# ESRL: Efficient Sampling-Based Reinforcement Learning for Sequence Generation

**Chenglong Wang[1], Hang Zhou[1], Yimin Hu[1], Yifu Huo[1], Bei Li[1], Tongran Liu[3], Tong Xiao[1,2]* and Jingbo Zhu[1,2]**

[1] School of Computer Science and Engineering, Northeastern University, Shenyang, China
[2] NiuTrans Research, Shenyang, China
[3] CAS Key Laboratory of Behavioral Science, Institute of Psychology, CAS, Beijing, China
{clwang1119, stceum}@gmail.com, {xiaotong, zhujingbo}@mail.neu.edu.cn

## Abstract

Applying Reinforcement Learning (RL) to sequence generation models enables the direct optimization of long-term rewards (*e.g.,* BLEU and human feedback), but typically requires large-scale sampling over a space of action sequences. This is a computational challenge as presented by the practice of sequence generation problems, such as machine translation, where we often deal with a large action space (*e.g.,* a vocabulary) and a long action sequence (*e.g.,* a translation). In this work, we introduce two-stage sampling and dynamic sampling approaches to improve the sampling efficiency during training sequence generation models via RL. We experiment with our approaches on the traditional sequence generation tasks, including machine translation and abstractive summarization. Furthermore, we evaluate our approaches in RL from human feedback (RLHF) through training a large language model using the reward model. Experimental results show that the efficient sampling-based RL, referred to as ESRL, can outperform all baselines in terms of both training efficiency and memory consumption. Notably, ESRL yields consistent performance gains over the strong REINFORCE, minimum risk training, and proximal policy optimization methods. The code is available at https://github.com/wangclnlp/DeepSpeed-Chat-Extension/examples/esrl.

## Introduction

The use of Reinforcement Learning (RL) in training sequence generation models has gained significant attention in recent years. This is primarily due to the fact that sequence generation is inherently a long-term decision-making problem and RL is particularly well-suited for optimizing long-term rewards, such as sequence-level scores (Wieting et al. 2019; Donato et al. 2022) and human feedbacks (Nguyen, Daumé III, and Boyd-Graber 2017; Stiennon et al. 2020; Ouyang et al. 2022; OpenAI 2022). Additionally, by training with sampled sequences, using RL to train sequence generation models can significantly mitigate the *exposure bias* problem (Ranzato et al. 2016; Wang and Sennrich 2020).

The RL training process typically involves two steps: (1) sampling a number of candidate sequences with a pre-trained model given an input (call it *exploration*), and (2) using an RL method, such as REINFORCE (Williams 1992) and Proximal Policy Optimization (PPO) (Schulman et al. 2017), to optimize the model with the long-term rewards given the sampled sequences (call it *optimization*). This paradigm has achieved promising results on several sequence generation tasks, such as machine translation (Wieting et al. 2019; Yehudai et al. 2022; Donato et al. 2022), abstractive summarization (Celikyilmaz et al. 2018; Stiennon et al. 2020), and dialogue generation (Hsueh and Ma 2020). Moreover, it has been proved to have a promising potential for guiding a large language model (LLM) to learn from human feedbacks (Ouyang et al. 2022; OpenAI 2022).

Despite such successes, applying RL to NLP is not low-hanging fruit. In practical applications of sequence generation, we often deal with a large action space (*e.g.,* a vocabulary) and a long action sequence (*e.g.,* a translation). This poses a serious computational challenge to the exploration procedure (Keneshloo et al. 2019), and is an important factor motivating the design of sophisticated sampling approaches.

To mitigate this problem, we investigate strategies for reducing the computational burden of exploration when applying RL to sequence generation models. In this work, we propose an **E**fficient **S**ampling-based **RL** (ESRL) method, which enables more efficient exploration by using the following two approaches. For one, we use a two-stage sampling framework to implement the exploration. It can take full advantage of the Transformer's parallelism computation, so the excessive computational graph storage requirements disappear. Furthermore, we propose a dynamic sampling approach that can reduce redundant sampling by considering the capability of a model. The motivation is that heavy sampling is simply not necessary because pre-trained generation models have already acquired some ability of generation.

We experiment with the proposed ESRL on machine translation and summarization tasks based on Transformer (Vaswani et al. 2017). Experimental results show that ESRL can surpass both the REINFORCE (Williams 1992; Kiegeland and Kreutzer 2021) and minimum risk training (Shen et al. 2016) in terms of generation quality, training time, and memory consumption. Notably, compared to REINFORCE, it can reduce 47% of the memory consumption and 39% of the training time on the machine translation task. Additionally, our ESRL significantly outperforms the vanilla Transformer over 1.04 BLEU points on the IWSLT'14 De-En and WMT'14 En-De test sets. It also significantly outperforms

*Corresponding author.

all baselines on the abstractive summarization task. Furthermore, we evaluate our ESRL in RLHF (Christiano et al. 2017) with LLaMA-7B-LoRA (Hu et al. 2022; Touvron et al. 2023). The results demonstrate that ESRL remains significantly more memory-efficient and faster in RLHF while achieving an improvement of +30.00 points on the total score of Vicuna-80 benchmark, as evaluated by GPT-4 (Chiang et al. 2023), compared to the robust PPO.

## Related Work

While reinforcement learning (RL) has long been appreciated in robotics and other fields, it has recently emerged as a promising approach to advance sequence generation models (Ranzato et al. 2016; Celikyilmaz et al. 2018; Yehudai et al. 2022; Donato et al. 2022). For example, Edunov et al. (2018) compared objective functions commonly used in RL for sequence generation models. Choshen et al. (2020) and Kiegeland and Kreutzer (2021) examined the limitations of RL in neural machine translation. Moreover, Kiegeland and Kreutzer (2021) conducted experiments on in-domain and cross-domain adaptation setups to highlight the significance of exploration during RL training. It is also an upward trend in using RL to train large language models with human feedbacks (Nguyen, Daumé III, and Boyd-Graber 2017; Stiennon et al. 2020; Ouyang et al. 2022; OpenAI 2022).

