Generative Cooperative Networks for Natural Language Generation

Sylvain Lamprier, Thomas Scialom, Antoine Chaffin, Vincent Claveau, Ewa Kijak, Jacopo Staiano, Benjamin Piwowarski

▶ **To cite this version:**

Sylvain Lamprier, Thomas Scialom, Antoine Chaffin, Vincent Claveau, Ewa Kijak, et al.. Generative Cooperative Networks for Natural Language Generation. ICML 2022 - 39th International Conference on Machine Learning, Jul 2022, Baltimore, MD, United States. pp.11891–11905. hal-03736116

HAL Id: hal-03736116
https://hal.science/hal-03736116

Submitted on 22 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Generative Cooperative Networks for Natural Language Generation

Sylvain Lamprier [*1] **Thomas Scialom** [*1 2] **Antoine Chaffin** [3 4] **Vincent Claveau** [3 5] **Ewa Kijak** [3] **Jacopo Staiano** [2] **Benjamin Piwowarski** [1 5]

Abstract

Generative Adversarial Networks (GANs) have known a tremendous success for many continuous generation tasks, especially in the field of image generation. However, for discrete outputs such as language, optimizing GANs remains an open problem with many instabilities, as no gradient can be properly back-propagated from the discriminator output to the generator parameters. An alternative is to learn the generator network via reinforcement learning, using the discriminator signal as a reward, but such a technique suffers from moving rewards and vanishing gradient problems. Finally, it often falls short compared to direct maximum-likelihood approaches. In this paper, we introduce Generative Cooperative Networks, in which the discriminator architecture is cooperatively used along with the generation policy to output samples of realistic texts for the task at hand. We give theoretical guarantees of convergence for our approach, and study various efficient decoding schemes to empirically achieve state-of-the-art results in two main NLG tasks.

1. Introduction

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have known a tremendous success for many generation tasks. In GANs, a *discriminator* network is trained to distinguish real data from fake ones, the latter being generated via a *generator* network trained to fool the discriminator. Both networks are trained as a min-max two-player game, which is referred to as adversarial training. Under some strong assumptions, (Goodfellow et al., 2014) gives theoretical guarantees of convergence of the generator towards the distribution underlying observed training data. Empirically, in continuous domains such as for image generation, these architectures have shown impressive abilities to generate realistic – unobserved – data, and have been extensively studied.

However, for discrete domains such as Natural Language Generation (NLG), optimizing GANs remains an open-problem, as no gradient flow can be properly back-propagated from the discriminator output to the generator parameters. Language GANs require to be optimized via Reinforcement Learning (RL) methods, with rewards provided by discriminator networks (de Masson d'Autume et al., 2019). Compared to classical NLG methods, such approaches have the potential to 1) avoid the well-known exposure bias plaguing the traditional MLE (teacher-forcing) training mode (Ranzato et al., 2015) and 2) automatically discover useful metrics to optimize via RL – compared to manually designed ones (Ranzato et al., 2015; Paulus et al., 2017; Scialom et al., 2019). Still, GAN-based approaches suffer from both high variance and non-stationary reward distributions, leading to many instabilities, and therefore usually fall short compared to traditional MLE approaches (Caccia et al., 2020).

Theoretically sound attempts such as (Che et al., 2017) proposed to augment the discriminator scores with maximum-likelihood signals in order to stabilize rewards, but still suffer from high variance in practice (Caccia et al., 2020). Other attempts such as (Scialom et al., 2020a) proposed to rather pick training samples close to generative distribution modes to smooth the learning process and thus prevent abrupt changes in the reward function. However, this approach, if not employed with a carefully designed learning rate scheduler, still exhibits high instabilities when training until convergence – making it harder to adapt the learning process for new NLG tasks or datasets.

Relying on the assumption that discrimination is easier than generation, some recent approaches, such as (Deng et al., 2020), (Scialom et al., 2020b) or (Scialom et al., 2021b), have proposed to employ a cooperative decoding scheme where the discriminator network is used along with the generator to output more realistic samples. (Scialom et al., 2020b) bias the standard beam search with scores provided by the discriminator network, to favor sequences that are

[*]Equal contribution [1]ISIR - Sorbonne Université, Paris, France [2]ReciTAL, Paris, France [3]IRISA, Rennes, France [4]IMATAG, Rennes, France [5]CNRS. Correspondence to: Sylvain Lamprier <sylvain.lamprier@isir.upmc.fr>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

classified as human-like texts. (Scialom et al., 2021b) builds upon that idea, but updates the generator at each step based on sequences generated from this augmented beam search process, in an expert-iteration (Anthony et al., 2017) learning scheme. However, while such a kind of cooperative approach to produce accurate imitation learning samples is appealing, we argue in Section 2 that it may reveal particularly unstable.

To address these shortcomings, we propose to take inspiration from (Norouzi et al., 2016) which introduced *Reward-augmented Maximum Likelihood* (RML), where samples to imitate are produced from a Boltzmann distribution $q(x) \propto exp(f(x)/\tau)$, where $f(x)$ is an effectiveness metric of a sample x (e.g. the BLEU metric). Adapting this framework with more flexible and learned quality metrics, like in GANs, approaches such as (Scialom et al., 2021b) employ, at each step t of the optimization process, a metric f mainly depending on the current discriminator $D_t(x)$ for any sample x. In this paper, we propose to rather consider $f(x) = \log(p_{t-1}(x)D_t(x))$ with p_{t-1} the previous generator distribution and D_t the discriminator at step t (trained on samples from p_{t-1}), which allows us to avoid instability issues (notably due to possible catastrophic forgetting) and to present convergence guarantees under similar assumptions as those considered in (Goodfellow et al., 2014) for the continuous case. Then, we consider various efficient cooperative decoding approaches, which enable the practical optimization of such training processes, mainly based on Monte-Carlo techniques and importance sampling.

Our contribution is threefold:

- We propose a novel formulation of GANs for the discrete setting, which exhibits unpublished theoretical convergence guarantees;
- We propose practical efficient NLG training algorithms relying on these theoretical results, based on various sampling schemes and corresponding re-weightings;
- We present state-of-the-art results for two important NLG tasks: Abstractive Summarization and Question Generation.

2. Generative Cooperative Networks

Let $p_d : \mathcal{Y} \rightarrow [0\,;1]$ be a target generative distribution, and assume we have access to training samples $y \sim p_d(y)$. The goal is to propose a training algorithm that computes a sequence of distributions $p_t(y)$ converging towards $p_d(y)$. In the following, we note $p_t : \mathcal{Y} \rightarrow [0;1]$ a generator distribution obtained at iteration t of the algorithm, and $D_t : \mathcal{Y} \rightarrow [0\,;1]$ a discriminator that outputs the likelihood for an outcome $y \in \mathcal{Y}$ of having been generated from p_d rather than from p_{t-1}.

Based on those definitions, a generic training process is given in Algorithm 1, where KL stands for the Kullback-Leibler divergence and h is a composition function that outputs a sampling distribution q_t depending on distributions given as its arguments. This training process unifies many different discrete GANs (e.g., MaliGAN, SelfGAN, ColdGAN), as well as our present work, through the choice of function h applied to the current discriminator D_t and the previous generator p_{t-1}. Line 3 aims at finding the best possible discriminator D_t given distributions p_d and p_{t-1}, according to the classical objective to be maximized in GANs. Following the RML paradigm introduced by (Norouzi et al., 2016), line 4 seeks to optimize the generator distribution p_t by considering the minimization of the KL divergence $KL(q_t||p_t)$, according to a fixed behavior distribution q_t including feedback scores to be optimized (in our case, discriminator outputs). For cases where it is possible to efficiently sample from q_t, this is more efficient than considering a more classical reinforcement learning objective implying the reversed $KL(p_t||q_t)$, usually subject to high variance (e.g., via score function estimators).

