Approximation Gradient Error Variance Reduced Optimization

Wei-Ye Zhao[1,2], **Yang Liu**[2], **Xiaoming Zhao** [2], **Jie-Lin Qiu**[3], **Jian Peng**[2]
[1] University of California, Berkeley
[2] University of Illinois at Urbana-Champaign
[3] Carnegie Mellon University
caesarzwy@berkeley.edu, liu301@illinois.edu, xz23@illinois.edu, jielinq@andrew.cmu.edu, jianpeng@illinois.edu

Abstract

Recent advances in deep learning have achieved human-level performance on a variety of real-world applications. However, the current algorithms still suffer from poor gradient estimation with excessive variance, resulting in unstable training and poor sample efficiency. In our paper, we proposed an innovative optimization strategy by reducing the Approximation Gradient Error Variance in gradient estimation. We evaluate the performance of our approach on the 20 games of the challenging Arcade Learning Environment, and report significant improvement in both reward scores and learning speed.

1 Introduction

The recent advances of supervised deep learning methods have tremendously improved the performance on challenging tasks in computer vision, speech recognition and natural language processing. Artificial neural networks is the core idea of deep learning, which is used to model complex hierarchical data abstractions and representations from raw input data. With the help of deep learning, reinforcement learning (RL) [Sutton and Barto, 1998] has recently achieved remarkable success on massive real-world applications, such as human-computer interaction [Maes and Kozierok, 1993], video games [Mnih *et al.*, 2015], visual navigation [Zhu *et al.*, 2017], goal-oriented autonomous decision making [Frank and Claus, 2006] and autonomous driving [Dai *et al.*, 2005].

Q-learning [Watkins and Dayan, 1992] is one of the most popular reinforcement learning algorithms, where the policy is learnt by adjusting the parameters at each training iteration to reduce the mean-squared error in the Bellman equation so as to optimize the cumulative future reward, resulting in sequences of well-defined optimization problems. A standard method to solve optimization problems is gradient descent [Kingma and Ba, 2014]. Since it is expensive to compute the full expectation in the gradient, stochastic methods are often used to optimize the loss function based on gradients of small batches of samples. Despite these successes, the inaccurate estimation of gradient as well as huge variance arisen from RL training procedure is still the key problem of these stochastic optimization methods, the inexact approximate gradient estimation can be viewed as the distorted gradient direction. In large scale deep Q learning problem, the Q value is represented with deep Q network with proper tuned network parameters. The DQN learning process can be viewed as iteratively optimizing network parameters process according to gradient direction of the loss function at each stage. Therefore, the Approximation Gradient Error (AGE) with a large variance can largely deteriorate the representation performance of deep Q network by driving the network parameter deviated from the optimal setting, causing the large variability of DQN performance. On the other hand, if we assume the network parameter of DQN is θ, the core learning step of deep Q learning is to minimize the gap between the estimated maximum Q value ($y_(s,a)$) given state s and action a and current Q value($Q(s,a;\theta)$) using the operation that $\hat{\theta} = \mathrm{argmin}_\theta \mathbb{E}\|y_(s,a) - Q(s,a;\theta)\|^2$. It is noteworthy that $\hat{\theta}$ is obtained with gradient descent, thus if the AGE variance is wild, it requires more iterations of argmin operation such that θ could reach $\hat{\theta}$, which means large gradient variance will postpone the process when DQN gets local optima.

Here we propose Stochastic Variance Reduction for Deep Q-learning (SVR-DQN) optimization to accelerate the convergence for deep Q-learning by reducing the Approximation Gradient Error variance. We evaluate our proposed algorithm using Arcade learning environment [Bellemare *et al.*, 2013]. Our experiments show that SVR-DQN optimization algorithm can significantly reduce the delay before the performance gets off the ground, and further lead to aggressive sample efficiency at initial training stage. Our new strategy outperforms Adam in both reward scores and training time on 18 out of 20 games.

2 Background

Reinforcement learning (RL) considers agents operating in an uncertain environment, where agents interact with environment to perform sequential actions. At each time step, the agents react according to the observation from environment, receiving a scalar reward from environment. The RL algorithm aims to search a policy in order to maximize the final cumulative rewards. To be specific, for time step $t = \{1, \ldots, T\}$, the agents sample action $a_t \sim \pi(a_t|s_t)$ based on observation s_t. Then the agents get reward r_t and next step observation s_{t+1} generated. The goal is to maximize

$\sum_{t=1}^{T} \gamma^{t-1} r_t$, where γ is the discounting factor for convergence.

2.1 Q-Learning

Q-learning is a satisfied method for solving sequential decision problems. It defines optimal value for each action as the expected future rewards when the optimal policy $\pi^\star$ starting from that action, namely

$$Q_{\pi^\star}(s,a) = \mathbb{E}(\sum_{t=1}^{T} \gamma^{t-1} r_t | s_0 = s, a_0 = a, \pi^\star) \quad (1)$$