As another line of research, the researchers focused on exploring better reward functions to enhance the learning of generation models, such as the use of semantic similarity (Li et al. 2016; Wieting et al. 2019; Yasui, Tsuruoka, and Nagata 2019) and the design of learnt reward functions (Shi et al. 2018; Böhm et al. 2019; Shu, Yoo, and Ha 2021). More recent work aimed at addressing the challenge of large action spaces in sequence generation models (Hashimoto and Tsuruoka 2019; Yehudai et al. 2022).

Although previous work improves the performance of RL on sequence generation tasks, they are often hindered by the inefficient exploration problem. Researchers have been aware of this (Keneshloo et al. 2019), but it is still rare to see studies on this issue.

## Our Method

In this section, we firstly recall the preliminaries of using RL in training sequence generation models. Then, we present our efficient sampling-based RL method. Last, we introduce our optimization algorithm.

### Preliminaries

**Sequence Generation Model** Given an input $x$ such as a text, a sequence generation model generates a sequence of $N$ tokens $y = \{y_1, \ldots, y_N\}$, where each token $y_t$ is drawn from a vocabulary. At the training stage, the model learns the probability:

$$p_\theta(y|x) = \prod_{t=1}^{N} p_\theta(y_t|y_{<t}, x) \tag{1}$$

where $y_{<t}$ is the prefix $\{y_1, y_2, \ldots, y_{t-1}\}$, and $\theta$ is a set of model parameters. In this process, the standard training objective is to maximize the likelihood over all the tokens of the target sequence, *i.e., maximum likelihood estimation (MLE)* (Myung 2003). At the inference stage, we generate tokens sequentially according to probability $p_\theta$. In this paper, we consider the tasks of neural machine translation, abstractive summarization, and RLHF and use them as instances of the above model.

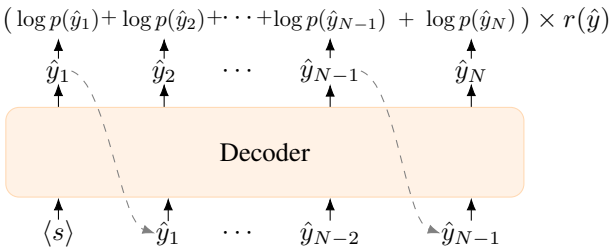



Figure 1: An illustration of the traditional RL loss calculation. In this process, we need to store each of the computational graphs produced by $\{p(\hat{y}_1), p(\hat{y}_2), \cdots, p(\hat{y}_N)\}$ to calculate the gradients. Thus, the memory footprint grows drastically as the sampled sequence becomes longer.

**Long-term Reward Optimization** Given a pre-trained sequence generation model, we can use RL to train this model. RL seeks to maximize the long-term reward, written as $\arg\max_\theta \mathbb{E}p_\theta(\hat{y}|x)[r(\hat{y})]$, where $\hat{y}$ is a generated sequence and $r(\cdot)$ is a reward function computing the long-term reward for $\hat{y}$. $r(\cdot)$ is typically defined to be a standard metric function, such as BLEU (Papineni et al. 2002) and ROUGE (Lin 2004). The corresponding RL loss for this training instance is then given by:

$$\mathcal{L}_{\mathrm{RL}} = \sum_{\hat{y} \in \Omega(x)} p_\theta(\hat{y}|x) r(\hat{y}) \tag{2}$$

where $\Omega(x)$ is the output space which comprises all possible candidate target sequences for input $x$.

**Exploration** However, computation of Eq. 2 is intractable because the size of $\Omega(x)$ grows exponentially with the size of the vocabulary and the lengths of the target sequences. To address this challenge, RL usually performs exploration to approximate $\Omega(x)$. A commonly-used method to solve Eq. 2 is the Monte Carlo method (Williams 1992). For each training instance, a number of sequences are sampled from a multinomial distribution defined by a Softmax layer with a temperature factor (Choshen et al. 2020). Here, both the sampling size and the sampling temperature can be used to control to what extent we explore the space. For example, a larger sampling size means more candidates involved in sampling, and a higher temperature means a larger diversity of sampled sequences (Kiegeland and Kreutzer 2021).

**Policy Gradient** To optimize the model with the long-term rewards of sampled sequences, policy gradient methods, such as REINFORCE (Williams 1992) and minimum risk training (MRT) (Shen et al. 2016), are often used.

