Nested-Wasserstein Distance for Sequence Generation

Ruiyi Zhang[1], **Changyou Chen**[2], **Zhe Gan**[3], **Zheng Wen**[4], **Wenlin Wang**[1], **Lawrence Carin**[1]
[1] Duke University, [2] University at Buffalo, [3] Microsoft Research, [4] DeepMind

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 $\boldsymbol{s}_0$, a recurrent neural network (RNN) produces a sequence of states $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T)$ given an input sequence-feature representation $(e(y_1), \ldots, e(y_T))$, where $e(\cdot)$ 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*: $\boldsymbol{s}_t = h_\theta(\boldsymbol{s}_{t-1}, e(y_t))$, where θ 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^s from a (trained) model, one iteratively applies the following operations:

$$y_{t+1}^s \sim \text{Multi}(\text{softmax}(g(\boldsymbol{s}_t))), \;\; \boldsymbol{s}_t = h(\boldsymbol{s}_{t-1}, e(y_t^s))\,, \tag{1}$$

where Multi$(\cdot)$ denotes a multinomial distribution. In conditional generation, $\boldsymbol{s}_0$ is initialized with Enc(X), where Enc$(\cdot)$ encodes the relevant information from the context [3, 11]. For unconditional generation, one typically draws $\boldsymbol{s}_0$ 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) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the deterministic environment dynamics and $r(\boldsymbol{s}, y)$ is a reward function. The policy π_θ, parameterized by θ, maps each state $\boldsymbol{s} \in \mathcal{S}$ to a probability distribution over $\mathcal{A}$. The objective is to maximize the expected reward, defined as:

$$J(\pi_\theta) = \mathbb{E}_{Y \sim \pi_\theta}\left[r(Y)\right] = \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{s}_t, y_t) \sim \pi_\theta}\left[r(\boldsymbol{s}_t, y_t)\right]\,, \tag{2}$$

where $Y \triangleq (\boldsymbol{s}_1, y_1, \cdots, \boldsymbol{s}_T, y_T)$ is a trajectory from policy π_θ 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 $r(\boldsymbol{s}_t, y_t)$. 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.

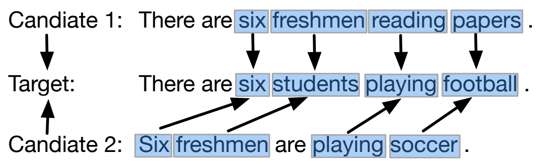
Optimal transport on discrete domains The optimal transport (OT) distance $W_c(\boldsymbol{\mu}, \boldsymbol{\nu})$ is a discrepancy score that measures the distance between two probability distributions $\boldsymbol{\mu}(\cdot)$ and $\boldsymbol{\nu}(\cdot)$ w.r.t. a cost function $c(\cdot, \cdot)$. Specifically, we consider two discrete distributions $\boldsymbol{\mu} \triangleq \sum_{i=1}^n u_i \delta_{\boldsymbol{z}_i}$ and $\boldsymbol{\nu} \triangleq \sum_{j=1}^m v_j \delta_{\boldsymbol{z}'_j}$ with $\delta_{\boldsymbol{z}}$ the Dirac delta function centered on $\boldsymbol{z}$. The weight vectors $\boldsymbol{u} = \{u_i\}_{i=1}^n \in \Delta_n$ and $\boldsymbol{v} = \{v_j\}_{j=1}^m \in \Delta_m$ 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:

$$W_c(\boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{\mathbf{T} \in \Gamma(\boldsymbol{\mu}, \boldsymbol{\nu})} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{T}_{ij} \cdot c(\boldsymbol{z}_i, \boldsymbol{z}'_j) = \min_{\mathbf{T} \in \Gamma(\boldsymbol{\mu}, \boldsymbol{\nu})} \langle \mathbf{T}, \mathbf{C} \rangle\,, \tag{3}$$

where $\sum_{j=1}^n \mathbf{T}_{ij} = \frac{1}{m}$ and $\sum_{i=1}^m \mathbf{T}_{ij} = \frac{1}{n}$ are the constraints, $\langle \cdot, \cdot \rangle$ represents the Frobenius dot-product, and $\mathbf{C}$ is the cost matrix defined by $\mathbf{C}_{ij} = c(\boldsymbol{z}_i, \boldsymbol{z}'_j)$. Intuitively, the OT distance is the minimal cost of transporting mass from $\boldsymbol{\mu}$ to $\boldsymbol{\nu}$.