However, it is intractable to compute the optimal $Q_{\pi^\star}$. Instead, we need to estimate optimal action values through temporal difference learning. The parameterized target with θ is formalized as following

$$Y_t^Q = r_t + \gamma \max_a Q_\pi(s_{t+1}, a, \theta_t) \quad (2)$$

2.2 Deep Q-Network

A deep Q network (DQN) is a multi-layered neural network parameterized with θ which outputs a vector $Q_\pi(s,\cdot)$ of action values when given a observation state s. If observation space has m dimension and action space has n dimension, the DQN is a mapping from $\mathbb{R}^m$ to $\mathbb{R}^n$. The standard method for DQN[Mnih *et al.*, 2015] integrates usage of replay buffer and target network. Ror the experience replay, we store observed transition tuples (s_t, a_t, r_t, s_{t+1}) and uniformly sample from them in order to break the correlation between tuple pairs, which will enhance the performance of model. Meanwhile, the target network has exactly the same network structure as the online network except for its parameter θ^-. The parameters θ^- will be copied from online network every certain steps so that $\theta = \theta^-$, which will be kept unchanged on all other steps, namely

$$Y_t^{DQN} = r_t + \gamma \max_a Q_\pi(s_{t+1}, a, \theta_t^-) \quad (3)$$

On the other hand, the standard Q-learning and DQN, namely Eq.2 and Eq.3, maximize Q with the same values both to select and to evaluate an action, making it more likely to select overestimated values. This phenomenon will cause overoptimistic value estimates. In order to resolve the overoptimism, we use Double Q-learning[Van Hasselt *et al.*, 2016] to get the following modified target:

$$Y_t^{DoubleQ} = r_t + Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a, \theta), \theta')$$

3 Stochastic Variance Reduction for Deep Q-learning Optimization

Many machine learning problems are considering the a finite-sum optimization problem as following:

$$\min_w f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \quad (4)$$

let $w^* = \arg\min_w f(w)$ denote the optimal solution for Eq.4, a lot of researches in optimization algorithm are motivated to find solution w such that $f(w) - f(w^*) \leq \epsilon$. For large-scale problems in form of Eq.4, randomized variance reduced first-order methods are especially efficient for their low per iteration cost. In order to develop fast stochastic first-order methods, we should make sure that when the iteration gets closer to optimum, the variance of randomized updating direction decreases.

3.1 Adam

An extension to stochastic gradient descent algorithm called Adam [Kingma and Ba, 2014] has recently been adopted for a broad range of deep learning models. Adam is an efficient stochastic optimization method to update network weights iteratively based on first-order gradients information. Once we have gradients of objective function, Adam can adaptively estimate the first and second moments of the gradients, and further compute adaptive learning rate.

Although Adam [Kingma and Ba, 2014] is a widely used state-of-the-art stochastic optimization algorithm in deep Q-learning achieving robust performance in a broad range of challenging tasks, the parameter update rule of Adam is solely based on first-order gradient information of stochastic sample batch, which is inaccurate due to variance caused by random sampling. To better exploit accurate gradient estimation and reduce the negative impact of noisy gradient, we design a new algorithm by leveraging both advantages from stochastic variance reduced gradient descent (SVRG) technique [Johnson and Zhang, 2013] and Adam.

3.2 Stochastic Variance Reduced Gradient

Stochastic variance reduced gradient (SVRG) [Johnson and Zhang, 2013] is an explicit variance reduction method for stochastic gradient descent which does not require gradient storage. SVRG enjoys very fast convergence rate using variate control and can be applied to complex problem such as neural network training.

To find approximate solution to optimization problem as Eq.4, a standard method is gradient descent, which is expensive since it requires n evaluations of derivatives at each iteration. A popular modification is SGD, which reduce the computation cost of standard gradient descent greatly by subsampling:

$$w_t = w_{t-1} - \eta_t \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(w_{t-1}, \varphi_t) \quad (5)$$

where φ_t is random variable depending on w_{t-1}, $t = 1, 2, \ldots$ are the subsequence time steps, m is the size of mini-batch sampled from n instances, and the expectation $\mathbb{E}[\sum_{i=1}^{m} \nabla f_{i_t}(w_{t-1}, \varphi_t) | w_{t-1}] = \sum_{i=1}^{m} \nabla f_i(w_{t-1})$. However, large random variances will arise due to the variance of $\sum_{i=1}^{m} \nabla f_{i_t}(w_{t-1}, \varphi_t)$, which slows down the convergence rate.

SVRG maintains the snapshot of estimated $\tilde{w}$ which is close to the optimal w every certain iteration. Given the preserved $\tilde{w}$, SVRG pre-calculates an average gradient $\tilde{\mu} =$

$\frac{1}{B}\sum_{i=1}^{B}\nabla f_i(\tilde{w})$ as an anchor point, where B is a subset of training samples. SVRG modifies Eq.5 as:

$$w_t = w_{t-1} - \eta_t \cdot (\frac{1}{m}\sum_{i=1}^{m}\nabla f_i(w_{t-1}) - \frac{1}{m}\sum_{i=1}^{m}\nabla f_i(\tilde{w}) + \tilde{\mu}) \tag{6}$$

Note that when w is close to $\tilde{w}$, the difference $\nabla f(w) - \nabla f(\tilde{w})$ is small. When both w_t and $\tilde{w}$ converge to the same optimal parameter w^*, then $\tilde{\mu} \to 0$ and $\nabla f(w) \to \nabla f(\tilde{w})$. Therefore, the variance of SVRG in update rule Eq.6 is reduced and SVRG can find the more accurate gradient direction estimation.

3.3 Stochastic Variance Reduced Deep Q-learning Optimization

In order to improve the performance of Adam optimization, we apply SVRG to find the accurate gradient direction based on the small stochastic training subset and propagate the optimized first-order information to Adam. Our main algorithm, Stochastic Variance Reduction for Deep Q-learning Optimization (SVR-DQN), is summarized in Algorithm 1.

