Latent Code and Text-based Generative Adversarial Networks for Soft-text Generation

Md. Akmal Haidar, Mehdi Rezagholizadeh, Alan Do-Omri and Ahmad Rashid
Huawei Noah's Ark Lab, Montreal Research Center, Canada
md.akmal.haidar@huawei.com, mehdi.rezagholizadeh@huawei.com
alan.do.omri@huawei.com, ahmad.rashid@huawei.com

Abstract

Text generation with generative adversarial networks (GANs) can be divided into the text-based and code-based categories according to the type of signals used for discrimination. In this work, we introduce a novel text-based approach called Soft-GAN to effectively exploit GAN setup for text generation. We demonstrate how autoencoders (AEs) can be used for providing a continuous representation of sentences, which we will refer to as soft-text. This soft representation will be used in GAN discrimination to synthesize similar soft-texts. We also propose hybrid latent code and text-based GAN (LATEXT-GAN) approaches with one or more discriminators, in which a combination of the latent code and the soft-text is used for GAN discriminations. We perform a number of subjective and objective experiments on two well-known datasets (SNLI and Image COCO) to validate our techniques. We discuss the results using several evaluation metrics and show that the proposed techniques outperform the traditional GAN-based text-generation methods.

1 Introduction

Text generation is an active research area and has many real-world applications, including, but not limited to, machine translation (Bahdanau et al., 2014), AI chat bots (Li et al., 2017), image captioning (Xu et al., 2014), question answering and information retrieval (Wang et al., 2017). Recurrent neural network language models (RNNLMs) (Mikolov et al., 2010) is the most popular approach for text generation which rely on maximum likelihood estimation (MLE) solutions such as teacher forcing (Williams and Zipser, 1989) (i.e. the model is trained to predict the next word given all the previous predicted words); however, it is well-known in the literature that MLE is a simplistic objective for this complex NLP task (Li et al., 2017). It is reported that MLE-based methods suffer from exposure bias (Huszr, 2015), which means that at training time the model is exposed to gold data only, but at test time it observes its own predictions. Hence, wrong predictions quickly accumulate and result in poor text generation quality.

However, generative adversarial networks (GANs) (Goodfellow et al., 2014) which are based on an adversarial loss function suffers less from the mentioned problems of the MLE solutions. The great success of GANs in image generation framework (Salimans et al., 2016) motivated researchers to apply its framework to NLP applications as well. GANs have been extensively used recently in various NLP applications such as machine translation (Wu et al., 2017; Yang et al., 2017a), dialogue models (Li et al., 2017), question answering (Yang et al., 2017b), and natural language generation (Gulrajani et al., 2017; Rajeswar et al., 2017; Press et al., 2017; Kim et al., 2017; Zhang et al., 2017; Cifka et al., 2018; Spinks and Moens, 2018; Haidar and Rezagholizadeh, 2019; Gagnon-Marchand et al., 2019; Rashid et al., 2019). However, applying GAN in NLP is challenging due to the discrete nature of the text. Consequently, back-propagation would not be feasible for discrete outputs and it is not straightforward to pass the gradients through the discrete output words of the generator.

Traditional methods for GAN-based text generation can be categorized according to the type of the signal used for discrimination into two categories: text-based and code-based techniques. Code-based methods such as adversarial autoencoder (AAE) (Makhzani et al., 2015) and adversarially regularized AE (ARAE) (Kim et al., 2017) derive a latent space representation of the text using an AE and attempt to learn data manifold of that latent space (Kim et al., 2017) instead of mod-

Proceedings of NAACL-HLT 2019, pages 2248–2258
Minneapolis, Minnesota, June 2 - June 7, 2019.

eling text directly. Text-based solutions, such as reinforcement learning (RL) based methods or approaches based on continuous approximation of discrete sampling, focus on generating text directly from the generator. RL-based methods treat the distribution of GAN generator as a stochastic policy and hold the discriminator responsible for providing proper reward for the generator's actions. However, the RL-based methods often need pre-training and are computationally more expensive compared to the methods of the other two categories. In the continuous approximation approach for generating text with GANs, the goal is to find a continuous approximation of the discrete sampling by using the Gumbel Softmax technique (Kusner and Hernández-Lobato, 2016) or approximating the non-differentiable argmax operator with a continuous function (Zhang et al., 2017; Gulrajani et al., 2017).

In this paper, we introduce Soft-GAN as a new solution for the main bottleneck of using GAN for text generation. Our solution is based on an AE to derive a soft representation of the real text (i.e. soft-text). This soft-text is fed to the GAN discriminator instead of the conventional one-hot representation used in (Gulrajani et al., 2017). Furthermore, we propose hybrid latent code and text-based GAN (LATEXT-GAN) approaches and show that how we can improve code-based and text-based text generation techniques by considering both signals in the GAN framework. We summarize the main contributions of this paper as:

- We introduce a new text-based solution Soft-GAN using the above soft-text discrimination. We also demonstrate the rationale behind this approach.
- We introduce LATEXT-GAN approaches for GAN-based text generation using both latent code and soft-text discrimination. To the best of our knowledge, this is the first time where a GAN-based text generation framework uses both code and text-based discrimination.
- We evaluate our methods using subjective and objective evaluation metrics. We show that our proposed approaches outperform the conventional GAN-based text generation techniques that do not need pre-training.