Algorithm 1 *RML-GAN*

1: **Input:** a generator $p_0 \in \mathcal{G}$, a discriminator family $\mathcal{D}$.
2: **for** iteration t from 1 to T **do**
3: $\quad D_t \leftarrow \arg\max_{D \in \mathcal{D}} \left[\begin{array}{c} \mathbb{E}_{y \sim p_d(y)} [\log D(y)] + \\ \mathbb{E}_{y \sim p_{t-1}(y)} [\log(1 - D(y))] \end{array} \right]$
4: $\quad p_t \leftarrow \arg\min_{p \in \mathcal{G}} KL(q_t = h(p_{t-1}, D_t)||p)$
5: **end for**

Let us first consider a setting where $q_t \triangleq h(p_{t-1}, D_t) \propto exp(D_t)$, i.e. the sampling distribution only considers outputs from the discriminator. This corresponds to a direct application of the work from (Norouzi et al., 2016) for the GAN setting. For the sake of analysis, we consider the case where, at a given step t, the generator distribution is optimal, i.e. $p_t = p_d$ over the whole support $\mathcal{Y}$. In the next step $t+1$, the optimal D_{t+1} is equal to 0.5 for any sample from $\mathcal{Y}$. In this case, optimizing $KL(q_{t+1}||p_{t+1})$ with $q_{t+1} \propto exp(D_{t+1})$ makes the generator diverge from the optimum p_d, forgetting all information gathered until that point. This shows that the direct adaptation of GAN to discrete outputs is fundamentally unstable. While this is not exactly what is performed in approaches such as SelfGAN (Scialom et al., 2021b),[1] this extreme setting illustrates instabilities that can occur with this family of recent state-of-the-art approaches. Discrimination cannot be all you need.

[1] Since SelfGAN employs a pre-filter based on its generator to avoid complexity issues.

Therefore, we rather propose to consider a slightly different and yet much smoother optimization scheme, where both the generator and the discriminator *cooperate* to form the target distribution: $q_t \propto p_{t-1}D_t$. Such a choice for q_t allows us to prove the following theorem, which gives theoretical convergence guarantees for our collaborative training process (proof given in Appendix A.1).

Theorem 2.1. *With $q_t \propto p_{t-1}D_t$, if the generator and discriminator architectures have enough capacity, and if at each iteration of Algorithm 1 both optimization problems reach their respective optimum (i.e., $D_t(y) = \frac{p_d(y)}{p_d(y)+p_{t-1}(y)}$ for any $y \in \mathcal{Y}$ (line 3) and $KL(q_t \propto p_{t-1}D_t||p_t) = 0$ (line 4)), then, starting from p_0 such that $p_0(y) > 0$ whenever $p_d(y) > 0$, p_t converges in distribution to p_d when $t \to +\infty$.*

As for classic continuous GANs, the neural architectures used to define generator and discriminator function sets $\mathcal{G}$ and $\mathcal{D}$ in practice represent a limited family of distributions, depending of their depth and width. However, the given theorem allows us to expect reasonable behavior for sufficiently powerful architectures. The following theorem relaxes the constraint on optimal discriminator (proof in Appendix A.2).

Theorem 2.2. *With $p_t \propto p_{t-1}D_t$, and if the discriminator is sufficiently trained, i.e. we have $\log \eta = \min\left(\mathop{\mathbb{E}}_{y\sim p_d(y)}[\log(D_t(y)], \mathop{\mathbb{E}}_{y\sim p_{t-1}(y)}[\log(1-D_t(y))]\right)$, with $\eta \in]\frac{1}{2}; 1[$, then we have at each iteration of Algorithm 1: $\Delta_t \triangleq KL(p_d||p_t) - KL(p_d||p_{t-1}) \leq \log(\frac{1}{\eta} - 1) < 0$.*

In other words, it suffices that both parts of the discriminator objective exceed the random accuracy (i.e., $1/2$) in expectation to make $q_t \propto p_{t-1}D_t$ a useful target to be approximated at each step. Even with only a few gradient steps at each iteration, we can reasonably assume that the parameters space is smooth enough to guarantee the convergence of the algorithm, with almost only useful gradient steps. We also note that the better discriminator (i.e., higher η), the more useful is a move from p_{t-1} to p_t (in terms of KL).

Getting back to Algorithm 1, at line 4, optimization can be performed via gradient descent steps $\nabla_{p_t} KL(q_t||p_t)$, which can be rewritten via Importance Sampling as:

$$\nabla_{p_t} KL(q_t||p_t) = -\mathop{\mathbb{E}}_{y\sim q_t(y)\propto p_{t-1}(y)D_t(y)}[\nabla_{p_t} \log p_t(y)] \quad (1)$$

$$= -\mathop{\mathbb{E}}_{y\sim p_{t-1}(y)}\left[\frac{q_t(y)}{p_{t-1}(y)}\nabla_{p_t} \log p_t(y)\right]$$

$$= -\frac{1}{Z_t}\mathop{\mathbb{E}}_{y\sim p_{t-1}(y)}[D_t(y)\nabla_{p_t} \log p_t(y)] \quad (2)$$

with $Z_t = \sum_{y\in\mathcal{Y}} p_{t-1}(y)D_t(y)$ the partition function of q_t. Note that, to the exception of the partition score Z_t that acts as a scale at each step, the considered gradient is closely similar to what is optimized in classic discrete GANs via reinforcement learning (i.e., policy gradient optimization of p_t and the discriminator score as reward, as described for instance in (Scialom et al., 2020a)), when only one gradient update is performed at each iteration.

The effect of this scaling factor can be seen when written as an expectation, i.e. $Z_t = \mathbb{E}_{y\sim p_{t-1}(y)}[D_t(y)]$. From this, it is clear that Z_t is maximized when the generator distribution coincides with D_t, i.e. when p_{t-1} allocates best probability mass for samples judged as the most realistic by the current discriminator. In the absence of such a normalization term, classic GAN approaches need to set an arbitrary learning rate scheduling to avoid the explosion of gradient magnitude as p_t gets closer to p_d. Our approach, naturally stabilized by Z_t, does not require such a difficult tuning to ensure convergence – as verified empirically in Section 4.

3. Cooperating for NLG

Many NLG tasks (e.g., translation, summarization, question generation, etc.) imply a context as input. This section first presents the extension of Algorithm 1 to this setting, and then discusses its practical implementation and the sampling strategies that enable its efficient use in real-world settings.

3.1. Learning algorithm

Let Γ be a training set of N samples (x^i, y^i) where each $x^i \in \mathcal{X}$ is a (possibly empty) context (assumed to be sampled from a hidden condition distribution p_x) and $y^i \sim p_d(y^i|x^i)$ is the corresponding observation. Algorithm 2 gives the practical implementation of Algorithm 1 for large scale NLG tasks. It considers parametric distributions p_θ and D_ϕ, implemented as deep neural networks,[2] with respective parameters θ and ϕ. Thus, $p_\theta : \mathcal{X} \times \mathcal{Y} \to [0\,;1]$ is the generative conditional distribution, where $p_\theta(y|x) = \prod_{j=1}^{|y|} p_\theta(y_j|x, y_{0:j-1})$ with $p_\theta(y_j|x, y_{0:j-1})$ the categorical distribution for token j of sequence y over the vocabulary, given the context x and the sequence history $y_{0:j-1}$. Also, $D_\phi : \mathcal{X} \times \mathcal{Y} \to [0\,;1]$ is the conditional discriminative distribution, where $D_\phi(x, y)$ returns the probability for sequence y of having been generated from p_d rather than p_θ given the context x.

The discriminator is trained at line 5 of Algorithm 2, on a batch of m samples of contexts, associated with corresponding sequences y from the training set and generated sequences $\hat{y}$ from the current generator. Consistently with (Scialom et al., 2020b), to effectively drive the cooperative decoding process in guided sampling strategies $\hat{q}$ (see below), the discriminator is trained, using a classical left-

[2] Transformer T5 (Raffel et al., 2019) in our experiments

to-right mask, on every possible starting sub-sequence $y_{1:j}$ in each sample y (i.e., taken from its start token to its j-th token), with $j \leq l$ and l standing for the max length for any decoded sequence. This enables discriminator predictions for unfinished sequences (allowing to avoid complex rollouts in MCTS, see below).

