ICLR 2018 Challenge TD LEARNING WITH CONSTRAINED GRADIENTS (Paper ID -Bj-ofQZRb)

Raihan Seraj - raihan.seraj@mail.mcgill.ca - 260752605
Alan An - chengyun.an@mail.mcgill.ca - 260683828
Zahra Khani - zahra.khanidahaj@mail.mcgill.ca - 260816351
Team-**ZRA**

Abstract— **In efforts to ensure that published results are reliable and reproducible in Machine Learning research, we investigated the reproducibility of empirical results of a paper submitted to the International Conference on Learning Representations 2018. The paper is under the title, *TD LEARNING WITH CONSTRAINED GRADIENTS* (Paper ID -Bj-ofQZRb) in which the authors proposed a new algorithm that puts constraints on the gradient updates of the parameters for Temporal Difference Learning with function approximation. In this work we provide a detailed methodology that we have adapted while trying to reproduce the experiments in the paper itself.**

I. INTRODUCTION

Reinforcement learning is the study of how an agent interact with the environment in order to come up with a policy π that is a mapping from state $\mathcal{S}$ to actions $\mathcal{A}$ that would maximize expected cumulative rewards [1].Reinforcement learning has experienced dynamic growth in attention and interest due to promising results achieved in continuous control task such as playing atari or beating a human expert in the game of GO. However [2] addresses the difficulties in reproducing DeepRL experiments as the literature presents a wide range of results with the same baseline algorithms.

Since reproducibility can be affected by both the extrinsic parameters (code bases and hyper parameters) and intrinsic factors(effects of random seeds and environment), a detailed analysis is required in order to provide sufficient justification of the reproducibility of the experiments. In this work we investigate the reproducibility of the paper titled *'TD Learning with Constrained Gradients'* which was submitted at ICLR 2018.
In the following sections we outline the main contributions of the paper and comment on the clarity of the techniques used in the paper which plays an important role in reproducing any scientific papers. We further outline the methodologies that we have incorporated and show detailed analysis of our effort to reproduce the experiments in the paper.

II. PAPER OUTLINE

A. *Technical Summary*

The paper on "*TD Learning with Constrained Gradients*" emphasize on putting a constraint on the update of gradients for TD learning with function approximation. The authors propose a constrained update that incorporates a gradient which causes minimal changes to the values of the next state sharing common features. Thus the proposed constrained update reduces the TD error in Temporal Difference Learning learning methods and at the same time causes minimal changes to the value of the next state. The paper therefore mentions that TD learning with function approximation incorporating the constraint gradients have better convergence properties, compared to baseline algorithms.In particular with the constrained gradients one does not need a target network to stabilize deep networks for approximating value functions.

B. *Governing Equations*

The main equations for constraining the updates is as follows

$$g_{update}(s_t) = g_{TD}(s_t) - \prod g_{TD}(s_t) \tag{1}$$

$$\hat{g_v}(s_{t+1}) = \frac{g_v(s_{t+1})}{\|g_{TD}(s_{t+1})\|} \tag{2}$$

$$\prod g_{TD}(s_t) = (g_{TD}(s_t) \cdot \hat{g_v}(s_{t+1})) \times \hat{g_v}(s_{t+1}) \tag{3}$$

$$\theta_{t+1} = \theta_t - \alpha g_{update}(s_t) \tag{4}$$

Where $g_{TD}(s_t)$ is the gradient of the TD loss with respect to the state s_t and parameters θ_t, $\hat{g_v}(s_{t+1})$ is the gradient at s_{t+1} that will change the value most and $g_{update}(s_t)$ is the gradient update orthogonal to $g_v(s_{t+1})$

C. *Experimental Results*

This paper presents experimental results in three domains.