2 Background

Generative adversarial networks include two separate deep networks: a generator and a discriminator. The generator G_θ takes in a random variable, z following a distribution $P_z(z)$ and attempt to map it to the real data distribution $P_x(x)$. The output distribution of the generator is expected to converge to the real data distribution during the training. On the other hand, the discriminator f_w is expected to discern real samples from generated ones by outputting zeros and ones, respectively. During training, the generator and discriminator generate samples and classify them, respectively by adversarially affecting the performance of each other. In this regard, an adversarial loss function is employed for training (Goodfellow et al., 2014):

$$\min_\theta \max_w (E_{x\sim P_x(x)} f_w(x) + E_{z\sim P_z(z)}(1 - f_w(G_\theta(z)))) \quad (1)$$

As stated, using GANs for text generation is challenging because of the discrete nature of text. The main bottleneck is the *argmax* operator which is not differentiable and blocks the gradient flow from the discriminator to the generator.

$$\min_\theta E_{z\sim P_z(z)}(1 - f_w(\text{argmax}(G_\theta(z)))) \quad (2)$$

3 Related Work

Text-based Solutions Generating text using pure GANs was inspired by improved Wasserstein GAN (IWGAN) work (Gulrajani et al., 2017). In IWGAN, a character level language model was developed based on adversarial training of a generator and a discriminator. Their generator is a convolution neural network (CNN) generating fixed-length texts. The discriminator is another CNN receiving 3D tensors as input sentences. The real sentences and the generated ones are represented using one-hot and softmax representations, respectively. A similar approach to IWGAN was proposed in (Rajeswar et al., 2017) with a recurrent neural network (RNN) based generator. In (Press et al., 2017), RNN is trained to generate text with GAN using curriculum learning (Bengio et al., 2009). The TextGAN (Zhang et al., 2017) method was proposed to alleviate the mode-collapsing problem by matching the high-dimensional latent feature distributions of real and synthetic sentences (Salimans et al., 2016).

Morever, several versions of the RL-based techniques using GAN have been introduced in the literature including Seq-GAN (Yu et al., 2017), RankGAN (Lin et al., 2017), MaliGAN (Che et al., 2017), LeakGAN (Guo et al., 2017), and MaskGAN (Fedus et al., 2018).

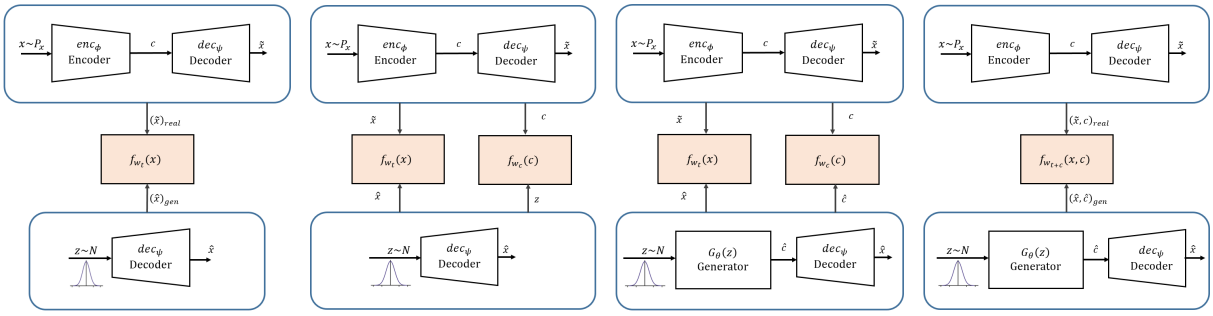
Code-based Solutions AEs have been exploited along with GANs in different architectures for computer vision application such as AAE (Makhzani et al., 2015), ALI (Dumoulin et al., 2016), and HALI (Belghazi et al., 2018). Similarly, AEs can be used with GANs for generating text. For instance, ARAE was proposed in (Kim et al., 2017) where it employs a discrete auto-encoder to learn continuous codes based on discrete inputs with a WGAN objective to learn an implicit probabilistic model over these codes. ARAE aims at exploiting GANs ability to push the generator to follow a continuous code space corresponding to the encoded real text in the AE. The generated code is then used by the decoder to generate the synthetic texts. A different version of the ARAE method was also introduced in (Spinks and Moens, 2018). In (Subramanian et al., 2018), sentence embeddings were learned by a generator to model a pre-trained sentence representation obtained by a general-purpose sentence encoder. A temperature sweeping framework was discussed in (Caccia et al., 2018) to evaluate the text generation models over whole quality-diversity spectrum and pointed out the fundamental flaws of quality-only evaluation. The Variational autoencoders (VAEs) (Kingma and Welling, 2013) were also applied for text generation in (Bowman et al., 2015; Hu et al., 2017).

