IMPROVING CONDITIONAL SEQUENCE GENERATIVE ADVERSARIAL NETWORKS BY STEPWISE EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Conditional sequence generation is a widely researched topic. One of the most important tasks is dialogue generation, which is composed of input-output pairs with the one-to-many property. Given the recent success of generative adversarial networks (GANs), GANs have been used for sequence generation. However, there is still limited work on conditional sequence generation. We investigate the influence of GAN on conditional sequence generation with three artificial grammars which share similar properties with dialogue generation. We evaluate state-of-the-art GAN-related algorithms using accuracy and answer coverage measures. Moreover, we propose stepwise GAN (SGAN) for conditional sequence generation, which predicts the reward at each time-step. SGAN can be seen as the general version of SeqGAN, which includes MCMC and REGS. We also explore energy-based SGAN (EBSGAN). These solve the sparse rewards problem in reinforcement learning. In addition, with weighted rewards for every step, we include the advantage of curriculum learning. Experimental results show that SGAN and EBSGAN are usually comparable with state-of-the-art algorithms and sometimes outperform them.

1 INTRODUCTION

Conditional sequence generation is the task of generating the correspondent response given an input sequence. One of the most important applications is dialogue generation. Dialogue generation is one-to-many; that is, there can be many acceptable responses for a specific input. In previous work, the sequence-to-sequence based dialogue generation model is trained using maximum likelihood estimation, and achieves promising results in terms of both meaning and coherence (Vinyals & Le, 2015). Despite this success, the generated responses given the inputs are still sometimes broken and are often general (for example, *I don't know*). Reinforcement learning was therefore proposed to preserve sequence-level quality as opposed to predicting each word given the sequence history (Ranzato et al., 2015; Kandasamy et al., 2017; Bahdanau et al., 2016).

More recently, generative adversarial networks have been applied to sequence generation, especially for natural language. The discrete nature of random variables for natural language precludes the use of back-propagation. To solve this problem, several approaches have been proposed, such as policy gradient (Yu et al., 2017; Li et al., 2017), Gumbel-Softmax (Kusner & Hernández-Lobato, 2016), MaliGAN (Che et al., 2017), and directly connected WGAN-GP (Gulrajani et al., 2017; Rajeswar et al., 2017; Press et al., 2017). In most previous work, multiple assistant methods are used to stabilize training and often introduce improvements, for example, Monte-Carlo search (Yu et al., 2017; Li et al., 2017; Che et al., 2017) and curriculum learning (Rajeswar et al., 2017; Press et al., 2017). Furthermore, modifications of the GAN objective function have been proposed to improve quality in text generation (Zhang et al., 2017; Lin et al., 2017).

In this paper, we propose stepwise GAN (SGAN) and energy-based stepwise GAN (EBSGAN). Both approaches predict a score for each time step when decoding. SGAN includes weighted factors that change the relative importance of each time step. Note that this factor makes SGAN a general version of SeqGAN. Furthermore, step-time-decreasing weight factors simulate curriculum learning by focusing on generating the head of a sequence. After the first subsequence is fit, further

improvement are found in later subsequences. Experiments show that step-time-decreasing weight factors indeed improve performance. EBSGAN, in turn, uses cross entropy to evaluate the energy of the generated sequence at each time step. As the discriminator here is actually the same as the generator, the generator and discriminator can share the same pre-trained model.

We construct artificial grammars to assist the GANs in conditional sequence generation. For this task, we calculate the accuracy and the coverage of the generated conditioned sequence to evaluate the quality of the model. The coverage is the percentage of the conditioned sequences sampled from the model distribution over all the probable responses. While accuracy reflects coherence and meaningfulness, coverage measures the diversity of the responses. Further, we present examples of the generated dialogue and the results of human evaluation. The proposed models are comparable with or even outperform state-of-the-art algorithms.