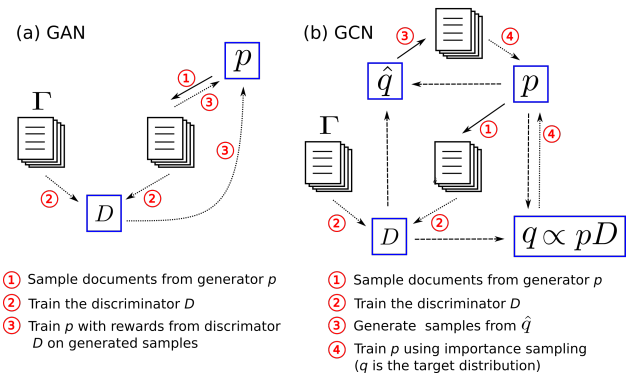

Figure 1. Training GANs vs GCNs. Solid arrows stand for sampling, dashed arrows depict dependence between distributions and dotted ones denote training. (a) Classical discrete GAN where: 1) the generator p samples sequences; 2) the discriminator D is updated according to these generated sequences and those from the training set Γ; 3) the scores given by the discriminator D are used as reward in a policy gradient update of p. (b) Our GCN approach where $\hat{q}$ and q respectively stand as the behavior and target distributions, which are defined by a cooperative scheme between D and p. After sampling from $\hat{q}$ in 3), q is used in an importance sampling weight for the update of p in 4), which corresponds to the minimization of $KL(q||p)$. The distribution q is set as $q(x) \propto p(x)D(x)$ to ensure convergence, and $\hat{q}$ can take various forms, the closer to q the lower the variance.

Line 7 of Algorithm 2 performs a gradient descent step for the generator, according to samples provided by a sampling strategy $\hat{q}$. Ideally, consistently with Eq.(1), training samples should be provided by $q_{\theta,\phi}(y|x) \propto p_\theta(y|x)D_\phi(y|x)$. However, directly sampling from this distribution is intractable. Various sampling strategies can be considered, using a weighted importance sampling scheme to unbias gradient estimators in line 7. For the task of unconditional generation (i.e., empty contexts x) and the case where $\hat{q} = p_\theta$, we can show that this is equivalent, up to a constant factor, to the gradient estimator given in Eq. (2), with expectations estimated on the current batch, since in that case w^i reduces to $D_\phi(x^i, \hat{y}^i)$. However, more efficient sampling strategies $\hat{q}$ can be employed, as discussed in the following. The process is illustrated by Figure 1, which compares the architectures of classical GANs with our GCN approach.

3.2. Efficient Sampling

To minimize the variance of gradient estimators, we need to sample sequences following a distribution as close as possible to $q_{\theta,\phi}(y|x) \propto p_\theta(y|x)D_\phi(x,y)$. While directly sampling from such a non-parametric distribution is difficult, and given that rejection-sampling or MCMC methods are very likely to be particularly inefficient in the huge associated support domain, it is possible to build on recent advances in guided decoding for providing methods for sampling informative sequences (Scialom et al., 2020b; 2021b), that are both likely for the generator p_θ, and realistic for the discriminator D_ϕ. Note that an alternative would have been to exploit the maximum entropy principle (Ziebart, 2010) to learn a neural sampling distribution $\hat{q}_\gamma$ as $\arg\max_{\hat{q}_\gamma} \mathbb{E}_{y\sim\hat{q}_\gamma(y|x)}[\log p_\theta(y|x) + \log D_\phi(x,y)] + \mathcal{H}_{\hat{q}_\gamma(.|x)}$, with $\mathcal{H}_q$ the entropy of distribution q. This would however imply a difficult learning problem at each iteration of Algorithm 2, and a sampling distribution $\hat{q}_\gamma$ that lags far behind $q_{\theta,\phi}$ if only few optimization steps are performed.

Algorithm 2 *Generative Cooperative Networks*

1: **Input:** generator p_θ with parameters θ, discriminator D_ϕ with parameters ϕ, training set Γ, sampling strategy $\hat{q}$, batch size m, max sequence length l.
2: **for** $t = 1, \ldots, T$ **do**
3: Sample $\{(x^i, y^i)\}_{i=1}^m$ from Γ
4: $\forall i \in [\![1\,;m]\!]$: Sample $\hat{y}^i \sim p_\theta(\hat{y}^i|x^i)$;
5: $\phi \leftarrow \phi + \epsilon_\phi \sum_{i=1}^{m} \sum_{j=1}^{l} \begin{bmatrix} \nabla_\phi \log D_\phi(x^i, y^i_{0:j-1})] + \\ \nabla_\phi \log(1 - D_\phi(x^i, \hat{y}^i_{0:j-1})) \end{bmatrix}$
6: $\forall i \in [\![1\,;m]\!]$: Sample $\hat{y}^i \sim \hat{q}(\hat{y}^i|x^i)$;
7: $\theta \leftarrow \theta + \epsilon_\theta \left[\frac{1}{\sum_{i=1}^m w^i} \sum_{i=1}^m w^i \nabla_\theta \log p_\theta(\hat{y}^i|x^i)\right]$ with $w^i = \frac{p_\theta(\hat{y}^i|x^i) D_\phi(x^i,\hat{y}^i)}{\hat{q}(\hat{y}^i|x^i)}$
8: **end for**

3.2.1. Sampling Mixtures

Before presenting our cooperative decoding strategy, we consider the use of variance reduction techniques when sampling from the generator distribution, which can be long-tailed, thus leading to unreliable sequence samples. In particular, Nucleus Sampling (Holtzman et al., 2019) has been shown to produce higher quality texts than more classic sampling strategies, including beam search and low temperature-based sampling (Scialom et al., 2020a). Its principle is to sample tokens at each decoding step only from the nucleus $V_p^{(\sigma)}$ of the considered generative distribution p, containing a specified amount σ of the probability mass. More precisely, let $V_p^{(\sigma)}$ be the minimal set of tokens from the vocabulary $\mathcal{V}$ whose total probability mass is greater than or equal to σ (i.e., $V_p^{(\sigma)} = \arg\min_{V\subseteq\mathcal{V}, \sum_{w\in V} p(w)\geq\sigma} |V|$). We denote in the following $p^{nucleus=\sigma}$ the truncation of distribution p on the set of tokens $V_p^{(\sigma)}$.

Using this technique for defining $\hat{q}$ in our Algorithm 2 could

allow to avoid usual text degeneration issues (Holtzman et al., 2019), which would benefit to our generative learning process by providing better formed sequences to the discriminator. However, Importance Sampling (IS) demands that $\hat{q}(y) > 0$ for any $y \in \mathcal{Y}$ such that $q(y) > 0$. A direct use of nucleus sampling as $\hat{q} = p^{nucleus=\sigma}$, or even more a classic beam search, cannot guarantee this property, which might involve ignoring many useful parts of $\mathcal{Y}$ in the gradient estimation, hence implying biases.

To cope with this, we propose to follow (Scialom et al., 2020a), which considers sampling distributions $\hat{q}$ as mixtures, ensuring that both properties, i.e. IS consistency and high quality samples, are verified. Formally, we use

$$\hat{q}_\theta(y|x) = \epsilon p_\theta(y|x) + (1-\epsilon) p_\theta^{nucleus=\sigma}(y|x) \quad (3)$$

where ϵ stands for a small probability for sampling from the true generator distribution rather than using a nucleus decoding ($\epsilon = 0.1$ and $\sigma = 0.1$ in our experiments), thus ensuring the validity of our IS estimator. Please also note that, using such mixture trick, each IS weight is upper-bounded by $D_\phi(y|x)/\epsilon$, which greatly limits gradient explosion issues usually associated with the use of IS in RL (or overweighting of unlikely sequences in weighted IS).

3.2.2. Guided Sampling

Next, we propose to consider cooperative decoding strategies to get a sampling distribution closer to $q_{\theta,\phi}$. More specifically, we propose to employ a Monte Carlo Tree Search strategy (MCTS), as recently considered for NLG in (Scialom et al., 2021b; Leblond et al., 2021; Chaffin et al., 2021). Using left-to-right decoding strategies, it can happen that all sequence candidates are judged as unrealistic by the discriminator, avoiding any useful learning signal for the generator. MCTS allows to deal with this strong limitation of myopic decoding, by anticipating the final utility of the successive decisions. In MCTS, a tree is built throughout decoding by repeating the four following steps: selection, expansion, evaluation, and back-propagation.

Step 1: Selection corresponds to following a path in a tree of already explored decisions for future tokens, from its root located at the current state of the sequence to be decoded, to a leaf s of the tree, for which a value $V(s)$ has not been set yet. At each node s of the tree, the child node s' is selected following the PUCT algorithm (Rosin, 2011; Silver et al., 2017):

$$s' = \underset{\hat{s} \in child(s)}{\arg\max} \left(V(\hat{s}) + c_{puct} p_\theta(\hat{s} \mid s) \sqrt{\frac{N(s)}{1+N(\hat{s})}} \right)$$

where $p_\theta(\hat{s} \mid s)$ corresponds to the conditional probability of sampling the next token to form sequence of $\hat{s}$ from the sequence corresponding to node s, according to the current generator probability. For children nodes $\hat{s}$ that have never been selected yet, their value $V(\hat{s})$ equals 0. c_{puct} is an hyper-parameter that controls the exploitation/exploration trade-off of the selection process, with $N(s)$ standing for the number of times node s has been selected in simulations.

