
CoT: Cooperative Training for Generative Modeling

Sidi Lu

Shanghai Jiao Tong University
steve_lu@apex.sjtu.edu.cn

Lantao Yu

Shanghai Jiao Tong University
yulantao@apex.sjtu.edu.cn

Weinan Zhang

Shanghai Jiao Tong University
wnzhang@apex.sjtu.edu.cn

Yong Yu

Shanghai Jiao Tong University
yyu@apex.sjtu.edu.cn

Abstract

We propose Cooperative Training (CoT) for training generative models that measure a tractable density function for target data. CoT coordinately trains a generator G and an auxiliary predictive mediator M . The training target of M is to estimate a mixture density of the learned distribution G and the target distribution P , and that of G is to minimize the Jensen-Shannon divergence estimated through M . CoT achieves independent success without the necessity of pre-training via Maximum Likelihood Estimation or involving high-variance algorithms like REINFORCE. This low-variance algorithm is theoretically proved to be unbiased for both generative and predictive tasks. We also theoretically and empirically show the superiority of CoT over most previous algorithms, in terms of generative quality and diversity, predictive generalization ability and computational cost.

1 Introduction

Generative modeling is essential in many scenarios, including continuous data modeling (*e.g.* image generation [Goodfellow *et al.*, 2014; Arjovsky *et al.*, 2017], stylization [Ulyanov *et al.*, 2016], semi-supervised classification [Radford *et al.*, 2015]) and sequential discrete data modeling (*e.g.* neural text generation [Bahdanau *et al.*, 2014; Yu *et al.*, 2017; Lu *et al.*, 2018]).

For data with tractable density like natural language, generative models are predominantly optimized through Maximum Likelihood Estimation (MLE), inevitably introducing *exposure bias* [Ranzato *et al.*, 2015], which results in that given a **finite** set of observations, the optimal parameters of the model trained via MLE do not correspond to the ones maximizing the generative quality. Specifically, the model is trained on the data distribution of inputs and tested on a different distribution of inputs, namely, the distribution of model output. This discrepancy implies that in the training stage, the model is never exposed to its own errors and thus in the test stage, the errors made along the way will quickly accumulate.

On the other hand, for general generative modeling tasks, an effective framework, named Generative Adversarial Network (GAN) [Goodfellow *et al.*, 2014], was proposed to mimic an intractable implicit density for continuous data. GAN introduces a discriminator D_ϕ parametrized by ϕ to distinguish the generated samples from the real ones. As is proved in [Huszár, 2015], GAN essentially optimizes an approximately estimated Jensen-Shannon divergence (JSD) between the currently learned distribution and the target distribution. GAN shows promising results in many unsupervised and semi-supervised learning tasks. The success of GAN results in the naissance of a new paradigm of deep generative models, *i.e.* adversarial networks.

However, since the gradient computation requires backpropagation through the generator’s output, GAN can only model the distribution of continuous variables, making it non-applicable for discrete

sequences generation. Researchers then proposed Sequence Generative Adversarial Network (SeqGAN) [Yu *et al.*, 2017], which uses model-free policy gradient algorithm to optimize the original GAN objective. With SeqGAN, the expected JSD between current and target discrete data distribution is minimized if the training being perfect. SeqGAN shows observable improvements in many tasks. Since then, many variants of SeqGAN have been proposed to improve its performance. However, SeqGAN is not an ideal algorithm for this problem, and current algorithms based on it cannot show stable, reliable and observable improvements that covers all scenarios, according to a previous survey [Lu *et al.*, 2018]. The reason will be discussed in detail in Section 3.

In this paper, we propose Cooperative Training (CoT), an efficient, low-variance, bias-free algorithm for training likelihood-based models by directly optimizing a well-estimated Jensen-Shannon divergence. CoT coordinately trains a generative module G , and an auxiliary predictive module M , called *mediator*, for guiding G in a cooperative fashion. For theoretical soundness, we derive the proposed algorithm directly from the definition of JSD. We further empirically and theoretically show the remarkable superiority of our algorithm over many strong baselines in terms of generative performance, generalization ability and computational performance in both synthetic and real-world scenarios.

2 Background

Notations. θ denotes the parameters of the generative module. For those that incorporate auxiliary predictive modules as density estimators, ϕ denotes the parameters of these additional modules. s stands for a complete sample from the training dataset or a generated complete sequence, depending on detailed declaration and context. When something is referred to as s_t , it means the t -length prefix of the original sequence, *i.e.* an incomplete sequence of length t . x denotes a token, and x_k stands for a token that appears in the k -th place of a sequence.

