EFFICIENT DIALOG POLICY LEARNING VIA POSITIVE MEMORY RETENTION

Rui Zhao, Volker Tresp

Ludwig Maximilian University, Oettingenstr. 67, 80538 Munich, Germany

Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany

ABSTRACT

This paper is concerned with the training of recurrent neural networks as goal-oriented dialog agents using reinforcement learning. Training such agents with policy gradients typically requires a large amount of samples. However, the collection of the required data in form of conversations between chat-bots and human agents is time-consuming and expensive. To mitigate this problem, we describe an efficient policy gradient method using positive memory retention, which significantly increases the sample-efficiency. We show that our method is 10 times more sample-efficient than policy gradients in extensive experiments on a new synthetic number guessing game. Moreover, in a real-word visual object discovery game, the proposed method is twice as sample-efficient as policy gradients and shows state-of-the-art performance.

***Index Terms*—** Goal-Oriented Dialog System, Deep Reinforcement Learning, Recurrent Neural Network

1. INTRODUCTION

In recent years, advances in Deep Learning (DL) and Reinforcement Learning (RL) have led to tremendous progress across many areas of natural language processing (NLP) and gameplay [1, 2, 3, 4, 5]. This progress, in turn, generated an emerging research area, the learning of goal-oriented dialogs [6]. This research involves agents that conduct a multi-turn dialogue to achieve some task-specific goal, such as locating a specific object in a group of objects [7], inferring which image the user is thinking about [8], and providing customer services and restaurant reservations [6]. All these tasks require that the agent possesses the ability to conduct a multi-round dialog and to track the inter-dependence of each question-answer pair. Eventually, the agent learns an optimal policy through trial-and-error. The reward signal of each trail is delayed, and is only available at the end of the dialog. Also the reward signal is very sparse compared with a vocabulary size that often exceeds several thousands. Due to these challenges, in practice, policy gradient methods [9] perform more favorably than Q-learning methods [10].

Consider a simple goal-oriented dialog example from our synthetic dataset in Figure 1. We initialize three roles in this number guessing game, i.e. a *questioner*, an *answerer*, and a *guesser*. The questioner and the guesser try to infer which number the answerer is thinking about. First, the questioner asks questions about the target digit given the image, such as the color of the digit, the background color, the style of the digit, and also the number itself. Then the answerer responses with a yes/no answer. The questioner needs to reason based on the history dialog and keeps querying with meaningful questions. At the end, when the maximum number of questions is reached, the guesser analyzes the whole conversation along with the image, and takes a guess. If the guess is correct, then the task is completed successfully, and the questioner gets a positive reward signal. Otherwise the task is counted as a failure, and the questioner gets a non-positive reward signal.

#	Question	Answer
1	Is it 2 in the image?	No
2	Is it in a yellow background?	No
3	Is it 9 in the image?	Yes
4	Is it in a white background?	Yes
5	Is it a stroke style digit?	Yes
6	Is it a digit in blue?	No
	Guess: row 1 column 3	✔

Fig. 1: MNIST GuessNumber dataset example: Each sample consists of an image (left), a set of sequential questions with answers (right), and a target digit. The goal of this game is to find out the target digit by a multi-round question-answering.

The training of chat-bots using on-policy policy gradient methods requires numerous training samples. When the samples are generated through human-machine-interaction, e.g. by using the Amazon Mechanical Turk or in real-world applications, the collection of the data is time-consuming and expensive [11]. Hence, sample-efficiency receives increasingly more attention in dialog policy learning. We improve sample-efficiency using a novel on-off-policy policy gradient method relying on a biologically inspired mechanism [12], termed positive memory retention. This mechanism employs a bounded importance weight proposal on past positive trajectories, i.e. the behavior policy [13], to train the target policy network. The retention stops automatically when no further improvement occurs in a predefined number of iterations. The

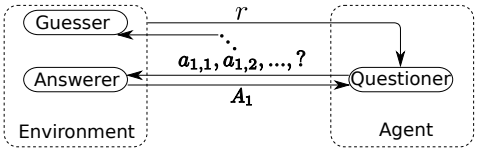

Fig. 2: GuessNumber Gameplay: The gameplay of the guessing game involves three plays, a questioner, an answerer, and a guesser. In our settings, we first pretrain all three models, then post-train only the questioner using RL. For each gameplay, the questioner first asks a question word by word, $a_{1,1}$, $a_{1,2}$, ..., ?, then the answerer responses with an answer, A_1. This question-answering repeats for a predefined number of rounds. Finally, the guesser reads in the dialog and makes a guess. If the guess is correct, then the questioner receives a reward $r = 1$, otherwise, $r = 0$.

bounded importance weight proposal tackles the problem of high variance in importance sampling. To reduce variance even further, we use recent behavior policies to update the probability in the memory buffer. An early-stopping mechanism within each epoch provides a trade-off between the sample-efficiency and the computational cost.