Step 2: Expansion corresponds to the creation of child nodes for the identified leaf s, if s is not terminal (end-of-sentence token). This is done in our case by restricting to tokens from the nucleus $V_{p_\theta}^{(\sigma)}$ of p_θ, as presented above. This allows to restrict the width of the tree to the most likely tokens, hence improving efficiency.

Step 3: Evaluation of the selected leaf s is usually done in MCTS via a direct sampling (rollout) from s to a terminal node. In our case, this is likely to imply a high variance. We thus replace rollouts by the evaluation of the corresponding unfinished sequence (i.e., $V(s) \leftarrow D_\phi(s)$).

Step 4: Back-propagation consists in updating values of parent nodes of s, to favor most promising nodes in the following selection steps of the process. Consistently with (Scialom et al., 2021b), the value of each parent node $\tilde{s}$ of s is updated as the maximal score back-propagated to $\tilde{s}$: $V(\tilde{s}) \leftarrow \max(V(\tilde{s}), D_\phi(s))$. This led to better results than using the more classic average score from children.

At the end of the N rounds of these four steps ($N = 50$ in our experiments) from a given root r, the next token n is selected as the root's child that was the most visited (i.e., $\arg\max_{s \in child(r)} N(s)$). Note that, for unconditional text generation, where no context x is given to the decoder, we rather sample a child proportionally to its number of visits to maintain enough diversity during learning. This process is repeated using n as the new root until reaching a terminal token or the maximum sequence length (512 in our experiments).

Cooperative Learning with MCTS To use this MTCS process to guide the generator decoding toward sequences of high discriminator scores, in our learning Algorithm 2, we re-use the same mixture trick as for Nucleus Sampling discussed above:

$$\hat{q}_\theta(y|x) = \epsilon p_\theta(y|x) + (1-\epsilon) p_\theta^{mcts}(y|x) \quad (4)$$

where $p_\theta^{mcts}(y|x)$ is a Dirac centered on the decoded sequence from the MCTS process in the conditional case (when contexts x are available), and the MCTS sampling distribution (according to number of visits, as described in the MCTS decoding process) in the unconditional case. Again, $\hat{q}_\theta(y|x) > 0$ whenever $y \in \mathcal{Y}$ such that $q_{\theta,\phi}(y|x) > 0$, and the IS weights are upper-bounded by $D_\phi(y|x)/\epsilon$.

4. Experiments

4.1. Experimental Setting

To evaluate the framework, we experiment on standard complementary unconditional and conditional NLG tasks, with the following datasets:

Unconditional NLG – Following the same setup as in many related studies (e.g. (Scialom et al., 2020a; Caccia et al., 2020)), we first compare our approaches with NLG baselines on the task of unconditional text generation, where the aim is to reproduce a given unknown generative distribution of texts from samples, on the EMNLP2017 News dataset.

Question Generation – The task consists in generating the question corresponding to a given text and answer. For this task, we use the SQuAD dataset (Rajpurkar et al., 2016), composed of 100K triplets of Wikipedia paragraphs, factual questions, and their answers.

Abstractive Summarization – The aim of this standard sequence-to-sequence task is to produce an abstract given an input text. We use the CNN/Daily Mail dataset (CNNDM) (Nallapati et al., 2016), composed of 300K news article/summaries pairs. Target summaries consist of of multiple sentences, allowing us to evaluate models on longer texts than for the Question Generation task.

To compare the models, we consider the standard BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) metrics. They both are defined as an overlap ratio between n-grams from the generated text and the ground truth. BLEU is precision oriented, while ROUGE is recall oriented.

For the task of unconditional NLG, where diversity is of crucial importance, we follow (Caccia et al., 2020), who proposed to plot results as curves of BLEU (i.e., with samples classically compared to ground truth references, measuring accuracy) vs. self-BLEU (i.e., with generated samples compared to themselves, measuring diversity). This is done by sampling texts for various temperature settings (i.e. temperature of the softmax on top of the generator).

We compare our models with the following baselines:

MLE – We naturally consider as an important baseline the T5 model trained via Teacher Forcing. It is furthermore used as a starting point for *all* models (unless specified).

ColdGAN – This model was one of the first GANs to outperform MLE for NLG tasks (Scialom et al., 2020a). Its main contribution was to introduce the use of a sampling strategy with lowered softmax temperature during training, with the objective of stabilizing the training process. We use its best reported version, which considers a mixture with Nucleus Sampling.

SelfGAN – The work presented in (Scialom et al., 2021b) uses an expert-iteration algorithm in combination with various different cooperative decoding strategies. In the following, we report results from its version using a MCTS process, which recently obtained state-of-the-art results on the three considered NLG tasks.

GCN – Our Generative Cooperative Networks which we introduce in this paper. Three versions of Algorithm 2 are considered in the experiments: GCN$^{\hat{q}=p}$, which corresponds to a classic GAN with implicit dynamic scheduler induced by partition $z_t = \sum_i w^i$, GCN$^{\hat{q}=Nucleus}$, which considers a mixture with Nucleus Sampling as defined in Eq. (3), and GCN$^{\hat{q}=MCTS}$, which considers a mixture with a discriminator-guided MCTS, as defined by Eq. (4).

GAN – For ablation study purposes, we also consider similar versions of our implementation of Algorithm 2 but without the use of a normalization, respectively called GAN$^{\hat{q}=p}$, GAN$^{\hat{q}=Nucleus}$ and GAN$^{\hat{q}=MCTS}$. The normalization is replaced by a linear learning rate scheduler tuned on a validation set for GAN$^{\hat{q}=p}_{+scheduler}$, GAN$^{\hat{q}=Nucleus}_{+scheduler}$ and GAN$^{\hat{q}=MCTS}_{+scheduler}$.

For each model, any decoding method could be applied at inference time, independently of the training scheme. In the following, unless specified otherwise, we report results obtained with a classic Beam Search decoding (with a beam size of 3) for all the experiments.

In all our experiments, our models are initialized with the seq2seq T5 model (Raffel et al., 2019), trained via Teacher Forcing. Unless specified otherwise, we use the T5-small architecture (60M parameters), as implemented in the HuggingFace library (Wolf et al., 2019). For our best setup, we also report the results using T5-large (3 billion parameters), denoted as T5-3B. Using 4 Nvidia V100 SXM2 GPUs, *GCN*$^{\hat{q}=MCTS}$ training took 32 hours for summarization, and 8 hours for QG. This is comparable to the state-of-the-art SelfGAN model. *GCN*$^{\hat{q}=Nucleus}$ only required 8 hours for training on summarization, and 2 hours for QG.

4.2. Results and Discussion

Unconditional Text Generation Figure 2 reports results for the unconditional NLG task. First, we observe the crucial importance of the scheduler for the GAN baselines: all of its versions without scheduler (and any normalization as in vanilla discrete GANs) strongly diverge since the first training epoch, obtaining significantly weaker results than MLE (which is the starting point of all curves from the left graph). However, we see that our GCNs are naturally implicitly scheduled, with results comparable to the scheduled version of GANs, thanks to its self-normalized IS. This is an important result, since tuning the rate scheduler from a validation set is tricky and resource consuming. We also note the significantly better and comparable be-

havior of GCN$^{\hat{q}=Nucleus}$ and GCN$^{\hat{q}=MCTS}$ compared to GCN$^{\hat{q}=p}$. This validates that the use of smarter sampling helps training, although the space of correct sequences is too large to fully benefit from the MCTS guided sampling. The right graph from Figure 2 plots accuracy vs diversity curves. Here again, we observe the significant impact of scheduling, which is naturally implied in our GCN approach, not only for the sample quality, but also on the coverage of the induced distribution. For completeness, the graph also reports curves for previous GAN approaches, including (Che et al., 2017; Yu et al., 2017), as given by (de Masson d'Autume et al., 2019) for the same setting. While they are not directly comparable since they do not use the same generative architecture, they all have been shown to fall short compared to their respective MLE counter-part (refer for instance to (Caccia et al., 2020), Figures 3 and 4), which is clearly not the case with our cooperative approach. Note also that SeqGAN with T5 is very similar to GAN$^{\hat{q}=p}$, using the same kind of incremental discriminator.