2.1 Maximum Likelihood Estimation

Maximum likelihood Estimation is equivalent to minimizing the KL divergence using the samples from the real distribution as

$$\min_{\theta} \mathbb{E}_{s \sim p_{\text{data}}} [-\log G_{\theta}(s)], \quad (1)$$

where $G_{\theta}(s)$ is the estimated probability of s by G_{θ} and p_{data} is the underlying real distribution.

2.2 Sequence Generative Adversarial Network

SeqGAN incorporates two modules, *i.e.* the generator and discriminator, parametrized by θ and ϕ respectively, as in the settings of GAN. By alternatively training these two modules, SeqGAN optimizes such an adversarial target:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{s \sim p_{\text{data}}} [\log(D_{\phi}(s))] + \mathbb{E}_{s \sim G_{\theta}} [\log(1 - D_{\phi}(s))]. \quad (2)$$

The objectives of generator G_{θ} and discriminator D_{ϕ} in SeqGAN can be formulated as

$$\text{Generator: } \min_{\theta} -\mathbb{E}_{s \sim G_{\theta}} \left[\sum_{t=1}^n Q_t(s_t, x_t) \cdot \log G_{\theta}(x_t | s_t) \right] \quad (3)$$

$$\text{Discriminator: } \max_{\phi} \mathbb{E}_{s \sim p_{\text{data}}} [\log(D_{\phi}(s))] + \mathbb{E}_{s \sim G_{\theta}} [\log(1 - D_{\phi}(s))], \quad (4)$$

where $s \sim G_{\theta} = [x_1, \dots, x_n]$ denotes a complete sequence sampled from the generator and the action value $Q_t(s_t, x_t) = \mathbb{E}_{s \sim G_{\theta}(\cdot | s_{t+1})} [D_{\phi}(s)]$ is the expectation of the output of discriminator given prefix $s_{t+1} = [s_t, x_t]$, which can be approximated via Monte Carlo search.

3 On the Limitations of Previous Algorithms

3.1 Limitations of MLE

MLE is essentially equivalent to optimizing a directed Kullback–Leibler (KL) divergence between the target distribution P and the currently learned distribution G , denoted as $KL(P||G)$. However,

since KL divergence is asymmetric, given **finite** observations this target is actually not ideal. As stated in Arjovsky and Bottou [2017], MLE tries to minimize

$$KL(P\|G) = \sum_s P(s) \log \frac{P(s)}{G(s)}. \quad (5)$$

- When $P(s) > 0$ and $G(s) \rightarrow 0$, the KL divergence grows to infinity, which means MLE assigns an extremely high cost to the “mode dropping” scenarios, where the generator fails to cover some parts of the data.
- When $G(s) > 0$ and $P(s) \rightarrow 0$, the KL divergence shrinks to 0, which means MLE assigns an extremely low cost to the scenarios, where the model generates some samples that do not locate on the data distribution.

Likewise, optimizing $KL(G\|P)$ will lead to exactly the reversed problems of the two situations. An ideal solution is to optimize a **symmetrized** and **smoothed** version of KL divergence, *i.e.* the Jensen-Shannon divergence (JSD), which is defined as

$$JSD(P\|G) = \frac{1}{2} (KL(P\|M) + KL(G\|M)), \quad (6)$$

where $M = \frac{1}{2}(P + G)$. However, directly optimizing JSD is conventionally considered as an intractable problem. JSD cannot be directly evaluated and optimized since the equally interpolated distribution M is usually considered unconstructable, as we only have access to the learned model Q instead of P .

3.2 Limitations of SeqGAN & its Variants

First, SeqGAN is an algorithm of high variance, which relies on pre-training via Maximum Likelihood Estimation as a variance reduction procedure. Besides, during the adversarial epochs, even if with variance reduction techniques such as Actor-Critic methods [Sutton, 1984], the fact that SeqGAN is essentially based on model-free reinforcement learning makes it a non-trivial problem for SeqGAN to converge well. As a result, SeqGAN tends to get stuck in some sub-optimal. Specifically, although discriminator can distinguish the output of generator easily, it is not able to effectively guide the generator because of the vanishing gradient, as is discussed in a recent survey [Lu *et al.*, 2018]. Although this problem can be alleviated by RankGAN [Lin *et al.*, 2017] or BRA [Guo *et al.*, 2017], they are more technical workarounds than essential solutions.

Second, it is common that SeqGAN suffers from the “mode collapse” problem, which is similar to the original GAN. That is to say, the learned distribution “collapse” to the other side of KL divergence, *i.e.* $KL(G\|P)$, which leads to the loss of diversity of generated samples. In other words, SeqGAN trains the model for better generative quality with the cost of diversity.