2 Related Work

Conditional sequence generation using the seq2seq model (Sutskever et al., 2014; Vinyals & Le, 2015) has been widely studied, especially for dialogue generation. The model can be learned by maximum-likelihood estimation (MLE), which minimizes the word-level cross-entropy between the true data distribution and the generated approximation. Although this method yields reasonable responses, it suffers from exposure bias and does not take into account sequence-level structure (Ranzato et al., 2015). Exposure bias is introduced because of inconsistent conditions between the training and testing stages: while the ground-truth words are fed to the seq2seq model in the training stage, generated words are used in the testing stage.

To solve these problems with MLE, (Ranzato et al., 2015) propose the REINFORCE and MIXER algorithms for sequence generation. By providing a task-specific score for the generated sequence, the REINFORCE algorithm (Williams, 1992) guides the seq2seq model to reach higher scores. Because the score is evaluated based on the whole generated sequence, both MLE problems are solved. However, as the REINFORCE algorithm is not easily trained from scratch, the MIXER algorithm is proposed to integrate MLE and REINFORCE. In this process, we first train the whole sequence using MLE, after which we accumulate the number of last words trained by REINFORCE. For further improvements, (Bahdanau et al., 2016) adopt another reinforcement learning (Sutton & Barto, 1998) based approach – the actor-critic architecture. They train a critic to predict the expected value of each time step to guide the actor. These algorithms outperform the original MLE algorithm on the task-specific score (BLEU) for text generation. Nonetheless, there is no evidence that these task-specific scores are strongly correlated with human prior knowledge. In particular, the relationship between the scores and human evaluation has been proven weak for dialogue generation (Liu et al., 2016).

Recently, the significant success of generative adversarial networks (GAN) for image processing has led researchers to use GANs for natural language. However, this has seen limited success because of the difficulty of backpropagation through discrete random variables. To address this problem, (Yu et al., 2017) use policy gradients on text generation. The reward is provided by a discriminator with Monte-Carlo search. In addition, (Li et al., 2017) adopt the same idea for dialogue generation. They also propose Reward for Every Generation Step (REGS), which is more time-efficient but is outperformed by Monte-Carlo search. Another way to use GAN for natural-language tasks is by using Gumbel-Softmax (Kusner & Hernández-Lobato, 2016), which can simulate the discrete argmax outputs, and be directly backpropagated from the discriminator. Also, MaliGAN (Che et al., 2017) derives the objective function irrelevant to backpropagation through the discrete outputs of the generator. More recently, the improved Wasserstein GAN (WGAN-GP) (Gulrajani et al., 2017) has shown success for text generation by directly feeding the softmax layer to the discriminator, even without pre-training. This breakthrough then inspired (Press et al., 2017) and (Rajeswar et al., 2017) to further investigate WGAN-GP for better performance on text generation.

3 Conditional Sequence Generation

In conditional sequence generation, we generate an output sequence $x \in X$ given an input condition $y \in Y$. When the output sequence is generated from a model, it is denoted x^G; when the output sequence is from real data (training examples), it is instead denoted x^R. In dialogue generation, both

x and y are sequences of words. They can be written as

$$\begin{aligned} x &= \{x_t\}_{t=1}^{T}, x_t \in V \\ y &= \{y_t\}_{t=1}^{T}, y_t \in V. \end{aligned} \tag{1}$$

Words x_t and y_t represent the word at time step t in the interval T of the specific sequences x and y. Set V is the vocabulary set from which the words are selected. In this paper, we generate x given y using the seq2seq model as the generator G (Sutskever et al., 2014; Vinyals & Le, 2015). From Sections 3.1 to 3.3, we introduce maximum likelihood estimation, the REINFORCE algorithm for sequence generation, and GAN for sequence generation. In Section 4, we introduce the proposed approaches.

3.1 Maximum Likelihood Estimation

The basic idea of maximum likelihood estimation (MLE) for conditional sequence generation is to find the parameters for model G that maximize the likelihood of generating the training data. When using MLE to train the generator model G, the objective function is

$$G^* = \arg\max_G E_{(x^R,y)\sim P^R(X,Y)}[\sum_{t=1}^{T} log(P^G(x_t^R|y, x_{1...t-1}^R))], \tag{2}$$

where $P^R(X, Y)$ is the joint distribution of the (x, y) pairs in the training data, and T is the length of x^R. In the training stage, in Eq. (2), the prediction is learned based on $< y, x_{1...t-1}^R >$, but the condition in the testing stage is $< y, x_{1...t-1}^G >$. This is known as *exposure bias*. Moreover, the likelihood is estimated at the word level (Ranzato et al., 2015) only as opposed to the whole sequence.

