Abstract

Reinforcement learning (RL) has been widely studied for improving sequence-generation models. However, the conventional rewards used for RL training typically cannot capture sufficient semantic information and therefore render model bias. Further, the sparse and delayed rewards make RL exploration inefficient. To alleviate these issues, we propose the concept of nested-Wasserstein distance for measuring the distance between two policy distributions. Based on this, a novel nested-Wasserstein self-imitation learning framework is developed, encouraging the model to exploit historical high-rewarded sequences for deeper explorations and better semantic matching. Our solution can be understood as approximately executing proximal policy optimization with nested-Wasserstein trust-regions. Experiments on a variety of unconditional and conditional sequence-generation tasks demonstrate the proposed approach consistently leads to improved performance.

1 Introduction

Sequence generation is an important research topic in machine learning, covering a wide range of applications, including machine translation [4, 10, 52], image captioning [1, 57, 63], and text summarization [41, 45]. Standard sequence generation follows an auto-regressive model design under maximum likelihood estimation (MLE) learning [24, 52, 60]. That is, models are trained to maximize the expected log-likelihood of the next word conditioned on its preceding ground-truth partial sentence. However, when testing, the generated partial sequence is fed to the generator to draw the next token. Such a discrepancy between training and testing, commonly known as exposure bias, leads to accumulated approximation errors along the sequence-generation trajectory [6, 43].

To address exposure bias, reinforcement learning (RL) techniques have been introduced [43]. Unlike MLE, which only leverages training examples, RL can also exploit samples drawn from the current policy. Improvements are gained from reinforcing the training towards more-plausible generations, typically based on a user-specified reward function [43, 65]. However, the manually designed rewards often target specific desirable properties in sequence generation (e.g., matching n-gram overlap between generated sequences and ground-truth references), which unintentionally induces extra bias and is often criticized as a bad proxy for human evaluation [58]. Concerns have also been raised w.r.t. efficient exploration in sequence generation. In existing RL-based methods for sequence generation [3, 43, 44], all experiences are treated as equivalent. However, merely relying on policy samples to explore often leads to forgetting a high-reward trajectory, unless it can be re-sampled frequently [31]. This problem becomes more severe in the sparse-reward setting used in sequence generation, i.e., the reward is only available after the whole sentence is generated.

Motivated by the above observations, we present a novel nested-Wasserstein Self-Imitation Learning (WSIL) framework for sequence generation. Specifically, we maintain an experience replay buffer to store historical high-reward sequences, and employ the nested-Wasserstein distance between the behavior policy and the artificial policy defined by the replay buffer to encourage self-imitation. WSIL is inspired by and derived from the policy optimization with Wasserstein trust-regions [66]. Specifically, it provides a novel reward function to match the generated sequences with the high-reward sequences in the replay buffer, encouraging semantic matching rather than simple n-gram
overlapping. Based on the nested-Wasserstein distance, two novel schemes are proposed for self-imitation learning, depending on whether the historical sequences interact with the policy directly or indirectly.

The main contributions of this paper are summarized as follows. (i) A novel nested-Wasserstein self-imitation learning framework is developed for sequence generation, exploiting historical good explorations for better future exploration. (ii) A novel Wasserstein reward is introduced when calculating the nested-Wasserstein distance for sequence generation, effectively alleviating the model training bias imposed by conventional rewards. (iii) Extensive empirical evaluation is performed on both unconditional and conditional text generation tasks, demonstrating consistent performance improvement over existing state-of-the-art approaches.

2 Background

Sequence-generation model  We consider the problem of discrete sequence generation, which learns to generate a sequence \( Y = (y_1, \ldots, y_T) \) of length \( T \), possibly conditioned on context \( X \). Here each \( y_t \) is a token from vocabulary \( \mathcal{A} \). Pairs \((X, Y)\) are used for training a sequence-generation model. We are particularly interested in applications to text generation, where \( Y \) is a sentence and each \( y_t \) is a word. Starting from the initial state \( s_0 \), a recurrent neural network (RNN) produces a sequence of states \((s_1, \ldots, s_T)\) given an input sequence-feature representation \((\epsilon(y_1), \ldots, \epsilon(y_T))\), where \( \epsilon(\cdot) \) denotes a word embedding mapping a token to its \( d \)-dimensional feature representation. The states are recursively updated with a function known as the cell: \( s_t = h_\theta(s_{t-1}, \epsilon(y_t)) \), where \( \theta \) denotes the model parameters. Popular implementations include Long Short-Term Memory (LSTM) [21] and the Gated Recurrent Unit (GRU) [11]. In order to generate sequence \( Y \), one typically draws \( s_0 \) from a standard Gaussian distribution.