4 Methodology

In this section, we introduce a new text-based solution by discriminating the reconstructed output of an AE (i.e., soft-text) with the synthesized generated text and we call it as Soft-GAN. Here, we use the decoder of the AE as the generator to generate the synthesized texts from random noise data z. The model is described in Figure 1a. We demonstrate the rationale behind this soft-GAN approach, which is to make the discrimination task of the discriminator between the real and synthetic texts more difficult and consequently providing a richer signal to the generator. We also introduce three LATEXT-GAN approaches where both code and text-based signals will be used in the GAN framework. We introduce LATEXT-GAN I approach on top of the AAE method. LATEXT-GAN II and III approaches will be proposed based on the ARAE approach. In the LATEXT-GAN I and II techniques, we use separate discriminators for the code and text discriminations. In the LATEXT-GAN III approach, the concatenation of the synthetic code and the synthetic text tries to mimic the concatenation of the latent code and the soft-text using a discriminator. The schematic diagram of the LATEXT-GAN I, II, and III approaches are described in Figures 1b, 1c, and 1d respectively. We share the parameters of the decoder of the AE to generate the synthesized text $\hat{x}$. In order to train these approaches, we train the auto-encoder and the GAN alternatingly by minimizing their loss functions using the WGAN-gradient penalty (WGAN-GP) approach (Gulrajani et al., 2017). In each iteration, the first step is the AE training in all of these techniques followed by the GAN loss functions. The autoencoder can be trained by using a cross-entropy or mean-squared loss functions. The input x to the AE is mapped to a latent code representation c which is decoded to soft-text $\tilde{x}$. In our experiments, we train the auto-encoder using mean-squared loss $\min_{(\phi,\psi)} ||x - \tilde{x}||^2$. We describe the training details of the Soft-GAN, LATEXT-GAN I, II, and III methods in the following subsections where the term critic and discriminator used interchangeably. The notations that are used in this paper are described in Table 1.

x	One-hot representation of the training text
$\tilde{x}$	Soft-text: Reconstructed output of the AE
$\hat{x}$	Synthesized generated text
$\bar{x}$	$[\bar{x} \sim P_{\bar{x}}] \leftarrow \alpha\,[\tilde{x} \sim P_{\tilde{x}}] + (1-\alpha)\,[\hat{x} \sim P_{\hat{x}}]$
$z \sim N$	Random data drawn from a normal distribution
c	Latent code representation of the training text
$\hat{c}$	Synthesized generated code
$\bar{c}$	$[\bar{c} \sim P_{\bar{c}}] \leftarrow \alpha\,[c \sim P_c] + (1-\alpha)\,[\hat{c} \sim P_{\hat{c}}]$
$\bar{cz}$	$[\bar{cz} \sim P_{\bar{cz}}] \leftarrow \alpha\,[c \sim P_c] + (1-\alpha)\,[z \sim P_z]$
w_{t+c}	Parameters of the combination of text and code-based discriminator
w_t	Parameters of the text-based discriminator
w_c	Parameters of the code-based discriminator
ϕ	Parameters of the encoder
ψ	Parameters of the decoder
θ	Parameters of the generator
λ	Gradient penalty co-efficient
∇	describes gradient

Table 1: Notations that are used in this paper



(a) Soft-GAN (b) LATEXT-GAN I (c) LATEXT-GAN II (d) LATEXT-GAN III

Figure 1: Proposed Approaches

4.1 Soft-GAN

As stated, in conventional text-based discrimination approach IWGAN (Gulrajani et al., 2017), the real and the synthesized generated text are described by the one-hot and the softmax represenation respectively. A disadvantage of this technique is that the discriminator is able to tell apart the one-hot input from the softmax input very easily. One way to avoid this issue is to derive a continuous representation of words rather than their one-hot and train the discriminator to differentiate between the continuous representations. We use a conventional AE to replace the one-hot representation with softmax reconstructed output ($\tilde{x}$), which we refer to as soft-text. This soft-text representation is used as the real input to the discriminator. The synthetic generated text $\hat{x}$ is obtained by inputting the random noise data z to the shared decoder. We define the proposed method as Soft-GAN. In each iteration, the model is trained using the following steps after the AE training step:

- Train the text-based discriminator f_{w_t} for k times and the decoder once using the loss L_{critic_t} to maximize the ability of the f_{w_t} to discriminate between $\tilde{x}$ and $\hat{x}$:

$$L_{critic_t} = \min_{(w_t,\psi)} (-E_{\tilde{x}\sim P_{\tilde{x}}}[f_{w_t}(\tilde{x})] + E_{\hat{x}\sim P_{\hat{x}}} [f_{w_t}(\hat{x})] + \lambda E_{\bar{x}\sim P_{\bar{x}}}[(||\nabla_{\bar{x}} f_{w_t}(\bar{x})||_2 - 1)^2]) \quad (3)$$

- Train the decoder based on the loss L_{gen} to fool the discriminator with improving the representation $\hat{x}$:

$$L_{gen} = \min_{\psi}(-E_{\hat{x}\sim P_{\hat{x}}}[f_{w_t}(\hat{x})] + E_{\tilde{x}\sim P_{\tilde{x}}} [f_{w_t}(\tilde{x})]) \quad (4)$$

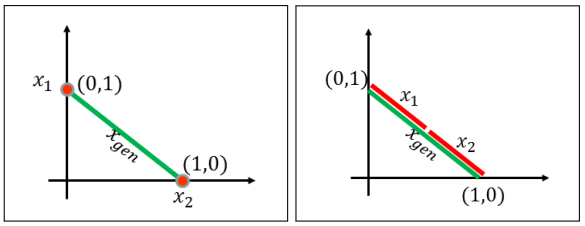


Figure 2: Locus of the input vectors to the discriminator f_{w_t} for a two-word language; Left panel: IWGAN, Right panel: Soft-GAN

4.1.1 Rationale: Why Soft-GAN should Work Better than IWGAN?

Suppose we have a language of vocabulary size of two words: x_1 and x_2. In the IWGAN approach, the one-hot representation of these two words (as two points in the Cartesian coordinates) and the span of the generated softmax outputs (as a line segment connecting them) is depicted in the left panel of Figure 2. As evident graphically, the task of the critic is to discriminate the points from the line connecting them, which is a rather simple very easy task, which makes it more prone to vanishing gradient.