4 Cooperative Training for Generative Modeling

4.1 Motivation

To be consistent with the goal that the target distribution should be well-estimated in both **quality** and **diversity** senses, an ideal algorithm for such models should be able to optimize a symmetric divergence or distance.

Since for sequential discrete data, the predicted probability for each sample will always be positive and we do not require gradient availability, the failures of JSD, as discussed in [Arjovsky *et al.*, 2017], will not appear in this case. Thus the choice to optimize JSD is acceptable. However, to our knowledge, no previous algorithms provide an unbiased, direct optimization of JSD. In this paper, we propose Cooperative Training (CoT), as shown in Algorithm 1, to directly optimize a well-estimated unbiased JSD for training such models.

4.2 Algorithm Derivation

Each iteration of Cooperative Training mainly consists of two parts. The first part is to train a *mediator* M_ϕ , which is a predictive module that measures a mixture distribution of the learned generative

Algorithm 1 Cooperative Training

Require: Generator G_θ ; mediator M_ϕ ; Samples from real data distribution P ; Hyper-parameter m .

- 1: Initialize G_θ , M_ϕ with random weights θ, ϕ .
 - 2: **repeat**
 - 3: **for** m steps **do**
 - 4: Collect a mini-batch of mixed balanced samples $\{s\}$ from both G_θ and P
 - 5: Update mediator M_ϕ with $\{s\}$ via Eq. (9)
 - 6: **end for**
 - 7: Generate a mini-batch of sequences $\{s\} \sim G_\theta$
 - 8: Update generator G_θ with $\{s\}$ via Eq. (13)
 - 9: **until** CoT converges
-

distribution G_θ and target latent distribution $P = p_{\text{data}}$ as

$$M_\phi \simeq \frac{1}{2}(P + G_\theta). \quad (7)$$

Since the mediator is only used as a **predictive** module during training, the directed KL divergence is now bias-free for measuring M_ϕ and P . Denote $\frac{1}{2}(P + G_\theta)$ as M^* , we have:

Lemma 1 (Mixture Density Decomposition)

$$\begin{aligned} \nabla_\phi J_m(\phi) &= \nabla_\phi KL(M^* \| M_\phi) \\ &= \nabla_\phi \mathbb{E}_{s \sim M^*} \left[\log \frac{M^*(s)}{M_\phi(s)} \right] \\ &= \nabla_\phi \left(- \mathbb{E}_{s \sim M^*} [\log M_\phi(s)] \right) \\ &= \nabla_\phi \frac{1}{2} \left(\mathbb{E}_{s \sim G_\theta} [-\log(M_\phi(s))] + \mathbb{E}_{s \sim P} [-\log(M_\phi(s))] \right) \end{aligned} \quad (8)$$

By Lemma 1, for each step, we can simply mix balanced samples from training data and the generator, then train the mediator via Maximum Likelihood Estimation with the mixed samples. The objective for the mediator M parametrized by ϕ therefore becomes

$$J_m(\phi) = \frac{1}{2} \left(\mathbb{E}_{s \sim G_\theta} [-\log(M_\phi(s))] + \mathbb{E}_{s \sim P} [-\log(M_\phi(s))] \right). \quad (9)$$

Since the objective of MLE is bias-free for predictive purposes, the estimated M_ϕ is also bias-free when adopted for estimating JSD. The training techniques and details will be discussed in Section 5.

After each iteration, the mediator is exploited to optimize an estimated Jensen-Shannon divergence for G_θ :

$$\begin{aligned} \nabla_\theta J_g(\theta) &= \nabla_\theta \left(-J\hat{S}D(G_\theta \| P) \right) \\ &= \nabla_\theta \left(-\frac{1}{2} [KL(G_\theta \| M_\phi) + KL(P \| M_\phi)] \right) \\ &= \nabla_\theta \left(-\frac{1}{2} \mathbb{E}_{s \sim G_\theta} \left[\log \frac{G_\theta(s)}{M_\phi(s)} \right] - \frac{1}{2} \mathbb{E}_{s \sim P} \left[\log \frac{P(s)}{M_\phi(s)} \right] \right) \\ &= \nabla_\theta \left(-\frac{1}{2} \mathbb{E}_{s \sim G_\theta} \left[\log \frac{G_\theta(s)}{M_\phi(s)} \right] \right) \quad (\text{one step update}). \end{aligned} \quad (10)$$

For any sequence or prefix of length t , we have:

Lemma 2 (Markov Backward Reduction)

$$\nabla_\theta \left(-\frac{1}{2} \mathbb{E}_{s_t \sim G_\theta} \left[\log \frac{G_\theta(s_t)}{M_\phi(s_t)} \right] \right) \quad (11)$$

$$= \nabla_\theta \left(-\frac{1}{2} \mathbb{E}_{s_{t-1} \sim G_\theta} \left[\sum_{s_t} G_\theta(s_t | s_{t-1}) \log \frac{G_\theta(s_t | s_{t-1})}{M_\phi(s_t | s_{t-1})} \right] - \frac{1}{2} \mathbb{E}_{s_{t-1} \sim G_\theta} \left[\log \frac{G_\theta(s_{t-1})}{M_\phi(s_{t-1})} \right] \right). \quad (12)$$

The detailed derivations can be found in the Appendix. Note that Lemma 2 can be applied recursively. That is to say, given any sequence s_t of arbitrary length t , optimizing s_t 's contribution to the expected JSD can be decomposed into optimizing the first term of Eq. (12) and solving an isomorphic problem for s_{t-1} , which is the longest proper prefix of s_t . When $t = 1$, since in Markov decision process the probability for initial state is always 1.0, it is trivial to prove that the final second term becomes zero.

Therefore, Eq. (10) can be reduced through recursively applying Lemma 2. After removing the constant multipliers and denoting the predicted probability distribution over the action space, *i.e.* $G_\theta(\cdot|s_t)$ and $M_\phi(\cdot|s_t)$, as $\pi_g(s_t)$ and $\pi_m(s_t)$ respectively, the gradient for training generator via Cooperative Training can be formulated as

$$\nabla_\theta J_g(\theta) = \nabla_\theta \mathbb{E}_{s \sim G_\theta} \left[\sum_{t=0}^{n-1} \pi_g(s_t)^\top (\log \pi_m(s_t) - \log \pi_g(s_t)) \right]. \quad (13)$$

The overall objective of CoT can be formulated as

$$\max_{\theta} \max_{\phi} \mathbb{E}_{s \sim p_{\text{data}}} [\log(M_\phi(s))] + \mathbb{E}_{s \sim G_\theta} [\log(M_\phi(s))]. \quad (14)$$

Note the strong connections and differences between Eq. (14) and Eq. (2). Figure 1 illustrates the whole Cooperative Training process.

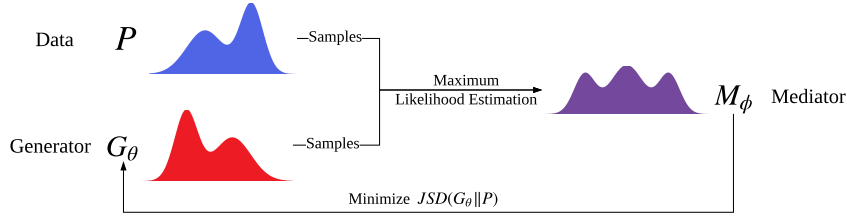


Figure 1: Process of Cooperative Training.

4.3 Relation to Adversarial Training & EM Algorithm

Both Cooperative Training and Adversarial Training can be regarded as extended variants of the Expectation Maximization algorithm. The most remarkable difference is that during the *Expectation* pass of each iteration, the optimal discriminator of GANs estimates a residual relative density, *i.e.* $D^* = 1 - \frac{p_{\text{fake}}}{p_{\text{real}} + p_{\text{fake}}} = \frac{p_{\text{real}}}{p_{\text{fake}} + p_{\text{real}}}$, as is discussed in the analysis of GAN. In CoT, however, the optimal mediator directly estimates a balanced mixture density of both distributions $M^* = \frac{p_{\text{real}} + p_{\text{fake}}}{2}$.

Note that in this sense, for *Maximizing* the expected likelihood of real data with respect to the estimator, the two modules of GANs have opposite optimization directions. This is what the word *adversarial* stands for. By contrast, in CoT, the estimated density by M^* is always $\frac{p_{\text{real}} + p_{\text{fake}}}{2}$ and will converge to p_{real} in ideal case. Even if M_ϕ is not trained to be optimal, since its training exploits equal numbers of samples from both distributions, it is still a well-balanced interpolation between P and G_θ .

4.4 Convergence Analysis

CoT has theoretical guarantee for its convergence.