Sequence generation as an RL problem  Sequence generation can be considered as an RL problem with deterministic state transition and sparse reward. It can be formulated as a Markov decision process (MDP) \( \mathcal{M} = (S, A, P, r) \), where \( S \) is the state space, \( A \) is the action space, \( P \) is the deterministic environment dynamics and \( r(s, a) \) is a reward function. The policy \( \pi_\theta \), parameterized by \( \theta \), maps each state \( s \in S \) to a probability distribution over \( A \). The objective is to maximize the expected reward, defined as:

\[
J(\pi_\theta) = \mathbb{E}_{Y \sim \pi_\theta}[r(Y)] = \sum_{t=1}^{T} \mathbb{E}_{(s_t, y_t) \sim \pi_\theta}[r(s_t, y_t)],
\]

(2)

where \( Y = (s_1, y_1, \ldots, s_T, y_T) \) is a trajectory from policy \( \pi_\theta \) with \( y_t \in \mathcal{A} \), and \( r(Y) \) represents the reward for a sentence \( Y \), which can be decomposed into a summation of the rewards over state-action pairs \( r(s_t, y_t) \). RL seeks to learn an optimal policy, that maximizes \( J(\pi_\theta) \). In practice, we use the first objective, since the reward is only revealed after the whole sequence is generated.

Optimal transport on discrete domains  The optimal transport (OT) distance \( W_c(\mu, \nu) \) is a discrepancy score that measures the distance between two probability distributions \( \mu(\cdot) \) and \( \nu(\cdot) \) w.r.t. a cost function \( c(\cdot, \cdot) \). Specifically, we consider two discrete distributions \( \mu \triangleq \sum_{i=1}^{n} u_i \delta_{z_i} \) and \( \nu \triangleq \sum_{j=1}^{m} v_j \delta_{z_j} \) with \( \delta_{z} \) the Dirac delta function centered on \( z \). The weight vectors \( u = \{u_i\}_{i=1}^{n} \in \Delta_n \) and \( v = \{v_j\}_{j=1}^{m} \in \Delta_m \) respectively belong to the \( n \) and \( m \)-dimensional simplex, i.e., \( \sum_{i=1}^{n} u_i = \sum_{j=1}^{m} v_j = 1 \). Accordingly, Wasserstein distance is equivalent to solving the following minimization problem:

\[
W_c(\mu, \nu) = \min_{T \in \Gamma(\mu, \nu)} \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij} \cdot c(z_i, z_j) = \min_{T \in \Gamma(\mu, \nu)} \langle T, C \rangle,
\]

(3)

where \( \sum_{j=1}^{m} T_{ij} = \frac{1}{m} \) and \( \sum_{i=1}^{n} T_{ij} = \frac{1}{n} \) are the constraints, \( \langle \cdot, \cdot \rangle \) represents the Frobenius dot-product, and \( C \) is the cost matrix defined by \( C_{ij} = c(z_i, z_j) \). Intuitively, the OT distance is the minimal cost of transporting mass from \( \mu \) to \( \nu \).
3 Distributional Semantic Matching for Sequence Generation

We consider evaluating the sentence from syntactic and semantic perspectives. Conventional metric rewards (e.g., BLEU) can capture the syntactic structure better, where the exact matching of words (or short phrases) to the reference sequences is encouraged, which induces strong bias in many cases. As such, we focus on the semantic matching and propose nested-Wasserstein distance, which defines the distance between two sequence distributions. Nested-Wasserstein distance provides a natural way to manifest semantic matching compared with the conventional rewards used in existing RL-based sequence models. Alternatively, we can train a discriminator to learn the reward model, but empirically it only rewards high-quality generations, even though they may be characterized by mode collapse [20], implying poor diversity in generation; while diversity is also an important aspect in evaluation. To better understand the issue, consider the following example on sentence matching:

![Figure 1: Comparison of different rewards (higher is better).](image)

It is clear that while the first candidate sentence has a similar syntactic structure to the reference, the second candidate sentence is more semantically consistent with the reference. However, popular hard-matching metrics [39, 55] consistently indicate the first candidate is a better match to the reference (see Figure 1). The above contradiction can be alleviated if the reward metric is more semantic-aware. So motivated, the remainder of this section is devoted to a discussion of design and implementation of Wasserstein rewards. The general idea is to match the semantic features via minimizing the Wasserstein distance between hypothesis sentences and their references in the semantic space. It will automatically match semantically similar words as shown in Figure 1, where the matching (black arrows) is determined by the optimal transport matrix $T$, and weights are determined by $C$ in (3).