Contributions: (1) We introduce positive memory retention for efficient dialog policy learning, which uses bounded importance sampling, probability updating, and adaptively adjusts the retention times via early stopping within epochs. (2) We perform a comprehensive study about the performance of our method for goal-oriented dialog tasks using a new synthetic number guessing game and verify the high sample-efficiency of our algorithm. (3) The proposed model is also tested on a real-world benchmark GuessWhat?! game [14] and shows state-of-the-art performance, and an increased sample-efficiency by a factor of two.

2. BACKGROUND

This section introduces recurrent language models, Markov decision process, policy gradient, and importance sampling.

Recurrent Language Models: The goal of a recurrent neural network (RNN) based language model in NLP is to produce an output sequence $y = [a_1, a_2, ..., a_T]$ given a context x as input [15]. Here $a_i \in \mathcal{A}$ where $\mathcal{A}$ is the word vocabulary. For each step, the recurrent unit processes the previous word along with the context, and outputs a new word. At each time step t, the state s_t is the context input x and the words $y_{t-1} = [a_1, ..., a_{t-1}]$ produced by the RNN so far, i.e. $s_t = (x, y_{t-1})$. We sample the next word a_t from this probability distribution $\pi(\cdot|s_t)$, then update our state $s_{t+1} = (x, y_t)$ where $y_t = [y_{t-1}, a_t]$, and repeat in a similar fashion.

Markov Decision Process and Policy Gradient: We formalize a simplified Markov decision process (MDP) to our setting. In the MDP, an agent takes an action a in a state s and transitions to a new state s'. A trajectory τ refers to a sequence of transitions until the agent enters a terminal state where it receives a reward from the environment.

In our guessing games, a trajectory τ is (x, $a_{1,1}$, $a_{1,2}$, ..., ?, A_1, $a_{2,1}$, $a_{2,2}$, ..., ?, A_2, ..., G, r), where x is the context (image); a_i is the word sequence in question i; ? is the question mark that only occurs at the end of each question; A_i is the answer to question i; G is the output of the guesser; r is the reward for being correct or incorrect. See also Figure 2.

Formally, the simplified MDP is a triple of $(\mathcal{S}, \mathcal{A}, R)$ where $\mathcal{S}$, $\mathcal{A}$, and R represent a set of states, actions, and rewards, respectively. A policy π is a function that chooses an action at a given state, e.g. $\pi : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, where $\pi(a|s)$ refers to the probability of executing action a at the state s. When we sample an action $a_t \sim \pi(\cdot|s_t)$, we transition into state $(x, [y_{t-1}, a_t])$. We overload notation and let $R(\tau) = \sum_{t=1}^{T} R(s_t)$ be the accumulated reward of a trajectory τ. We define that $R(\tau) = 1$, if the task is a success; $R(\tau) = 0$, if the task is a failure. The policy network π is a recurrent language model, parametrized by a vector $\theta \in \mathbb{R}^n$, i.e. π_θ. The expected return of a policy π_θ is:

$$J(\theta) = \mathbb{E}[R(\tau)|\pi_\theta].$$

Our goal is to learn θ to maximize the expectation of the return $J(\theta)$. The objective function can be optimized with an on-policy policy gradient method, known as REINFORCE [9]. The gradient is calculated as:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}\left[(R(\tau) - b)\log\pi_\theta(a_t|s_t)\right]$$

where b is an optional baseline function used to reduce the variance of the gradient estimate [9].

Importance Sampling: Importance sampling (IS) is a general technique to estimate an integral $\int f(x)p(x)dx$ of a function $f(x)$, with distribution $p(x)$ [16]. IS samples from an appropriate proposal distribution $q(x)$, and then uses the samples to estimate the integral:

$$I = \mathbb{E}[f] = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{N}\sum_{n=1}^{N}\omega_n f(x^{(n)}) = \hat{I},$$

where $\omega_n = p(x^{(n)})/q(x^{(n)})$ are the importance weights. The goal is to minimize the variance of the estimate $\hat{I}$, which is proportional to $\mathrm{var}_q[f(x)\omega(x))] = \mathbb{E}_q[f^2(x)\omega^2(x)] - I^2$. Since the last term is independent of q, we can ignore it. Using Jensen's inequality, we have the following lower bound:

$$\mathbb{E}_q[f^2(x)\omega^2(x)] \geqslant (\mathbb{E}_q[|f(x)\omega(x)|])^2 = \left(\int |f(x)|p(x)dx\right)^2$$

The lower bound is obtained when using the optimal importance distribution: $q^*(x) = |f(x)|p(x)/\int |f(x')|p(x')dx'$. Interestingly, to estimate an integral, it is more efficient to sample from $q(x) \propto |f(x)|p(x)$ than to sample from $p(x)$. Of course, if $f(x)$ is unknown, it is best to sample from $q(x) = p(x)$ [17].