Conditional Text Generation More important are the results for conditional text generation, for which applications are numerous. On both considered tasks, we observe from Figure 3 the same trends as for unconditional NLG, with a dramatic divergence of classic GAN approaches. We note a significant improvement of our GCN approaches compared to their GAN scheduled counterparts on both tasks, with a clear advantage for GCN$^{\hat{q}=MCTS}$ on summarization, where the discriminator guided sampling process obtains very stable results, significantly greater than those of other considered approaches. This result confirms that using MCTS to sample during the learning process is key to produce long texts of better quality.

These trends on the BLEU metrics are confirmed by numerical results from Table 1, where GCN$^{\hat{q}=MCTS}$ obtains the best results on both tasks over three metrics, with more than 2 ROUGE-L points gained over the very recent state-of-the-art approach SelfGAN (which also uses MCTS sampling) on QG[3]. Note that these results were obtained without the complex variance reduction techniques that other RL-based GAN approaches require for obtaining results comparable to MLE, which underlines further the interest of our approach. For completeness, we also report results using MCTS for decoding at test time, denoted as GCN$^{\hat{q}=MCTS}_{decod=MCTS}$, which shows some further improvements, consistently with (Scialom et al., 2021b).

Finally, our experiment on scaling $GCN^{\hat{q}=MCTS}$ to a larger model (i.e. T5 3B instead of T5 Small) allows us to further improve the results, indicating the scaling potential for GCN, and establishing a new state-of-the-art for QG and summarization. Please note that, consistently with ColdGAN and SelfGAN, we used a beam-search of size $b = 3$ and no length penalty $\alpha = 0$ (which are set to $b = 4$ and $\alpha = 0.6$ in the original paper T5 paper (Raffel et al., 2019)). Despite these lighter decoding settings, we observe that our $GCN^{\hat{q}=MCTS}_{T5-3B}$ significantly outperforms T5 3B and 11B from the original paper, without any specific tuning.

To summarize, we observe that GCN always obtains significantly better results than its GAN counterparts, regardless of the sampling distribution used, in both considered conditional NLG tasks. Moreover, the required tuning of the training scheduler for text GANs is a very difficult task, that implies a very costly grid search process (more than 10 trials required for each setting of GANs in our experiments), which is not needed in our approach. While the approach requires however to re-sample sequences twice per iteration (cf. line 4 and 6 of Algorithm 2), this additional cost is clearly counterbalanced by the ease of deployment and the important accuracy gains. At last, this additional cost can be removed by re-using samples from line 4 at line 7, with an appropriate IS term (no significant accuracy difference in our experiments).

	QG			Summarization		
	B	R-1	R-L	B	R-1	R-L
MLE	19.7	45.2	41.1	15.9	42.3	40.4
ColdGAN	19.9	45.2	41.4	16.3	42.8	40.7
SelfGAN	20.5	46.6	42.6	17.0	42.8	41.5
GAN$^{\hat{q}=p}_{+scheduler}$	19.3	45.3	41.2	15.5	40.0	38.8
GAN$^{\hat{q}=p}$	11.2	26.3	23.9	9.8	23.3	22.5
GCN$^{\hat{q}=p}$	19.7	46.2	42.0	15.9	40.8	39.5
GAN$^{\hat{q}=Nucleus}_{+scheduler}$	20.1	47.3	43.0	16.0	41.8	40.4
GAN$^{\hat{q}=Nucleus}$	11.3	26.6	24.1	10.2	23.5	22.7
GCN$^{\hat{q}=Nucleus}$	20.9	47.7	44.5	16.6	43.2	41.8
GAN$^{\hat{q}=MCTS}_{+scheduler}$	20.4	47.9	43.5	16.4	42.2	40.9
GAN$^{\hat{q}=MCTS}$	11.7	27.5	25.0	11.7	24.3	23.4
GCN$^{\hat{q}=MCTS}$	**21.5**	**48.3**	**44.7**	**17.1**	**43.4**	**42.0**
GCN$^{\hat{q}=MCTS}_{decod=mcts}$	**21.6**	**48.7**	**45.2**	**17.6**	**43.7**	**42.3**
GCN$^{\hat{q}=MCTS}_{T5-3B}$	**21.8**	**49.8**	**45.9**	**19.2**	**44.2**	**43.8**

Table 1. Final results on QG and Summarization test sets, in terms of BLEU-4 (B), ROUGE-1 (R-1) and ROUGE-L (R-L). Scores in bold are significantly different from the best baseline (GAN$^{\hat{q}=MCTS}_{+scheduler}$) according to a 95%-Student-t-test.

5. Related Work

Under the most popular paradigm, sequence generative models (Sutskever et al., 2014) are usually trained with

[3]Note that, while curves in Figures 2 and 3 use the MLE pretrain from (Scialom et al., 2021a) as base model, we report here results obtained starting from the best performing MLE model that is used in (Scialom et al., 2021b), to obtain results comparable with this paper (results from the former MLE pre-trained model can be found in Appendix A.5).

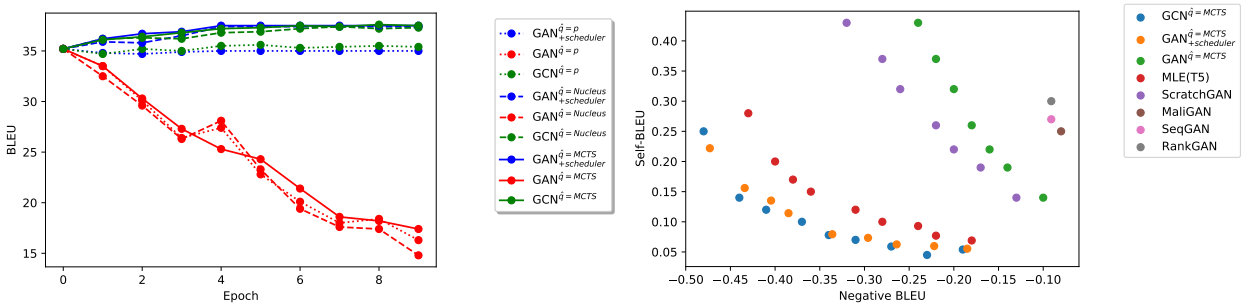

Figure 2. Results on the EMNLP 2017 dataset. Left: Evolution of BLEU results on tests sets w.r.t. training epochs (higher is better) – red: GAN without scheduler, blue: GAN with scheduler, green: GCN. Right: Curves of negative BLEU vs self BLEU (lower is better). Scores for previous studies are taken from (de Masson d'Autume et al., 2019).

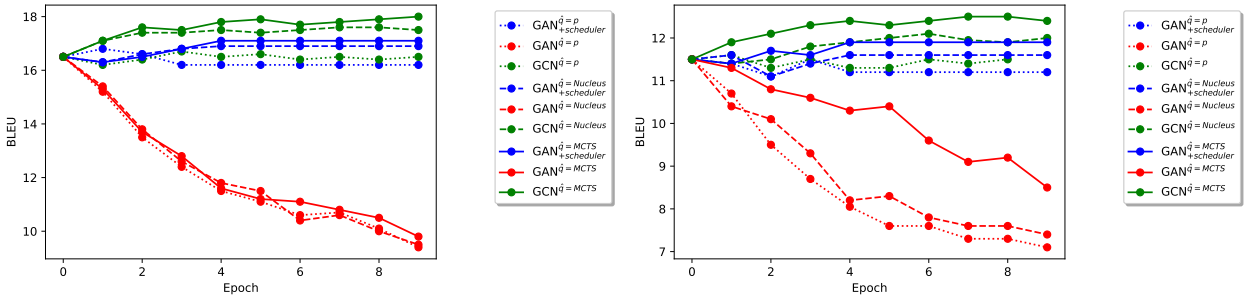

Figure 3. Evolution of performance on the test set w.r.t. training epochs (in term of BLEU, the higher the better), for conditioned NLG tasks. Left: Question Generation, Right: Summarization.

Maximum Likelihood Estimation (MLE), via Teacher Forcing (Williams & Zipser, 1989). Though MLE has lots of attractive properties, it is however prone to overfitting for auto-regressive generative models, due to a too strong exposition to the somehow limited ground-truth data. More importantly, MLE suffers from the mismatch between learning and simulation conditions, i.e. the well known exposure bias (Ranzato et al., 2015; Bengio et al., 2015). Namely, at inference, the model is conditioned on sequences of previously generated tokens which may have never been observed at training time. MLE also lacks a sequence-level loss to accurately optimize sequence probabilities (Welleck et al., 2019; Negrinho et al., 2018), resulting in often degenerated texts (e.g., prone to repetition) (Holtzman et al., 2019) .