1) **DQN in CartPole**
Experiments were done and a comparison was made with DQN and Constrained DQN in the CartPole environment. The results presented in the paper shows that Constrained DQN has better convergence properties and less variance over an average of 10 runs in the

experiments. The authors used RMS-Prop for their optimization method compared to adam. The following figure presents the results of the DQN and Constrained DQN experiments

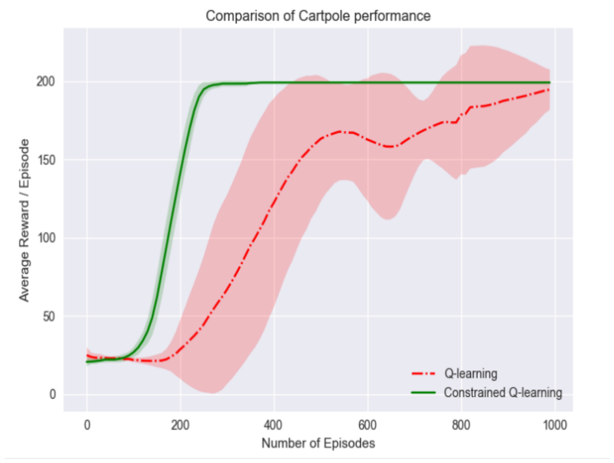


Fig. 1. Comparison of DQN and Constrained DQN for CartPole

2) **DQN in Grid World**
A 10×10 GridWorld domain has been used with a reward of 1 at $(0, 4)$ and 0 other place. A Deep Q network with 2 hidden units were used for approximation of the value function. The values estimated by the Deep Q Network and the Constrained Deep Q Network has been reported, the results were compared with the values obtained by policy evaluation on the GridWorld domain and it was shown that the mean squared error for the Constrained DQN is less compared to DQN.

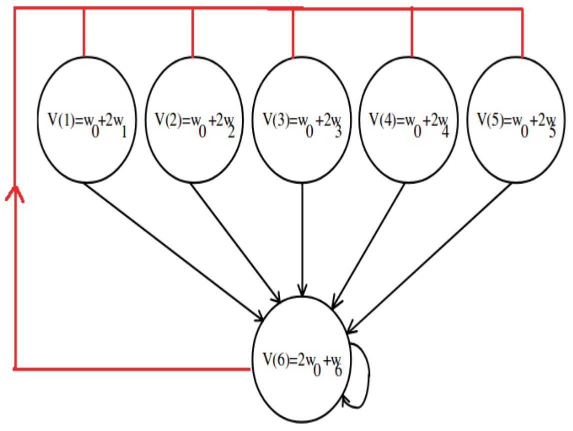


Fig. 2. Baird's counterexample illustration

3) **Baird's counterexample**

Baird's counterexample is the simplest system that TD gradient descent with function approximation would fail and diverge. More concretely,Baird's counterexample can be considered as an episodic six-state,two-action MDP shown in Figure 2. The red action takes the system to one of the upper states with equal probability, whereas the black action takes the system to the lower state. The behavior policy selects the action with probability $\frac{5}{6}$ and $\frac{1}{6}$ respectively.
The reward is zero everywhere and discount rate $\gamma = 0.99$. The target policy should always take the black action. The authors experimented the Baird's counterexample with regular Q-learning, residual gradient algorithms along with their proposed modification on these two algorithms.
It is reported that they made 10 independent runs with 2000 iterations per run. The author reported that with their constrained technique, both Q-learning and residual gradient converged and have an identical overlapping result graph. However, both of them failed to converge to the ideal value, zero, and the converged value remains unknown as it has not been reported.

III. Reproducibility Methodology

We follow the network structure as mentioned in the paper closely and try to reproduce all the three experiments that has been mentioned in the paper. For every experiment, we try to duplicate both the baseline results and results produced with the authors' technique. However, we faced many different difficulties in each experiment we ran, ranging from unavailabilities of the code base to missing report of hyper-parameters. It is therefore the best choice to report in a case by case fashion.

1) **CartPole**
First we tried the open-AI baseline on the CartPole environment. Then we made an effort to implement the code that incorporates the constrained updates with the information mentioned in the paper. The authors presented a neural network with 2 hidden layers of size $[5, 32]$ and used RMS-Prop as the optimization technique, however we did not find any concrete details pertaining to the learning rate and the batch size. Moreover, we had no information whether the convolution neural network required any form of regularization such as dropout or batch normalization.Apart from that the activation function used for each of the hidden nodes are also not reported in the paper.
We therefore assumed the default hyper parameters already present in the open-AI baselines and change the network architecture to 2 hidden units with size $[5, 32]$.An average over 10 runs were taken and the experiment was performed with 1000 episodes which is also similar to the configuration as outlined in the paper. We made an effort to incorporate constrained gradient updates in DQN for CartPole as well as other for other environments such as the Mountain Car environment. However, we did not make enough progress on this .