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 phases) 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:


Candiate 1: There are six freshmen reading papers .
Target: There are six students playing football .
Candiate 2: Six freshmen are playing soccer .

	BLEU	ROUGE-L	CIDEr	Wasserstein
Candidate 1	**36.8**	**50.0**	**163.7**	76.3
Candidate 2	0.0	35.8	55.9	**80.1**

Figure 1: Comparison of different rewards (higher is better).

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 $\mathbf{T}$, and weights are determined by $\mathbf{C}$ in (3).

Definition 1 (Wasserstein Distance between Sequence Pairs) *Consider sequence* $Y = (y_1, \ldots, y_T)$ *as a discrete distribution* $p_Y = \frac{1}{T}\sum_t \delta_{e(y_t)}$ *in the semantic space, with the length-normalized point mass placed at the semantic embedding, i.e.,* $\boldsymbol{z}_t = e(y_t)$ *of each token* y_t *from the sequence* Y*. Given a hypothesis sequence* Y *w.r.t. a reference sequence* Y'*, we define the Wasserstein distance as* $W_c(p_Y, p_{Y'}) \triangleq \min_{\mathbf{T}}\langle \mathbf{T}, \mathbf{C}\rangle$ *between* p_Y *and* $p_{Y'}$ *with cost* $c(\boldsymbol{z}, \boldsymbol{z}')$*. When the cosine distance* $c_{\cos}(\boldsymbol{z}, \boldsymbol{z}') = 1 - \frac{\boldsymbol{z}^\intercal \boldsymbol{z}'}{\|\boldsymbol{z}\|_2 \|\boldsymbol{z}'\|_2}$ *is used as our cost, we define the Wasserstein reward as* $r_s \triangleq \langle \mathbf{T}^*, 1 - \mathbf{C}\rangle$*, where* $\mathbf{T}^*$ *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 the 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* $\boldsymbol{Y} = \{Y_i\}_{i=1}^K$ *and* $\boldsymbol{Y}' = \{Y_j'\}_{j=1}^{K'}$ *drawn from two sequence distributions* $\mathbb{P}_{\boldsymbol{Y}}$ *and* $\mathbb{P}_{\boldsymbol{Y}'}$*, where* K *and* K' *are the number of sequences in* $\boldsymbol{Y}$ *and* $\boldsymbol{Y}'$*. The nested-Wasserstein distance, denoted as* $\mathcal{W}_{nc}(\mathbb{P}_{\boldsymbol{Y}}, \mathbb{P}_{\boldsymbol{Y}'})$*, is a metric measuring the distance between* $\mathbb{P}_{\boldsymbol{Y}}$ *and* $\mathbb{P}_{\boldsymbol{Y}'}$ *in a nested manner:*

$$\mathcal{W}_{nc}(\mathbb{P}_{\boldsymbol{Y}}, \mathbb{P}_{\boldsymbol{Y}'}) \triangleq \min_{T'} \sum_{i=1}^{K} \sum_{j=1}^{K'} T'_{ij} W_c(p_{Y_i}, p_{Y_j'}) \,, \tag{4}$$

where $T'_{ij} \geq 0$ *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* $\mathcal{W}_{nc}(\cdot,\cdot)$ *in algorithms.*

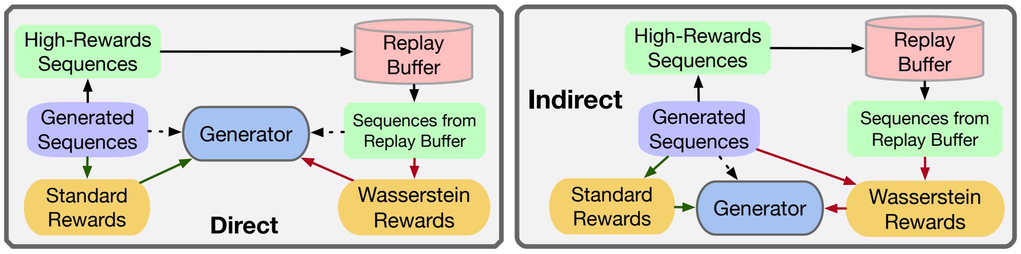


Figure 2: Illustration of the proposed nested-Wasserstein Self-Imitation Learning (WSIL) framework. **Left**: direct WSIL, where samples from the replay buffer are directly used as pseudo-samples to update the generator. **Right**: indirect WSIL, where Wasserstein self-imitation rewards are defined to encourage the generator to imitate samples from the replay buffer.