3. POSITIVE MEMORY RETENTION

This section contains our main contribution, the positive memory retention method. The reuse of past positive trajectories in policy learning is enabled by several contributions: first, sample efficiency is improved by concentration of past positive trajectories; secondly, the sampling bias is corrected by importance sampling in policy gradients; thirdly, the stability is ensured by introducing bounds on the important weights; fourthly, the variance is reduced by probability updating of the proposal distribution; finally, the stability is improved by policy search via early stopping.

Positive Memory Matters: In human memory retention, focusing on *rewarded* events has been discovered to be a preferred strategy in the post-learning phase happening in the hippocampus area of the brain [12]. We believe that this fact also intuitively applies to RL since non-rewarded trajectories do not contribute directly to the estimated gradient to increase the expected return, $\nabla_\theta J(\theta)$, since $R(\tau)$ is zero.

In a more general case, consider the policy updating with a baseline function, e.g. $0 < b < 1$. The gradient of non-rewarded trajectories is opposite to the direction of the gradient of the current policy, $\nabla_\theta \log \pi_\theta(a_t|s_t)$, because $R(\tau) - b < 0$. This means that the weights are changed in a way to depress the current policy π_θ, which is not necessarily equivalent to maximizing the expected return. However, it might be helpful, in a way to encourage exploration.

The *high efficiency* of positive memory retention can also be derived from importance sampling. Consider that the expectation we want to estimate is the expected return $\mathbb{E}[R(\tau)]$, so $R(\tau)$ assumes the role of $f(x)$ in Section 2. The proposal distribution q should thus be of the form $q(\tau) \propto R(\tau)p(\tau)$, thus we only need to consider successful memory trajectories.

Policy Gradient with Importance Sampling: However, memory trajectories cannot be directly applied to policy gradient methods. The main reason is that the training requires trajectories from the target policy $p(\tau|\pi_\theta)$ with the current parameter vector θ, whereas the memory trajectories were generated by $q(\tau|\pi_{\theta'}) = p(\tau|\pi_{\theta'})$ with a different parameter vector θ'. We again can use the concept of IS and obtain

$$\hat{J}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \frac{p(\tau^{(i)}|\pi_\theta)}{q(\tau^{(i)}|\pi_{\theta'})} R(\tau^{(i)}), \text{ with } \tau^{(i)} \sim q$$

where n is the number of trajectories used to estimate the expected return $J(\theta)$ [18]. In the equation above, we assume $q(\tau) = 0 \Rightarrow p(\tau) = 0$. This is readily true, since each action is sampled from the defined action space $\mathcal{A}$. The importance weights are evaluated using:

$$\omega(\tau^{(i)}) = \frac{p(\tau^{(i)}|\pi_\theta)}{q(\tau^{(i)}|\pi_{\theta'})} = \frac{\prod_{t=1}^{T} \pi_{\theta(a_t|s_t)}}{\prod_{t=1}^{T} \pi_{\theta'(a_t|s_t)}}$$

where $\prod_{t=1}^{T} \pi_{\theta(a_t|s_t)}$ needs to be calculated from the target policy, and $\prod_{t=1}^{T} \pi_{\theta'(a_t|s_t)}$ has already been calculated from the behavior policy. Finally, the importance weighted policy gradient is:

$$\nabla_\theta \hat{J}(\theta) = \nabla_\theta \mathbb{E}_q \left[\omega(\tau)(R(\tau) - b)\log \pi_\theta(a_t|s_t)\right]. \quad (1)$$

Bounded Importance Weight Proposal: This estimator is unbiased, but it suffers from very high variances because it involves a product of a series of unbounded importance weights. To prevent the importance weight from "exploding", the goal is now to select only samples that are *not far* from the target policy.