3.2 REINFORCE

Conditional sequence generation can be formulated as reinforcement learning. Similar to (Ranzato et al., 2015), we describe it as a Markov decision process ***MDP***, where state s, the history, consists of the condition and previous word sequence – in our case, $s = < y, x_{1...t-1}^G >$ – and an action a is the generated word conditioned on the current state – in our case, a is a word in the vocabulary. Each action is generated according to the policy, which is the generator model G. The reward function $R(x, y)$ evaluates the goodness of generating x given y. The generator G learns to maximize the expected reward

$$G^* = \arg\max_G E_{y\sim P^R(Y),x^G\sim P^G(X|y)}[R(x^G, y)], \tag{3}$$

where $P^R(Y)$ is the probability distribution of condition y in the training data, $P^G(X|y)$ is the probability of generating the sequence x^G given the generator G and condition y, and $R(x, y)$ is the reward function. Here $R(x, y)$ is assumed to be the summation of a sequence of step evaluation reward r_t, where r_t is the reward given to the current state-action pair. Reward r_t is an evaluation of the goodness of generating a specific word x_t^G given the conditions and the generated words $< y, x_{1...t-1}^G >$.

$$R(x^G, y) = \sum_{t=1}^{T} r_t, \tag{4}$$

where T is the length of x^G. Note that the main difference between Equations (2) and (3) is that the condition sequence here is $x_{1...t-1}^G$ rather than $x_{1...t-1}^R$. Moreover, each x_t^G here is sampled using softmax rather than argmax over vocabulary set V. As in (Ranzato et al., 2015), we use the policy gradient algorithm REINFORCE (Williams, 1992) for this task. The parameters of the generator θ_G are updated as

$$\theta_G \leftarrow \theta_G + \eta(\sum_{t'=t}^{T} r_t - b_t)\nabla log(p_G(x_t^G|y, x_{1...t-1}^G)), \tag{5}$$

where b_t is the baseline to reduce training variance (Sutton & Barto, 1998; Ranzato et al., 2015), and η is the learning rate.

3.3 GENERATIVE ADVERSARIAL NETWORK

A generative adversarial network (GAN) is composed of a generator and a discriminator (Goodfellow et al., 2014). The discriminator differentiates between real data and data from the generator, and the generator attempts to generate plausible data that will deceive the discriminator. Here we use GAN for conditional sequence generation by considering our model G as the generator and constructing a discriminator D sequentially fed with input-output pairs y and x (Mirza & Osindero, 2014; Li et al., 2017).

3.3.1 SEQGAN

Since the vocabulary V in the generated sequence is a discrete variable, we cannot backpropagate through the generator. In SeqGAN (Yu et al., 2017; Li et al., 2017), the generation task is formulated as a reinforcement learning scenario, similar to that described in Section 3.2, and the reward function is replaced with the discriminator in regular GAN. The discriminator D is then updated through backpropagation, while the generator G is updated using a policy gradient with a reward given by D. The optimization functions for D and G are

$$\begin{aligned} D^* &= \arg\max_D E_{y\sim P^R(Y),x^R\sim P^R(X|y)}[log(D(x^R|y))] + E_{y\sim P^R(Y),x^G\sim P^G(X|y)}[log(1-D(x^G|y))] \\ G^* &= \arg\max_G E_{y\sim P^R(Y),x^G\sim P^G(X|y)}[D(x^G|y)]. \end{aligned} \tag{6}$$

G in Eq. (6) is optimized using the REINFORCE algorithm. The formulation for optimizing G in Eq. (6) is the same as Eq. (3), except that $R(x^G, y)$ is replaced with $D(x^G|y)$.