4 Nested-Wasserstein Self-Imitation Learning

Purely adopting the nested-Wasserstein distance as the reward in a standard policy-gradient method is not effective, because the syntactic information is missing. Instead of combining the rewards with different weights [35, 40], we present the nested-Wasserstein Self-Imitation Learning framework, which provides a novel way to leverage both syntactic (metric) and semantic (Wasserstein) information.

The overall idea of the proposed nested-Wasserstein self-imitation learning (WSIL) is to define a nested-Wasserstein trust-region between the current policy (a.k.a. behavior policy) and the artificial policy defined by the replay buffer. Intuitively, the nested-Wasserstein trust-region encourages the self-imitation of historical high-reward sequences, which provides semantic signals to guide training, in addition to the stabilizing effect from trust-region optimization. Specifically, we consider sequences generated from a conditional behavior policy $\pi_{\theta,X}$, parameterized by θ with the conditional variable X. For example, in image captioning, each sequence is generated conditioned on a given image. For unconditional generation, the conditional variable is empty. Furthermore, a replay buffer is used to store high-reward historical sequences, whose induced conditional policy is denoted as $\pi_{\mathcal{B},X}$. Our new objective function with a nested-Wasserstein trust-region is defined as:

$$J(\pi_\theta) = \mathbb{E}_{X\sim p_d}\left\{\mathbb{E}_{Y^s\sim\pi_{\theta,X}}\left[r(Y^s)\right] - \lambda\cdot\mathcal{W}_{nc}(\pi_{\theta,X},\pi_{\mathcal{B},X})\right\}\ , \tag{5}$$

where $\mathcal{W}_{nc}$ is the nested-Wasserstein distance defined in Definition 1, and $r(\cdot)$ can be a metric reward between Y^s and the ground truth references $\boldsymbol{Y}$. With a little abuse of notation, but for conciseness, we use π_θ 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 $\pi_{\mathcal{B},X}$. 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 $\pi_{\mathcal{B}}$ 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 $\pi_{\mathcal{B}}$. 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 $\pi_{\mathcal{B},X}$ to be $\{Y^s\}$ and $\boldsymbol{Y}^b \triangleq \{Y^b_j\}_{j=1}^{K'}$, with sizes of 1 and K', respectively. Here $Y^s \sim \pi_{\theta,X}$ and

$Y_j^b \sim \pi_{\mathcal{B},X}$, $\forall j$. $\{Y^s\}$ and $\{Y^b\}$ will be used in calculating the nested-Wasserstein distance. Let $r_{ns}(Y^s, \boldsymbol{Y}^b) \triangleq \sum_j T'_j r_s(Y^s, Y_j^b)$ be the nested-Wasserstein reward, with $\mathbf{T}' = \{T'_j\}$ the optimal weights. Based on (5), the objective of WSIL-I is adapted to be:

$$J_I(\pi_\theta) \triangleq \mathbb{E}_{X \sim p_d} \mathbb{E}_{Y^s \sim \pi_{\theta,X}} \left[r(Y^s) + \lambda r_{ns}(Y^s, \boldsymbol{Y}^b) \right] , \tag{6}$$

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 π_θ and introducing a baseline b, the gradient estimate of WSIL-I can be expressed as:

$$\nabla_\theta J_I(\pi_\theta) \approx -\sum_{k=1}^{K} \left[(r(Y_k^s) - b) \nabla_\theta \log \pi_\theta(Y_k^s) + \lambda r_{\text{ns}}(Y_k^s, \boldsymbol{Y}^b) \nabla_\theta \log \pi_\theta(Y_k^s) \right] , \tag{7}$$