**Definition 1 (Wasserstein Distance between Sequence Pairs)** Consider sequence $Y = (y_1, \ldots, y_T)$ as a discrete distribution $p_Y = \frac{1}{T} \sum_i \delta(y_i)$ in the semantic space, with the length-normalized point mass placed at the semantic embedding, i.e., $z_i = c(y_i)$ of each token $y_i$ from the sequence $Y$. Given a hypothesis sequence $Y'$ w.r.t. a reference sequence $Y$, we define the Wasserstein distance as $W_c(p_Y, p_{Y'}) \triangleq \min_T \langle T, C \rangle$ between $p_Y$ and $p_{Y'}$ with cost $c(z, z')$. When the cosine distance $c_{\cos}(z, z') = 1 - \frac{z \cdot z'}{\|z\|_2 \|z'\|_2}$ is used as our cost, we define the Wasserstein reward as $r_c \triangleq \langle T^*, 1 - C \rangle$, where $T^*$ is the optimal transport matrix.

**Remark 1** Wasserstein distance aims at semantic matching and does not consider the syntactic information. To incorporate both types of information, we propose the self-imitation scheme.

**Nested-Wasserstein distance** Note our ultimate goal is to measure distance between two policy distributions instead of sequence pairs. Given two sets of sequences from two policies, one aims to incorporate the semantic information between sequences into the distance measure. Directly applying Wasserstein distance between two distributions with limited samples is not accurate enough, which, in addition, does not consider semantic information directly. To deal with this issue, we propose the nested-Wasserstein distance in Definition 2.

**Definition 2 (Nested-Wasserstein Distance)** Consider two sets of sequences $Y = \{Y_i\}_{i=1}^K$ and $Y' = \{Y'_j\}_{j=1}^{K'}$ drawn from two sequence distributions $P_Y$ and $P_{Y'}$, where $K$ and $K'$ are the number of sequences in $Y$ and $Y'$. The nested-Wasserstein distance, denoted as $W_{nc}(P_Y, P_{Y'})$, is a metric measuring the distance between $P_Y$ and $P_{Y'}$ in a nested manner:

$$W_{nc}(P_Y, P_{Y'}) \triangleq \min_{T_{ij}} \sum_{i=1}^K \sum_{j=1}^{K'} T_{ij} W_c(p_{Y_i}, p_{Y'_j}),$$  \hspace{1cm} (4)

where $T_{ij}$ satisfies $s_i \geq 0$ satisfies $\sum_i T_{ij} = \frac{1}{K}$ and $\sum_j T_{ij} = \frac{1}{K'}$, and $W_c(\cdot, \cdot)$ denotes the $c$-Wasserstein distance defined in (3).

**Remark 2** The word “nested” comes from the definition in (4), which essentially consists of two nested levels of Wasserstein distances. The proposed nested-Wasserstein distance brings in the semantic information via the distance measure $W_c$ in the first level distance. Note that we have omitted the expectation over samples in (4) for simplicity, as we essentially use a single set of samples to approximate $W_{nc}(\cdot, \cdot)$ in algorithms.
we replace the Wasserstein distance $W$

As illustrated in Figure 2, we propose two self-imitation schemes towards (5): (i) we use $\pi$ where $W$
The overall idea of the proposed nested-Wasserstein self-imitation learning (WSIL) is to define a

$\pi$ is the nested-Wasserstein distance defined in Definition 1, and $\lambda$ is the nested Wasserstein distance degenerates
to denote both the policy and the distribution over the sequences. Distinct from classic trust-region policy optimization, which defines the trust region based on KL-divergence [48], WSIL defines the trust region based on the nested-Wasserstein distance between the behavior policy $\pi_{\theta,X}$ and the artificial policy $\pi_{B,X}$. Note when $K = K' = 1$, the nested Wasserstein distance degenerates to the definition of Wasserstein distance between two sequences.

As illustrated in Figure 2, we propose two self-imitation schemes towards (5): (i) direct WSIL, where samples from $\pi_B$ are directly used as pseudo-samples to update the policy (sequence generator); and (ii) indirect WSIL, where an additional reward is defined to encourage the matching between the current policy and $\pi_B$. In general, the two methods differ on how samples from the policies are realized. Algorithm 2 in the Appendix describes the general implementation procedure of the two schemes. For simplicity, we may sometimes omit the first expectation $\mathbb{E}_{X \sim p_d}$ in the following sections.

**Indirect nested-Wasserstein Self-Imitation Learning** We seek to use historical high-reward sequences to define a “self-imitation” reward function, which is then combined with the original reward function to update the generator with policy gradient methods. The word “indirect” comes from the mechanism that historical sequences interact with the policy indirectly via the self-imitation reward. Intuitively, higher self-imitation rewards are achieved when the generated sequences are close to historical high-reward sequences. Thus the generator is guided to perform self imitation and we call this method indirect nested-Wasserstein self-imitation learning (WSIL-I). WSIL-I incorporates a self-imitation reward, denoted $r_s(Y^s, Y^b)$, into the objective function. Here $Y^b$ denotes a sample from the replay buffer and $Y^s$ denotes a sample from the current policy. To this end, we replace the Wasserstein distance $W_c$ in the nested-Wasserstein distance with $r_s(Y^s, Y^b)$ in the general objective (5). Specifically, we define the two sets of sample sequences from $\pi_{\theta,X}$ and $\pi_{B,X}$ to be $\{Y^s\}$ and $Y^b \triangleq \{Y^b_{j=1}^{K'}\}$, with sizes of 1 and $K'$, respectively. Here $Y^s \sim \pi_{\theta,X}$ and