To overcome the above shortcomings of MLE, recently, many sequential GANs for discrete outputs have been proposed in the literature (Yu et al., 2017; Guo et al., 2018), in which generators are typically trained to fool a learned discriminator via reinforcement learning (e.g., Policy Gradient such as the REINFORCE algorithm). While these methods allow to fill the learning-simulation gap, they usually suffer from high variance, partly due to the non-stationarity of their reward distribution. Until recently with some advances on smoother sampling techniques and the use of control variates (Scialom et al., 2020a), GAN approaches usually under-performed MLE training in most real-world tasks (Caccia et al., 2020), with resulting sharp distributions that often sacrifice diversity for quality. Recent works based on cooperative decoding (Scialom et al., 2021b; 2020b) opened the way for more efficient approaches, that rely on the discriminator not only as reward, but also for sampling, as we do in this work. However, these approaches exhibit instabilities, as discussed in Section 2, which we dealt with in this paper, leveraging a more theoretically sound framework. Finally, approaches such as (Zhou et al., 2020) attempted to cope with reward sparsity and mode collapse issues of text-GANs, by employing specific discriminators (e.g., comparative classifiers in (Zhou et al., 2020)), but this usually prevents the theoretic study of convergence behavior, convergence of non-Bernoulli GANs remaining an open problem.

While not arising from the same perspective, our work on GANs for discrete outputs is strongly related to the MaliGAN approach, proposed in (Che et al., 2017). Like ours, this approach relies on the work of (Norouzi et al., 2016) that unified reinforcement learning and maximum likelihood, by considering a KL divergence loss between a reward-derived distribution q and the learned distribution p_t.

It extends this framework for the GAN setting by substituting to q a distribution based on a learned discriminator, to gain in flexibility compared to hand-defined metrics con-

sidered in (Norouzi et al., 2016). However, rather than iteratively driving the learning process towards the data distribution p_d as in this paper, (Che et al., 2017) attempts to directly model it with the assumption that the discriminator is close enough to the optimum. The approach consists in defining a reward function derived from the usual property of optimal discriminators in classic GANs, namely that $D^*(y) = p_d(y)/(p_d(y) + p_{gen}(y))$, to weight sequences (with IS) according to the unknown target distribution p_d. Note that this can be seen as a specific instance of our Algorithm 1, with $h(p_{t-1}, D_t)$ defined as $p_{t-1}D_t/(1 - D_t)$.

However, the optimality of the discriminator is far from being guaranteed at each step: in (Scialom et al., 2020a), discriminators are shown to be strongly specialized for the current generator distribution, with possibly many sequences out of that distribution being greatly over-estimated . We argue that as (Che et al., 2017) intrinsically relies on this optimality, it is exposed to a high variance of its IS estimator, as acknowledged by the many variance reduction techniques the authors employed. Note that even doing so, they obtain comparable results to our simple GCN$^{\hat{q}=p}$ (using a pure sampling approach) for conditional NLG tasks.

In this paper, we only rely on the optimal discriminator property in our proof of convergence, similarly to continuous GANs (Goodfellow et al., 2014), and show that even a decent discriminator drives the convergence process in the right direction. As a further improvement over (Che et al., 2017), we experimentally show, consistently with (Scialom et al., 2021b; Deng et al., 2020; Scialom et al., 2020b), that a more sophisticated discriminator-guided sampling process is highly beneficial. At last, we note the interesting Boundary Seeking GANs proposed in (Hjelm et al., 2017), which stands as a generalization of MALIGAN for various f-measures, but whose direct application for conditioned sequential generation looks difficult.

Finally, we note that the sampling distribution q_t we introduce in this work (i.e., $q_t \propto p_{t-1}D_t$) is quite similar to the energy-based generative model considered in (Bakhtin et al., 2021), which also deals with cooperative decoding for NLG but aims to transform sequence distributions from a constant generative language model p_ϕ, using an energy function learned by noise contrastive estimation (Ma & Collins, 2018). In theory, the resulting model should match the target data distribution p_d, but relies on the strong assumption that the base language model is accurate enough in the domain of p_d, residual learning always carrying strong liability to its base model. Moreover, the negative sampling considered is performed independently from the learned distribution, which might be particularly inefficient for long tailed distributions, with a strong divergence from the target p_d. Our work suggests that using cooperative sampling could be valuable in such a setting.

6. Discussion

The work presented in this paper sheds new light on discrete GAN approaches, and in particular on theoretically-sound approaches such as MaliGAN (Che et al., 2017). We give a new perspective for this approach, and introduce a slightly modified algorithm, with strong theoretical guarantees, which can be combined with cooperative sampling strategies to obtain state-of-the-art results on various NLG tasks, and focused on GAN-like approaches, based on a learned discriminator to drive the generator.

This work can also be seen as an unification of SelfGAN (Scialom et al., 2021b) and ColdGAN (Scialom et al., 2020a) in a same theoretical framework. ColdGAN proposed to consider mixtures of behavioral policies, allowing the use of specific - e.g. deterministic - decoding strategies while ensuring that importance sampling holds. SelfGAN proposed to employ an expert-iteration learning scheme, that uses discriminator scores to guide the expert decoding strategy (e.g., MCTS). We borrow both of these ideas that we combine to leverage their benefits (in terms of sample efficiency), while alleviating their drawbacks regarding instability issues, in a theoretically well-sounded framework. The key differences with these approaches are: 1) the use of Reward-augmented MLE rather than Expert-Iteration, that allows avoiding the biased gradient estimation of SelfGAN while keeping its sample efficiency; 2) the introduction of a target distribution, based on discriminator scores, which ensures theoretical convergence guarantees, which were impossible to obtain for both previous approaches. Our approach leverages MCTS to drive the sampling distribution with a well-adapted normalization term. We claim that this is a very important contribution with strong implications on the stability of learning.

Now, it would be interesting to study how our cooperative mechanisms could apply in the context of approaches based on density ratio estimators, such as promising ones proposed in (Lu et al., 2019; Song et al., 2020). Hybrid approaches, based on ratio estimators between current densities and expected ones, that can be derived from our theoretical results in optimal GAN conditions, also constitute a promising research perspective, for measuring model drift and discovering new regularization objectives. We believe our work paves the way for new formulations of GANs for discrete settings. Notably, our assumption in Theorem 2.2 suggests possibly effective modifications for the discriminator loss, to gain in learning stability.

7. Acknowledgments

This work was partly funded by the ANR-21-CE23-0007 ACDC project. Experiments were performed using HPC resources from GENCI-IDRIS (Grant 2021-AD011012318).

References

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. *arXiv preprint arXiv:1705.08439*, 2017.

Bakhtin, A., Deng, Y., Gross, S., Ott, M., Ranzato, M., and Szlam, A. Residual energy-based models for text. *J. Mach. Learn. Res.*, 22:40–1, 2021.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., and Charlin, L. Language gans falling short. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJgza6VtPB.

Chaffin, A., Claveau, V., and Kijak, E. PPL-MCTS: Constrained Textual Generation Through Discriminator-Guided Decoding. *CoRR*, abs/2109.13582, 2021. URL https://arxiv.org/abs/2109.13582.

Che, T., Li, Y., Zhang, R., Hjelm, R. D., Li, W., Song, Y., and Bengio, Y. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

de Masson d'Autume, C., Mohamed, S., Rosca, M., and Rae, J. Training language gans from scratch. In *Advances in Neural Information Processing Systems*, pp. 4302–4313, 2019.

Deng, Y., Bakhtin, A., Ott, M., Szlam, A., and Ranzato, M. Residual energy-based models for text generation. *arXiv preprint arXiv:2004.11714*, 2020.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., and Wang, J. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Hjelm, R. D., Jacob, A. P., Che, T., Trischler, A., Cho, K., and Bengio, Y. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*, 2017.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Leblond, R., Alayrac, J.-B., Sifre, L., Pislar, M., Lespiau, J.-B., Antonoglou, I., Simonyan, K., and Vinyals, O. Machine translation decoding beyond beam search. *arXiv preprint arXiv:2104.05336*, 2021.