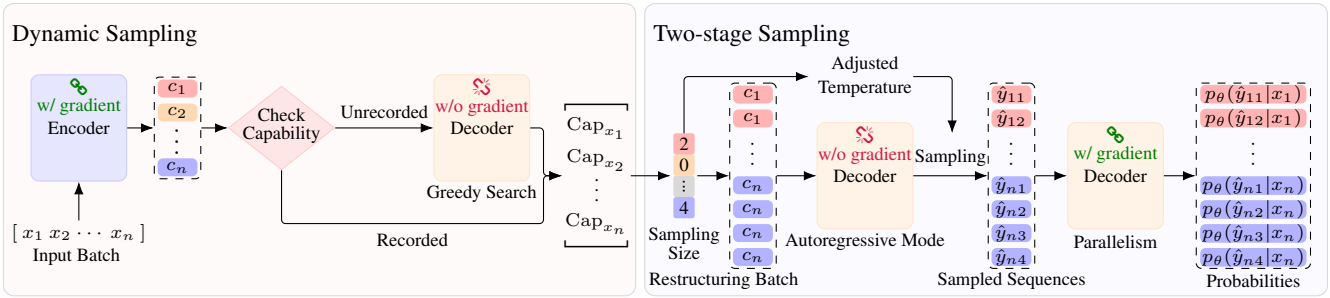



Figure 2: Architecture of ESRL. We introduce two-stage sampling and dynamic sampling approaches to design ESRL, which enables it to be much more memory-efficient and much faster in training a sequence generation model. For two-stage sampling, we take full advantage of the Transformer's parallelism computation to avoid the excessive computational graph storage. During the dynamic sampling, based on the estimated capability, we dynamically adjust the size and temperature of sampling to eliminate unnecessary exploration. Here encoder portion is used only by the encoder-decoder sequence generation model.

Specifically, REINFORCE uses log derivatives to define the loss function:

$$\mathcal{L}_{\text{REINFORCE}} = -\sum_{\hat{y}\in S(x)} \log p_\theta(\hat{y}|x) r(\hat{y}) \tag{3}$$

where $S(x)$ is an approximated space, which consists of these sampled sequences. The calculation process is also illustrated in Figure 1. Since each sequence is sampled by an autoregressive mode (Vaswani et al. 2017; Xiao and Zhu 2023), RL requires the storage of computational graphs on the order of $N$ times that of MLE training.

Unlike REINFORCE, MRT method uses these sampled sequences to approximate a posterior distribution with renormalization:

$$Q_\theta(\hat{y}|x) = \frac{p_\theta(\hat{y}|x)^\alpha}{\sum_{\hat{y}\in S(x)} p_\theta(\hat{y}|x)^\alpha} \tag{4}$$

where $\alpha$ is a smoothness parameter and $Q_\theta(\hat{y}|x)$ is a distribution defined on the approximated space. Based on the $Q_\theta$ distribution, MRT gives a new loss function:

$$\mathcal{L}_{\text{MRT}} = \sum_{\hat{y}\in S(x)} Q_\theta(\hat{y}|x)[-r(\hat{y})] \tag{5}$$

In some cases, MRT can achieve better performance compared with REINFORCE (Kiegeland and Kreutzer 2021). But the exploration process of MRT requires an enormous amount of memory to store the computational graphs used in renormalization.

## Efficient Sampling-based RL (ESRL)

In this work, our aim is to reduce the computational cost of applying RL to sequence generation models. We propose the ESRL to achieve this. The overview of ESRL is depicted in Figure 2. As shown in the figure, we present two-stage sampling and dynamic sampling in ESRL achieve our goal. In the following subsections, we will describe them in detail.

**Two-stage Sampling** In response to excessive computational graph storage requirements produced by the sampling process, we use a two-stage framework that effectively mitigates this issue. Stage one is to sample the candidate sequences via an autoregressive mode. Note that this stage is not involved in backpropagation. It thus does not require the storage of computational graphs. Stage two is to calculate the probabilities of the sampled candidate sequences, *i.e.,* $p_\theta(\hat{y}|x)$ in Eqs. 3 and 4. At this stage, due to the presence of the complete output sequence, we can use Transformer's parallelism computation instead of an autoregressive mode. It allows this calculation to be done with just one forward pass. Compared to the conventional RL sampling, the two-stage sampling incurs additional time costs due to an extra forward pass. However, with the help of two-stage sampling, it can effectively reduce the memory footprint. Generally, the conventional RL sampling needs to store the computational graphs of $N$ forward passes, while the two-stage sampling only stores the computational graph of one forward pass. It is noteworthy that the two-stage sampling approach has been adopted in several open-source projects, such as `TRL`* and `TRLX`†, attributable to its training efficiency.

**Dynamic Sampling** We propose a dynamic sampling approach to further improving the efficiency of RL training. In our dynamic sampling approach, we first estimate the model capability, then adjust the sampling size and temperature according to this estimated capability so that we can perform sampling in an adequate and efficient way.

For the model capability estimation, we reuse old sequences sampled at the previous epoch. Specifically, given an input $x$, following the sampling of candidate sequences, we employ these sampled sequences to estimate the model's generation capability of the input. Then, the estimated model capability is then recorded and used in the subsequent epoch to adjust the sampling size for the same input. Taking the machine translation task as an instance, we use the entropy (Settles 2009) and BLEU (Papineni et al. 2002) to estimate the model capability. When using BLEU to estimate the

*https://github.com/huggingface/trl
†https://github.com/CarperAI/trlx

model capability, the capability score is given by:

$$Cap_x = \frac{1}{m} \sum_{\hat{y} \in S(x)} \mathrm{BLEU}(\hat{y}, y) \tag{6}$$

where $m$ is the sampling size of the input $x$ and $\mathrm{BLEU}(\cdot)$ is the *sacre*BLEU (Post 2018). When considering entropy as another estimation of the model capability, written as:

$$Cap_x = 1 + \frac{1}{N \times m} \sum_{\hat{y} \in S(x)} \sum_{t=1}^{N} p_\theta(\hat{y}_t | \hat{y}_{<t}, x) \log p_\theta(\hat{y}_t | \hat{y}_{<t}, x) \tag{7}$$

There are other choices to define $Cap_x$ for specific tasks. For instance, we can replace BLEU with ROUGE (Lin 2004) in the abstractive summarization task. Note that when the model's capability for a given input $x$ is not recorded, *i.e.,* no sampling operation has been performed on the input, we employ a greedy search algorithm to generate an optimal sequence quickly. Then we use this generated sequence to estimate the model capability.