3.3.2 WASSERSTEIN GAN WITH GRADIENT PENALTY (WGAN-GP)

In recent work, WGAP-GP has been successfully used for sequence generation (Gulrajani et al., 2017; Rajeswar et al., 2017; Press et al., 2017). Instead of using more complicate methods such as a policy gradient, they directly feed the softmax layer into the discriminator. The generator can therefore be updated through backpropagation. We then formulate the conditional version of WGAN-GP as

$$\begin{aligned} D^* &= \arg\max_D E_{y\sim P^R(y),x^R\sim P^R(X|y)}[D(x^R|y)] - E_{y\sim P^R(y),x^G\sim P(X|y)}[D(x^G|y)] \\ G^* &= \arg\max_G E_{y\sim P^R(Y),x^G\sim P^G(X|y)}[D(x^G|y)]. \end{aligned} \tag{7}$$

In contrast to Eq. (6), G in Eq. (7) is directly optimized by backpropagation.

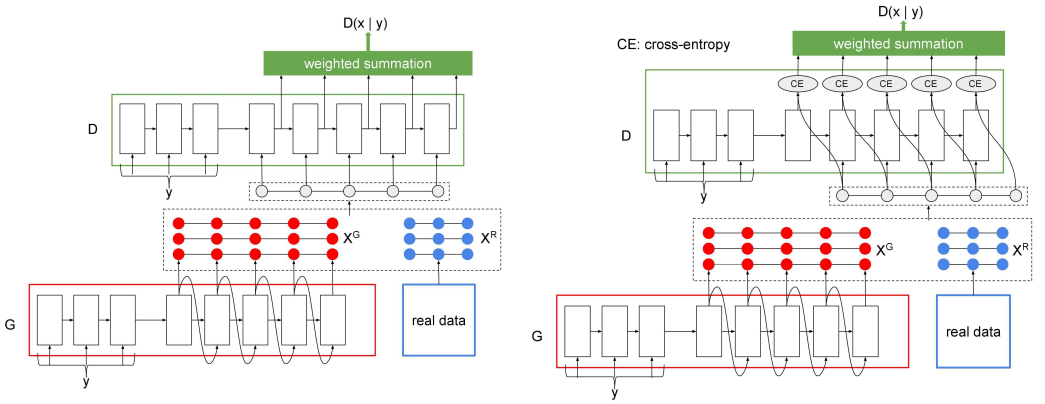
4 PROPOSED APPROACH

Inspired by the actor-critic architecture in (Bahdanau et al., 2016), we seek to predict D at each time step. For this we propose two different approaches: stepwise GAN (SGAN) and energy-based SGAN (EBSGAN).

4.1 STEPWISE GAN

The basic idea of stepwise GAN (see Fig. 1a) is to construct a sequence-to-sequence model as discriminator D. At each time step of D's decoder, the hidden vector is passed to a fully-connected layer. The discriminator D then outputs the evaluation score for each subsequence $\langle y, x_{1\dots t}\rangle$, denoted as $D(x_{1\dots t}|y)$. With discriminator D, we seek to minimize the summation of $D(x^G_{1\dots t}|y)$ over the time steps when input the generated sequences x^G. Simultaneously, D maximizes the summation of $D(x^R_{1\dots t}|y)$ when input the real data x^R. The optimization of D is thus

$$\begin{aligned} D(x|y) &= \sum_{t=1}^{T} \alpha_t^D D(x_{1\dots t}|y) \\ D^* &= \arg\max_D E_{y\sim P^R(y),x^R\sim P^R(X|y)}[D(x^R|y)] - E_{y\sim P^R(y),x^G\sim P(X|y)}[D(x^G|y)], \end{aligned} \tag{8}$$