On the other hand, the output locus of the soft-GAN decoder would be two red line segments as depicted in Figure 2 (Right panel) instead of two points (in the one-hot case). The two line segments lie on the output locus of the generator, which will make the generator more successful in fooling the critic.

4.2 LATEXT-GAN I

In the LATEXT-GAN I approach (Figure 1b), we deploy two critics: one for the soft-text discrimination and the other for the latent code discrimination. The text-based discriminator f_{w_t} is used to discriminate the soft-text output $\tilde{x}$ with the synthesized text $\hat{x}$ which is obtained by inputting the

random noise data to the shared decoder. The code-based discriminator f_{w_c} is used to discriminate the random noise data z with the latent code c in the AAE setting which was explored for image generation. In the AAE setting (Makhzani et al., 2015), the encoder enhances its representation to a prior distribution z. It can be seen that the LATEXT-GAN I can be obtained by adding the above code-based discriminator f_{w_c} into the Soft-GAN. In each iteration, the model is trained using the following steps after the AE training step and the Equation 3 step of the Soft-GAN in section 4.1:

- Train the code-based discriminator f_{w_c} for k times using the loss L_{critic_c} to maximize the ability of the f_{w_c} to discriminate between c and z:

$$L_{critic_c} = \min_{w_c}(E_{c\sim P_c}[f_{w_c}(c)]- E_{z\sim P_z}[f_{w_c}(z)]+ \lambda E_{\bar{cz}\sim P_{\bar{cz}}}[(||\nabla_{\bar{cz}} f_{w_c}(\bar{cz})||_2 - 1)^2]) \quad (5)$$

- Train the encoder and the decoder neural networks once with the loss L_{gen} to fool the discriminators f_{w_c} and f_{w_t} with improving the representations c and $\hat{x}$ respectively:

$$L_{\text{gen}} = \min_{(\phi,\psi)}(-E_{\hat{x}\sim P_{\hat{x}}}[f_{w_t}(\hat{x})] + E_{\tilde{x}\sim P_{\tilde{x}}} [f_{w_t}(\tilde{x})] + E_{z\sim P_z}[f_{w_c}(z)] - E_{c\sim P_c}[f_{w_c}(c)]) \quad (6)$$

4.3 LATEXT-GAN II

The LATEXT-GAN II approach (Figure 1c) is similar to the LATEXT-GAN I approach except the training of the code-based discriminator f_{w_c} is done as the ARAE training. The critic f_{w_c} is used to discriminate the synthetic code $\hat{c}$ with the latent code c. Here, the synthetic code is formed by using the ARAE method (Spinks and Moens, 2018). For each iteration in the model training, the AE training step and the Equation 3 step of the Soft-GAN in section 4.1 are carried out first. Then the following two steps are performed:

- Train the code-based discriminator f_{w_c} for k times and the encoder once using the loss L_{critic_c} to maximize the ability of the f_{w_c} to discriminate between c and $\hat{c}$:

$$L_{critic_c} = \min_{(w_c,\phi)}(-E_{c\sim P_c}[f_{w_c}(c)] + E_{\hat{c}\sim P_{\hat{c}}} [f_{w_c}(\hat{c})] + \lambda E_{\bar{c}\sim P_{\bar{c}}}[(||\nabla_{\bar{c}} f_{w_c}(\bar{c})||_2 - 1)^2]) \quad (7)$$

- Train the generator and the decoder neural networks once using the loss L_{gen} to fool the discriminators f_{w_t} and f_{w_c} with improving the representations $\hat{x}$ and $\hat{c}$ respectively:

$$L_{\text{gen}} = \min_{(\theta,\psi)}(-E_{\hat{x}\sim P_{\hat{x}}}[f_{w_t}(\hat{x})] + E_{\tilde{x}\sim P_{\tilde{x}}} [f_{w_t}(\tilde{x})] - E_{\hat{c}\sim P_{\hat{c}}}[f_{w_c}(\hat{c})] + E_{c\sim P_c}[f_{w_c}(c)]) \quad (8)$$

4.4 LATEXT-GAN III

In the third approach (Figure 1d), the combination of latent code c generated by ARAE (Spinks and Moens, 2018) and the soft-text output $\tilde{x}$ of an AE is used to signal the discriminator. We performed this combination by getting inspiration from an Adversarially Learned Inference (ALI) paper (Dumoulin et al., 2016) introduced for image generation. We call it as LATEXT-GAN III. Here, the discriminator $f_{w_{t+c}}$ tries to determine which combination of the samples derive from the latent code and the soft-text, ($\tilde{x}$,c), and which ($\hat{x}$, $\hat{c}$) are generated from the noise z. After the AE training step $\min_{(\phi,\psi)} ||x - \tilde{x}||^2$, the LATEXT-GAN III model is trained using the next two steps in each iteration:

- Train the discriminator $f_{w_{t+c}}$ for k times, the encoder and the decoder once using the loss $L_{critic_{t+c}}$ to maximize the ability of the discriminator network to discriminate between ($\tilde{x}$,c) and ($\hat{x}$, $\hat{c}$):

$$L_{critic_{t+c}} = \min_{(w_{t+c},\phi,\psi)}(-E_{(\tilde{x},c)\sim P_{\tilde{x}},P_c} [f_{w_{t+c}}(\tilde{x},c)] + E_{(\hat{x},\hat{c})\sim P_{\hat{x}},P_{\hat{c}}}[f_{w_{t+c}}(\hat{x},\hat{c})]+ \lambda E_{(\bar{x},\bar{c})\sim P_{\bar{x}},P_{\bar{c}}}[(||\nabla_{(\bar{x},\bar{c})} f_{w_{t+c}}(\bar{x},\bar{c})||_2 - 1)^2]) \quad (9)$$

- Train the generator and the decoder once based on L_{gen} to fool the discriminator $f_{w_{t+c}}$ with improving the representation ($\hat{x}$, $\hat{c}$):

$$L_{\text{gen}} = \min_{(\theta,\psi)}(-E_{(\hat{x},\hat{c})\sim P_{\hat{x}},P_{\hat{c}}}[f_{w_{t+c}}(\hat{x},\hat{c})] + E_{(\tilde{x},c)\sim P_{\tilde{x}},P_c}[f_{w_{t+c}}(\tilde{x},c)]) \quad (10)$$

5 Experiments

5.1 Dataset and Experimental Procedures

We do our experiments on two different datasets: the Stanford Natural Language Inference (SNLI) corpus [1], which contains 714,667 sentences for

[1] https://github.com/aboev/arae-tf/tree/master/data_snli

training and 13323 sentences for testing, and the Image COCO [2] dataset's image caption annotations, where we sample [3] 10,000 sentences as training set and another 10,000 as test set (Zhu et al., 2018). We perform word-based experiments. For the SNLI dataset, we use a vocabulary size of 10000 words and use the maximum sentence length of size 15. For the COCO dataset, we use a vocabulary size of 5000 and perform experiments using the maximum sentence length of sizes 15 and 20. We train a simple AE using one layer with 512 LSTM cells (Hochreiter and Schmidhuber, 1997) for both the encoder and the decoder. For decoding, the output from the previous time step is used as the input to the next time step. We use the hidden code c from the last time step of the encoder and applied as an additional input at each time step of decoding. We normalize the code and then added an exponentially decaying noise before decoding. The greedy search approach is applied to get the best output. We train the auto-encoder using Adam (Diederik and Jimmy, 2014) optimizer with learning rate = 0.001, β_1= 0.9, and β_2= 0.999. We use CNN-based generator and discriminator with residual blocks (Gulrajani et al., 2017). The $tanh$ function is applied on the output of the ARAE generator (Kim et al., 2017). We train the generator and the discriminator using Adam optimizer with learning rate = 0.0001, β_1= 0.5, and β_2= 0.9. We do not apply any kind of attention mechanisms (Bahdanau et al., 2014; Zhang et al., 2018) and pre-training (Zhu et al., 2018) in our experiments. We use the WGAN-GP (Gulrajani et al., 2017) approach with 5 discriminator updates for every generator update and a gradient penalty co-efficient of λ=10 unlike a setup in (Zhu et al., 2018). For the AAE-based experiments, we normalize the data drawn from a prior distribution z. We train the models for 200000 iterations where in each iteration we sample a random batch and train the networks of the models.

5.2 Quantitative Evaluation

We use the frequently used BLEU metric (Papineni et al., 2002) to evaluate the word similarity between sentences and the perplexity to evaluate our techniques. We calculate BLEU-n scores for n-grams without a brevity penalty (Zhu et al., 2018). The results with the best BLEU-n scores in the synthesized generated texts are reported. To calculate the BLEU-n scores, we generate ten batches of sentences as candidate texts, i.e. 640 sentences and use the entire test set as reference texts. As the GAN-based models usually suffer from mode collapse (i.e., generating same samples over and over again), evaluating models by only BLEU metric is not appropriate. So, we also calculate recently proposed self-BLEU scores for the COCO dataset using maximum sentence length of size 20 and 10k synthetic sentences to evaluate the diversity (Zhu et al., 2018). Using one synthetic sentence as hypothesis and others as reference, the BLEU is calculated for every synthetic sentence, and define the average BLEU score as the self-BLEU (Zhu et al., 2018). A higher self-BLEU score describe less diversity. For the perplexity evaluations, we generate 100k and 10k sentences for the SNLI and the COCO datasets respectively using the models of the last iteration.

