptiGAN-OnlyGAN | $75.96 \pm 0.71$ | $53.79 \pm 0.99$ | $34.34 \pm 0.86$ | $21.51 \pm 0.56$ | $\mathbf{0.735 \pm 0.080}$ |
| OptiGAN (RL+GAN) | $\mathbf{76.42 \pm 0.70}$ | $\mathbf{54.40 \pm 0.99}$ | $\mathbf{35.06 \pm 0.90}$ | $\mathbf{22.25 \pm 0.66}$ | $0.737 \pm 0.082$ |

TABLE III
BLEU SCORES AND NLL VALUES ON EMNLP NEWS 2017 DATASET

| | BLEU-2 ↑ | BLEU-3 | BLEU-4 | BLEU-5 | NLL ↓ |
|---|---|---|---|---|---|
| SeqGAN | $\mathbf{76.05 \pm 1.67}$ | $47.60 \pm 1.51$ | $23.88 \pm 0.88$ | $12.05 \pm 0.40$ | $2.359 \pm 0.272$ |
| OptiGAN-OnlyRL | $79.16 \pm 2.18$ | $53.57 \pm 4.00$ | $31.26 \pm 4.66$ | $16.67 \pm 3.14$ | $2.267 \pm 0.154$ |
| OptiGAN-OnlyGAN | $73.15 \pm 2.35$ | $48.00 \pm 1.32$ | $26.12 \pm 1.00$ | $13.77 \pm 0.71$ | $2.234 \pm 0.152$ |
| OptiGAN (RL+GAN) | $74.03 \pm 1.69$ | $\mathbf{48.73 \pm 1.08}$ | $\mathbf{26.64 \pm 1.07}$ | $\mathbf{14.05 \pm 0.79}$ | $\mathbf{2.226 \pm 0.148}$ |

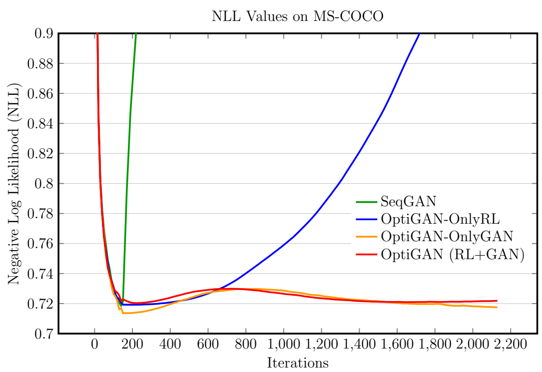


Fig. 3. NLL values on MS-COCO Dataset. Unlike SeqGAN, OptiGAN does not sacrifice output diversity.

TABLE IV
SENTENCES FROM OPTIGAN-ONLYRL. PURE RL LOOSES STRUCTURE WITH BLEU REWARDS, REPEATING CERTAIN N-GRAMS

| Generated sentences from **OptiGAN-OnlyRL** |
|---|
| the party has pledged a plate coach and don ' t think it was good to the real head . |
| and i ' ve been - it ' i ' i ' i have that thing a couple to hear the end but i ' m , " it ' s really people were going to do , ' it ' he work . |
| i ' d like , when i ' d like to give them to that it , there , and it ' ll happen looking to doubt that everybody challenges ... |
| if i ' ve got no evidence , it ' s it ' i expect ' there ' he ' he ' it ' he ' it ' out , and information it ' it ' ... |
| ' i ' i ' it ' we ' i think i ' we spent a escape i ' i ' ve been on the ... |

In the case of OptiGAN-OnlyRL, we find that pure reinforcement learning can achieve a higher BLEU score than other models (with very high variance). However, it has worse NLL values, which means it has worse diversity than our model. Fig. 3 shows that OptiGAN-OnlyRL fails to converge to low NLL, unlike our model, which has competitive NLL values with OptiGAN-OnlyGAN.

Moreover, although pure reinforcement learning can reach high BLEU scores, yet the sentences mostly are not realistic. We show in Table IV sentences from OptiGAN-OnlyRL, where we find that many of the generated sentences are unrealistic repetitions of certain n-grams in the test set. In the case of MS-COCO dataset, the generated sentences lengths are shorter than the average length of the dataset. This behavior possibly means that in the absence of the GAN objective part of the loss, pure reinforcement learning does not have incentive to generate sentences close to the real data distribution. In this case, the model only has to achieve high BLEU score to reduce the optimization loss.