At the beginning of the algorithm, we form a training sample batch B^s sampling from the whole training instances, and fix it for the whole optimization process in s-th outer loop. We calculate the average gradient using samples from B^s to perform the current anchor point $\tilde{\mu}^s$, the difference with the standard SVRG process is that we don't use the whole training samples to construct anchor, which is inspired by Sallinen's Practical SVRG [Harikandeh *et al.*, 2015] that SVRG can solve optimization problem inexactly to calculate $\tilde{\mu}$ with a subset of training examples, and the convergence rate is unchanged. In the inner loop iteration (SVRG variance reduction), we reduce the variance with the average of randomly selected mini-batch b^t from B^s and update parameter according to updating rule Eq.6, since the usage of individual training sample has a great variance and is also computational inefficient. In order to thoroughly leverage the information from samples in B^s to find optimal gradient direction estimation, the ideal selection of mini-batch size b and SVRG inner loop iteration number m should satisfy the constraint: $b \times m \geq B$.

After SVRG variance reduction process, we have the updated parameter w_m and previous stored snapshot $\tilde{w}^s$, the variance reduce gradient estimation g_s is calculated as $w_m - \tilde{w}^s$. Note that it is unnecessary to rescale g_s, since effective step size in Adam is invariant to the scale of the gradients [Kingma and Ba, 2014]. With g_s calculated, we follow the standard Adam procedure to construct bias-corrected first moment estimate and second raw moment estimate and further finalize updating parameters for this training iteration leveraging the more accurate gradient direction estimation.

4 Approximation Gradient Error Variance Reduction

4.1 Approximation Gradient Error Variance

The Approximation Gradient Error(AGE) is the error in the gradient direction estimation of cost function $f(\tilde{w})$, where $\tilde{w}$ is the hyper-parameters of this function, which are optimized with gradient descent methods iteratively by minimize the DQN loss(Algorithm 1 line 29). Given the certain learning samples preserved in experience buffer, the ideal gradient estimation of loss function is supposed to give the accurate learning direction (derivation value) leveraging the current information provided by those learning samples, thus the agent (DQN) can quickly converge to policy optimas by optimizing hyper-parameter at the gradient direction. However, AGE appears in gradient estimation process.

AGE is a result of several factors: Firstly, the suboptimality of current hyper-parameters $\tilde{w}$ due to inexact minimization. Secondly, the constrained representing strength of DQN. Thirdly, the limited representation number of the samples we used for deriving the gradients. Lastly, representation error due to unseen(un-stored) state transitions and policies caused by finite storage of Experience-Replay buffer. The AGE can cause the distortion of the gradient estimation, thus derive the agent policy to a worse one. The AGE can also cause a large variability of DQN performance and postpone the process when DQN gets to local optima. To analyse the AGE variance we first propose the variance of approximation gradient for one single sample.

We suppose the gradient estimation from one single sample is $\nabla f_i(\tilde{w}) = AGE_i + \nabla f_i(w_*)$, where i is one training sample from the replay buffer, w_* denotes the exact minimized parameters from current stored samples. We also suppose $\nabla f(w_*)$ denotes the optimal gradient direction given current stored samples, $\mathbb{E}(AGE_i) = 0$, $\mathrm{Var}(AGE_i) = \sigma^2$:

$$\begin{aligned}\mathrm{Var}(\nabla f_i(\tilde{w})) &= \mathrm{Var}(AGE_i + \nabla f_i(w_*)) \\ &= \mathrm{Var}(AGE_i) + \mathrm{Var}(\nabla f_i(w_*)) \\ &= \sigma^2.\end{aligned}$$

To give the argument of approximation gradient variance reduction, we begin by deriving the bound of the variance of $\nabla f_i(\tilde{w})$, suppose that each ∇f_i is L-Lipschitz continuous:

$$\begin{aligned}f_i(\tilde{w}) \geq f_i(w_*) &+ \langle \nabla f_i(\tilde{w}), \tilde{w} - w_* \rangle \\ &+ \frac{1}{2L}\|\nabla f_i(\tilde{w}) - \nabla f_i(w_*)\|^2.\end{aligned}$$

by summing this inequality above over all the training sample i, and divide LHS and RHS by $\frac{1}{n}$, we obtain the bound of approximation gradient error variance $\mathrm{Var}(\nabla f_i(\tilde{w}))$ that:

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(\tilde{w}) - \nabla f_i(w_*)\|^2 \leq 2L(f(\tilde{w}) - f(w_*)) \tag{7}$$

4.2 SVR-DQN in reducing AGE Variance

We continue with Stochastic Variance Reduction for Deep Q-learning, recall the optimized gradient estimation procedure