Theorem 3 (Jensen-Shannon Consistency) *If in each step, the mediator M_ϕ of CoT is trained to be optimal, *i.e.* $M_\phi = M^* = \frac{1}{2}(G_\theta + P)$, then optimization via Eq. (14) leads to minimization of $JSD(G_\theta \| P)$.*

Theorem 4 (Jensen-Shannon Efficiency) *If in each step, the mediator M_ϕ of CoT is trained to be optimal, *i.e.* $M_\phi = M^* = \frac{1}{2}(G_\theta + P)$, then optimization via Eq. (14) is **one-order optimal** for minimizing $JSD(G_\theta \| P)$.*

Proof. Let p denote the intermediate states. All we need to show is

$$\nabla_\theta \mathbb{E}_{s \sim G_\theta} \left[\sum_{t=1}^n \pi_g(s_t)^\top (\log \pi_m(s_t) - \log \pi_g(s_t)) \right] \propto \nabla_\theta JSD(P \| G_\theta). \quad (15)$$

By inversely applying Lemma 2, the left term in Eq. (15) can be recovered as

$$\nabla_{\theta} \left(\frac{1}{2} \mathbb{E}_{s \sim G_{\theta}} \left[\log \frac{G_{\theta}(s)}{M_{\phi}(s)} \right] \right), \quad (16)$$

which is equivalent to

$$\nabla_{\theta} \left(\mathbb{E}_{s \sim G_{\theta}} \left[\log \frac{G_{\theta}(s)}{M_{\phi}(s)} \right] + \mathbb{E}_{s \sim P} \left[\log \frac{P(s)}{M_{\phi}(s)} \right] \right). \quad (17)$$

Since now mediator is trained to be optimal, *i.e.* $M_{\phi} = M^*$, we have

$$\begin{aligned} (17) &= \nabla_{\theta} \left(\mathbb{E}_{s \sim G_{\theta}} \left[\log \frac{G_{\theta}(s)}{M^*(s)} \right] + \mathbb{E}_{s \sim P} \left[\log \frac{P(s)}{M^*(s)} \right] \right) \\ &= 2\nabla_{\theta} JSD(P \| G_{\theta}) \propto \nabla_{\theta} JSD(P \| G_{\theta}). \end{aligned} \quad (18)$$

4.5 Discussion

CoT has several practical advantages over previous methods, including MLE, Scheduled Sampling (SS) [Bengio *et al.*, 2015] and adversarial methods like SeqGAN [Yu *et al.*, 2017].

First, although CoT and GAN both aim to optimize an estimated JSD, CoT is exceedingly more stable than GAN. This is because the two modules, namely generator and mediator, have similar tasks, *i.e.* to approach the same data distribution **generatively** and **predictively**. In ideal case, when the generator converges to P , the optimal mediator $M^* = \frac{1}{2}(G + P)$ will converge to the same target $M^* = \frac{1}{2}(P + P) = P$. Also, the superiority of CoT over inconsistent methods like Scheduled Sampling is obvious, since CoT theoretically guarantees the training effectiveness. Compared with methods that require extra computations in order to reduce variance like SeqGAN [Yu *et al.*, 2017], CoT is computationally cheaper. More specifically, under recommended settings, CoT has the same computational complexity as MLE.

Second, CoT is very practical in conditional generation. Since generator and mediator have similar targets, they can share some parameters like word embeddings and the encoder in sequence-to-sequence tasks (*e.g.*, neural machine translation), without extra efforts to, for example, building purified control signals for generator and discriminator as in Conditional GAN [Mirza and Osindero, 2014], or fine-tuning in order to alleviate exposure bias when using MLE.

Third, CoT works independently. In practice, it does not require model pre-training via conventional methods like MLE. This is the first time we manage to do unbiased unsupervised learning on discrete sequential data without using supervised approximation for variance reduction or sophisticated smoothing as in Wasserstein GAN with gradient penalty (WGAN-GP) [Gulrajani *et al.*, 2017].

5 Experiments

5.1 Universal Sequence Modeling in Synthetic Turing Test

Following the synthetic data experiment proposed in [Yu *et al.*, 2017] and refined by Zhu *et al.* [2018], we design a synthetic Turing test¹, in which NLL_{oracle} is calculated for evaluating the quality of samples from the generator. Particularly, to support our claim that our method causes little mode collapse, we calculated NLL_{test} , which is to sample an extra batch of samples from the oracle LSTM, and to calculate the negative log-likelihood measured by the generator. We show that under this more reasonable setting, our proposed algorithm reaches the state-of-the-art performance with exactly the same network architecture. Note that models like LeakGAN [Guo *et al.*, 2017] contain architecture-level modification, which is orthogonal to our approach, thus it will not be included in this part. The results are shown in Table 1.