To evaluate the distance, we use a symmetric version of the KL-divergence, i.e. the Jensen-Shannon divergence [19]:

$$\mathrm{JS}(p, q) = 0.5\mathbb{KL}(p \parallel 0.5(p+q)) + 0.5\mathbb{KL}(q \parallel 0.5(p+q)).$$

We now derive a formulation of the JS-divergence, as a distance metric, which is related to the importance weight ω:

$$\begin{aligned}\mathrm{JS}(p, q) &\approx 0.5\sum_{k=1}^{K} p_k \log \frac{2p_k}{p_k + q_k} + 0.5\sum_{k=1}^{K} q_k \log \frac{2q_k}{p_k + q_k} \\ &= 0.5\sum_{k=1}^{K} p_k \log \frac{2}{1 + \omega_k} + 0.5\sum_{k=1}^{K} q_k \log \frac{2}{\frac{1}{\omega_k} + 1}\end{aligned}$$

We can see that the distance between the proposal distribution q and the optimal solution p depends on both ω_k and $1/\omega_k$. To limit the variance of the importance sampling, we limit the importance weight as $\omega_k \leq \omega_{max}$ and its inverse as $1/\omega_k \leq \omega_{max}$. Subsequently, we define a trust region of importance weights, $\omega_k \in [1/\omega_{max}, \omega_{max}]$ and only use trajectories whose importance weights fall within this range.

Essentially, we use the importance weight ω as a value to select high quality trajectories, filtering out those that deviate far from the current policy. In this way, we shape the proposal distribution into a safe one. The bound ω_{max} of the distribution can be selected by observing the learning curve during training.

Probability Updating: Another way to reduce the variance is to *adapt* the proposal distribution, $q(x)$, to make it as close as possible to $p(x)$. After updating the target policy with Equation 1, we use the current target policy as a behavior policy for retention in the future. In this way, the memory buffer is being continuously updated and the proposal distribution is also updated.

Policy Search via Early Stopping: In order to make the best use of the memory, the learning process is verified using a group of validation samples. During the training process, the model remembers the positive trajectories within the current epoch for later retention. During the retention phase, the model first goes through the memory buffer, and updates the model using Equation 1 with the bounded importance weight proposal. After each iteration, the model verifies the learned policy on a validation set. If the policy becomes better than the previous policy, then it is saved. If the policy has not been

Algorithm 1 Positive Memory Retention (PMR)

Require: pretrained RNN language model π_θ

for iteration in range(max iterations) **do**
  for $t = 1$ to T **do**
    $(a_t, p_t) = \pi_\theta(x, y_{t-1})$, $y_t = (y_{t-1}, a_t)$
  $r = \mathrm{R}(y) \leftarrow$ Environment
  for $t = T$ to 1 **do**
    $\theta = \theta + \alpha(r - b)\nabla_\theta \log \pi_\theta(a_t|(x, y_{t-1}))$
  if $r > 0$ **then**
    memory $\leftarrow \tau = (x, y, p, r)$
validating(π_θ), $\theta' = \theta$
for trajectory τ in memory **do** # memory retention
  $x, y, p, r \leftarrow \tau$
  for $t = 1$ to T **do**
    $q_t = p_t$, $(a_t, p_t) = \pi_\theta(x, y_{t-1})$
    memory $\leftarrow p_t$ # probability updating
  $\omega = \prod_{t=1}^{T} p_t / \prod_{t=1}^{T} q_t$
  if $\omega \in [1/\omega_{max}, \omega_{max}]$ **then**
    for $t = T$ to 1 **do**
      $\theta = \theta + \alpha\omega(r - b)\nabla_\theta \log \pi_\theta(a_t|(x, y_{t-1}))$
validating(π_θ)
if no improvement for n_{max} iterations **then**
  $\theta = \theta'$
if improvement **then**
  $\theta' = \theta$

improved for a limited number of iterations n_{max}, then memory retention is stopped and the training of the model with REINFORCE continues. This mechanism helps the model make the best use of the past training samples and makes the learning more stable.

Complete Training Algorithm: We summarize the complete method in Algorithm 1. Note that, before training the language model with RL, we pretrained the model in a supervised fashion for a kickstart policy.

4. EXPERIMENTS

We conduct two sets of experiments to verify the proposed method. To highlight the methods' ability to boost sample-efficiency, we first create and experiment with a synthetic dataset [1]. We then show that the method also works well on a real-word benchmark, GuessWhat?! [14].

4.1. MNIST GuessNumber Dataset

Experiment setting: We create a synthetic dataset, named MNIST GuessNumber, which is designed for quick testing and analysis of RL methods in the task of visual-grounded goal-oriented dialog systems. The creation of the dataset is inspired by [20]. Each image in MNIST GuessNumber contains a 3×3 grid of MNIST digits and each MNIST digit in the grid has four randomly sampled attributes, i.e. color $= \{\text{red, blue, green, purple, brown}\}$, bgcolor $= \{\text{cyan, yellow, white, silver, salmon}\}$, number $= \{x | 0 \leq x \leq 9\}$ and style $= \{\text{flat, stroke}\}$, as illustrated in Figure 1.