Algorithm 1 Stochastic Variance Reduction for Deep Q-learning Optimization

procedure STOCHASTIC VARIANCE REDUCTION FOR DEEP Q-LEARNING OPTIMIZATION($B, \eta, m, \alpha, \beta_1, \beta_2, b$)
  Inputs:
  B: Training sample batch size
  η: SVRG learning rate
  m: Number of SVRG inner loop iteration
  α: Adam stepsize
  $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rate for moment estimates
  b: mini-batch size
  Initialization:
  Initialize $\tilde{w}^0 = 0$ (Initialize parameter vector)
  Initialize $m_0 = 0$ (Initialize first moment vector)
  Initialize $v_0 = 0$ (Initialize second moment vector)
  for $s = 0, 1, 2, ...$ **do**
    $B^s = B$ elements sampled without replacement from all training samples ▷ training sample batch
    Calculate the anchor point:
    $\tilde{\mu}^s = \frac{1}{B} \sum_{i \in B^s} \nabla f_i(\tilde{w}^s)$
    $w_0 = \tilde{w}^s$
    for $t = 1, 2, ..., m$ **do** ▷ SVRG variance reduction
      Draw a mini-batch b^t uniformly random from B^s ▷ mini-batch
      Reduce variance and update parameter with mini-batch b_t:
      $w_t = w_{t-1} - \eta(\frac{1}{b} \sum_{i \in b^t} \nabla f_i(w_{t-1}) - \frac{1}{b} \sum_{i \in b^t} \nabla f_i(\tilde{w}^s) + \tilde{\mu}^s)$
    end for
    Calculate the more accurate estimation of gradient direction from sample batch B^s:
    $g_s = w_m - \tilde{w}^s$
    $m_{s+1} = \beta_1 \cdot m_s + (1 - \beta_1) \cdot g_s$ (Update biased first moment estimate) ▷ Adam process
    $v_{s+1} = \beta_2 \cdot v_s + (1 - \beta_2) \cdot g_s^2$ (Update biased second raw moment estimate)
    $\hat{m}_{s+1} = m_{s+1}/(1 - \beta_1^{s+1})$ (Compute bias-corrected first moment estimate)
    $\hat{v}_{s+1} = v_{s+1}/(1 - \beta_2^{s+1})$ (Compute bias-corrected second raw moment estimate)
    $\tilde{w}^{s+1} = \tilde{w}^s - \alpha \cdot \hat{m}_{s+1}/(\sqrt{\hat{v}_{s+1}} + \epsilon)$ (Update parameters)
  end for
  return $\tilde{w}^s$
end procedure

in Algorithm 1:

$$
\begin{aligned}
g_{\mathrm{SVR-DQN}} &= w_m - \tilde{w} \\
&= w_{m-1} - \eta(\frac{1}{b} \sum_{i \in b} \nabla f_i(w_{m-1}) \\
&\quad - \frac{1}{b} \sum_{i \in b} \nabla f_i(\tilde{w}) + \tilde{\mu}) - \tilde{w} \\
&= w_{m-2} - \cdots - \tilde{w} \\
&= w_0 - \sum_{i=0}^{m-1} \tau_i - \tilde{w} \\
&= - \sum_{i=0}^{m-1} \tau_i.
\end{aligned}
$$

where $w_0 = \tilde{w}$ and τ_i denotes $\eta(\frac{1}{b} \sum_{j \in b} \nabla f_j(w_i) - \frac{1}{b} \sum_{j \in b} \nabla f_j(\tilde{w}) + \tilde{\mu})$. Since at each inner iteration mini-batch b is uniform-sampled, for $i \neq j$: $\mathrm{Cov}(\tau_i, \tau_j) = 0$. Therefore,

$$
\begin{aligned}
\mathrm{Var}(g_{\mathrm{SVR-DQN}}) &= \mathrm{Var}(- \sum_{i=0}^{m-1} \tau_i) \\
&= \mathrm{Var}(\sum_{i=0}^{m-1} \tau_i) \\
&= \sum_{i=0}^{m-1} \mathrm{Var}(\tau_i).
\end{aligned}
$$

In order to give the bound of $\mathrm{Var}(g_{\mathrm{SVR-DQN}})$, we derive the bound of $\mathrm{Var}(\tau_i)$:

$$
\begin{aligned}
\mathrm{Var}(\tau_i) &= \mathbb{E}\|\tau_i - \mathbb{E}(\tau_i)\|^2 \\
&\leq \mathbb{E}\|\tau_i\|^2.
\end{aligned}
$$

this inequality is followed the definition of variance that $\mathbb{E}\|X - Y\|^2$ reaches its minority only if $Y = \mathbb{E}(X)$, the above inequality is obtained by setting Y to zero. Next, we assume the anchor point $\tilde{\mu}$ represents $\nabla f(\tilde{w}) = \mathbb{E}(\nabla f_i(\tilde{w}))$, then we have $\tilde{\mu} = \mathbb{E}(\nabla f_i(\tilde{w}) - \nabla f_i(w_*))$, note that $\mathbb{E}(\nabla f_i(w_*)) = 0$

since w_* denotes the optimal solution given current samples. We have

$$\begin{aligned}
\mathbb{E}\|\tau_i\|^2 &= \mathbb{E}\|\eta(\frac{1}{b}\sum_{j\in b}\nabla f_j(w_i) - \frac{1}{b}\sum_{j\in b}\nabla f_j(\tilde{w}) + \tilde{\mu})\|^2 \\
&= \eta^2\mathbb{E}\|\frac{1}{b}\sum_{j\in b}(\nabla f_j(w_i) - \nabla f_j(w_*) \\
&+ \nabla f_j(w_*) - \nabla f_j(\tilde{w}) + \tilde{\mu})\|^2 \\
&\leq \frac{2\eta^2}{b^2}\sum_{j\in b}\mathbb{E}\|\nabla f_j(w_i) - \nabla f_j(w_*)\|^2 \\
&+ \frac{2\eta^2}{b^2}\sum_{j\in b}\mathbb{E}\|\nabla f_j(\tilde{w}) - \nabla f_j(w_*) - \tilde{\mu}\|^2 \\
&= \frac{2\eta^2}{b^2}\sum_{j\in b}\mathbb{E}\|\nabla f_j(w_i) - \nabla f_j(w_*)\|^2 \\
&+ \frac{2\eta^2}{b^2}\sum_{j\in b}\mathbb{E}\|\nabla f_j(\tilde{w}) - \nabla f_j(w_*) \\
&- \mathbb{E}(\nabla f_j(\tilde{w}) - \nabla f_j(w_*))\|^2 \\
&\leq \frac{2\eta^2}{b^2}\sum_{j\in b}\mathbb{E}\|\nabla f_j(w_i) - \nabla f_j(w_*)\|^2 \\
&+ \frac{2\eta^2}{b^2}\sum_{j\in b}\mathbb{E}\|\nabla f_j(\tilde{w}) - \nabla f_j(w_*)\|^2 \\
&\leq \frac{4L\eta^2}{b}(f(w_i) - f(w_*)) + \frac{4L\eta^2}{b}(f(\tilde{w}) - f(w_*)) \\
&\leq \frac{8L\eta^2}{b}(f(\tilde{w}) - f(w_*)).
\end{aligned}$$