5.2.1 BLEU-n Scores evaluation

The BLEU score results for the n-grams of the synthesized texts are depicted in Table 2 and 3 with maximum sentence length of 15 for the SNLI and the COCO datasets respectively. We also report experimental results with a longer maximum sentence length of 20 using the COCO dataset to differentiate the effectiveness of code and text-based solutions (in Table 4). Furthermore, we report the BLEU and self-BLEU score results of our proposed approaches in Table 5 and 6 respectively for the COCO dataset to compare with the results of the existing approaches reported in (Zhu et al., 2018).

Model	B-2	B-3	B-4	B-5
AAE	0.797	0.614	0.449	0.294
ARAE	0.73	0.575	0.431	0.297
IWGAN	0.70	0.518	0.369	0.246
Soft-GAN	0.849	0.648	0.446	0.252
LATEXT-GAN I	**0.87**	**0.679**	**0.508**	0.336
LATEXT-GAN II	0.793	0.631	0.48	**0.338**
LATEXT-GAN III	0.782	0.617	0.466	0.33

Table 2: BLEU-n (B-n) scores results using SNLI dataset and 640 synthetic sentences

From tables 2, 3, and 4, we can see that our proposed approaches outperform the standalone code (AAE or ARAE) and text-based (IWGAN) solutions. For the maximum sentence length of size 15 experiments, the LATEXT-GAN I is better than LATEXT-GAN II and III for shorter length text (e.g., 2,3-grams). The performance of the LATEXT-GAN II and III degrades with increasing maximum sentence length to 20. This is be-

[2] http://cocodataset.org

[3] https://github.com/geek-ai/Texygen/tree/master/data

Model	B-2	B-3	B-4	B-5
AAE	0.733	0.477	0.284	0.156
ARAE	0.676	.457	0.287	0.172
IWGAN	0.636	0.417	0.258	0.155
Soft-GAN	**0.781**	0.492	0.296	0.155
LATEXT-GAN I	0.758	**0.496**	0.294	0.155
LATEXT-GAN II	0.707	0.489	**0.316**	**0.198**
LATEXT-GAN III	0.701	0.483	0.311	0.193

Table 3: BLEU-n (B-n) scores results for COCO dataset with maximum sentence length of size 15 and 640 synthetic sentences

Model	B-2	B-3	B-4	B-5
AAE	0.751	0.475	0.287	0.167
ARAE	0.665	0.447	0.279	0.162
IWGAN	0.669	0.454	0.294	0.178
Soft-GAN	**0.799**	**0.520**	**0.317**	**0.190**
LATEXT-GAN I	0.77	0.503	0.314	0.185
LATEXT-GAN II	0.687	0.456	0.283	0.174
LATEXT-GAN III	0.680	0.466	0.292	0.178

Table 4: BLEU-n (B-n) scores results for COCO dataset with maximum sentence length of size 20 and over 640 synthetic sentences

cause for longer sequence length experiments, the hidden code of the last time step might not be able to keep all the information from the earlier time steps. On the other hand, the LATEXT-GAN I and the Soft-GAN improve their performance with increasing maximum sentence length to 20. This might be because of the encoder enhances its representation better to the prior distribution, z from which the text is generated. Furthermore, the Soft-GAN outperforms all the proposed LATEXT GAN approaches.

We also compare our proposed approaches with TextGAN (Zhang et al., 2017), some RL-based approaches (SeqGAN (Yu et al., 2017), RankGAN (Lin et al., 2017), MaliGAN (Che et al., 2017)) and MLE approach described in a benchmark platform (Zhu et al., 2018) where they apply pre-training before applying adversarial training. We evaluate the BLEU and Self-BLEU score results on 10k synthetic sentences using the maximum sentence length of size 20 for the COCO dataset with a vocabulary of size 5000 as in (Zhu et al., 2018). The BLEU and the self-BLEU score results are reported in Table 5 and 6 respectively. From Table 5, it can be noted that our proposed approaches show comparable results to the RL-based solutions for the BLEU score results. We can also see that our proposed LATEXT-GAN III approach gives lower self-BLEU scores in Table 6. From the above experimental results, we can note that LATEXT-GAN III can generate real-like and more diverse sentences compare to some approaches reported in (Zhu et al., 2018) and our other proposed approaches.

Model	B-2	B-3	B-4	B-5
TextGAN	0.593	0.463	0.277	0.207
SeqGAN	0.745	0.498	0.294	0.180
RankGAN	0.743	0.467	0.264	0.156
MaliGAN	0.673	0.432	0.257	0.159
MLE	0.731	0.497	0.305	0.189
Soft-GAN	0.787	0.496	0.286	0.150
LATEXT-GAN I	0.736	0.447	0.258	0.146
LATEXT-GAN II	0.672	0.430	0.257	0.147
LATEXT-GAN III	0.660	0.435	0.260	0.149

Table 5: *BLEU-n (B-n)* scores results for COCO dataset using sequence length 20 and 10k generated sentences

Model	B-2	B-3	B-4	B-5
TextGAN	0.942	0.931	0.804	0.746
SeqGAN	0.950	0.840	0.670	0.489
RankGAN	0.959	0.882	0.762	0.618
MaliGAN	0.918	0.781	0.606	0.437
MLE	0.916	0.769	0.583	0.408
Soft-GAN	0.988	0.950	0.847	0.612
LATEXT-GAN I	0.991	0.957	0.854	0.613
LATEXT-GAN II	0.896	0.755	0.523	0.263
LATEXT-GAN III	**0.874**	**0.706**	**0.447**	**0.205**