5.1.1 Discussion

Hyper-parameter Robustness A wide-ranged hyper-parameter tuning experiment (all regularizations are removed) on synthetic data experiment shows that our approach is less sensitive to hyper-parameter choices than adversarial training, as shown in Figure 2.

¹Experiment code: <https://github.com/desire2020/Cooperative-Training>

Table 1: Likelihood-based benchmark for synthetic Turing test. ‘-(MLE)’ means the best performance is acquired during MLE pre-training.

Model/Algorithm	NLL_{oracle}	NLL_{test} (final/best)	best $NLL_{oracle+test}$
MLE	9.08	8.97/7.60	9.43 + 7.67
SeqGAN	8.68	10.10/-(MLE)	(The same as MLE)
RankGAN [Lin <i>et al.</i> , 2017]	8.37	11.19/-(MLE)	(The same as MLE)
MaliGAN [Che <i>et al.</i> , 2017]	8.73	10.07/-(MLE)	(The same as MLE)
CoT (ours)	8.19	8.03/7.54	8.19 + 8.03

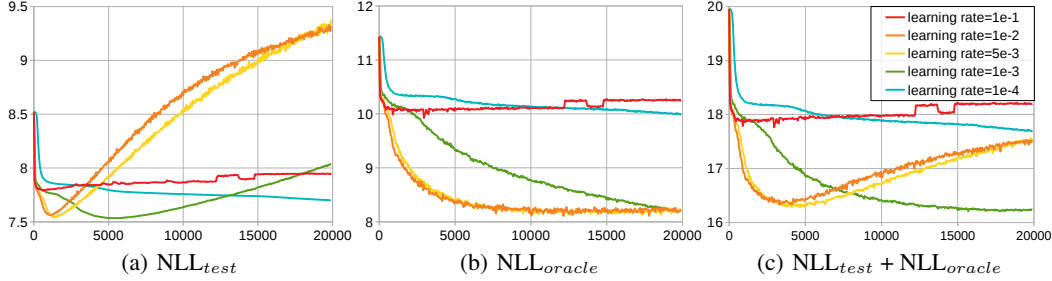


Figure 2: Curves of evaluation on NLL_{test} , NLL_{oracle} and $NLL_{test} + NLL_{oracle}$ during iterations of CoT under different training settings. Note that a model trained via CoT is insensitive to hyper-parameters in terms of generative performance (NLL_{oracle}), yet good hyper-parameters do help in obtaining better generalization ability in terms of predictive performance (NLL_{test}).

Self-estimated Training Progress Indicator We find that the training loss of the mediator, namely *balanced NLL*, can be a real-time training progress indicator as shown in Figure 3. To be specific, in a wide range, balanced NLL is a good estimation of real $JSD(G||P)$ with a steady translation (*i.e.* the entropy of G_θ and P).

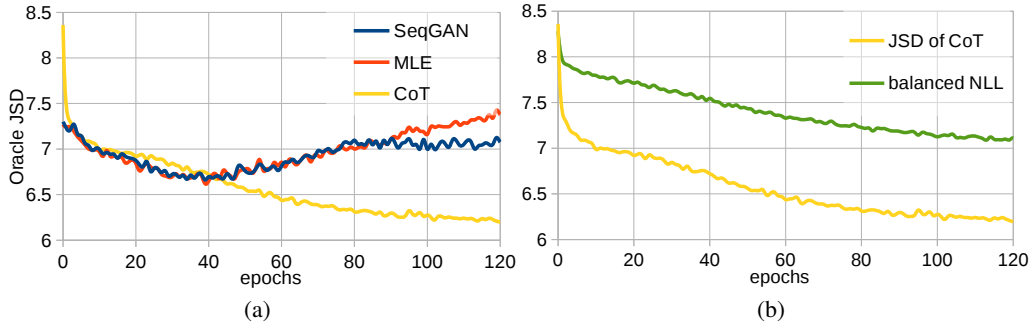


Figure 3: (a) Curves of training time $JSD(G||P)$ for MLE, SeqGAN and CoT. (b) Curves of balanced NLL and real JSD.

5.2 TextCoT: Zero-prior Long & Diversified Text Generation

As an important sequential data modeling task, zero-prior text generation, especially long and diversified text generation, is a good testbed for evaluating the performance of a generative model.

Following the experiment proposed in LeakGAN [Guo *et al.*, 2017], we choose EMNLP 2017 WMT News Section as our dataset, with maximal sentence length limited to 51. We pay major attention to both **quality** and **diversity**. To keep the comparison fair, we present two implementations of CoT, namely CoT-basic and CoT-strong. As for CoT-basic, the generator follows the settings of that in MLE, SeqGAN, RankGAN and MaliGAN. As for CoT-strong, the generator is implemented with the similar architecture in LeakGAN.