Given the generated image from MNIST GuessNumber, we automatically generate questions and answers about a set of the digits in the grid that focus on one of the four attributes. During question generation, the target subset for a question is selected based on the previous target subset referred by the previous QA, as shown in Figure 1. For answer generation, we generate a yes/no answer based on whether the questioned attribute matches with the target digit. The QA is repeated until there is only one digit in the target subset. We generated 30 K, 10 K, and 10 K images for training, validating, and testing, respectively, and one successful game for each unique image. In each grid image, there are nine cells. Each cell contains four attributes, including the color, bgcolor, number, and the style.

Model details and pretraining: There are three roles in this number guessing game, a questioner, an answerer, and a guesser. The word and the image embeddings are trained end-to-end using lookup table layers. The questioner model that we used is a long short-term memory (LSTM) [21] of 256 units conditioned on a given image. The answerer model takes in the question along with the target digit and outputs a yes/no answer. The answer model is based on an LSTM with 64 units. The last one is the guesser model. The guesser uses an LSTM with 64 units to process the whole dialog, and compares it with each digit using a dot product on their respective latent representations. The prediction is the most similar digit. We train all three models for 30 epochs using the maximum likelihood criterion. The pretrained models obtain a game success rate of 63.09% on the test split with a maximum of four rounds of QA.

Reinforced training: After pretraining, we keep the answerer and the guesser fixed and train the questioner model with RL. Given the unique images in the training set, for each game, the answerer randomly picks a digit in the image as the target and lets the questioner ask. The baseline method is the REINFORCE. For positive memory retention, we set the parameter $\omega_{max} = 10$, and the early stopping threshed $n_{max} = 2$. The ω_{max} is selected on the validation set. An extensive evaluation of the impact of ω_{max} is shown in Table 1 lower part. When we use $\omega_{max} = 10$, we observe that about 65% of the positive trajectories are used for weight updating. So, ω_{max} can be considered as a trade-off between sample reuse ratio and the variance introduced by importance sampling. The early stopping threshed n_{max} is a trade-off between sample-efficiency and computational power. Our implementation [2] uses Torch7 [22].

Results: From Figure 3 left, we can see that at the 10^{th}

[1] https://github.com/ruizhaogit/MNIST-GuessNumber

[2] https://github.com/ruizhaogit/PositiveMemoryRetention

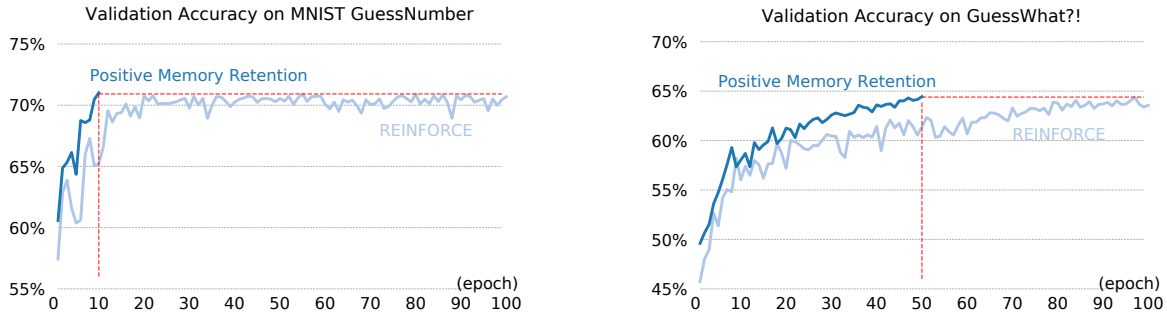

Fig. 3: Experimental results: The left figure shows that at the 10^{th} epoch, the model trained with positive memory retention obtains about 70% accuracy on the validation set, which is is equivalent to the same model trained with REINFORCE for 100 epochs, which obtains 69.92% accuracy. The right figure shows that the model trained with REINFORCE for 100 epochs, obtains 63.39% accuracy on the validation split. At the 50^{th} epoch, the same model trained with our proposed method has a better result, 63.44%.

epoch, the model trained with positive memory retention, obtains about 70% accuracy on the validation set, which is comparable with the same model trained with REINFORCE for 100 epochs with an accuracy of 69.92%. After training, we evaluated the best model on the test set. REINFORCE and positive memory retention obtain 69.86% and 70.27% accuracy on the test split, respectively. However, the memory positive attention only needs one-tenth of the training samples. The experiment on MNIST GuessNumber shows that the sample-efficiency has been improved by a factor of 10.