### Sentences quality of OptiGAN

Table V shows generated sentences of OptiGAN. The sentences generally look meaningful, structured and diverse, hence showing the capacity of OptiGAN in generating good and diverse sentences.

TABLE V
GENERATED SAMPLE SENTENCES FROM OUR MODEL

| **Samples from MS-COCO** |
|---|
| a roadside vendor sells food to passersby on there are two multitcolored towels . |
| an older man sitting at a kitchen with stainless steel appliances . |
| a woman standing in a field with mountains in the view of a field and a bus stop . |
| a clean bathroom with a blue toilet . |
| a group of people is watching buses next to a tall building . |
| a woman in a white shirt and jeans walking up a air gondola wears a costume decorations in a red jacket hides building |
| three small dogs under a towel rack . |
| a bathroom with a toilet and a large mirror |
| a city street with cars vehicles parked on the ground . |
| a large passenger jet flying through the air flying a kite and an airport . |
| **Samples from EMNLP News** |
| people had gone in a few weeks ago , it ' s really very quiet tonight to do . |
| she is me but it ' s a concept , the girls can build high strength . |
| i ' ve got a shock for their parents law to the same offence . |
| they ' re going to acknowledge that their football leader will be able to get every most republicans . |
| a tory source said : ' the 22 fall in the family in all of the newcastle day . at cbs, that is more difficult, this is that the social stuff isn't quite any tribute for britain |
| it ' s a safe model from making a book of the first lady who had to respond . |

### C. Air-Combat Trajectory Generation

*1) Evaluation metrics:* We use the McGrew score[19] which measures how good is the aircraft positioned in an attempt to get behind the other aircraft. McGrew score is well-known by domain experts in air-combat maneuvers.

*2) Datasets:* For trajectory generation task, we used simulated data from *ACE-Zero* simulator [37]. We created simulated trajectory data for the Stern Conversion maneuver [36] (Fig. 1) with two fighters; the blue and the red. We created 6,000 trajectories under this scenario [2].Each trajectory contains 16 featuresfor each of the two fighters.

*3) Experimental settings and results:* In all of our experiments, we use 40 simulation time steps (tokens) for each fighter trajectory. We use 256 units hidden layer for LSTM unit with 2 hidden layers. For the VAE part of the model, we use 12 hidden units and latent dimension of size 10. We pretrained the generator for 80 epochs before starting the adversarial and policy gradients training. In all experiments we set $\sigma = 0$ for sampling $x_t$. We show samples of the training data trajectories and generated trajectories by our model in supplementary materials. We can see from the generated trajectories is that the model is able to capture the correct behavior, were the blue trajectory tries to get behind the red aircraft.

TABLE VI
BLUE FIGHTER ENGAGEMENT SCORES (MCGREW SCORE)

| | $\lambda$ | $\alpha$ | McGrew Score |
|---|---|---|---|
| SeqGAN* | | | N/A |
| LSTM | – | – | 6.21 |
| OptiGAN-OnlyGAN | 1.0 | – | 7.34 |
| OptiGAN | 0.2 | 0.75 | **8.41** |
| *ACE0 Simulator Dataset* | – | – | *8.53* |

**SeqGAN works only with discrete data, not real-valued data*

TABLE VII
EFFECT OF HYPER-PARAMTER $\lambda$ FOR GAN-ONLY TRAINING

| | $\lambda$ | McGrew Score |
|---|---|---|
| OptiGAN-OnlyGAN | 1.0 | 7.34 |
| | 0.2 | 6.79 |

**Score optimization**

We want the generated trajectories to be more optimized towards better engagement position against the red fighter. The desired outcome is a higher McGrew score, which means better engagement positions along the generated trajectory. We evaluate the effect of using policy gradients on the McGrew score of the blue aircraft and show the results in Table VI. In all experiments, we use $\gamma = 0.9$.

We compare with three baselines, Our model for real-valued data without adversarial training or policy gradients (**LSTM**), GAN without policy gradients (**OptiGAN-OnlyGAN**), and the average McGrew score of the training data (from simulator).