For the sampling size adjustment, our main aim is to eliminate unnecessary exploration. Specifically, when $Cap_x$ is high, we consider that the model has ability to get a great long-term reward and thus decrease the sampling size. By contrast, when $Cap_x$ is low, we increase the sampling size to have a larger-scale exploration. It allows the model to learn from a sufficient number of possible generated sequences per input and to improve its own capability. Here we use the following function to achieve this goal:

$$k_x = \lceil k_{max} - \beta \cdot n \cdot k_{max} \cdot \frac{Cap_x}{\sum_{x \in I} Cap_x} \rceil \tag{8}$$

where $k_x$ and $k_{max}$ denote the adjusted sampling size and the maximum sampling size, respectively. $\beta$ is a ratio of eliminated samples within the range of [0, 1), relative to the total number of samples. Here we use batch-level elimination strategy that reduces the sampling size of the input with higher capability score within the current batch's distribution. Thus $I$ denotes an input set consisting of all inputs in the current batch and $n$ denotes the number of inputs.

Considering that the sampling temperature also impacts the exploration, we adopt a simple strategy to control the exploration by adjusting the temperature: when the capacity score is low, we use a higher temperature to encourage exploration. We dynamically adjust the temperature in the interval $[\tau_{min}, \tau_{max}]$ based on the adjusted sampling size to further control the exploration. The rule of temperature adjustment is given by:

$$\tau_x = \tau_{min} + k_x \times \frac{\tau_{max} - \tau_{min}}{k_{max}} \tag{9}$$

where $\tau_x$ is the adjusted temperature for $x$.

After adjusting the size and temperature of sampling, we sample $k_x$ candidate sequences for each input. Following Kiegeland and Kreutzer (2021)'s work, we use a restructuring batch trick which restructures a new batch by repeating the encoder representations to act as the input of the decoder (see Figure 2), to take advantage of the parallel computation.

## Optimization

We replace the standard policy method with the fusion of MRT and REINFORCE in computing the loss. Specifically, we use $\mathcal{L}_{\mathrm{MRT}}$ to serve as the loss when $k_x > 1$. When $k_x = 1$, since the renormalization is not feasible, we instead use $\mathcal{L}_{\mathrm{REINFORCE}}$ to serve as the loss. This design combines the strengths of MRT and REINFORCE and makes full use of the sampled sequences to optimize the model.

**FIFO-based Baseline Reward** The *baseline reward* technique (Sutton and Barto 2018) has been shown to be effective to improve the generalization of sequence generation models (Kreutzer, Sokolov, and Riezler 2017). The ideal baseline value is an average of the long-term rewards of all possible candidate sequences. Again, this is intractable because there is an exponentially large number of candidate sequences in sequence generation tasks. Although some works attempt to estimate this ideal baseline value (Hashimoto and Tsuruoka 2019), they involve complex training. Inspired by the idea of using a queue to proxy the global in Wang et al. (2021)'s work, we propose a FIFO-based baseline reward approach, which employs a First-In-First-Out (FIFO) reward queue $\mathcal{Q}$ to compute the baseline value. We use $\mathcal{Q}_{size}$ to denote the reward queue size. At each training step, we push rewards of all sampled sequences into $\mathcal{Q}$ and pop out the 'Oldest' rewards. Then we compute the average of the rewards in $\mathcal{Q}$ to serve as the baseline value $b$. By using this baseline reward, we replace the reward function in Eqs. 3 and 4 with $r(\hat{y}, y) - b$.

# Experiments

We evaluated our ESRL method on the traditional sequence generation tasks, including machine translation and abstractive summarization. We also evaluated ESRL in RLHF with LLaMA-7B-LoRA.

## Experimental Setups

**Datasets** The datasets used for each task are as follows:

- *Machine Translation*: We conducted experiments on two machine translation datasets, including a small-scale IWSLT'14 German-English (De-En) dataset and a large-scale WMT'14 English-German (En-De) dataset. We preprocessed the datasets following the same setup in Hu et al. (2021)'s work.
- *Abstractive Summarization*: We also tested the ESRL's capability to train the abstractive summarization model on the CNN/DM dataset (Hermann et al. 2015). Our data preprocess method was the same as in Li et al. (2022).
- *RLHF*: We integrated data from Alpaca data (52k training instances) and GPT-4 Alpaca data (Peng et al. 2023; Taori et al. 2023) to perform the supervised fine-tuning (SFT) and RLHF. We used the GPT-4 Comparison English dataset[‡] to train our reward model.