Ablation tests: A summary of the ablation tests is shown in Table 1. We can see that the bounded importance weight proposal is critical for policy gradient with importance sampling. Without this component, the model *diverges* quickly, shown in Table 1 (# 3-4). The other proposed innovations all improve model performance as well, as shown in Table 1 (# 5-9). One can also see that the choice of the bound parameter ω_{max} has a major influence on the performance.

4.2. GuessWhat?! Game

Experiment settings: In the GuessWhat?! dataset [14] the dialogs are collected using Amazon Mechanical Turk with respect to MS-COCO [23] images. Each game is composed of an image, a target object in the image, the spatial information of the objects, the category of the objects, and the QA-dialogs. Unlike the MNIST GuessNumber, the questions in the training set are in free form text. The answers are still in the yes/no form. This dataset is more challenging due to its large vocabulary size (5 K), and long dialog sequences. Due to the large action space and the extremely delayed reward signals, the importance sampling estimates have very large variances. In our experiment, when we first attempted to retain with all past trajectories, and without the weight bound, the model *diverged* quickly, as in the MNIST GuessNumber experiments, see Table 1 (# 3). Our proposed method reduces the variance and makes the sample reuse possible for real-word sceneries.

Model details and pretraining: Each game in the GuessWhat?! contains three roles, a questioner, an answerer, and a guesser. As our aim is to compare the sample-efficiency of our proposed model with other strong baselines, we use the same model structure as was used in [7]. The questioner model is a one layer LSTM with 512 units and conditioned on the VGG16-CNN-FC8 [24] features of the image. The answerer model deploys an LSTM with 512 units to process the question along with spatial and categorical information of the target object. The guesser model uses an LSTM to process the whole dialog and can consider all the spatial and categorical information of the objects in the image. The guesser compares the similarities between the dialog representation and each object representation with a dot product, and then takes the guess. All these three models are pretrained with MLE for 30 epochs for a kickstart policy. We reproduced the paper's experimental results using Torch7 [22], and obtained 41.41% accuracy on the test split after supervised training.

Reinforced training: With the pretrained models, we keep the answerer and the guesser fixed and train the questioner model. We train the model for 100 epochs, using REINFORCE with a learning rate α of 0.0001 and a running average as the baseline b. Our reimplementation using Torch obtains 62.61% accuracy on the test split, about 2% higher than their result of 60.3% from the original implementation [25], due to some technical differences. We use our reimplementation as the REINFORCE baseline, in Figure 3, to eliminate the influence of these technical differences for a fair comparison. For positive memory retention, Algorithm 1, we use weight bound $\omega_{max} = 5$, so that $\omega \in [1/5, 5]$, to stabilize the training. We use the early stopping threshed in each epoch as $n_{max} = 2$. We observe that with $\omega_{max} = 5$, about 85% of the trajectories in the memory contribute to the model weight updates. This high ratio of reuse is also due to the probability updating mechanism, which bridges the gap between the behavior policies and the target policy.

Results: From Figure 3 right, we can see that the model trained with REINFORCE obtains 63.39% accuracy on the validation split after training for 100 epochs. However, at the

Table 1: Ablation tests on MNIST GuessNumber: Notations: RF denotes the REINFORCE; IS is importance sampling; PM means using positive memory only, otherwise all memory; UB denotes the upper bound ω_{max}; LB represents the lower bound $1/\omega_{max}$; PB is the probability updating trick: ES means the early stopping within epochs, if use ES, then the early stopping threshold is 2, otherwise train for 3 iterations; (%) is the accuracy on the test split using the best model selected via validation split during 10 epochs of training. The **upper part** above the middle horizontal line shows the ablation test of different components in the positive memory retention. Here, $\omega_{max} = 10$, and (#1) is the performance of the kickstart policy after supervised training. Note that (#3) and (#4) *diverges* quickly, which means that the testing accuracies are lower than 20.0% after 10 epochs of training. UB makes the stable training of RF+IS possible, shown in (#5). PM, LB, PB, ES, contribute 0.76%, 1.18%, 0.73%, and 1.54%, respectively. The **lower part** below the middle horizontal line shows the extensive evaluation regarding to the upper bound parameter ω_{max}.