where $\boldsymbol{Y}^s \triangleq \{Y_k^s\}_{k=1}^K$, $Y_k^s \sim \pi_{\theta,X}$, $\forall k$. In practice, $\mathcal{I}\left[r(Y^b) > r(Y^s)\right]$ will be combined with the nested-Wasserstein rewards, where $\mathcal{I}(\cdot) = 1$ 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 $\mathcal{I}(\cdot)$ 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 $\mathcal{B}$. The sequences from the replay buffer are directly used as pseudo-samples to update the generator [31]. Similarly, define $r_{ns}(Y^s, \boldsymbol{Y}) \triangleq \sum_j T'_j r_s(Y^s, Y_j)$, with $\mathbf{T}' = \{T'_j\}$ the optimal weights. to be the nested-Wasserstein reward between the sequence Y^s and ground-truth references $\boldsymbol{Y}$. The general objective (5) is then extended to be the objective for WSIL-D, as

$$J_D(\pi_\theta) \triangleq \mathbb{E}_{Y^s \sim \pi_{\theta,X}} \left[r(Y^s) \right] + \lambda \mathbb{E}_{Y^b \sim \pi_{\mathcal{B},X}} \left[r_{ns}(Y^b, \boldsymbol{Y}) \pi_\theta(Y^b) \right] , \tag{8}$$

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 λ that controls the update frequency:

$$\nabla_\theta J_D(\pi_\theta) \approx -\sum_{k=1}^{K} \left[(r(Y_k^s) - b) \nabla_\theta \log \pi_\theta(Y_k^s) \right] - \lambda \sum_{j=1}^{K'} \left[\left(r_{ns}(Y_j^b, \boldsymbol{Y}) - b_s \right)_+ \nabla_\theta \log \pi_\theta(Y_j^b) \right] \tag{9}$$

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 $\mathbf{T}$ via iterative optimization, *i.e.*, $\mathbf{T}^{(t+1)} = \arg\min_{\mathbf{T} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \left\{ \langle \mathbf{T}, \mathbf{C} \rangle + \gamma \cdot \mathbb{D}_{\text{KL}}(\mathbf{T}, \mathbf{T}^{(t)}) \right\}$, where $1/\gamma > 0$ is the generalized step size and the generalized KL-divergence $\mathbb{D}_{\text{KL}}(\mathbf{T}, \mathbf{T}^{(t)}) = \sum_{i,j} \mathbf{T}_{ij} \log \frac{\mathbf{T}_{ij}}{\mathbf{T}_{ij}^{(t)}} - \sum_{i,j} \mathbf{T}_{ij} + \sum_{i,j} \mathbf{T}_{ij}^{(t)}$ 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, λ 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.

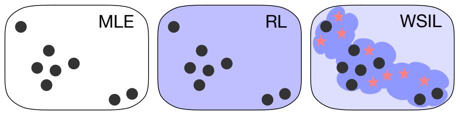


Figure 3: Exploration space of different methods. Circle: ground truth; Star: high-reward sequences.

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 incorporating 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.

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

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

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

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 EMNNLP2017 WMT News dataset using Amazon Mechnical 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.

Methods	**MLE** [7]	**LeakGAN**	**SIL-D**	**SIL-I**
Human scores	2.97±0.05	2.63±0.05	2.54±0.05	2.55±0.05
Methods	**Real**	**WSIL-D**	**WSIL-I**	-
Human scores	4.11±0.04	3.49±0.05	3.41±0.05	-

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.

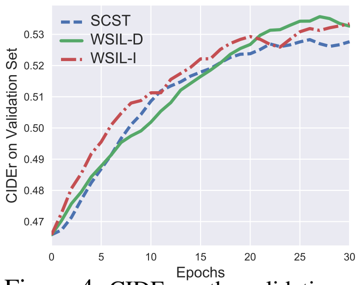


Figure 4: CIDEr on the validation set.

Method	BLEU-4	METEOR	ROUGE-L	CIDEr
ED-LG [64]	35.2	25.2	-	-
SA-LSTM [62]	36.6	25.9	-	-
SCST [40]	40.5	28.4	61.4	51.7
MBP [59]	41.3	28.7	61.7	48.0
OUR IMPLEMENTATIONS				
MLE	39.2	27.8	59.8	46.6
MIXER [43]	40.2	27.9	60.8	50.3
SCST [44]	40.7	27.9	61.6	51.3
WSIL-D	**42.5**	**29.0**	**62.4**	52.1
WSIL-I	41.6	28.4	62.0	**52.2**

Table 4: Video captioning results on MSR-VTT.