(a) Stepwise GAN

(b) Energy-based stepwise GAN

Figure 1: Illustration of stepwise GAN (SGAN) and energy-based stepwise GAN (EBSGAN)

where α_t^D is a weighted factor with $\sum_{t=1}^{T} \alpha_t^D = 1$. We set $\alpha_t^D = \frac{1}{T}$ in the experiments, but it is possible to give different steps different weights. SGAN is a generalized version of SeqGAN. If we set $\alpha_T^D = 1$, and $\alpha_t^D = 0$ for $t < T$, then the proposed approach is equivalent to SeqGAN without MCMC. If we take $D(x_{1...t}|y)$ as the value obtained by MCMC in SeqGAN, then the proposed approach is the same as SeqGAN with MCMC. However, the proposed approach is more efficient than MCMC because the time-consuming sampling step is not needed. Also, the REGS algorithm (Li et al., 2017) can be induced by set $\alpha_t^D = 1$ at a randomly chosen time step.

The parameters of the generator G, θ_G, are updated using the REINFORCE algorithm. If the objective function of G is

$$G^* = \arg\max_G E_{y\sim P^R(Y), x^G\sim P^G(X|y)}[D(x^G|y)], \tag{9}$$

then the formulation for parameter update is

$$\theta_G \leftarrow \theta_G + \eta\alpha_t^G(\sum_{t'=t}^{T} D(x_{1...t'}|y))\nabla log(p_G(x_t^G|y, x_{1...t-1}^G)). \tag{10}$$

Using factor α_t^G, we diversify training by arbitrarily weighting the importance of each time step[1]. We explore the influence of different values of α_t^G in the experiments below, in which we call this SGAN-Seq. For θ_G, the update formulation is

$$\theta_G \leftarrow \theta_G + \eta\alpha_t^G D(x_{1...t}|y)\nabla log(p_G(x_t^G|y, x_{1...t-1}^G)). \tag{11}$$

We compare the two update strategies in the experiments.

4.2 Energy-based SGAN

We propose another type of GAN to predict values for each time step called energy-based SGAN (EBSGAN), which is mainly inspired by energy-based GAN (Zhao et al., 2016) but has a completely different structure. As depicted in Fig. 1b, the discriminator D here has the same architecture as generator G, which is a seq2seq model. The benefit of EBSGAN is that we can use a seq2seq model pre-trained using MLE to initialize the discriminator D. The discriminator predicts a probability distribution over vocabulary V, in order to find a model D that maximizes the likelihood of real data and minimizes the likelihood of generated sequences. At the same time, the generator G attempts to maximize the likelihood of discriminator D. Therefore, the optimization functions are written as

$$\begin{aligned} D^* = &\arg\max_D E_{(x^R,y)\sim P^R(X,Y)}[\sum_{t=1}^{T} log(P^D(x_t^R|y, x_{1...t-1}^R))] \\ &+maximum(0, \beta - E_{y\sim P^R(Y), x^G\sim P^G(X|y)}[\sum_{t=1}^{T} log(P^D(x_t^G|y, x_{1...t-1}^G))]) \end{aligned} \tag{12}$$

[1] α_t^G needs not be equivalent to α_t^D.

Table 1: Grammar definitions and examples

Grammar	Definition	Examples
Sequence	Continue the sequence for a random length	123: 4, 45, ...
Counting	Randomly choose a digit, and then calculate the left- and right-hand lengths	123: 012, 121, 230
Addition	Randomly partition, and then add the two numbers	123: 15, 24

$$G^* = \arg\max_G E_{y\sim P^R(Y), x^G\sim P^G(X|y)}[\sum_{t=1}^{T} log(P^D(x_t^G|y, x_{1\ldots t-1}^G))], \tag{13}$$

where β is the threshold for preventing the discriminator D from distinguishing generated data from real data too easily. $P^D(x_t^G|y, x_{1\ldots t-1}^G)$ in Eq. (13) is the probability of generating x_t^G given $< y, x_{1\ldots t-1}^G >$ based on the seq2seq model of D. When $P(X^G|Y)$ is far from $P(X^R|Y)$, the threshold turns off the second term in Eq. (12). The aim of discriminator D is therefore the same as maximum-likelihood estimation. Eq. (13) then leads the generator to reproduce the distribution learned by the discriminator.