#	RF	IS	PM	UB	LB	PB	ES	(%)
1	–	–	–	–	–	–	–	63.09
2	✓	–	–	–	–	–	–	65.40
3	✓	✓	–	–	–	–	–	< 20.
4	✓	✓	✓	–	–	–	–	< 20.
5	✓	✓	–	✓	–	–	–	66.06
6	✓	✓	–	✓	✓	–	–	67.24
7	✓	✓	✓	✓	✓	–	–	68.00
8	✓	✓	✓	✓	✓	✓	–	68.73
9	✓	✓	✓	✓	✓	✓	✓	**70.27**
10	✓	✓	✓	1	✓	✓	✓	66.24
11	✓	✓	✓	5	✓	✓	✓	65.69
12	✓	✓	✓	10	✓	✓	✓	**70.27**
13	✓	✓	✓	20	✓	✓	✓	69.38
14	✓	✓	✓	30	✓	✓	✓	69.09
15	✓	✓	✓	100	✓	✓	✓	65.39

50^{th} epoch, the same model trained with positive memory retention reaches 63.44% validation accuracy. After training, we evaluated the best model on the test set. REINFORCE and positive memory retention obtain 62.61% and 63.17% accuracy on the test split, respectively. We can see that the proposed method provides state-of-the-art performance with double sample-efficiency on the GuessWhat?! dataset.

5. RELATED WORK

Goal-Oriented Dialogs: Recently, researchers started exploring intensively deep RL for goal-oriented dialogs [6, 7, 8, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40], focusing on learning to achieve a goal via dialog. Bordes and Weston [6] pointed out that the recent success in chit-chat dialogs may not carry over to goal-oriented settings. Strub et al. [7] and Das et al. [8] conduct the task-oriented conversation over image guessing games. In [7], the dialog aims at object discovery through a series of yes/no QAs. Policy gradient is used to improve performances of dialog agents in terms of task completion rate. Das et al. [8] use policy gradient to train two chat-bots to play image guessing games and show that they establish their own communication style. Both works use on-policy policy gradient methods. Sample-efficient on-off-policy learning methods, as developed in this paper, have not yet been explored in the field of goal-oriented dialog.

Sample-Efficient RL: While the goal-oriented dialog using RL is a recent research direction, control tasks via RL have been studied extensively and importance sampling based actor-critic methods have been known to be beneficial for sample-efficiency [18, 13, 41, 42, 43, 44]. However, the control tasks are inherently different from dialog tasks in the aspect of action space. For example, in Atari games, the agents normally have less than 20 actions to explore; in contrast, the action space, i.e. the vocabulary, contains thousands of words in dialogs. Moreover, the reward of a dialog is only available at the end, which is much more sparse and delayed than in Atari games. In these games, there are intermediate rewards prior to the game ending. The long trajectories in dialog tasks make the often observed problem of exploding importance weights even more extreme. Even if an explosion does not occur, the variance of the importance sampling increases. To overcome these challenges, new solutions must be introduced.

Memory Retention: The use of positive memory retention is inspired by recent neuroscience research, which concludes that the brain prioritizes those high-reward memories, which might be the most important for obtaining future rewards [12]. Tresp et al. [45] argue that the brain's memory functions might inspire technical solutions requiring memory traces. Biologically-inspired experience replay [46], was used to stabilize the training process in RL and and thereby was quite successful. These papers used Q-learning, which is an off-policy method that is able to use the past trajectories directly. However, on-policy policy gradients cannot reuse past samples directly [18, 13]. The main contribution of this paper is that we show that our extensions permit an efficient reuse of past samples in on-policy policy gradients methods. These extensions also work well in dialog settings, which are challenging due to the sparse reward and the large action space.

6. CONCLUSION

We proposed a novel positive memory retention mechanism for improving sample-efficiency in dialog policy learning, using past positive trajectories and low-variance importance sampling estimates. The model reuses past positive samples as behavior policies, samples from a bounded importance weight proposal, and updates the target policy with an importance weight correction. We tested on both synthetic and real-word datasets and illustrated dramatically improved sample-efficiency. We demonstrate that policy gradient can successfully be trained using past trajectories in dialog tasks.

7. REFERENCES

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

[2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra, "Deal or no deal? end-to-end learning for negotiation dialogues," *arXiv preprint arXiv:1706.05125*, 2017.

[5] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[6] Antoine Bordes and Jason Weston, "Learning end-to-end goal-oriented dialog," in *International Conference on Learning Representations (ICLR)*, 2017.

[7] Florian Strub, Harm de Vries, Jérémie Mary, Bilal Piot, Aaron C. Courville, and Olivier Pietquin, "End-to-end optimization of goal-driven and visually grounded dialogue systems," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[8] Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *International Conference on Computer Vision (ICCV)*, 2017.