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Lu, S., Yu, L., Feng, S., Zhu, Y., and Zhang, W. Cot: Cooperative training for generative modeling of discrete data. In *International Conference on Machine Learning*, pp. 4164–4172. PMLR, 2019.

Ma, Z. and Collins, M. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*, 2018.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Negrinho, R., Gormley, M., and Gordon, G. J. Learning beam search policies via imitation learning. In *Advances in Neural Information Processing Systems*, pp. 10652–10661, 2018.

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., Schuurmans, D., et al. Reward augmented maximum likelihood for neural structured prediction. *Advances In Neural Information Processing Systems*, 29:1723–1731, 2016.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Paulus, R., Xiong, C., and Socher, R. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Rosin, C. D. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3): 203–230, 2011.

Scialom, T., Lamprier, S., Piwowarski, B., and Staiano, J. Answers unite! unsupervised metrics for reinforced summarization models. *CoRR*, abs/1909.01610, 2019. URL http://arxiv.org/abs/1909.01610.

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., and Staiano, J. Coldgans: Taming language gans with cautious sampling strategies. *Advances in Neural Information Processing Systems*, 2020a.

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., and Staiano, J. Discriminative adversarial search for abstractive summarization. *arXiv preprint arXiv:2002.10375*, 2020b.

Scialom, T., Dray, P., Lamprier, S., Piwowarski, B., and Staiano, J. To beam or not to beam: That is a question of cooperation for language gans. *CoRR*, abs/2106.06363, 2021a. URL https://arxiv.org/abs/2106.06363.

Scialom, T., Dray, P.-A., Staiano, J., Lamprier, S., and Piwowarski, B. To beam or not to beam: That is a question of cooperation for language gans. *Advances in Neural Information Processing Systems*, 34, 2021b.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, 2018.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Song, Y., Miao, N., Zhou, H., Yu, L., Wang, M., and Li, L. Improving maximum likelihood training for text generation with density ratio estimation. In *International Conference on Artificial Intelligence and Statistics*, pp. 122–132. PMLR, 2020.

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

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Yu, L., Zhang, W., Wang, J., and Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Zhou, W., Ge, T., Xu, K., Wei, F., and Zhou, M. Self-adversarial learning with comparative discrimination for text generation. *arXiv preprint arXiv:2001.11691*, 2020.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, figshare, 2010.

A. Appendix

A.1. Proof for Theorem 2.1

Let p_d the data distribution that we seek at approximating, and p_0 be the initial generator, of same support $\mathcal{Y}$ as p_d and which is not null everywhere p_d is not null.

As considered in Algorithm 1, let consider each step t the learning the following discriminator optimization:

$$D_t \leftarrow \arg\max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{y \sim p_d(y)} [\log D(y)] + \mathop{\mathbb{E}}_{y \sim p_{t-1}(y)} [\log(1 - D(y))]$$

Thus, following the proof in (Goodfellow et al., 2014), if $\mathcal{D}$ has enough capacity, $D_t(y) = \dfrac{p_d(y)}{p_d(y) + p_{t-1}(y)}$ for every $y \in \mathcal{Y}$.

Also, at each step t of Algorithm 1, we set:

$$p_t \leftarrow \arg\min_{p \in \mathcal{G}} KL(q_t || p)$$

With $q_t(y) \triangleq \dfrac{D_t(y) p_{t-1}(y)}{z_t}$ for each $t > 0$ and all $y \in \mathcal{Y}$, where z_t is the partition function of distribution q_t.

Thus, if $\mathcal{G}$ has enough capacity and p_t is sufficiently trained, we have for every $y \in \mathcal{Y}$ and every $t \geq 1$:

$$p_t(y) \propto D_t(y) p_{t-1}(y) = \frac{p_d(y) p_{t-1}(y)}{p_d(y) + p_{t-1}(y)} = \frac{p_d(y)}{(p_d(y)/p_{t-1}(y)) + 1} \triangleq \tilde{p}_t(y) \tag{5}$$

With $z_t \triangleq \sum_{y \in \mathcal{Y}} \tilde{p}_t(y)$, we have $p_t(y) = \dfrac{\tilde{p}_t(y)}{z_t}$.

In the following, we consider, for all $y \in \mathcal{Y}$, the sequence $\hat{z}_t(y)$ defined as:

$$\hat{z}_t(y) = \begin{cases} p_d(y)/p_0(y), & \text{if } t = 0; \\ z_t(\hat{z}_{t-1}(y) + 1), & \forall t \geq 1. \end{cases}$$

Lemma A.1. *At every step t of Algorithm 1, we have for all $y \in \mathcal{Y}$:*

$$\tilde{p}_{t+1}(y) = \frac{p_d(y)}{\hat{z}_t(y) + 1}$$

Proof. Let consider a proof by induction.

First consider the base case where $t = 0$. From eq.(5), we have $\tilde{p}_1(y) = \dfrac{p_d(y)}{(p_d(y)/p_0(y)) + 1}$ and thus, $\tilde{p}_1(y) = \dfrac{p_d(y)}{\hat{z}_0(y) + 1}$.

Let now assume that $\tilde{p}_t(y) = \dfrac{p_d(y)}{\hat{z}_{t-1}(y) + 1}$ is true at any step $t > 0$. We need to show that this relation still holds for $t + 1$ to prove the lemma.

Under this assumption, starting from Eq.(5), we have:

$$\begin{aligned} \tilde{p}_{t+1} &= \frac{p_d(y) p_t(y)}{p_d(y) + p_t(y)} = \frac{p_d(y) \tilde{p}_t(y)}{p_d(y) z_t + \tilde{p}_t(y)} = \frac{p_d(y) \tilde{p}_t(y)}{p_d(y) z_t + p_d(y)/(\hat{z}_{t-1}(y) + 1)} \\ &= \frac{\tilde{p}_t(y)(\hat{z}_{t-1}(y) + 1)}{z_t(\hat{z}_{t-1}(y) + 1) + 1} = \frac{p_d(y)}{z_t(\hat{z}_{t-1}(y) + 1) + 1} = \frac{p_d(y)}{\hat{z}_t(y) + 1} \end{aligned}$$

□

Lemma A.2. *For every step $t > 1$ of Algorithm 1, $z_t < 1$.*

Proof. For every step $t > 0$, using lemma A.1 on the second and fourth equality (below), we have:

$$\begin{aligned} z_{t+1} &= \sum_{y\in\mathcal{Y}} \tilde{p}_{t+1}(y) = \sum_{y\in\mathcal{Y}} \frac{p_d(y)}{\hat{z}_t(y)+1} = \sum_{y\in\mathcal{Y}} \frac{p_d(y)}{\hat{z}_{t-1}(y)+1}\frac{\hat{z}_{t-1}(y)+1}{\hat{z}_t(y)+1} = \sum_{y\in\mathcal{Y}} \tilde{p}_t(y)\frac{\hat{z}_{t-1}(y)+1}{\hat{z}_t(y)+1} \\ &= \sum_{y\in\mathcal{Y}} p_t(y)\frac{z_t(\hat{z}_{t-1}(y)+1)}{\hat{z}_t(y)+1} = \sum_{y\in\mathcal{Y}} p_t(y)\frac{\hat{z}_t(y)}{\hat{z}_t(y)+1} = \mathbb{E}_{y\sim p_t(y)}\left[\frac{\hat{z}_t(y)}{\hat{z}_t(y)+1}\right] \end{aligned}$$

Thus, since $\hat{z}_t(y) \geq 0$ for all $y \in \mathcal{Y}$ and all $t \geq 0$, $z_{t+1} < 1$ for all $t > 0$.

□

Then, to prove theorem 1 (convergence of p_t to p_d in law), let us rewrite $\hat{z}_t$ (using its definition for $t > 0$) as:

$$\hat{z}_t(y) = z_t(\hat{z}_{t-1}(y)+1) = \prod_{s=1}^{t} z_s(\frac{p_d(y)}{p_0(y)}) + \sum_{s=1}^{t}\prod_{s'=s}^{t} z_s$$

For any pair $(y, y') \in \mathcal{Y}^2$, we thus have:

$$\hat{z}_t(y) - \hat{z}_t(y') = (\frac{p_d(y)}{p_0(y)} - \frac{p_d(y')}{p_0(y')})\prod_{s=1}^{t} z_s$$

Since from Lemma A.2 we know that $z_t < 1$ for any $t > 1$, we have: $\lim_{t\to+\infty}\prod_{s=1}^{t} z_s = 0$ and thus, $\hat{z}_t(y) - \hat{z}_t(y')$ converges to 0 for any pair $(y, y') \in \mathcal{Y}^2$, ensuring that $\hat{z}_t(y)$ converges to a constant K, which shows that

$$\tilde{p}_t(y) \underset{+\infty}{\to} \frac{p_d(y)}{1+K}$$

which in turn implies our final conclusion, i.e. that p_t converges in distribution to p_d.