[‡]https://github.com/Instruction-Tuning-with-GPT-4/GPT-4-LLM

| Method | SS | IWSLT'14 De-En | | | | WMT'14 En-De | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | BLEU | COMET-22 | Time (hours) | Memory (G) | BLEU | COMET-22 | Time (hours) | Memory (G) |
| MLE | - | 33.77 | 79.32 | - | - | 26.73 | 83.36 | - | - |
| REINFORCE | 1 | 33.91 | 79.52 | 4.52 | 3.31 | 26.97 | 83.45 | 7.70 | 5.32 |
| ESRL-Random | 1 | 33.71 | 79.47 | 2.69 | 2.63 | 26.85 | 83.38 | 6.45 | 4.84 |
| ESRL-BLEU | 1 | **34.02** | **79.62** | 3.15 | **2.54** | **27.02** | **83.55** | **6.19** | 4.55 |
| ESRL-Entropy | 1 | 33.96 | 79.56 | **2.73** | 2.66 | 26.95 | 83.43 | 6.26 | **4.13** |
| REINFORCE | 5 | 34.05 | 79.58 | 7.04 | 8.66 | 27.10 | 83.52 | 12.68 | 13.83 |
| MRT | 5 | 34.17 | 79.66 | 8.96 | 16.60 | 27.12 | 83.60 | 13.76 | 14.93 |
| ESRL-Random | 5 | 33.68 | 79.23 | **3.73** | 5.26 | 26.87 | 83.41 | 10.34 | 12.39 |
| ESRL-BLEU | 5 | 34.37 | 79.81 | 4.34 | 5.49 | **27.25** | **83.68** | 11.14 | 11.79 |
| ESRL-Entropy | 5 | **34.40** | **79.85** | 3.99 | **5.19** | 27.18 | 83.59 | **10.29** | **11.75** |
| REINFORCE | 10 | 34.22 | 79.63 | 8.19 | 15.61 | 27.21 | 83.72 | 15.20 | 20.85 |
| MRT | 10 | 34.31 | 79.71 | 10.26 | 23.03 | 27.26 | 83.80 | 16.90 | 22.14 |
| ESRL-Random | 10 | 33.75 | 79.31 | **4.63** | 10.15 | 27.05 | 83.52 | 13.26 | 15.87 |
| ESRL-BLEU | 10 | 34.53 | 80.02 | 4.85 | **9.86** | **27.43** | 83.87 | **13.03** | 16.12 |
| ESRL-Entropy | 10 | **34.63** | **80.13** | 5.11 | 10.02 | 27.39 | **83.90** | 13.82 | **15.58** |
| REINFORCE | 15 | 34.41 | 79.88 | 9.91 | 22.25 | - | - | - | >24.00 |
| MRT | 15 | - | - | - | >24.00 | - | - | - | >24.00 |
| ESRL-Random | 15 | 33.61 | 79.12 | 6.91 | 11.86 | 27.22 | 83.71 | 15.53 | 17.15 |
| ESRL-BLEU | 15 | **34.79** | 80.24 | 6.10 | 12.53 | **27.54** | **83.98** | **15.46** | **16.97** |
| ESRL-Entropy | 15 | 34.68 | **80.33** | **6.03** | **11.78** | 27.45 | 83.93 | 16.08 | 17.72 |
| REINFORCE | 20 | - | - | - | >24.00 | - | - | - | >24.00 |
| MRT | 20 | - | - | - | >24.00 | - | - | - | >24.00 |
| ESRL-Random | 20 | 33.78 | 79.34 | 7.25 | 17.16 | 27.18 | 83.68 | **20.13** | 21.09 |
| ESRL-BLEU | 20 | **34.95** | **80.56** | **7.04** | **16.35** | **27.67** | **84.12** | 21.36 | **19.45** |
| ESRL-Entropy | 20 | 34.83 | 80.42 | 7.32 | 16.98 | 27.58 | 84.05 | 21.65 | 20.76 |

Table 1: Results on the machine translation task using different sampling sizes. The best results for each group are in bold. The suffix "-Random", "-BLEU", and "-Entropy" denote that we use random-based, BLEU-based, and entropy-based strategies to adjust the sampling size, respectively. SS: sampling size; Time: training time; Memory: maximum memory consumption.

**Setups** For machine translation and abstractive summarization tasks, we pre-trained a standard Transformer base model (Vaswani et al. 2017) using the MLE until convergence. Here we employed BLEU and ROUGE-L as the reward functions during RL training. For RLHF, we fine-tuned a LLaMA-7B model using LoRA approach. Following Ouyang et al. (2022)'s work, we trained a reward model using a LLaMA-7B model to predict rewards during RL training. More training setups are shown in our arXiv version[§].

**Evaluation Metrics** We measured the translation quality in terms of BLEU. Here, we employed *sacre*BLEU to calculate the BLEU scores. We measured the summary quality by calculating ROUGE-L scores for the CNN/DM dataset. To further evaluate the performance of the model, two model-based metrics, COMET-22 (Rei et al. 2022) and BARTScore (Yuan, Neubig, and Liu 2021), were employed for measuring machine translation and summarization tasks, respectively. Additionally, we used Vicuna-80 benchmark[¶] to evaluate the performance of RLHF, where the scores were assessed by GPT-4 following Zheng et al. (2023)'s work. For training efficiency and memory consumption, we tested ESRL on four TITAN RTX GPUs. Specifically, we used a global batch size (per GPU) of 1,024 tokens, 2048 tokens, and 4 samples for the machine translation, abstractive summarization, and RLHF, respectively. We also used the restructuring batch in MRT to make a fair comparison.

**Baselines** Our baseline is the standard **MLE**. Additionally, we compare ESRL with commonly used sampling-based (on-policy) RL methods, including **REINFORCE** (Ranzato et al. 2016) and **MRT** (Shen et al. 2016), across various sampling sizes. For REINFORCE, following Kiegeland and Kreutzer (2021), we implemented it using the moving average baseline with the temperature $\tau = 0.95$. In RLHF, we compare ESRL with the standard **SFT** and **PPO**. We also chose **ESRL-Random** method as an additional baseline to evaluate the effectiveness of ESRL. In ESRL-Random, we randomly adjusted the size and temperature of sampling during dynamic sampling. Furthermore, we compare with off-policy RL methods, including **GOLD-$s$** and **GOLD-$p$** (Pang and He 2021), as shown in Table 5.