Table 6: Self-BLEU scores results for COCO dataset using sequence length 20 and 10k generated sentences

5.2.2 Perplexity Evaluation

The forward and reverse perplexities of the LMs trained with maximum sentence length of 15 and 20 using the SNLI and the COCO datasets respectively are described in Table 7. The forward perplexities (F-PPL) are calculated by training an RNN language model (Zaremba et al., 2015) on real training data and evaluated on the synthetic samples. This measure describe the fluency of the synthetic samples. We also calculate the reverse perplexities (R-PPL) by training an RNNLM on the synthetic samples and evaluated on the real test data. We can easily compare the performance of the LMs by using the forward perplexities while it is not possible by using the reverse perplexities as the models are trained using the synthetic samples with different vocabulary sizes. The perplexities of the LMs using real data are 16.01 and 67.05 for the SNLI and the COCO datasets respectively reported in F-PPL column. From the tables, we can note the models with lower forward perplexities (higher fluency) for the synthetic samples tend to have higher reverse perplexities except the AAE-based models (Cifka et al.,

2018) and/or the IWGAN. The forward perplexity for the IWGAN is the worst which means that the synthetic sentences of the IWGAN model are not fluent or real-like sentences. For the SNLI dataset, we can note that the LATEXT-GAN II and III approaches can generate more fluent sentences than the other approaches. For the COCO dataset, it can be seen that the forward perplexity of the LATEXT-GAN I (51.39) is far lower than the real data (67.05) which means the model suffers from mode-collapse. The Soft-GAN, the LATEXT-GAN II and III approaches suffer less from the mode-collapse.

Models	F-PPL SNLI	R-PPL SNLI	F-PPL COCO	R-PPL COCO
Real	16.01	-	67.05	-
AAE	74.56	48.04	83.34	257.21
ARAE	66.70	315.58	64.63	185.20
IWGAN	193.30	53.96	94.95	80.99
Soft-GAN	110.95	142.13	61.42	170.17
LATEXT-GAN I	86.24	56.78	**51.39**	158.21
LATEXT-GAN II	**53.22**	144.80	63.32	203.39
LATEXT-GAN III	54.80	143.42	71.60	214.76

Table 7: Forward (F) and Reverse (R) perplexity (PPL) results for the SNLI and COCO datasets using synthetic sentences of maximum length 15 and 20 respectively.

5.3 Human Evaluation

The subjective judgments of the synthetic sentences of the models trained using the COCO dataset with maximum sentence length of size 20 is reported in Table 8. We used 20 different random synthetic sentences generated by using the last iteration of each model and gave them to a group of 5 people. We asked them to rate the sentences based on a 5-point Likert scale according to their fluency. The raters are asked to score 1 which corresponds to gibberish, 3 corresponds to understandable but ungrammatical, and 5 correspond to naturally constructed and understandable sentences (Cifka et al., 2018). From Table 8, we can note that the proposed LATEXT-GAN III approach get the higher rate compare to the other approaches. From all the above different evaluations, we can note that the synthetic sentences by using the LATEXT-GAN II and III approaches are more balanced in diversity and fluency compare to the other approaches. We also depicted some examples of the synthetic sentences for the COCO and the SNLI datasets in Table 9 and 10 respectively.

Model	Fluency
Real	4.32
AAE	1.72
ARAE	2.60
IWGAN	1.58
Soft-GAN	1.66
LATEXT-GAN I	1.82
LATEXT-GAN II	2.48
LATEXT-GAN III	**2.70**

Table 8: Human Evaluation on the synthetic sentences

6 Conclusion and Future Work

In this paper, we introduced Soft-GAN as a new solution for the main bottleneck of using GAN for generating text, which is the discontinuity of text. This is based on applying soft-text to the GAN discriminator instead of the one-hot representation in the traditional approach. We also introduced three LATEXT-GAN approaches by combining the reconstructed output (soft-text) of an auto-encoder to the latent code-based GAN approaches (AAE and ARAE) for text generation. LATEXT-GAN I is formed on top of AAE method. LATEXT-GAN II and III approaches were formed based on ARAE. The LATEXT-GAN I and II approaches used separate discriminators for the synthetic text and the synthetic code discriminations. The LATEXT-GAN III used the combination of the soft-text and the latent code to compare with the concatenation of the synthetic text and the synthetic code by using a single discriminator. We evaluated the proposed approaches over the SNLI and the COCO datasets using subjective and objective evaluation metrics. The results of the experiments are consistent with different evaluation metrics. We showed the superiority of our proposed techniques, especially the LATEXT-GAN III method over other conventional GAN-based techniques which does not require pre-training. Finally, we summarize our plan for future work in the following:

1. We trained the GAN using WGAN-GP approach. Spectral normalization technique (Miyato et al., 2018) can be applied to stabilize the GAN training which could generate more diverse text in our settings.