![Figure 2: Illustration of the proposed nested-Wasserstein Self-Imitation Learning (WSIL) framework. Left: direct WSIL, where samples from the replay buffer are directly used as pseudo-samples to update the generator. Right: indirect WSIL, where Wasserstein self-imitation rewards are defined to encourage the generator to imitate samples from the replay buffer.](image-url)
where \( Y_j^b \sim \pi_{B,X}, \forall j \). \( \{Y^s\} \) and \( \{Y^b\} \) will be used in calculating the nested-Wasserstein distance. Let \( r_{ns}(Y^s, Y^b) \triangleq \sum Y_i T_j r_s(Y^s, Y^b) \) be the nested-Wasserstein reward, with \( T' = \{T_j'\} \) the optimal weights. Based on (5), the objective of WSIL-I is adapted to be:

\[
J_I(\pi_\theta) \triangleq \mathbb{E}_{X \sim p_\nu} \mathbb{E}_{Y^s \sim \pi_{\theta,X}} \left[ r(Y^s) + \lambda r_{ns}(Y^s, Y^b) \right],
\]

where \( r \) is the original RL reward; \( r_{ns} \) is the nested-Wasserstein reward. Since not all historical explored samples are helpful for updating the current policy, we only consider a subset of the high-reward sequences when performing self-imitation. Using \( K \) trajectories sampled \( i.i.d. \) from \( \pi_\theta \) and introducing a baseline \( b \), the gradient estimate of WSIL-I can be expressed as:

\[
\nabla_\theta J_I(\pi_\theta) \approx - \sum_{k=1}^{K} \left[ (r(Y_k^s) - b) \nabla_\theta \log \pi_\theta(Y_k^s) + \lambda r_{ns}(Y_k^s, Y^b) \nabla_\theta \log \pi_\theta(Y_k^s) \right],
\]

where \( Y^s \triangleq \{Y_k^s\}_{k=1}^K \). In practice, \( \mathbb{I} \left[ r(Y^b) > r(Y^s) \right] \) will be combined with the nested-Wasserstein rewards, where \( \mathbb{I}(\cdot) = 1 \) if the condition is satisfied, and 0 otherwise; \( b \) is the baseline to stabilize training. If the reward of a historical high-reward sequence is greater than the current one (i.e., \( r(Y^b) > r(Y^s) \)), the generator learns to imitate this high-reward sequence. Otherwise, the update based on the historical sequence is not performed due to the \( \mathbb{I}(\cdot) \) operator. This encourages the agent to only imitate its good historical explorations.

**Direct nested-Wasserstein Self-Imitation Learning** Direct Wasserstein self-imitation learning (WSIL-D) weights the original rewards with outputs from the behavior policy for sequences in the replay buffer \( B \). The sequences from the replay buffer are directly used as pseudo-samples to update the generator [31]. Similarly, define \( r_{ns}(Y^s, Y) \triangleq \sum J_k T_j r_s(Y^s, Y_j) \), with \( T' = \{T_j'\} \) the optimal weights. Based on the objective of (5), the general objective (8) is then extended to be the objective for WSIL-D, as

\[
J_D(\pi_\theta) \triangleq \mathbb{E}_{Y^s \sim \pi_{\theta,X}} \left[ r(Y^s) + \lambda \mathbb{E}_{Y^s \sim \pi_{B,X}} \left[ r_{ns}(Y^b, Y) \pi_\theta(Y^s) \right] \right],
\]

where \( r \) is the original RL reward; \( r_{ns} \) is the nested-Wasserstein reward. Based on the objective of (8), we update the generator with standard RL loss and the self-imitation loss alternatively, with a hyperparameter \( \lambda \) that controls the update frequency:

\[
\nabla_\theta J_D(\pi_\theta) \approx - \sum_{k=1}^{K} \left[ (r(Y_k^s) - b) \nabla_\theta \log \pi_\theta(Y_k^s) \right] - \lambda \sum_{j=1}^{K'} \left[ (r_{ns}(Y_j^b, Y) - b_s) \nabla_\theta \log \pi_\theta(Y_j^b) \right],
\]

where \((\cdot)_+ = \max(\cdot, 0)\) and \( b_s \) and \( b \) are the baselines to reduce the variance of gradient estimates. In practice, \((\cdot)_+\) means that WSIL-D only imitates the sequences in the replay buffer with the higher rewards. Intuitively, direct self-imitation implicitly imposes larger weights on good simulated data for training, to exploit good historical explorations. The main difference between WSIL-D and its indirect counterpart is that sequences from the replay buffer are not used to compute the self-imitation rewards, but used to evaluate the policy. Intuitively, WSIL-D changes the data distribution to explore the good history more efficiently.