5 Experiments

We use a recurrent neural network for both the discriminator and generator due to its strong sequential correlation (Press et al., 2017; Rajeswar et al., 2017). Specifically, we use gated recurrent units (GRUs) (Chung et al., 2014) in our experiments. We view the noise feature in GAN as the random process of sampling from the softmax layer distribution.

5.1 Artificial Grammars

To better evaluate GANs for conditional sequence generation, we define three artificial grammars: sequence, counting, and addition. The three grammars are described in Table 1. For the sequence grammar, the aim is to generate a continued consecutive number sequence behind the input Y. For example, for input $\langle 1, 2, 3\rangle$, the answer would be a consecutive number sequence of any length starting with 4, such as $\langle 4, 5, 6, 7, 8\rangle$. The counting grammar is more complicated. The generated sequence should contain exactly 3 words, where the median is a randomly selected word from the input sequence. The first generated word should be the number of words on the left-hand side of the selected median, while the last generated word should be the number of words on the right-hand side. For example, when the input is $\langle 5, 9, 2, 8, 3, 2, 9, 1\rangle$, one permissible generated sequence is $\langle 0, 5, 7\rangle$. Last, for the addition grammar we generate the addition of two numbers randomly segmented from the input sequence. That is, for input $\langle 8, 1, 3, 4\rangle$, then one permissible output is the addition of 8 and 134 – thus $\langle 1, 4, 2\rangle$. Note that both the input and output numbers for this grammar are represented in terms of their corresponding digits.

The purpose of these design is to imitate major properties in dialogue generation, such as variable-length, repeated beginning-sequence, the same sequence space, one-to-many, and many-to-one. The variable-length property means there is no fixed length for the input and output sequences. The repeated beginning-sequence property means the beginning subsequences are usually shared by many data, for instance *What* and *I am* in natural language. The same sequence space here means that the input and output have the same structure and as such are sampled from the same space. Finally, one-to-many and many-to-one are quite common in dialogue generation. For example, when asking *How are you?*, responses vary from *I'm fine* to *Great! How are you?*. Also, the same response can be paired with multiple questions, such as for *My name is Paul* in response to *What's your name?* and *Who are you?*.

We then randomly generate 100,000 samples as training data, 10,000 as development data, and 10,000 samples as testing data. The architectures are set to one layer with 128 hidden units. Since the artificial grammars are easy to evaluate, we evaluate our results using the three measures in

Table 2: SGAN with different weight factors. The dash (-) here notes that the weight factors yield neither improvements nor deterioration.

		VLGAN-RL			VLGAN-VRL		
		Acc	**AccS**	**Cov**	**Acc**	**AccS**	**Cov**
Sequence	Uniform	97.19	75.92	66.08	97.11	74.85	67.49
	Increase	97.17	75.33	66.75	97.18	73.30	68.19
	Decrease	97.16	74.73	67.14	97.09	74.34	67.48
Counting	Uniform	77.23	71.43	70.18	74.91	70.70	69.84
	Increase	74.50	69.66	70.86	74.54	**70.74**	**70.11**
	Decrease	**81.98**	**72.24**	69.02	**75.47**	70.64	69.82
Addition	Uniform	44.26	32.39	**31.91**	44.77	32.34	31.65
	Increase	-	-	-	-	-	-
	Decrease	**44.94**	**32.19**	31.67	**45.55**	**32.49**	**32.01**

Table 3: Results of artificial grammars with different backpropagation methods. Evaluation label Acc (%) is the accuracy of argmax samples, AccS (%) is the accuracy of softmax samples, and Cov (%) is the coverage of softmax samples over probable answers. The dash (-) here indicates that the algorithm introduced no improvements based on the pre-trained model.