## Experimental Results

**Results of Machine Translation** Figure 1 summarizes the results of machine translation. In terms of training time and

[§]https://arxiv.org/abs/2308.02223

[¶]https://lmsys.org/blog/2023-03-30-vicuna/

| Method | SS | RG-L | BS | Time (hours) | Memory (G) |
|---|---|---|---|---|---|
| MLE | - | 37.06 | -1.65 | - | - |
| REINFORCE | 1 | 37.58 | -1.54 | 4.38 | 9.75 |
| ESRL-Random | 1 | 37.23 | -1.63 | 2.52 | 4.14 |
| ESRL-ROUGE | 1 | **37.72** | **-1.48** | **2.46** | 4.26 |
| ESRL-Entropy | 1 | 37.64 | -1.50 | 2.58 | **4.01** |
| REINFORCE | 5 | - | - | - | >24.00 |
| MRT | 5 | - | - | - | >24.00 |
| ESRL-Random | 5 | 37.38 | -1.61 | 4.05 | 6.72 |
| ESRL-ROUGE | 5 | **38.13** | **-1.42** | **3.87** | **6.59** |
| ESRL-Entropy | 5 | 37.98 | -1.46 | 4.28 | 7.02 |

Table 2: Results on the abstractive summarization task. RG-L: ROUGE-L; BS: BARTScore.

memory consumption, our ESRL consistently outperforms REINFORCE and MRT on different sampling sizes. For instance, ESRL can reduce about 47% of memory consumption and 39% of training time on training IWSLT model with a sampling size of 15. It demonstrates that ESRL can efficiently achieve RL training on the machine translation task, while also showing its ability to conduct larger-scale sampling with identical settings on resource-constrained devices. In terms of translation quality, ESRL achieves the best result in training translation models compared to all the baselines. Notably, ESRL yields a +0.98 BLEU improvement on the WMT En-De dataset compared to MLE, when using the sampling size of 20. Compared to REINFORCE and MRT, our ESRL can also gain a better translation quality. We attribute this to the fact that ESRL benefits from the appropriate exploration obtained by the dynamic sampling at each training step (see an analysis from Section **Balancing Exploration and Exploitation**).

**Results of Abstractive Summarization** We also evaluated the proposed ESRL on the abstractive summarization task. The results are presented in Table 2. We can see that ESRL outperforms MLE by a large margin (*e.g.,* 1.07 ROUGE-L and 0.23 BARTScore benefits). Due to the excessively long input (*i.e.,* an article), REINFORCE and MRT necessitate a huge training footprint to train a summarization model while sampling multiple sequences. However, in this case, ESRL still achieves an efficient RL training as the sampling process receives benefits from both two-stage sampling and dynamic sampling approaches.

**Results of RLHF** As shown in Table 3, we evaluated our ESRL in RLHF with a sampling size of 1. The experimental results indicate that compared to the conventional PPO, our ESRL can still be more memory-efficient and faster in RLHF. Notably, ESRL-Entropy can outperform SFT by a substantial margin of 56.00 points on Vicuna-80's total score. Additionally, compared to the PPO with two-stage sampling, our ESRL method yields an improvement of +30.00 points. This result demonstrates that our dynamic sampling approach not only improves the training efficiency but also contributes to the generation quality in RLHF.

| Method | Score | Time (hour) | Memory (G) |
|---|---|---|---|
| SFT | 560.00 | - | - |
| PPO | - | - | >24.00 |
| PPO w/ TS | 596.00 | 13.87 | 23.14 |
| ESRL-Random | 571.00 | 10.81 | 19.57 |
| ESRL-Reward | 619.00 | 10.53 | **19.36** |
| ESRL-Entropy | **626.00** | **10.19** | 20.03 |

Table 3: Vicuna benchmark's total scores evaluated by GPT-4. "-Reward" denotes that we use the predicted reward score to estimate model capability. TS: two-stage sampling.

Furthermore, compared to ESRL-Random, we observe that ESRL-BLEU, ESRL-Reward, and ESRL-Entropy can achieve better generation quality on all tasks. This finding illustrates that adjustments based on model capacity are superior to those made randomly. Additionally, we investigate the performance gain of different capacity estimation strategies. From the results, we find that both the entropy-based estimation and the BLEU/ROUGE/Reward-based estimation can contribute to the generation quality improvement over the baselines on all tasks.

## Ablation Study

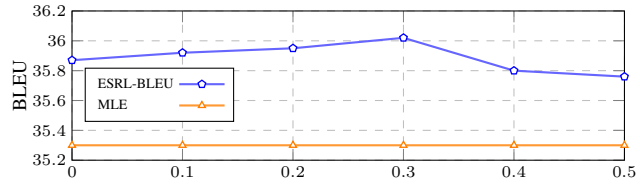
We present detailed ablation studies to explore effects of two-stage sampling, dynamic sampling, and FIFO-based baseline reward with a sampling size of 20. The experiments were conducted on the IWSLT dataset, and the impacts of removing each approach are thoroughly examined. The results are summarized in Table 4. Through the results, we see that the two-stage sampling approach can significantly reduce the training time cost and memory consumption, which makes it feasible to RL training on resource-constrained devices. We also see that without dynamic sampling, ESRL fails to gain a well-performing translation model. Furthermore, to investigate the impact of temperature adjustment, we attempt to employ ESRL to train a translation model with removing this factor, specifically by solely adjusting the sampling size during the dynamic sampling process. The results show that temperature adjustment can improve generation quality without bringing additional computational costs. Additionally, we see that using the FIFO-based baseline reward can train a better model. It shows the effectiveness of using FIFO to compute baseline value.