**Empirical estimation of nested-Wasserstein rewards** Computing the exact nested-Wasserstein distance is computationally intractable [2, 16, 46], and therefore we employ the recently proposed IPOT algorithm [61] to obtain an efficient approximation of the Wasserstein reward. Specifically, IPOT considers the following proximal gradient descent to solve the optimal transport matrix \( T \) via iterative optimization, \( i.e. \),

\[
T^{(t+1)} = \arg \min_{T \in \Pi(\mu, \nu)} \left\{ \langle T, C \rangle + \gamma \cdot D_{KL}(T, T^{(t)}) \right\},
\]

where \( 1/\gamma > 0 \) is the generalised step size and the generalised KL-divergence \( D_{KL}(T, T^{(t)}) = \sum_{i,j} T_{ij} \log \frac{T_{ij}}{T_{ij}^{(t)}} - \sum_{i,j} T_{ij} + \sum_{i,j} T_{ij}^{(t)} \) is used as the proximity metric. Standard Sinkhorn iterations [12] are used to solve the above sub-problem. The full approach is summarized as Algorithm 1 in Appendix A.

**Exploration Efficiency** The exploration space of MLE is the examples in the training set [54], \( i.e. \), no exploration is performed in supervised training. In contrast, standard policy optimization [43] basically allows the whole exploration space. However, the exploration may become inefficient since it may be too flexible, and some good sequences observed in history tend to be less explored and imitated due to the sparse rewards. Our proposed WSIL aims to provide more efficient and systematic exploration. It allows the whole-space exploration, but re-weights the exploration space to focus more on the exploration which may provide better performance with the Wasserstein trust-region.
Implementation Details  A few key techniques are required for successful model training. (i) The reward from a greedy-decoding sentence is used as the baseline [44] in conditional text generation; in unconditional text generation, a constant baseline is used. (ii) In unconditional text generation, a variational autoencoder framework [26] is further adopted, where $X$ is the target sentence itself in the training, and random noise in the testing. (iii) A single large replay buffer is maintained for unconditional generation, and multiple replay buffers are maintained for different conditions in conditional generation. (iv) According to the theory of Wasserstein gradient flows [56], $1/\lambda$ can be interpreted as a generalized decaying learning rate. With more explorations, $\lambda$ becomes larger, and the algorithm should focus more on the self-imitation learning, providing a guideline to balance the standard RL training and self-imitation learning. More details are provided in Appendix A. Practically, Wasserstein rewards provide weak supervision focusing on semantic matching, which is reasonable since the historical high-reward sequences contain some noises.

5 Related Work

Self-Imitation Learning  Experience replay has been widely considered in RL. Deterministic policy gradient [51, 32] performs experience replay, but is limited to continuous control. Actor-critic approaches [27] can also utilize a replay buffer to improve performance. Prioritized experience replay [47] samples trajectories based on the time-difference error, and we adopt it in our implementation. These approaches indiscriminately buffer all experiences, while the approach proposed here only buffers high-reward experience. Further, episodic control [29] can be regarded as an extreme way of exploiting past experience, trying to reproduce its best past decisions, but retrieving states leads to poor efficiency and generalization in testing. Self-imitation learning was first applied in Atari games and Mujoco [38, 15], reporting performance improvement w.r.t. sparse rewards. Compared with that work, our solution considers two novel self-imitation learning schemes in the context of sequence generation.

RL for Sequence Generation  RL techniques have been explored in detail for sequence generation. For example, a Seq2Seq model can be trained by directly optimizing the BLEU/ROUGE scores via policy gradient [43, 3]. Furthermore, Rennie, et al. [44] baselines the actor with the reward of a greedy-decoding sequence for the REINFORCE method. Sequence generation with RL can also adopt generative adversarial imitation learning and use a learned discriminator (or, critic) to provide sequence-level guidance. By constructing different objectives, previous work [65, 33, 19, 13] combine the policy-gradient algorithm with the original GAN training procedure. However, mode-collapse problems make the training of these methods challenging. The soft-argmax trick is used in [67, 22] instead of REINFORCE to get more-stable training, at the sacrifice of losing exploration. These methods treat all explorations equivalently. Compared with them, we propose the use of self-imitation learning, and maintain a replay buffer to exploit past good explorations. A memory buffer is maintained in [31], where high-reward samples are used for program synthesis. The motivation is similar to ours, but only direct self-imitation learning is considered in their case.