Method	BLEU-4	METEOR	ROUGE-L	CIDEr
Show & Tell [57]	27.7	23.7	-	85.5
OT [9]	31.0	24.6	-	94.7
Adaptive [36]	33.2	26.6	-	108.5
Top-Down [1]	33.3	26.3	55.3	111.4
OUR IMPLEMENTATIONS				
MLE	28.8	24.4	52.0	91.3
MIXER [43]	30.8	24.7	52.9	101.2
SCST [44]	**32.1**	25.4	53.9	105.5
WSIL-D	31.8	**25.7**	**54.0**	107.4
WSIL-I	32.0	25.6	53.9	**107.6**

Table 5: Image captioning results on COCO.

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.

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.

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

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

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and vqa. In *CVPR*, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

[3] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *ICLR*, 2017.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[5] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL Workshop*, 2005.

[6] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015.

[7] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv:1811.02549*, 2018.

[8] Liqun Chen, Shuyang Dai, Chenyang Tao, et al. Adversarial text generation via feature-mover's distance. In *NeurIPS*, 2018.

[9] Liqun Chen, Yizhe Zhang, Ruiyi Zhang, et al. Improving sequence-to-sequence learning via optimal transport. *arXiv preprint arXiv:1901.06283*, 2019.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[12] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[13] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: Better text generation via filling in the _. *ICLR*, 2018.

[14] Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. Semantic compositional networks for visual captioning. In *CVPR*, 2017.

[15] Tanmay Gangwani, Qiang Liu, and Jian Peng. Learning self-imitating diverse policies. *arXiv:1805.10309*, 2018.

[16] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *AISTATS*, 2018.

[17] Xiaodong Gu, Kyunghyun Cho, Jungwoo Ha, and Sunghun Kim. Dialogwae: Multimodal response generation with conditional wasserstein auto-encoder. In *ICLR*, 2019.

[18] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *NIPS*, 2017.

[19] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *AAAI*, 2017.

[20] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *ICLR*, 2019.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[22] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Controllable text generation. In *ICML*, 2017.

[23] Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. Supervised word mover's distance. In *NIPS*, 2016.

[24] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[25] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[27] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *NIPS*, 2000.

[28] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[29] Máté Lengyel and Peter Dayan. Hippocampal contributions to control: the third way. In *NIPS*, 2008.

[30] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *NAACL*, 2018.

[31] Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. Memory augmented policy optimization for program synthesis with generalization. In *NeurIPS*, 2018.

[32] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, et al. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[33] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. In *NIPS*, 2017.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[35] Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. Improved image captioning via policy gradient optimization of spider. In *ICCV*, 2017.

[36] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, 2017.

[37] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *LREC 2018*, 2018.

[38] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *ICML*, 2018.

[39] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[40] Ramakanth Pasunuru, Mohit Bansal, and Mohit Bansal. Reinforced video captioning with entailment rewards. In *NAACL*, 2017.

[41] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *ICLR*, 2017.

[42] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. Style transfer through back-translation. In *ACL*, 2018.

[43] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *ICLR*, 2016.

[44] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *CVPR*, 2016.

[45] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv:1509.00685*, 2015.

[46] Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving GANs using optimal transport. In *ICLR*, 2018.

[47] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2015.

[48] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

[49] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *NIPS*, 2017.

[50] Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. Towards diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*, 2018.

[51] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[52] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[53] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[54] Bowen Tan, Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric Xing. Connecting the dots between mle and rl for sequence generation. *arXiv:1811.09740*, 2018.

[55] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.

[56] Cédric Villani. *Optimal transport: old and new*. Springer Science & Business Media, 2008.

[57] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[58] Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*, 2018.

[59] Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. Video captioning via hierarchical reinforcement learning. In *CVPR*, 2018.

[60] Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, 2016.

[61] Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. A fast proximal point method for Wasserstein distance. In *arXiv:1802.04307*, 2018.

[62] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*, 2016.

[63] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[64] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *CVPR*, 2015.

[65] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017.

[66] Ruiyi Zhang, Changyou Chen, Chunyuan Li, and Lawrence Carin. Policy optimization as wasserstein gradient flows. In *ICML*, 2018.

[67] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *ICML*, 2017.

[68] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *SIGIR*, 2018.