In all baselines, we generate 6,000 trajectories. . We can see that full OptiGAN model with the policy gradients achieve higher McGrew scores than other baselines, and closest to the real physics simulator. Although the GAN without policy gradients was able to achieve a slightly less score, the policy gradient model was run with the small value $\lambda = 0.2$. This means that the adversarial training did not contribute to the high score achieved by policy gradients model, rather, it was mainly the effect of policy gradients. As shown in Table VII, GAN with no PG model with $\lambda = 0.2$ did not achieve the same score as the the one with $\lambda = 1.0$.

**Trajectories quality of OptiGAN**

Fig. 4 shows the trajectories generated by OptiGAN compared to real trajectories. It can be observed that OptiGAN can generate high-quality trajectories resembling real data.

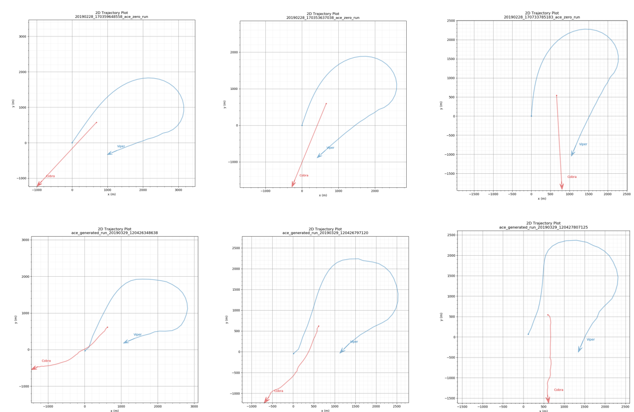

Fig. 4. Samples of the training data and generated trajectories from the model. *Top row:* samples from the training trajectories in 2D position plane. *Bottom row:* generated trajectories from the trained model (McGrew score = 6.03 ).

## VI. CONCLUSION

In this paper we presented a sequential deep generative model, OptiGAN, that integrates both generative adversarial networks and reinforcement learning for goal optimized generation. In many applications, goal optimization is a useful mechanism to give desired properties to generated outputs. We applied our model to text and air-combat trajectory generation tasks, and showed that the model generated high quality sentences with higher desired scores. In addition, OptiGAN preserves the diversity of outputs close to the real data. Our model serves as a general framework, that can be used for any GAN model to enable it to directly optimize a desired goal according to the given task.

In future work, we plan to improve the quality of real-valued outputs (e.g. trajectories) to make it more realistic to the data and physically constrained. We further look to incorporate latent space in the discrete case, which can be leveraged to guide the generation process along learned disentangled features of the data.


## ACKNOWLEDGMENTS

We gratefully acknowledge the partial support from the Defence Science and Technology (DST) Group for this work.


[2]All trajectory data is available at https://bit.ly/33k1AkT

## Appendix

### A. *Proof of the final objective function*

Consider this optimization problem:

$$\max_{G}\min_{D}\Big[\mathbb{E}_{X\sim p_d}\left[\log p_G\left(X\mid\theta\right)\right]- \mathbb{E}_{X\sim p_d}\left[\log D\left(X\right)\right]-\mathbb{E}_{z\sim p_z}\left[\log\left[1-D\left(G\left(z\right)\right)\right]\right]\Big]. \quad (8)$$

Given a generator $G$, the optimal $D^*\left(G\right)$ is determined as:

$$D_G^*\left(X\right)=\frac{p_d\left(X\right)}{p_G\left(X\right)+p_d\left(X\right)},$$

where $p_G\left(X\right)$ is the distribution induced from $G\left(X\right)$ where $X\sim p_d\left(X\right)$.

Substituting $D_G^*$ back to Eq. (8), we obtain the following optimization problem regarding $G$:

$$\max_G\left(\mathbb{E}_{p_d}\left[\log p_G\left(X\right)\right]-I_{JS}\left(P_d\|P_G\right)\right). \quad (9)$$

The objective function in Eq. (9) can be written as

$$\begin{aligned}&\mathbb{E}_{p_d}\left[\log p_G\left(X\right)\right]-I_{JS}\left(P_d\|P_G\right)\\ &=-I_{JS}\left(P_d\|P_G\right)-I_{KL}\left(P_d\|P_G\right)-\mathbb{E}_{p_d}\left[\log p_d\left(X\right)\right]\\ &=-I_{JS}\left(P_d\|P_G\right)-I_{KL}\left(P_d\|P_G\right)+\text{const.}\end{aligned}$$

Therefore, the optimization problem in Eq. (9) is equivalent to:

$$\min_G\left(I_{JS}\left(P_d\|P_G\right)+I_{KL}\left(P_d\|P_G\right)\right).$$

At the Nash equilibrium point of this game, we hence obtain: $p_G\left(X\right)=p_d\left(X\right)$.