Optimal transport (OT) in NLP  Optimal transport was first applied to NLP in [28], and proposed the word mover’s distance (WMD); OT has also been employed to improve topic modeling [23]. The transportation cost is usually defined as Euclidean distance, and OT distance is approximated by solving a Kantorovich-Rubinstein dual [18] or a less-accurate lower bound [28]. Our work considers nested-Wasserstein distance as rewards, presenting an efficient IPOT-based implementation for OT distance approximation [8], and successfully using it to guide sequence generation.

6 Experiments

We evaluate the proposed method on both unconditional and conditional text-generation tasks on standard benchmark datasets. Details of the datasets, experimental setup and model architectures are provided in Appendix C, due to limited space. Code for all our experiments will be made publicly available.

6.1 Unconditional Text Generation

We compare our approach with a number of related GAN models for unconditional text generation [19, 33, 65, 67]. Our implementation is developed based on the LeakGAN model, by incorpo-
rating Wasserstein self-imitation learning. All baseline experiments are performed on the texygen platform [68]. The corpus-level BLEU score is employed to quantitatively evaluate the generated sentences. Specifically, we follow the strategy in [65, 19] and adopt the BLEU score, referenced by test set (test-BLEU) and themselves (self-BLEU) to evaluate the quality of generated samples. Test-BLEU evaluates the goodness of generated samples, and self-BLEU measures their diversity. The BLEU scores for 1000 generated sentences are averaged to obtain the final score for each model. A good generator should achieve both a high test-BLEU score and a low self-BLEU score. Following previous work [19], we test the proposed method on the short and long text generation on Image COCO and EMNLP2017 WMT News datasets. The BLEU scores with different methods are provided in Tables 1 and 2. The example generated sentences are provided in Appendix G.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test-BLEU-2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Self-BLEU-2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE [7]</td>
<td>0.902</td>
<td>0.706</td>
<td>0.470</td>
<td>0.392</td>
<td>0.787</td>
<td>0.646</td>
<td>0.485</td>
</tr>
<tr>
<td>SeqGAN [65]</td>
<td>0.820</td>
<td>0.604</td>
<td>0.361</td>
<td>0.211</td>
<td>0.807</td>
<td>0.577</td>
<td>0.278</td>
</tr>
<tr>
<td>RankGAN [33]</td>
<td>0.852</td>
<td>0.637</td>
<td>0.389</td>
<td>0.248</td>
<td>0.822</td>
<td>0.592</td>
<td>0.230</td>
</tr>
<tr>
<td>TextGAN [67]</td>
<td>0.910</td>
<td>0.728</td>
<td>0.484</td>
<td>0.306</td>
<td>0.806</td>
<td>0.548</td>
<td>0.217</td>
</tr>
<tr>
<td>LeakGAN [19]</td>
<td>0.922</td>
<td>0.797</td>
<td>0.602</td>
<td>0.416</td>
<td>0.912</td>
<td>0.825</td>
<td>0.689</td>
</tr>
<tr>
<td>WSIL-D (ours)</td>
<td>0.917</td>
<td>0.774</td>
<td>0.576</td>
<td>0.393</td>
<td>0.797</td>
<td>0.569</td>
<td>0.284</td>
</tr>
<tr>
<td>WSIL-I (ours)</td>
<td>0.922</td>
<td>0.778</td>
<td>0.576</td>
<td>0.396</td>
<td>0.813</td>
<td>0.600</td>
<td>0.326</td>
</tr>
</tbody>
</table>

Table 1: Test-BLEU (↑) and Self-BLEU (↓) scores on Image COCO.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test-BLEU-2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Self-BLEU-2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE [7]</td>
<td>0.905</td>
<td>0.701</td>
<td>0.464</td>
<td>0.278</td>
<td>0.764</td>
<td>0.522</td>
<td>0.295</td>
</tr>
<tr>
<td>SeqGAN [65]</td>
<td>0.630</td>
<td>0.354</td>
<td>0.164</td>
<td>0.087</td>
<td>0.728</td>
<td>0.411</td>
<td>0.139</td>
</tr>
<tr>
<td>RankGAN [33]</td>
<td>0.723</td>
<td>0.440</td>
<td>0.210</td>
<td>0.107</td>
<td>0.672</td>
<td>0.346</td>
<td>0.119</td>
</tr>
<tr>
<td>TextGAN [67]</td>
<td>0.777</td>
<td>0.529</td>
<td>0.305</td>
<td>0.161</td>
<td>0.806</td>
<td>0.662</td>
<td>0.448</td>
</tr>
<tr>
<td>LeakGAN [19]</td>
<td>0.923</td>
<td>0.757</td>
<td>0.546</td>
<td>0.335</td>
<td>0.837</td>
<td>0.683</td>
<td>0.513</td>
</tr>
<tr>
<td>SIL-D (ours)</td>
<td>0.875</td>
<td>0.634</td>
<td>0.401</td>
<td>0.243</td>
<td>0.724</td>
<td>0.466</td>
<td>0.256</td>
</tr>
<tr>
<td>SIL-I (ours)</td>
<td>0.869</td>
<td>0.633</td>
<td>0.399</td>
<td>0.242</td>
<td>0.710</td>
<td>0.455</td>
<td>0.263</td>
</tr>
<tr>
<td>WSIL-D (ours)</td>
<td>0.931</td>
<td>0.736</td>
<td>0.503</td>
<td>0.317</td>
<td>0.795</td>
<td>0.553</td>
<td>0.299</td>
</tr>
<tr>
<td>WSIL-I (ours)</td>
<td>0.926</td>
<td>0.726</td>
<td>0.492</td>
<td>0.307</td>
<td>0.815</td>
<td>0.595</td>
<td>0.380</td>
</tr>
</tbody>
</table>