## Analysis

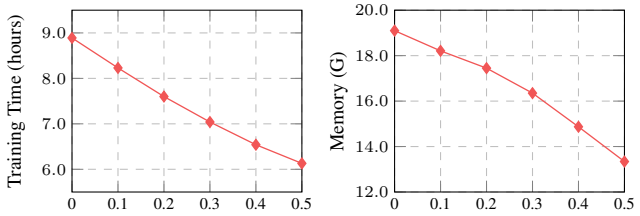
**Performance on Different Elimination Ratios** Based on the two-stage sampling with a sampling size of 20, we investigate the impact of using different elimination ratios. Figure 3 (top) compares ESRL-BLEU with MLE on the IWSLT dataset. We see that ESRL can achieve consistent BLEU improvements across various elimination ratios. Additionally, the results show an interesting observation that the elimination operation may bring certain benefits to our ESRL in terms of BLEU. We attempt to give a potential cause for this observation from the perspective of balancing exploration and exploitation. Figure 3 (bottom) shows the results

| Method | BLEU | COMET-22 | Time (hours) | Memory (G) |
|---|---|---|---|---|
| MLE | 35.30 | 80.88 | - | - |
| ESRL | 36.34 | 82.29 | 7.04 | 16.35 |
| w/o TS | - | - | - | >24.00 |
| w/o DS | 35.87 | 81.27 | 8.89 | 19.10 |
| w/o TA | 36.02 | 81.55 | 7.13 | 16.42 |
| w/o FBR | 36.17 | 82.01 | 7.02 | 15.87 |

Table 4: Ablation studies on the components of ESRL. The translation quality is tested on the IWSLT development set. DS: dynamic sampling; TA: temperature adjustment; FBR: FIFO-based baseline reward.




(a) Performance Comparison




(b) Efficiency Comparison

Figure 3: The comparison of performance and efficiency against different elimination ratios: 0, 0.1, 0.2, 0.3, 0.4, 0.5.

for training time and memory consumption using different elimination ratios. From the results, we can observe that our elimination operation can progressively diminish training time and memory consumption usage as increasing the elimination ratios. Considering the impact on BLEU and efficiency, we choose the elimination ratio of 0.3 to conduct our all experiments.

**Effect of Temperature Interval on Performance** We study the impact of using different temperature intervals. As shown in Figure 4 (left), we swept over different intervals: $\{[0.2, 0.6], [0.4, 0.8], [0.6, 1.0], [0.8, 1.2]\}$. From the results, we see that the use of different temperature intervals can result in different performance gains. We find that the optimal temperature interval is $[0.6, 1.0]$ which makes an appropriate diversity in the sampled sequences.

**Comparison with Off-policy RL Methods** Table 5 shows the performance of off-policy RL method on the IWSLT dataset. We can observe that ESRL is still better than strong GOLD (Pang and He 2021) under the evaluation of various metrics. Furthermore, our ESRL is orthogonal to the off-policy RL method. Here, we take GOLD-$s$ as an instance.

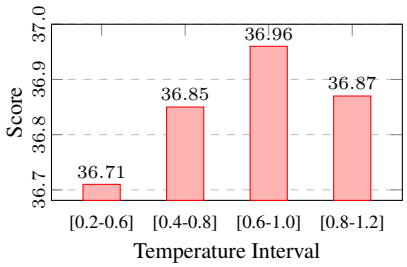



Figure 4: Performance of ESRL with different temperature intervals on the IWSLT dataset.

| Method | RL Type | BLEU | COMET-22 |
|---|---|---|---|
| MLE | - | 33.77 | 79.32 |
| ESRL | On-policy | 34.95 | 80.56 |
| GOLD-$p$ | Off-policy | 34.21 | 79.83 |
| GOLD-$s$ | Off-policy | 34,33 | 80.11 |
| ESRL+GOLD-$s$ | (On+Off)-policy | **35.12** | **80.82** |

Table 5: Performance on the IWSLT test set, using standard models trained with off-policy objectives.

Specifically, we first train a translation model with ESRL, and then use the trained model to perform GOLD-$s$ procedure. The experimental results show that the combined method can achieve superior performance.

**Balancing Exploration and Exploitation** Balancing exploration and exploitation has been proven to improve RL in the planning problem (Tokic 2010; Sutton and Barto 2018; Jiang and Lu 2020). Here, we aim to illustrate that our ESRL method can outperform all baseline approaches by effectively achieving a balance between exploration and exploitation, resulting in enhanced performance. When the model has a strong capacity and obtains high deterministic rewards, our ESRL exploits and reduces exploration as much as possible, *i.e.,* reducing the size and temperature of sampling. This allows the model to make full use of the current learned knowledge for decision-making and optimization. Instead, when the model has a weak capacity, ESRL increases the size and temperature of sampling to enhance exploration, which gathers more possible generated sequences to optimize the model. Thus, compared to baselines, using dynamic sampling approach enables ESRL to balance exploration and exploitation well and achieve better performance.

## Conclusion

In this paper, we focus on reducing the computational cost of RL training in sequence generation models. We have proposed an efficient sampling-based RL method (referred to as ESRL) via two-stage sampling and dynamic sampling approaches. Our extensive experiments show that our ESRL significantly outperforms all baselines in terms of both training efficiency and generation quality.

## Acknowledgments

This work was supported in part by the National Science Foundation of China (No.62276056), the National Key R&D Program of China, the Natural Science Foundation of Liaoning Province of China (2022-KF-16-01), the Fundamental Research Funds for the Central Universities (Nos. N2216016, N2216001, and N2216002), and the Program of Introducing Talents of Discipline to Universities, Plan 111 (No.B16009).

## References

Böhm, F.; Gao, Y.; Meyer, C. M.; Shapira, O.; Dagan, I.; and Gurevych, I. 2019. Better Rewards Yield Better Summaries: Learning to Summarise Without References. In *Proc. of EMNLP*.

Celikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep Communicating Agents for Abstractive Summarization. In *Proc. of NAACL*.

Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; Stoica, I.; and Xing, E. P. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality.