2. The proposed approaches are the pure GAN-based techniques for text generation and they are not very powerful in generating long sentences. RL or self-attention (Zhang et al., 2018) techniques can be used as a tool to ac-

ARAE	IWGAN	Soft-GAN
a motorcycle parked outside of a counter street .	a small hang bathroom with a park and side .	a man is sitting with a counter counter .
a plane scene with focus on a toilet .	a group of kitchen sits near a sliding .	a dog with a people above a street .
a cat standing down the of a bathroom .	a picture from a giraffe in a garlic .	a blue of cat sitting in a red wall .
a bathroom with a toilet, a wall and .	a man shorts on a large table over .	a woman is looking with a red .
a white cat sits in the kitchen bowl .	a car is hidden from a small kitchen .	a dog is sitting in a bathroom .
LATEXT-GAN I	LATEXT-GAN II	LATEXT-GAN III
a bathroom bathroom with with toilet and and mirror .	a bathroom with a sink and and white plane .	there are a park with ride on around display .
a man kitchen next to a kitchen cabinets and .	a kitchen filled with wooden sink and a large window .	a kitchen with white cabinets and a black tub .
a man standing at a kitchen in a .	a person is parked on a city road .	a bathroom has a sink and a toilet .
a man in in front a a bathroom with .	a cat sitting on a bench of a city street .	a woman riding a bike with a large on .
a plane is sitting with the sky at .	a group of people sitting on a city bench .	a picture of a kitchen with a a skies .

Table 9: Example generated sentences with models trained using COCO dataset with maximum sentence length of size 20

ARAE	IWGAN	Soft-GAN
A little girl is playing outside .	A car is sitting in a house .	A children of a blue a shirt while a .
A woman is near the water .	Someone are Indian in his trinkets .	A little girl is looking while a a .
A group of people are sitting .	A woman is outside .	A couple of people are in white a a .
The lady is in the water .	the man is in the house .	A couple in a blue white is is standing .
A lady sitting on a bench .	The kids are laughing Bull .	A man in a white shirt outside a a .
LATEXT-GAN I	LATEXT-GAN II	LATEXT-GAN III
A woman is holding a .	People are walking in the park .	A group of people are sitting in a field .
A man is playing a .	The girl is playing with a a .	A boy is outside in a field .
A man is standing on an beach .	There are motorcycles outside .	A man is riding a bike in the water .
A man is walking in .	The young boy is wearing a red shirt .	A woman is riding a bike .
A little girl is playing in the .	A woman is walking on the park .	The man is standing .

Table 10: Example generated sentences with models trained using SNLI dataset with maximum sentence length of size 15

commodate this weakness.

3. We used the hidden code from the last time step of the encoder. Attention mechanism can be applied for decoding. Furthermore, the powerful transformer (Vaswani et al., 2017) can be applied for the auto-encoder which could improve the performance of the proposed approaches. Pre-train the auto-encoder before adversarial training can also improve the performance.

Acknowledgments

We would like to thank Professor Qun Liu, Chief Scientist of Speech and Language Computing, Huawei Noah's Ark Lab for his valuable comments on the paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Mohamed Ishmael Belghazi, Sai Rajeswar, Olivier Mastropietro, Negar Rostamzadeh, Jovana Mitrovic, and Aaron Courville. 2018. Hierarchical adversarially learned inference. *arXiv preprint arXiv:1802.01071*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*, pages 41–48.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Massima Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language GANs falling short. *arXiv preprint arXiv:1811.02549*.

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

Ondrej Cifka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. 2018. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *arXiv preprint arXiv:1804.07972*.

P Kingma Diederik and Ba Jimmy. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. 2016. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. MaskGAN: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Jules Gagnon-Marchand, Hamed Sadeghi, Md. Akmal Haidar, and Mehdi Rezagholizadeh. 2019. SALSA-TEXT: Self attentive latent space based adversarial text generation. In *Canadian AI 2019*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*, pages 2672–2680.

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

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Md. Akmal Haidar and Mehdi Rezagholizadeh. 2019. TextKD-GAN: Text generation using knowledge distillation and generative adversarial networks. In *Canadian AI 2019*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation, 9(8):17351780*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*, pages 1587–1596.

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

Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*.

Diederik P Kingma and Max. Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

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

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

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

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Tom Mikolov, Martin Karafit, Luk Burget, Jan ernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *INTERSPEECH*.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

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

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

Ahmad Rashid, Alan Do-Omri, Md. Akmal Haidar, Qun Liu, and Mehdi Rezagholizadeh. 2019. Bilingual-GAN: A step towards parallel text generation. In *NeuralGen 2019*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training GANs. In *NIPS*, pages 2234–2242.

Graham Spinks and Marie-Francine Moens. 2018. Generating continuous representations of medical texts. In *NAACL-HLT*, pages 66–70.

Sandeep Subramanian, Sai Rajeswar, Alessandro Sordoni, Adam Trischler, Aaron Courville, and Christopher Pal. 2018. Towards text generation with adversarially learned neural outlines. *NeurIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Zhang Dell. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1(2):270280*.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, and Yoshua Bengio. 2014. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017a. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017b. Semi-supervised QA with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*.

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

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*.

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

Yaoming Zhu, Sidi Lu, Guo Jiaxian Zheng Lei, Zhang Weinan, Wang Jun, and Yu Yong. 2018. Texygen: A benchmarking platform for text generation models. *arXiv preprint arXiv:1802.01886*.