Table 2: Test-BLEU (↑) and Self-BLEU (↓) scores on EMNLP2017 WMT News.

Analysis  Compared with other methods, LeakGAN, WSIL-D and WSIL-I achieve comparable test-BLEU scores, demonstrating high-quality generated sentences. However, LeakGAN tends to over-fit on training data, leading to much higher (worse) self-BLEU scores. Our proposed methods, by contrast, show good diversity of the generated text with lower self-BLEU scores. Other baselines obtain both low self-BLEU and test-BLEU scores, leading to more random generations.

Ablation Study  We conduct ablation studies on long-text generation to investigate the improvements brought by each part of WSIL. First, we test the benefits of using two types of self-imitation schemes. We compare RL training with (i) self-imitation (SIL-D and SIL-I), where only a replay buffer and conventional rewards are employed; and (ii) Wasserstein self-imitation (WSIL-D and WSIL-I). Results are shown in Table 2. We observe that the self-imitation strategy, with specific replay buffer construction, can alleviate the discrepancies between reward model bias and conventional rewards (e.g., self-BLEU). Without Wasserstein rewards, we achieve lower self-BLEU at the sacrifice of test-BLEU. When combining with Wasserstein rewards, WSIL-D and WSIL-I show superior performance relative to the baselines. The randomly selected generated samples in Appendix D and human evaluations further validate this.

Human Evaluation  Simply relying on the above metrics is not sufficient to evaluate the proposed method [7]. Following previous work [19], we perform additional human evaluation on the EMNLP2017 WMT News dataset using Amazon Mechanical Turk. Previous work has shown higher scores of LeakGAN compared with other baselines, therefore we mainly focus on the comparison of our methods with LeakGAN. We randomly sampled 200 sentences from each model, and asked 5 different workers to score each sentence on a scale of 1 to 5, considering its readability and meaning. Results are shown in Table 3, which indicates better performance of the proposed WSIL.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MLE [7]</th>
<th>LeakGAN</th>
<th>SIL-D</th>
<th>SIL-I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human scores</td>
<td>2.97±0.05</td>
<td>2.63±0.05</td>
<td>2.54±0.05</td>
<td>2.55±0.05</td>
</tr>
</tbody>
</table>

Table 3: Results of human evaluation.
6.2 Conditional Text Generation

Video Captioning We conduct experiments on the MSR-VTT dataset [62] for video captioning. The MSR-VTT is a large-scale video dataset, consisting of 20 video categories. The dataset was split into 6513 and 3487 clips in the training and testing sets. Each video is annotated with about 20 captions. For each video, we sample at 3 fps and extract Inception-v4 [53] features from these sampled frames. We report BLEU-4 [39], CIDEr [55], and METEOR [5] scores. Results are summarized in Table 5. Consistent improvements are observed with the WSIL framework. WSIL-D performs slightly better than WSIL-I, both yielding much higher optimized CIDEr and METEOR scores than SCST. This indicates that Wasserstein self-imitation can improve the semantic matching between generated sentences and their references, while achieving reasonable exact-matching-based metric scores.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
<th>CIDEr</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED-LG [64]</td>
<td>35.2</td>
<td>25.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SA-LSTM [62]</td>
<td>36.6</td>
<td>25.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SCST [40]</td>
<td>40.5</td>
<td>28.4</td>
<td>61.4</td>
<td>51.7</td>
</tr>
<tr>
<td>MBP [59]</td>
<td>41.3</td>
<td>28.7</td>
<td>61.7</td>
<td>48.0</td>
</tr>
<tr>
<td>OUR IMPLEMENTATIONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLE</td>
<td>39.2</td>
<td>27.8</td>
<td>39.8</td>
<td>46.6</td>
</tr>
<tr>
<td>MIXER [43]</td>
<td>40.2</td>
<td>27.9</td>
<td>60.8</td>
<td>50.3</td>
</tr>
<tr>
<td>SCST [44]</td>
<td>40.7</td>
<td>27.9</td>
<td>61.6</td>
<td>51.3</td>
</tr>
<tr>
<td>WSIL-D</td>
<td>42.5</td>
<td>29.0</td>
<td>62.4</td>
<td>52.1</td>
</tr>
<tr>
<td>WSIL-I</td>
<td>41.6</td>
<td>28.4</td>
<td>62.0</td>
<td>52.2</td>
</tr>
</tbody>
</table>

Table 4: Video captioning results on MSR-VTT.