## References


[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in neural information processing systems*, 2014.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR*, 2014.

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, pp. 1–16, 2015.

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017.

[5] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," 2018.

[6] N. Dam, Q. Hoang, T. Le, T. D. Nguyen, H. Bui, and D. Phung, "Three-player Wasserstein GAN via amortised duality," in *International Joint Conference on Artificial Intelligence*, 2019.

[7] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 837–14 847.

[8] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[9] G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, and A. Aspuru-Guzik, "Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models," 2017.

[10] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proceedings of ICML*, vol. 70, 2017.

[11] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Machine Learning*, 1992, pp. 229–256.

[12] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, pp. 1–57, 2016.

[13] T. Nguyen, T. Le, H. Vu, and D. Phung, "Dual discriminator generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2017, pp. 2670–2680.

[14] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "MGAN: Training generative adversarial nets with multiple generators," in *International Conference on Learning Representations*, 2018.

[15] T. Le, H. Vu, T. Nguyen, and D. Phung, "Geometric enclosing networks," in *In Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[16] T. Le, Q. Hoang, H. Vu, T. D. Nguyen, H. Bui, and D. Phung, "Learning generative adversarial networks from multiple data sources," in *International Joint Conference on Artificial Intelligence*, 2019.

[17] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient." in *Proc. of AAAI*, 2017, pp. 2852–2858.

[18] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, 2002, pp. 311–318.

[19] J. S. McGrew, J. P. How, B. Williams, and N. Roy, "Air-combat strategy using approximate dynamic programming," *Journal of guidance, control, and dynamics*, vol. 33, no. 5, pp. 1641–1654, 2010.

[20] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2011.

[21] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *NIPS*, 2015.

[22] H. Le, T. Tran, T. Nguyen, and S. Venkatesh, "Variational memory encoder-decoder," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1508–1518.

[23] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," in *Advances in neural information processing systems*, 2016, pp. 2199–2207.

[24] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," *arXiv preprint arXiv:1803.05428*, 2018.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] S. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space." in *Proceedings of CoNLL.*, 2016.

[27] W. Fedus, I. Goodfellow, and A. M. Dai, "MaskGAN: Better text generation via filling in the," in *ICLR*, 2018, pp. 1–15.

[28] W. Nie, N. Narodytska, and A. Patel, "RelGAN: Relational generative adversarial networks for text generation," in *International Conference on Learning Representations*, 2019.

[29] L. Chen, S. Dai, C. Tao, D. Shen, Z. Gan, H. Zhang, Y. Zhang, and L. Carin, "Adversarial text generation via feature-mover's distance," in *NIPS*, 2018.

[30] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," in *ICML*, 2017.

[31] M. J. Kusner and J. M. Hernández-Lobato, "GANs for sequences of discrete elements with the gumbel-softmax distribution," *arXiv:1611.04051*, 2016.

[32] C. J. Maddison, A. Mnih, and Y. W. Teh, "The CONCRETE distribution: A continuous relaxation of discrete random variables," *arXiv:1611.00712*, 2016.

[33] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[34] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2019.

[35] E. Putin, A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A. V. Aladinskaya, A. Aliper, and A. Zhavoronkov, "Adversarial threshold neural computer for molecular de novo design," *Molecular Pharmaceutics*, vol. 15, no. 10, pp. 4386–4397, 2018.

[36] F. Austin, G. Carbone, H. Hinz, M. Lewis, and M. Falco, "Game theory for automated maneuvering during air-to-air combat," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 6, pp. 1143–1149, 1990.

[37] M. Ramirez, M. Papasimeon, L. Behnke, N. Lipovetzky, T. Miller, and A. R. Pearce, "Real–time uav maneuvering via automated planning in simulations," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 5243–5245.

[38] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[39] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.