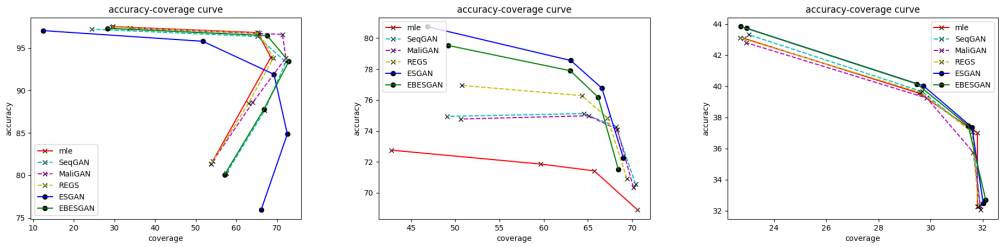
	Sequence			**Counting**			**Addition**		
	Acc	**AccS**	**Cov**	**Acc**	**AccS**	**Cov**	**Acc**	**AccS**	**Cov**
MLE	**97.43**	81.32	53.77	73.48	68.89	**70.63**	44.57	32.28	31.79
REINFORCE	*99.81*	*97.30*	*4.54*	*99.97*	*99.36*	*16.99*	*79.98*	*75.60*	*18.32*
WGAN-GP	-	-	-	-	-	-	-	-	-
MaliGAN	97.34	**81.66**	54.19	74.12	70.35	70.15	44.27	32.05	31.92
SeqGAN	97.20	80.28	57.61	74.59	70.56	70.39	44.87	32.30	31.90
MC-SeqGAN	97.20	80.98	55.49	72.96	68.10	70.54	44.72	32.28	31.83
REGS	97.42	81.36	53.82	75.99	70.93	69.40	44.64	32.32	32.01
SGAN	97.19	75.92	66.08	**81.98**	**72.24**	69.02	44.94	32.19	31.67
SGAN-Seq	97.11	74.85	**67.49**	75.47	70.64	69.82	45.55	32.49	32.01
EBSGAN	97.32	80.09	57.12	79.78	71.52	68.44	**45.74**	**32.68**	**32.11**

Table 3. The first is the accuracy of samples generated from the argmax policy (Acc), the second is that generated from the softmax probability (AccS), and the last is the coverage of softmax samples over all the permissible answers of the specific grammar (Cov). For all inputs in testing data, we generate 50 softmax samples for evaluation. Among the 50 samples, the correct ones are collected to evaluate the coverage. Note that the last two measures have not been used in previous work. We report them to evaluate whether the one-to-many property is being learned. To ensure a fair comparison, all the algorithms are based on the same pre-trained model: the MLE model listed in Table 3[2].

In Table 2 we compare different weight factors for SGAN. In practice, we find as that the α_t^D setting can actually be included in α_t^G, we set $\alpha_t^D = \frac{1}{T}$ for all the presented cases. As depicted in Table 2, the three grammars are trained using three sorts of weighted factors: uniform ($\alpha_t^G = 1$), increasing ($\alpha_t^G = t$), and decreasing ($\alpha_t^G = T - t + 1$). The results clearly show that the time-step-decreasing weight factors positively affect training. We believe this is because the training spirit of decreased weighted factor start from first correcting beginning subsequence, which is similar to curriculum learning. We also tried a weighted version of EBSGAN, but this yielded little change on the three grammars.

The best weight factors for SGAN and SGAN-Seq were then selected for comparison with other algorithms in Table 3. We find that in our architecture, using the SGD optimizer for the generator and the Adam optimizer for the discriminator improves both training speed and performance. In the

[2]The accuracy of REINFORCE here is taken as be the upper bound, since the given reward is the true accuracy.



(a) Sequence (b) Counting (c) Addition

Figure 2: Sampled accuracy and coverage curves

sequence grammar, all the GAN-based methods showed higher coverage with reduced accuracy. To investigate the corresponding accuracy and coverage trends, we plot the accuracy-coverage curve by sharpening the softmax layer. Fig. 2a reveals their inconsistent advantages along the curve. Therefore, we summarize there are no positive or negative impacts on the sequence grammar – this only changes to another curve. Also, the table clearly shows that SGAN (include SGAN-Seq) and EBSGAN outperform the previous algorithms, especially in argmax accuracy. To better understand the influence on coverage, see the curves in Figures 2b and 2c. In both grammars, the differences are more significant when sampling from a sharper distribution. Moreover, the coverage and softmax accuracy are both clearly improved.