[9] Ronald J Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, 1992.

[10] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction (Second Edition)*, MIT press, 2018.

[11] Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh, "Evaluating visual conversational agents via cooperative human-ai games," in *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2017.

[12] Matthias J Gruber, Maureen Ritchey, Shao-Fang Wang, Manoj K Doss, and Charan Ranganath, "Post-learning hippocampal dynamics promote preferential retention of rewarding events," *Neuron*, 2016.

[13] Thomas Degris, Martha White, and Richard S Sutton, "Off-policy actor-critic," in *International Conference on Machine Learning (ICML)*, 2012.

[14] Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville, "Guesswhat?! visual object discovery through multi-modal dialogue," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *Interspeech*, 2010.

[16] Art B. Owen, *Monte Carlo theory, methods and examples*, 2013.

[17] Kevin P Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012.

[18] Tang Jie and Pieter Abbeel, "On a connection between importance sampling and the likelihood ratio policy gradient," in *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[19] Jianhua Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, 1991.

[20] Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal, "Visual reference resolution using attention memory for visual dialog," in *Advances in Neural Information Processing Systems*, 2017.

[21] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[22] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[24] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] Florian Strub and Harm de Vries, "Guesswhat?! models," `https://github.com/GuessWhatGame/guesswhat/`, 2017.

[26] Jason D Williams and Geoffrey Zweig, "End-to-end lstm-based dialog control optimized with supervised and reinforcement learning," *arXiv preprint arXiv:1606.01269*, 2016.

[27] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng, "End-to-end reinforcement learning of dialogue agents for information access," *arXiv preprint arXiv:1609.00777*, 2016.

[28] Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng, "Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking," *arXiv preprint arXiv:1608.05081*, 2016.

[29] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky, "Deep reinforcement learning for dialogue generation," *arXiv preprint arXiv:1606.01541*, 2016.

[30] Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao, "End-to-end task-completion neural dialogue systems," *arXiv preprint arXiv:1703.01008*, 2017.

[31] Alexander Rudnicky and Wei Xu, "An agenda-based dialog management architecture for spoken language systems," in *IEEE Automatic Speech Recognition and Understanding Workshop*, 1999, vol. 13.

[32] Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker, "Reinforcement learning for spoken dialogue systems," in *Advances in Neural Information Processing Systems*, 2000, pp. 956–962.

[33] Oliver Lemon, Xingkun Liu, Daniel Shapiro, and Carl Tollander, "Hierarchical reinforcement learning of dialogue policies in a development environment for dialogue systems: Reall-dude," in *BRANDIAL'06, Proceedings of the 10th Workshop on the Semantics and Pragmatics of Dialogue*, 2006, pp. 185–186.

[34] Tiancheng Zhao and Maxine Eskenazi, "Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning," *arXiv preprint arXiv:1606.02560*, 2016.

[35] Rui Zhao and Volker Tresp, "Improving goal-oriented visual dialog agents via advanced recurrent nets with tempered policy gradient," *arXiv preprint arXiv:1807.00737*, 2018.

[36] Rui Zhao and Volker Tresp, "Learning goal-oriented visual dialog via tempered policy gradient," in *IEEE Spoken Language Technology (SLT) (forthcoming)*, 2018.

[37] Kavosh Asadi and Jason D Williams, "Sample-efficient deep reinforcement learning for dialog control," *arXiv preprint arXiv:1612.06000*, 2016.

[38] Gellért Weisz, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić, "Sample efficient deep reinforcement learning for dialogue systems with large action spaces," *arXiv preprint arXiv:1802.03753*, 2018.

[39] Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young, "Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management," *arXiv preprint arXiv:1707.00130*, 2017.

[40] Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu, "Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2454–2464.

[41] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare, "Safe and efficient off-policy reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 1054–1062.

[42] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas, "Sample efficient actor-critic with experience replay," in *International Conference on Learning Representations (ICLR)*, 2017.

[43] Audrunas Gruslys, Mohammad Gheshlaghi Azar, Marc G Bellemare, and Remi Munos, "The reactor: A sample-efficient actor-critic architecture," *arXiv preprint arXiv:1704.04651*, 2017.

[44] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al., "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561*, 2018.

[45] Volker Tresp, Cristóbal Esteban, Yinchong Yang, Stephan Baier, and Denis Krompaß, "Learning with memory embeddings," in *Advances in neural information processing systems (NIPS Workshop)*, 2015.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., "Human-level control through deep reinforcement learning," *Nature*, 2015.