Table 2: N-gram-level quality benchmark: BLEU on test data of EMNLP2017 WMT News

Model/Algorithm	BLEU-2	BLEU-3	BLEU-4	BLEU-5
MLE	0.781	0.482	0.225	0.105
SeqGAN [Yu <i>et al.</i> , 2017]	0.731	0.426	0.181	0.096
RankGAN [Lin <i>et al.</i> , 2017]	0.691	0.387	0.178	0.095
MaliGAN [Che <i>et al.</i> , 2017]	0.755	0.456	0.179	0.088
LeakGAN [Guo <i>et al.</i> , 2017]	0.835	0.648	0.437	0.271
TextCoT-basic (ours)	0.785	0.489	0.261	0.152
TextCoT-strong (ours)	0.800	0.501	0.273	0.200
TextCoT-strong ($\alpha = 1.5$) (ours)	0.856	0.701	0.510	0.310

Table 3: Diversity & Generalization benchmark: RS-BLEU and NLL_{test}

Model/Algorithm	RSBLEU-2	RSBLEU-3	RSBLEU-4	RSBLEU-5	NLL_{test}
MLE	0.972	0.893	0.728	0.595	2.365
SeqGAN [Yu <i>et al.</i> , 2017]	1.060	1.150	1.209	1.142	3.122
RankGAN [Lin <i>et al.</i> , 2017]	1.048	1.106	1.170	1.284	3.083
MaliGAN [Che <i>et al.</i> , 2017]	1.062	1.173	1.227	1.086	3.240
LeakGAN [Guo <i>et al.</i> , 2017]	1.095	1.341	1.744	2.198	2.327
TextCoT-basic (ours)	0.977	0.910	0.757	0.646	2.247
TextCoT-strong (ours)	0.983	0.923	0.837	0.789	2.144

For quality evaluation, we evaluated BLEU on a small batch of test data separated from the original dataset. For diversity evaluation, we calculated Relative Self-BLEU [Zhu *et al.*, 2018] (RSBLEU), which is the ratio of Self-BLEU of the generated samples with respect to that of the test data. To show the diversity in a reasonable range, good samples should have such properties when measured by RSBLEU:

- Less than 1.0 so that the generated text is at least as diversified as the original data;
- Closer to 1.0 so that the diversity is reasonable instead of encouraging exposure bias.

The results are shown in Table 2 and Table 3. In terms of generative quality, CoT-basic achieves state-of-the-art performance over all the baselines with the same architecture-level capacity, especially the long-term robustness at n-gram level (BLEU-4, BLEU-5). CoT-strong using a conservative generation strategy as in [Guo *et al.*, 2017] achieves the best performance over all compared models. In terms of generative diversity, the results show that our model achieves comparable performance to that of models trained via MLE. In terms of predictive generalization ability, our results reaches the state-of-the-art performance on NLL_{test} , which is the optimization target of MLE. This is because models trained by supervised training approaches like MLE tend to suffer from over-fitting.

6 Future Work & Conclusion

We proposed Cooperative Training, a powerful unbiased, low-variance, computationally efficient algorithm inspired by coordinate decent algorithms like GAN and Expectation Maximization. Models trained via CoT shows promising results in many sequential data modeling tasks.

As for future work, we plan to apply CoT to more types of data modeling tasks, e.g., discretized image generation [van den Oord *et al.*, 2016]. Another interesting direction of future work is Nested CoT. Although for predictive tasks MLE is unbiased, it raises the risk of mediator’s overfitting. Nested CoT, which is to train the mediator via CoT, can be used to avoid this. For scenarios that extremely rely on generalization, Nested CoT is very promising.

References

- Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.
- Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv:1701.07875*, 2017.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.
- Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179, 2015.
- Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv:1702.07983*, 2017.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NIPS*, pages 5769–5779, 2017.
- Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. *arXiv:1709.08624*, 2017.
- Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv:1511.05101*, 2015.
- Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. In *NIPS*, pages 3155–3165, 2017.
- Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. Neural text generation: Past, present and beyond. *arXiv preprint arXiv:1803.07133*, 2018.
- Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984.
- Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.
- Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv:1601.06759*, 2016.
- Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.
- Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *arXiv:1802.01886*, 2018.