Image Captioning We consider image captioning using the COCO dataset [34], with Karpathy’s split [25]. We follow the implementation of the SCST approach [44], and use extracted image tags [14] as image features (encoder). We report BLEU-k (k from 1 to 4) [39], CIDEr [55], and METEOR [5] scores. Results are summarized in Table 4. Compared with the XE baseline, RL-based methods significantly increase the overall performance under all evaluation metrics. We choose CIDEr as the optimizing metric, since it performs best [44]. Our proposed WSIL shows improvement on every metric compared with the SCST baseline, and WSIL-D performs slightly better than WSIL-I in most metrics. Examples of generated captions are provided in Appendix E.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
<th>CIDEr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show &amp; Tell [57]</td>
<td>27.7</td>
<td>23.7</td>
<td>-</td>
<td>85.5</td>
</tr>
<tr>
<td>OT [9]</td>
<td>31.0</td>
<td>24.6</td>
<td>-</td>
<td>94.7</td>
</tr>
<tr>
<td>Adaptive [36]</td>
<td>33.2</td>
<td>26.6</td>
<td>-</td>
<td>108.5</td>
</tr>
<tr>
<td>Top-Down [1]</td>
<td>33.3</td>
<td>26.3</td>
<td>55.3</td>
<td>111.4</td>
</tr>
<tr>
<td>OUR IMPLEMENTATIONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLE</td>
<td>28.8</td>
<td>24.4</td>
<td>52.0</td>
<td>91.3</td>
</tr>
<tr>
<td>MIXER [43]</td>
<td>30.8</td>
<td>24.7</td>
<td>52.9</td>
<td>101.2</td>
</tr>
<tr>
<td>SCST [44]</td>
<td>32.1</td>
<td>25.4</td>
<td>53.9</td>
<td>105.3</td>
</tr>
<tr>
<td>WSIL-D</td>
<td>31.8</td>
<td>25.7</td>
<td>54.0</td>
<td>107.4</td>
</tr>
<tr>
<td>WSIL-I</td>
<td>32.0</td>
<td>25.6</td>
<td>53.9</td>
<td>107.6</td>
</tr>
</tbody>
</table>

Table 5: Image captioning results on COCO.

Non-parallel Style Transfer Different from the captioning tasks, and for style transfer pair-wise information should be inferred from the training data, which becomes more challenging. We use the same data and split method described in [49]. The accuracy of transferred sentences is evaluated by a pretrained CNN classifier, which achieves an accuracy of 97.4% on the validation set. We also report the BLEU scores with original sentences (BLEU) and human references (Human) [30], to evaluate the content preservation and fluency of transferred sentences. Results are shown in Table 6. WSIL-I and WSIL-D boost the performance compared with the baseline; WSIL-I performs best, since it can explicitly create good transferring pairs in the self-imitation, while WSIL-D pays more attention to successfully transferred sentences.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc(%)</th>
<th>BLEU</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVAE [49]</td>
<td>73.9</td>
<td>20.7</td>
<td>7.8</td>
</tr>
<tr>
<td>Controllable [22]</td>
<td>86.7</td>
<td>58.4</td>
<td>-</td>
</tr>
<tr>
<td>Back-Translation [42]</td>
<td>91.2</td>
<td>2.8</td>
<td>2.0</td>
</tr>
<tr>
<td>DeleteAndRetrieval [30]</td>
<td>88.9</td>
<td>36.8</td>
<td>14.7</td>
</tr>
<tr>
<td>Baseline</td>
<td>90.5</td>
<td>50.4</td>
<td>20.1</td>
</tr>
<tr>
<td>WSIL-D</td>
<td>90.7</td>
<td>51.9</td>
<td>22.3</td>
</tr>
<tr>
<td>WSIL-I</td>
<td>91.5</td>
<td>52.0</td>
<td>25.6</td>
</tr>
</tbody>
</table>

Table 6: Style transfer results on test dataset with human references.

7 Conclusions We have proposed a novel Wasserstein self-imitation learning framework for sequence generation, to alleviate the sparse-rewards problem of RL methods and model-training bias imposed by conventional rewards. This has been done by encouraging self imitation and semantic matching. Two novel training schemes have been presented, directly or indirectly exploiting past good generated sequences. Further, our method can be approximately interpreted as policy optimization with Wasserstein trust-regions. Experiments on unconditional and conditional text generation demonstrate consistent performance improvement over strong baselines. For future work, the proposed method has the potential to be applied on other sequence-generation tasks, such as program synthesis [31].
References