5.2 Dialogue Generation

We use OpenSubtitle (Tiedemann, 2009) as the training data with a vocabulary of the top 4,000 most frequently occurring words. Both the generator and discriminator are 1-layer GRUs with a hidden dimension set to 1024. To compare the improvements introduced by all the algorithms, we first pre-trained the generator using MLE, after which we randomly chose 32,000 examples from the training data to further modify the model. Time-comparable algorithms such as SeqGAN, SGAN, and EBSGAN were then trained using the pre-training model with the training data subset. The SeqGAN and SGAN discriminators were pre-trained on real data and generated data from the pre-trained generator. The EBSGAN discriminator specifically used the same pre-trained model as the generator. Note that these models were all trained without any other auxiliary tricks such as MIXER, curriculum learning, or teacher forcing.

For human evaluation, we randomly selected 20 inputs from the development set. Each model then generated one argmax sample and two softmax samples given an input. We presented both an input and the generated outputs to 16 judges, who were asked to score each generated output from 1 to 5. We then normalized each judge's scores and calculated two measures from the results: Acc and AccS. Acc is the average score for the argmax samples, and AccS is the average score of the three generated samples. In addition, we evaluated the smoothness of the softmax layer by averaging the log probability of 10 softmax samples for each of 1,000 inputs randomly selected from development set. That is, a lower average log probability indicates a smoother distribution for the softmax layer. We also sought to evaluate their coverage, though there is no complete reference set for dialogue generation. Specifically, a lower average log probability with higher softmax accuracy for a model suggests that this model does have wider coverage, because while it generates a wide range of samples, most of the coverage is actually good.

In Table 5, SGAN outperforms the other three models in both Acc and AccS with a lower AvgLog-Prob than MLE. This indicates that SeqGAN and SGAN yield better coverage than MLE. When interacting with the model, it generates a greater number of proper responses to an input. On the other hand, although the EBSGAN achieves better performance in artificial grammars, it diverges from the MLE model in the dialogue generation task with its lower AccS and much lower AvgLog-Prob.

Table 4: Argmax examples of dialogue generation

Input	I don' t wanna know your name , your story .
MLE	I ' m not a _UNK .
SeqGAN	You ' re a _UNK _UNK .
SGAN	You wanna know what I mean ?
EBSGAN	I ' m not interested .
Input	We decide when we die and how .
MLE	We don ' t know .
SeqGAN	We don ' t want to die .
SGAN	No , you don ' t .
EBSGAN	We ' re not leaving .

Table 5: Human evaluation and average log probability for dialogue generation

	Acc (%)	**AccS (%)**	**AvgLogProb**
MLE	46.92	52.61	-2.201
SeqGAN	48.20	56.89	-2.383
SGAN	**54.38**	**60.92**	-2.281
EBSGAN	43.83	46.61	-2.652

6 CONCLUSION

In this paper we demonstrate the essence of GANs for conditional sequence generation. Therefore the experiments were conducted without other auxiliary approaches, such as the notable teacher forcing and curriculum learning. We show that the proposed SGAN and EBSGAN models perform equally to or outperform current GANs using the policy gradient scheme on artificial grammars. SGAN also significantly outperforms other approaches as evaluated by humans on a dialogue generation task.

Our proposed artificial grammars not only accurately reflect model coverage and accuracy but also boast clearly distinguishable styles. For example, the sequence style can be the length, the counting style can be the selected digit position, and the addition style can be the selected partition position. This property lends itself to investigating style transference for sequences, which is one of our aims for future work.

REFERENCES

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Kirthevasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David Carter. Batch policy gradient methods for improving neural conversation models. *arXiv preprint arXiv:1702.03334*, 2017.

Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. *arXiv preprint arXiv:1705.11001*, 2017.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*, 2016.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*, 2017.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*, 2017.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Jörg Tiedemann. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov (eds.), *Recent Advances in Natural Language Processing*, volume V, pp. 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria, 2009. ISBN 978 90 272 4825 1.

Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pp. 2852–2858, 2017.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*, 2017.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.