A Detailed Derivation of the Algorithm

$$\begin{aligned}
(10) &= \nabla_{\theta} \left(-\frac{1}{2} \mathbb{E}_{s_t \sim G_{\theta}} [\log G_{\theta}(s_t) - \log M_{\phi}(s_t)] \right) \\
&= \nabla_{\theta} \left(-\frac{1}{2} \mathbb{E}_{s_t \sim G_{\theta}} \left[\frac{G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1})}{G_{\theta}(s_t)} (\log G_{\theta}(s_t) - \log M_{\phi}(s_t)) \right] \right) \\
&= \nabla_{\theta} \left(-\frac{1}{2} \mathbb{E}_{s_t \sim G_{\theta}} \left[\frac{G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1})}{G_{\theta}(s_t)} (\log G_{\theta}(s_t|s_{t-1})G_{\theta}(s_{t-1}) - \log M_{\phi}(s_t|s_{t-1})M_{\phi}(s_{t-1})) \right] \right) \\
&= -\frac{1}{2} \nabla_{\theta} \left(\sum_{s_t} G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1}) (\log G_{\theta}(s_t|s_{t-1}) - \log M_{\phi}(s_t|s_{t-1})) \right. \\
&\quad \left. + \sum_{s_t} G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1}) \log \frac{G_{\theta}(s_{t-1})}{M_{\phi}(s_{t-1})} \right) \\
&= -\frac{1}{2} \nabla_{\theta} \left(\sum_{s_t} G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1}) (\log G_{\theta}(s_t|s_{t-1}) - \log M_{\phi}(s_t|s_{t-1})) \right. \\
&\quad \left. + \sum_{s_{t-1}} \sum_{s_t} G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1}) \log \frac{G_{\theta}(s_{t-1})}{M_{\phi}(s_{t-1})} \right) \\
&= -\frac{1}{2} \nabla_{\theta} \left(\sum_{s_t} G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1}) (\log G_{\theta}(s_t|s_{t-1}) - \log M_{\phi}(s_t|s_{t-1})) \right. \\
&\quad \left. + \sum_{s_{t-1}} \left(G_{\theta}(s_{t-1}) \log \frac{G_{\theta}(s_{t-1})}{M_{\phi}(s_{t-1})} \right) \sum_{s_t} G_{\theta}(s_t|s_{t-1}) \right) \\
&= -\frac{1}{2} \nabla_{\theta} \left(\sum_{s_t} G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1}) (\log G_{\theta}(s_t|s_{t-1}) - \log M_{\phi}(s_t|s_{t-1})) + \sum_{s_{t-1}} G_{\theta}(s_{t-1}) \log \frac{G_{\theta}(s_{t-1})}{M_{\phi}(s_{t-1})} \right) \\
&= -\frac{1}{2} \nabla_{\theta} \left(\sum_{s_t} G_{\theta}(s_{t-1})G_{\theta}(s_t|s_{t-1}) (\log G_{\theta}(s_t|s_{t-1}) - \log M_{\phi}(s_t|s_{t-1})) + \mathbb{E}_{s_{t-1} \sim G_{\theta}} \left[\log \frac{G_{\theta}(s_{t-1})}{M_{\phi}(s_{t-1})} \right] \right) \\
&= -\frac{1}{2} \nabla_{\theta} \left(\sum_{s_{t-1}} G_{\theta}(s_{t-1}) \sum_{s_t} G_{\theta}(s_t|s_{t-1}) (\log G_{\theta}(s_t|s_{t-1}) - \log M_{\phi}(s_t|s_{t-1})) + \mathbb{E}_{s_{t-1} \sim G_{\theta}} \left[\log \frac{G_{\theta}(s_{t-1})}{M_{\phi}(s_{t-1})} \right] \right) \\
&= (12)
\end{aligned}$$

B Further Discussions about the Experiment Results

The Optimal Balance for Cooperative Training We find that the same learning rate and iteration numbers for the generator and mediator seems to be the most competitive choice. As for the architecture choice, we find that the mediator needs to be slightly stronger than the generator. For the best result in the synthetic experiment, we adopt exactly the same generator as other compared models and a mediator whose hidden state size is twice larger (with 64 hidden units) than the generator.

Theoretically speaking, we can and we should sample more batches from G_{θ} and P respectively for training the mediator in each iteration. However, if no regularizations are used when training the mediator, it can easily over-fit, leading the generator's quick convergence in terms of $KL(G_{\theta}||P)$ or NLL_{oracle} , but divergence in terms of $JSD(G_{\theta}||P)$. Empirically, this could be alleviated by applying dropout techniques [Srivastava *et al.*, 2014] with 50% keeping ratio before the output layer of RNN. After applying dropout, the empirical results show good consistency with our theory that, more training batches for the mediator in each iteration is always helpful.

However, applying regularizations is not an ultimate solution and we look forward to further theoretical investigation on better solutions for this problem in the future.