A.2. Proof for Theorem 2.2

Let us consider the case of $p_t \propto p_{t-1}D_t$, and a discriminator sufficiently trained such that, i.e. such that for

$$\log\eta = \min\left(\mathop{\mathbb{E}}_{y\sim p_d(y)}[\log(D_t(y)], \mathop{\mathbb{E}}_{y\sim p_{t-1}(y)}[\log(1-D_t(y))]\right) \tag{6}$$

we have $\eta \in]\frac{1}{2}; 1[$

The difference of KL divergences of the target distribution p_d from the generator distribution taken at two successive steps is given as:

$$\begin{aligned} \Delta_t &\triangleq KL(p_d||p_t) - KL(p_d||p_{t-1}) \\ &= \mathop{\mathbb{E}}_{y\sim p_d(y)}[\log(p_{t-1}(y)) - \log(p_t(y))] \\ &= \mathop{\mathbb{E}}_{y\sim p_d(y)}[\log(p_{t-1}(y)) - \log(p_{t-1}(y)D_t(y))] + \log(\sum_{y'\in\mathcal{Y}} p_{t-1}(y)D_t(y)) \\ &= \mathop{\mathbb{E}}_{y\sim p_d(y)}[-\log(D_t(y))] + \log(\sum_{y\in\mathcal{Y}} p_{t-1}(y)D_t(y)) \\ &= \log(\mathop{\mathbb{E}}_{y\sim p_{t-1}(y)}[D_t(y)]) - \mathop{\mathbb{E}}_{y\sim p_d(y)}[\log(D_t(y))] \end{aligned}$$

From the assumption given in Eq.(6), we have:

$$\begin{aligned} \log\eta &\leq \mathop{\mathbb{E}}_{y\sim p_{t-1}(y)}[\log(1-D_t(y))] \\ &\leq \log(\mathop{\mathbb{E}}_{y\sim p_{t-1}(y)}[1-D_t(y)]) \end{aligned}$$

where the second inequality is obtained with the Jensen inequality on expectations of concave functions.

This equivalent to:

$$\log(1 - \mathop{\mathbb{E}}_{y \sim p_{t-1}(y)}[1 - D_t(y)]) \leq \log(1 - \eta)$$

And thus:

$$\log(\mathop{\mathbb{E}}_{y \sim p_{t-1}(y)}[D_t(y)]) \leq \log(1 - \eta)$$

From assumption of Eq.6, we also know that $\mathop{\mathbb{E}}_{y \sim p_d(y)}[\log(D_t(y))] \geq \log(\eta)$.

Thus, we have:

$$\Delta_t \leq \log(1 - \eta) - \log(\eta) = \log(\frac{1}{\eta} - 1) < 0$$

which concludes the proof.

A.3. Implementation Details

Our experimental settings are similar to those used in the SelfGAN paper (Scialom et al., 2021b). In all our experiments, we used the T5-small (Raffel et al., 2019) generator,[4] in which the positional embedding is relative. For the discriminators, we frame the classification task as a text2text task where the model has to generate either the token *human* or *machine*. This allows to use again T5-small for all experiments, removing possible bias from architecture differences between the generator and the discriminator. We start by training via Teacher Forcing a model corresponding to the MLE baseline. All our GANs are initialized from this MLE model. During this pre-training, we used a learning rate fixed to 5e-6 for both the discriminator and the generator, and a number of epochs set to 5.

In MCTS, sequence lengths are not aligned as in a standard left-to-right decoding algorithm. Therefore, we used a simple trick to enable efficient batching of sequences, that can be applied to any Language Model benefiting from a relative positional embedding (Shaw et al., 2018). We used a custom left padding that shifts the start of each sequences from a batch, so that all of their last tokens are aligned.

We tested on a validation set different values for our hyper parameter $C_{puct} \in \{1.0, 2.0, 3.0, 4.0\}$ and found that 3.0 gives the best results. We thus only report the results with $C_{puct} = 3.0$. For the budget allocated to the MCTS we tested different number of simulations per token for the MLE model with $n \in \{5, 10, 25, 50, 100\}$ and observed no significant improvement between 50 and 100. We hence used $n = 50$ for all our experiments.

A.4. Variance and Samples of our State-of-the-art Approach

	QG			Summarization		
	B	R-1	R-L	B	R-1	R-L
$\text{GCN}^{\hat{q}=MCTS}_{T5-3B}$	21.8 (0.06)	49.7 (0.16)	45.1 (0.09)	19.2 (0.04)	44.1 (0.1)	43.8 (0.09)

Table 2. Results with standard-deviation (in brackets) for our state-of-the-art $\text{GCN}^{\hat{q}=MCTS}_{T5-3B}$ approach.

Table 2 gives standard deviation of our state of the art approach over three seeds. This shows very stable results.

We report below some QG samples from this model for the following input text:

```
Input:
Super Bowl 50 was an American football game to determine the champion of the National
    Football League (NFL) for the 2015 season. The American Football Conference (AFC)
    champion Denver Broncos defeated the National Football Conference (NFC) champion
    Carolina Panthers 24 euros 10 to earn their third Super Bowl title. The game was
    played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa
```

[4] As implemented in HuggingFace transformers (Wolf et al., 2019).

```
    Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden
    anniversary" with various gold-themed initiatives, as well as temporarily suspending
    the tradition of naming each Super Bowl game with Roman numerals

Conditioned Answer: Super Bowl
MLE: What was the name of the game that would have been known as "Super Bowl
GCN: How is called the American football game which determines the NFL champion?

Conditioned Answer: golden anniversary
GCN: As this was the 50th Super Bowl, what was emphasized by the league?

Conditioned Answer: 50th Super Bowl
GCN: The league emphasized the "golden anniversary" during what Super Bowl?
```

A.5. Results with Former MLE baseline

As mentioned in section 4, results reported in table 1 are not using the same base model as for figures 2 and 3. For completeness, we report here results obtained using this former model used for the figures, which has the same architecture but uses a learning rate of $1e-3$ during 2 training epochs, while the new one used for table 1 uses a learning rate of $1e-4$ during 5 training epochs (which is slower but more accurate).

	QG			Summarization		
	B	R-1	R-L	B	R-1	R-L
MLE	16.5	43.9	40	11.5	36.8	34.9
ColdGAN	16.9	44.2	40.3	11.6	37.8	36.4
SelfGAN	17.2	44.3	40.6	12.3	38.6	36.7
$\text{GAN}^{\hat{q}=p}_{+scheduler}$	16.2	43.1	39.3	11.2	36.1	34.3
$\text{GAN}^{\hat{q}=p}$	9.4	25.0	22.8	7.1	21.0	19.9
$\text{GCN}^{\hat{q}=p}$	16.5	43.9	40.0	11.5	36.8	34.9
$\text{GAN}^{\hat{q}=Nucleus}_{+scheduler}$	16.9	45.0	41.0	11.6	37.7	35.7
$\text{GAN}^{\hat{q}=Nucleus}$	9.5	25.3	23.0	7.4	21.2	20.1
$\text{GCN}^{\hat{q}=Nucleus}$	17.5	45.3	42.4	12	39.0	37.0
$\text{GAN}^{\hat{q}=MCTS}_{+scheduler}$	17.1	45.5	41.5	11.9	38.1	36.2
$\text{GAN}^{\hat{q}=MCTS}$	9.8	26.1	23.8	8.5	21.9	20.7
$\text{GCN}^{\hat{q}=MCTS}$	**18**	**45.9**	**42.6**	**12.4**	**39.1**	**37.1**
$\text{GCN}^{\hat{q}=MCTS}_{decod=mcts}$	**18.4**	**46.3**	**43.1**	**12.7**	**39.4**	**37.4**
$\text{GCN}^{\hat{q}=MCTS}_{T5-3B}$	**21.8**	**49.8**	**45.9**	**19.2**	**44.2**	**43.8**

Table 3. Final results on QG and Summarization test sets, in terms of BLEU-4 (B), ROUGE-1 (R-1) and ROUGE-L (R-L), using the same base model as used in (Scialom et al., 2021a).