The first inequality above follows that $\mathbb{E}\|A+B\|^2 \leq 2\mathbb{E}\|A\|^2 + 2\mathbb{E}\|B\|^2$. The second inequality follows that $\mathbb{E}\|X - \mathbb{E}(X)\|^2 \leq \mathbb{E}\|X\|^2$. The third inequality follows eq. 7 from section 4.1. The fourth inequality follows that $f(w_i) - f(w_*) \leq \alpha^i(f(\tilde{w}) - f(w_*))$ where $\alpha \in (0,1)$, which is proved by Zhang [Johnson and Zhang, 2013].Therefore, we have derive the bound of $\mathrm{Var}(g_{\mathrm{SVR-DQN}})$ that

$$\begin{aligned}
\mathrm{Var}(g_{\mathrm{SVR-DQN}}) &= \sum_{i=0}^{m-1}\mathrm{Var}(\tau_i) \\
&\leq \sum_{i=0}^{m-1}\mathbb{E}\|\tau_i\|^2 \\
&\leq \frac{8Lm\eta^2}{b}(f(\tilde{w}) - f(w_*)).
\end{aligned}$$

Note that for Double DQN, where gradient direction is estimated as the average of B approximation gradients, the bound of its gradient estimation variance can be derived that

$$\begin{aligned}
\mathrm{Var}(g_{\mathrm{Double-DQN}}) &= \mathrm{Var}(\frac{1}{B}\sum_{i\in B}\nabla f_i(\tilde{w})) \\
&= \frac{1}{B^2}\sum_{i\in B}\mathrm{Var}(\nabla f_i(\tilde{w})) \\
&\leq \frac{2L}{B}(f(\tilde{w}) - f(w_*)),
\end{aligned}$$

meaning that SVR-DQN is theoretically more efficient in AGE variance reduction than Double-DQN, and at least $\frac{b}{4m\eta^2 B}$ times better than Double-DQN. Note that the the variance of $g_{\mathrm{SVR-DQN}}$ decreases as learning rate η decreases theoretically. Whereas the η shouldn't be too small or the sample efficiency can be too slow in practice. Therefore, a proper parameters setting should be carefully tuned, in our experiment setting $B = 512, b = 32, m = 32, \eta = 0.01$ respectively.

5 Experimental Results on Atari Games

To demonstrate our method's effectiveness, we evaluate our proposed algorithm on a collection of 20 games from Arcade Learning Environment [Bellemare *et al.*, 2013]. This environment is considered as one of the most challenging datasets because of its high-dimensional state representation [Van Hasselt *et al.*, 2016]. Due to its complexity, it is extremely demanding to find a good learning algorithm which is generally effective across all 20 games with game-specific hyperparameter tuning.

Similar to the previous work [Mnih *et al.*, 2015], we utilize a neural network as the approximation of action value, taking raw images as input. The network architecture is a convolutional neural network with three convolutional layers and a fully-connected layer. The filter sizes of three convolutional layers are 8×8, 4×4 and 3×3 and the strides are 4, 2 and 1, respectively. All the convolutional layers are followed by a rectifier nonlinear layer; The fully-connected layer contains 512 hidden units. To pre-process the raw image, we rescale it into 84×84 from the extracted Y channel.

Following the paper [Mnih *et al.*, 2015], we use the ϵ-greedy scheme for exploration ,where ϵ is annealed linearly from 1.0 to 0.1 over the first million frames. All the experienced transitions are stored in a sliding replay memory, and the algorithm performs gradient descent on mini-batches of 512 transitions sampled uniformly from the reply memory. We set the learning frequency to 128, which means the training process repeats every 128 mini-batches. We also apply a frame-skipping strategy where the network takes the four frames as an input. All experiments are performed on an NVIDIA GTX Titan-X 12GB graphics card. In this paper, we utilize the tunned version of Double DQN algorithm [Van Hasselt *et al.*, 2016], as it somehow resolves the over-estimation issue in Q-learning.

5.1 Evaluation

Our proposed algorithm can obtain more accurate gradient estimation through the same batch of training samples compared to baseline, and we assume that more accurate gradient evaluation should result in more aggressive learning curves