Choshen, L.; Fox, L.; Aizenbud, Z.; and Abend, O. 2020. On the Weaknesses of Reinforcement Learning for Neural Machine Translation. In *Proc. of ICLR*.

Christiano, P. F.; Leike, J.; Brown, T. B.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep Reinforcement Learning from Human Preferences. In *Proc. of NeurIPS*.

Donato, D.; Yu, L.; Ling, W.; and Dyer, C. 2022. MAD for Robust Reinforcement Learning in Machine Translation. *ArXiv preprint*.

Edunov, S.; Ott, M.; Auli, M.; Grangier, D.; and Ranzato, M. 2018. Classical Structured Prediction Losses for Sequence to Sequence Learning. In *Proc. of NAACL*.

Hashimoto, K.; and Tsuruoka, Y. 2019. Accelerated Reinforcement Learning for Sentence Generation by Vocabulary Prediction. In *Proc. of NAACL*.

Hermann, K. M.; Kociský, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching Machines to Read and Comprehend. In *Proc. of NeurIPS*.

Hsueh, C.-H.; and Ma, W.-Y. 2020. Semantic Guidance of Dialogue Generation with Reinforcement Learning. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Hu, C.; Wang, C.; Ma, X.; Meng, X.; Li, Y.; Xiao, T.; Zhu, J.; and Li, C. 2021. RankNAS: Efficient Neural Architecture Search by Pairwise Ranking. In *Proc. of EMNLP*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proc. of ICLR*.

Jiang, J.; and Lu, Z. 2020. Generative Exploration and Exploitation. In *Proc. of AAAI*.

Keneshloo, Y.; Shi, T.; Ramakrishnan, N.; and Reddy, C. 2019. Deep Reinforcement Learning for Sequence-to-Sequence Models. *IEEE Transactions on Neural Networks and Learning Systems*.

Kiegeland, S.; and Kreutzer, J. 2021. Revisiting the Weaknesses of Reinforcement Learning for Neural Machine Translation. In *Proc. of NAACL*.

Kreutzer, J.; Sokolov, A.; and Riezler, S. 2017. Bandit Structured Prediction for Neural Sequence-to-Sequence Learning. In *Proc. of ACL*.

Li, B.; Zheng, T.; Jing, Y.; Jiao, C.; Xiao, T.; and Zhu, J. 2022. Learning Multiscale Transformer Models for Sequence Generation. In *Proc. of ICML*.

Li, J.; Monroe, W.; Ritter, A.; Jurafsky, D.; Galley, M.; and Gao, J. 2016. Deep Reinforcement Learning for Dialogue Generation. In *Proc. of EMNLP*.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*.

Myung, I. J. 2003. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*.

Nguyen, K.; Daumé III, H.; and Boyd-Graber, J. 2017. Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback. In *Proc. of EMNLP*.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *ArXiv preprint*.

Pang, R. Y.; and He, H. 2021. Text Generation by Learning from Demonstrations. In *Proc. of ICLR*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*.

Peng, B.; Li, C.; He, P.; Galley, M.; and Gao, J. 2023. Instruction tuning with gpt-4. *ArXiv preprint*.

Post, M. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2016. Sequence Level Training with Recurrent Neural Networks. In *Proc. of ICLR*.

Rei, R.; C. de Souza, J. G.; Alves, D.; Zerva, C.; Farinha, A. C.; Glushkova, T.; Lavie, A.; Coheur, L.; and Martins, A. F. T. 2022. COMET-22: Unbabel-IST 2022 Submission for the Metrics Shared Task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *ArXiv preprint*.

Settles, B. 2009. Active learning literature survey.

Shen, S.; Cheng, Y.; He, Z.; He, W.; Wu, H.; Sun, M.; and Liu, Y. 2016. Minimum Risk Training for Neural Machine Translation. In *Proc. of ACL*.

Shi, Z.; Chen, X.; Qiu, X.; and Huang, X. 2018. Toward Diverse Text Generation with Inverse Reinforcement Learning. In *Proc. of IJCAI*.

Shu, R.; Yoo, K. M.; and Ha, J.-W. 2021. Reward optimization for neural machine translation with learned metrics. *ArXiv preprint*.

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D. M.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. F. 2020. Learning to summarize with human feedback. In *Proc. of NeurIPS*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford alpaca: An instruction-following llama model.

Tokic, M. 2010. Adaptive $\varepsilon$-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *ArXiv preprint*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Proc. of NeurIPS*.

Wang, C.; and Sennrich, R. 2020. On Exposure Bias, Hallucination and Domain Shift in Neural Machine Translation. In *Proc. of ACL*.

Wang, F.; Yan, J.; Meng, F.; and Zhou, J. 2021. Selective Knowledge Distillation for Neural Machine Translation. In *Proc. of ACL*.

Wieting, J.; Berg-Kirkpatrick, T.; Gimpel, K.; and Neubig, G. 2019. Beyond BLEU:Training Neural Machine Translation with Semantic Similarity. In *Proc. of ACL*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Xiao, T.; and Zhu, J. 2023. Introduction to Transformers: an NLP Perspective. *ArXiv preprint*.

Yasui, G.; Tsuruoka, Y.; and Nagata, M. 2019. Using Semantic Similarity as Reward for Reinforcement Learning in Sentence Generation. In *Proc. of ACL*.

Yehudai, A.; Choshen, L.; Fox, L.; and Abend, O. 2022. Reinforcement Learning with Large Action Spaces for Neural Machine Translation. In *Proc. of COLING*.

Yuan, W.; Neubig, G.; and Liu, P. 2021. BARTScore: Evaluating Generated Text as Text Generation. In *Proc. of NeurIPS*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *ArXiv preprint*.