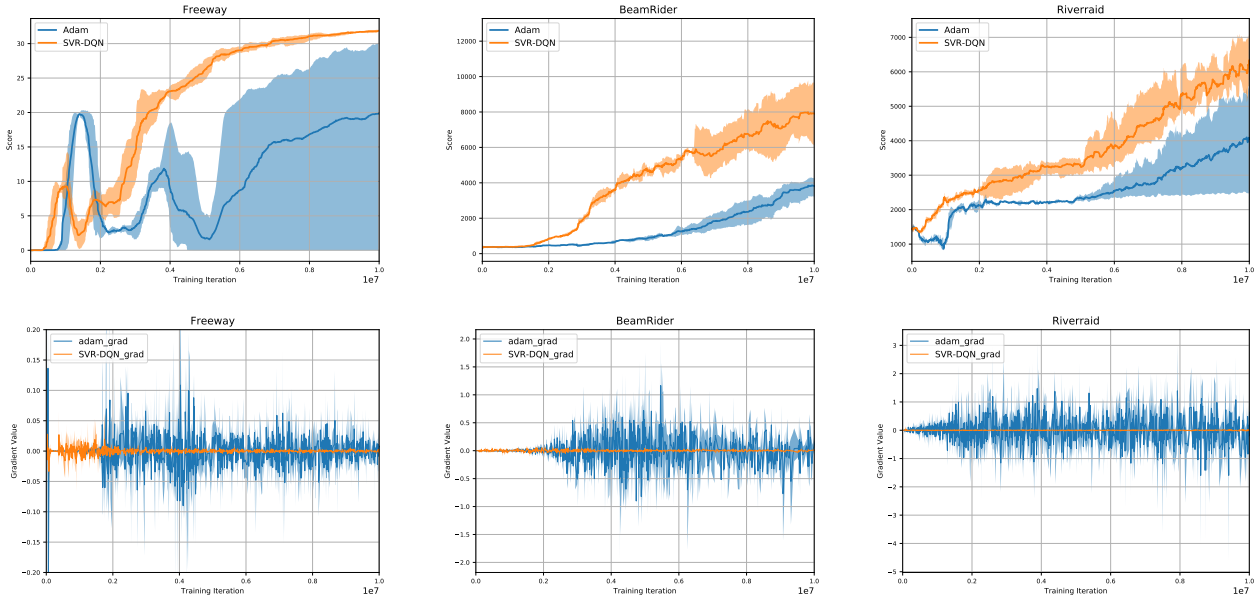

Figure 1: The top row shows the learning curves (in raw score) for the Double DQN with Adam optimizer (blue), SVR-DQN optimizer (yellow), on 3 games of the Atari benchmark suite. The bold lines are averaged over 6 independent learning trials (6 different seeds).The performance test using ϵ-greedy policy with 10 million iterations. The shaded area presents one standard deviation. The bottom row shows that when applied SVR-DQN, the variance of averaged gradient estimation is largely reduced, performance improves, and less variability is observed. 30 no-op evaluation is used and moving average over 4 points is applied. Here x-axis denotes the number of training frames while y-axis denotes the evaluation score in the game.

in the initial training stage. Though the previous work [Mnih *et al.*, 2015] trained their agent using 200 million (200M) frames or 50M training iteration for each game, we choose to train our agent within only 40M frames or 10M training iterations, due to time constraints. Note that regarding evaluating the performance of SVR-DQN, our main concerns focus on the performance in initial stage. Instead of using Double DQN baseline results for those 20 games published from previous work, to obtain fair comparison, we replicate the baseline results using the same hyper-parameter setting, code base, and random seed initialization as SVR-DQN for 10M training frames. The only difference is the optimizer we utilized to minimize the bellman error, where our gradient estimator could lead to a smaller variance. Our experiments could be finished within two days.

Our evaluation procedure follows the description by [Mnih *et al.*, 2015], at the beginning of each episode, the starting states are randomized by executing a random number special of no-op actions, which have no effect on the environment, for 30 times. This procedure is often referred as '30 no-op evaluation' to provide different starting points for the agent. Our agent is evaluated after a maximum of 5 minute gameplay, which contains $18,000$ frames, with the usage of ϵ-greedy policy where $\epsilon = 0.05$. The rewards are the average from 100 episodes. For each game, our agent is evaluated at the end of every epoch (160 epochs in total), and we select the best performance as the agent's final result. To compare the performance of our algorithm to the Double DQN baseline across games, we apply the normalization algorithm proposed by [Van Hasselt *et al.*, 2016] to obtain the normalized improvement score in percent as follows:

	Mean	Median
SVR-DQN	**139.75%**	**118.02%**
Double DQN	92.48%	63.13%

Table 1: Mean and median normalized scores.

$$\text{score}_{\text{normalized}} = \frac{\text{score}_{\text{agent}} - \text{score}_{\text{random}}}{|\text{score}_{\text{Double DQN}} - \text{score}_{\text{random}}|} \quad (8)$$

The detailed results could be found in 2 and 1.

In summary, we adopt the 'Double DQN' and 'random' score reported by [Mnih *et al.*, 2015], the results are demonstrated in Fig. 2. We observe a better performance on 19 out of 20 games, which demonstrates the effectiveness of our proposed algorithm. We also give the summary statistics in terms of mean and median score in Table 1. Compared to Adam, the median performance across 20 games increases from 63.13% to 118.02% and the mean performance increases from 92.48% to 139.75%. Noteworthy examples include Seaquest (from 27.42% to 267.94%), Gopher (from 53.22% to 145.42%).

We also conduct a comparison of the sample efficiency and results could be found in Fig.3. We observe that SVR-DQN boosts the performance on almost all games, and the sample efficiency of SVR-DQN is nearly twice as fast as original Double-DQN with Adam optimizer.

Also, the performances of 3 representative games are reported in Fig.1. The three games include 'BeamRider', 'Free-

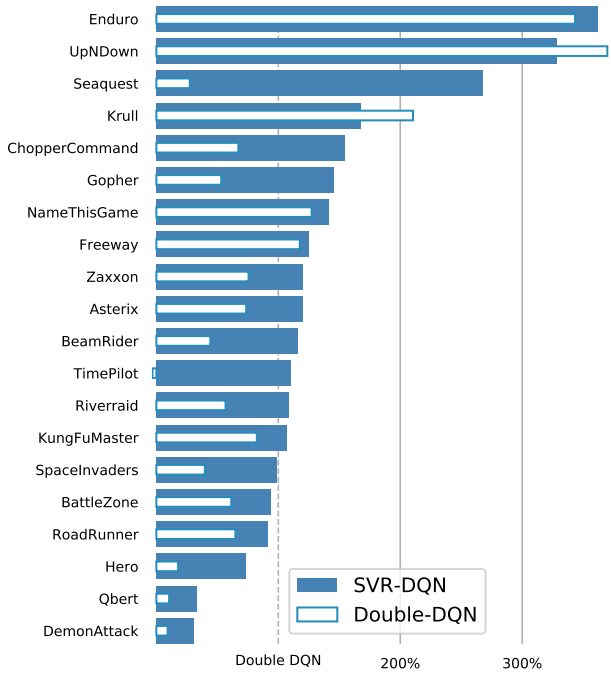

Figure 2: Normalized score on 20 Atari games, tested for 100 episodes per game. The blue bars denotes our SVR-DQN while the white bars denote the Adam optimizer, which is a baseline.

Figure 3: Summary plots of sample efficiency. Median over 20 games of the normalized score achieved so far. The normalized score is calculated in Equation 8.

way', 'Riverraid'. As can be seen in Fig.1 that out proposed SVR-DQN method results in significant lower average gradient estimates, and the variance of gradient is largely reduced. We also observe that our method outperforms the baselines with a significant margin on the majority of the games, and SVR-DQN leads to less variability between the runs of independent learning trials. For the game of Freeway, we see that the divergence of Double-DQN can be prevented by SVR-DQN. On the other hand, the performance of Double-DQN with Adam optimizer has a sudden deterioration in 4M iteration where the gradient variance suddenly increases.

It is noteworthy that SVR-DQN usually leads to aggressive performance improvement at the initial training stage. We believe that our method can be combined with other techniques developed for DQN, such as prioritized experience replay [Schaul *et al.*, 2015], dueling networks [Wang *et al.*, 2015] to further improve the effectiveness.

5.2 Understanding SVR-DQN

We take the investigation of our SVR-DQN algorithm in terms of the performance improvement brought by variance control technique and computation acceleration impacts. Notice all the experimental settings are pre-described in section 5 and are kept constant through all the experiments.

From Fig.1, it is clear that the major weapon helping SVR-DQN outperform significantly than Double-DQN is the usage of stochastic variance reduction strategy. Here the Double-DQN denotes we solely use Adam optimization with the vanilla gradient estimation, this choice allows for improvement in computation efficiency but causes a larger estimation variance due to mini-batch estimation noise, long horizon noise and unknown dynamics, etc. The reduction of the AGE variance is crucial for achieving faster convergence rate. As you can see for Freeway(Fig.1), Double-DQN with vanilla gradient estimation gets stuck at bad local optimal and difficult for the performance to get off the ground due to high variance inaccurate estimation. It can be beneficial for exploration of the parameter space and better performance around current policy with small gradient noise being controlled. However, if the gradient variance is very wild, the performance can be damaged greatly as illustrated in all the three games from Fig.1

Another major weapon is the subsampling strategy to accelerate the computation speed as illustrated throughout our experiments. Proved by [Harikandeh *et al.*, 2015], if we assume the sample variance of the gradients norms is bounded for each iteration, the convergence rate is the constant with the usage of full batch, when the sub-batch size $|B|$ is properly selected. Using subsampling strategy as a building block, we further propagate it into SVRG inner loop using mini-batch sample instead of individual training sample which has large variance among each other. Note that the sample complexity is what we concern more in reinforcement learning experiments, the computation cost of stochastic variance reduction step is negligible compared to simulation time, which is also confirmed throughout our experiments that both SVR-DQN and Double for 10 million iteration can be finished within 2 days across all the tested games, there is no significant training time difference between the two methods.

6 Related Work

In recent years, numerous techniques have been proposed to improve the convergence and stability of deep reinforcement learning and optimization method plays a critical role. The well-known **REINFORCE** [Williams, 1992] uses SGD method. To accelerate the convergence rate and solve the challenges aforementioned, some important improvements are explored. AdaGrad [Duchi *et al.*, 2011] adapts learning rate with respect to the frequency of parameters, and is well-suited for dealing with sparse data. RMSprop [Tieleman and Hinton, 2012] improves AdaGrad by resolving its radically diminishing learning rates. Adaptive Moment Estimate (Adam) [Kingma and Ba, 2014] combines the advantage of both AdaGrad and RMSprop while keeping momentum technique, empirically outperforming other adaptive learning-method algorithms. Under the mechanics of variance control, representative methods such as SAG [Roux *et al.*, 2012] and SDCA [Shalev-Shwartz and Zhang, 2013] are proposed, In terms of convergence rate with dramatic variance reduction and not requiring large gradient storage, Stochastic Variance Reduction [Johnson and Zhang, 2013] significantly outperforms SAG and SDCA. Recently, second-order statistics optimization algorithms are adopted [Battiti, 1992] [Wang and Zhang, 2017]. However, second-order methods are infeasible in practice for high-dimensional training, such as neural network.

6.1 Stochastic Gradient-based Optimization

By far, stochastic gradient descent is a common method for neural networks optimization [Kingma and Ba, 2014]. Many optimization problems can be summarized as finding the minima or maxima of scalar objective function $J(\theta)$. Gradient descent updates the parameters in the opposite direction of the gradient of $J(\theta)$ until reaching a minimum by evaluating the gradient over the full training set. However, objective functions are often stochastic as they are composed of different subfunctions [Kingma and Ba, 2014]. In such cases, stochastic gradient descent (SGD) improves gradient descent by computing gradients with a single or a few training examples and takes gradient steps through individual subfunctions [Bottou, 2010]. Although SGD exhibited its efficiency in many machine learning successful stories, there are still some key challenges wait to be solved, including choosing a proper learning rate schedule and avoiding to get trapped in numerous suboptimal local minima in non-convex neural networks training [Choromanska *et al.*, 2015]. Therefore, efficient stochastic optimization techniques are required.

6.2 Variance Reduction in Deep Q Learning

Since the existence of variance in deep Q learning procedure can largely deteriorate the performance of the network, there have been a number of techniques proposed to reduce varieties of variance in deep Q learning, such that the convergence, running time and stability are enhanced.

The well-known variance in DQN is the Q learning overestimation error, which is first investigated by [Baird III, 1993], who has showed that since action values contain random errors distributed in the interval $[-\epsilon, \epsilon]$. Since the DQN target is obtained using max operator, the expected overestimation errors are bound by $\gamma \in \frac{n-1}{n+1}$, where n is the applicable action numbers given current state s. the intuition nature of overestimation error is that it can cause asymptotically sub-optimal policies, as shown by [Baird III, 1993] and later by [Van Hasselt *et al.*, 2016] that noisy in Arcade Learning Environment can lead to overestimation. The Double DQN [Van Hasselt *et al.*, 2016] is a possible way to tackle overestimation error which replaces the positive bias with a negative one, where two Q-network are applied for Q action selection and Q function value calculation respectively.

Another variance in DQN is the Target Approximation Error (TAE), which is investigated by [Anschel *et al.*, 2016] that if we denote the term of gap between optimal estimated Q value and current Q value $\triangle = Q(s, a; \theta) - Q^*(s, a)$, and decompose it as $Q(s, a, \theta) - y_{s,a} + y_{s,a} - \hat{y}_{s,a} + \hat{y}_{s,a} - Q^*(s, a)$, where $y_{s,a}$ is the DQN target and $\hat{y}_{s,a}$ is the true target. And TAE is the gap between $Q(s, a, \theta)$ and $y_{s,a}$. The TAE is result of sub-optimality of θ due to inexact minimization and limited representation power of DQN. A efficient method to reduce TAE variance is Average DQN [Anschel *et al.*, 2016], the key idea is to use the K previously calculated Q-values to estimate the current action-value estimate. The averaging reduces the TAE variance and leads to more stability in learning process.

In addition to the aforementioned kinds of variance in deep Q learning, a recent explored variance is caused by noisy in reward signals in real experiment settings, which is investigated by [Romoff *et al.*, 2018]. In order to reduce reward signal variance, a direct reward estimator $\hat{R}(s_t)$ is proposed to update the discounted value function instead of sampled reward. Note that in discrete tabular case, the reward estimator is corresponded to using sampled mean.

Our stochastic variance reduction for deep Q learning method differs from all of the aforementioned approaches. The key idea of our method is to reduce the variance caused by approximate gradient estimation, and thus greatly improve the efficiency and performance.

7 Conclusion

In this paper we proposed an innovative optimization algorithm for Q-learning which reduces the variance in gradient estimation, our proposed optimization algorithm achieves significantly faster convergence than the Adam optimizer. Our method significantly improves the performance of Double DQN on the Atari 2600 domain. In the future, we plan to investigate the impact advanced constrained optimization and explore the potential synergy with other techniques.

References

[Anschel *et al.*, 2016] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. *arXiv preprint arXiv:1611.01929*, 2016.

[Baird III, 1993] Leemon C Baird III. Advantage updating. Technical report, WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.

[Battiti, 1992] Roberto Battiti. First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166, 1992.

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013.

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Choromanska *et al.*, 2015] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[Dai *et al.*, 2005] Xiaohui Dai, Chi-Kwong Li, and Ahmad B Rad. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 6(3):285–293, 2005.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[Frank and Claus, 2006] Michael J Frank and Eric D Claus. Anatomy of a decision: striato-orbitofrontal interactions in reinforcement learning, decision making, and reversal. *Psychological review*, 113(2):300, 2006.

[Harikandeh *et al.*, 2015] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečnỳ, and Scott Sallinen. Stopwasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pages 2251–2259, 2015.

[Johnson and Zhang, 2013] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

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

[Maes and Kozierok, 1993] Pattie Maes and Robyn Kozierok. Learning interface agents. In *AAAI*, volume 93, pages 459–465, 1993.

[Mnih *et al.*, 2015] 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*, 518(7540):529–533, 2015.

[Romoff *et al.*, 2018] Joshua Romoff, Alexandre Piché, Peter Henderson, Vincent Francois-Lavet, and Joelle Pineau. Reward estimation for variance reduction in deep reinforcement learning. *arXiv preprint arXiv:1805.03359*, 2018.

[Roux *et al.*, 2012] Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[Schaul *et al.*, 2015] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[Shalev-Shwartz and Zhang, 2013] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Tieleman and Hinton, 2012] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016.

[Wang and Zhang, 2017] Jialei Wang and Tong Zhang. Improved optimization of finite sums with minibatch stochastic variance reduced proximal iterations. *arXiv preprint arXiv:1706.07001*, 2017.

[Wang *et al.*, 2015] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[Zhu *et al.*, 2017] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE, 2017.