2) **GridWorld**
In order to reproduce DQN with Grid World environment we follow an approach of reproducing the exact environment as that mentioned in the paper. Thus

we started of in reproducing a 10×10 Grid World environment with a reward of only 1 present at the goal state $(0, 4)$. However since the start state of the agent was not explicitly mentioned in the paper, the agent always starts in the environment at $(0, 0)$.
We had some issue to incorporate the constrained algorithm in the DQN setting, so We only ran DQN and regular Q learning for this Grid World problem.
We then made an effort to compare the mean squared error between the value function approximated by our baseline DQN and baseline DQN of the authors' with respect to the value function calculated by policy evaluation in the Gridworld domain. We used a deep network with two hidden layers, each with 32 units to approximate the Q-values. The performance plot comparing Q learning and constraint Q learning was not given explicitly in the paper, we therefore followed a notion of plotting all the performance curves comparing the performance of DQN and Q learning without any function approximation.

3) **Baird's counterexample**
We found a github repo created by Zhang [4] to do the Baird counterexample simulation. We verified the code and modified based on the six-state MDP as mentioned in the paper Specifically, we implemented the TD learning and the constrained TD-learning ourselves based on the provided environment. We ran the algorithm on the Baird-6-state setup for 2000 steps for 10 independent runs and compared our results with the ones the authors claimed. To best match the experiment conducted by the authors, we run Baird's counterexample using TD off-policy learning with linear function approximation. We set the discount factor $\gamma = 0.99$, the learning rate $\alpha = 0.01$.
In addition we extracted the feature values from the Fig.2, for example the feature values for the lower state is a 7-dimension vector with sixth entry being 1 and seventh entry being 2. Note that the seventh entries of all the feature vectors correspond the coefficient of w_0 in every state. We initialized the weights according to what Sutton suggested in his book. [5] The main reproducibility difficulty we faced was similar to the ones we faced in other experiments. The authors did not report the hyper-parameters, namely the discount factor and learning rate they used. More importantly, they did not report how they initialized the weights as well. It is thus quite difficult to fully duplicate their experiments.

In addition, we found it quite difficult to understand the main update equations for their proposed methodology. In equation seven of their paper, they have used an expression $g_v(s_{t+1})$. However, in the paper itself, it was not clearly mentioned how this expression calculated exactly, yet this is the very key component to understand their algorithm. After some research, we found out that the authors have replaced the $g_v(s_{t+1})$ with $gTD(s_{t+1})$ in their revised version of this paper that was submitted to the Deep Reinforcement Learning Symposium at NIPS. Only after this discovery were we able to proceed with the implementation. Note that due to time constraint, we did not implement the residual gradient method and the constrained gradient method. The also mentions that their technique would not only work on the three experiments mentioned above, but also work on gradients of any TD(0) objective, and can be easily applied to both linear and non-linear function approximations. We incorporated the the constrained gradient updates for both Q learning and SARSA and tested this claim in Mountain Car environment with Linear function approximation.

4) **Mountain Car**
We ran simple Q learning and SARSA with Linear function approximation where we have used an RBF kernel to featurize the state. Since there were no experiments associated with linear function approximation with this environment we can only compare SARSA and Q learning with the constrained SARSA and Q learning as mentioned in the paper.

IV. Empirical results

1) **CartPole** We use the baselines and tried to reproduce the baseline results at first. We followed the network architecture comprising of 2 hidden units with sizes $[5, 32]$ and with RMS-Prob used as an optimization technique,since the batch size were not explicitly we took the default batch size and took 10 independent runs and averaged them to get the following results. We took the learning rate of 10^{-4} as outlined by the paper and had the following results.

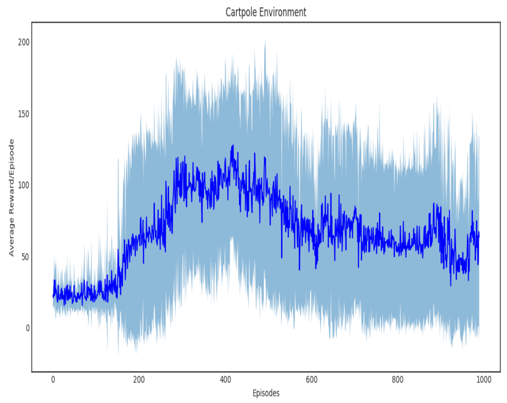
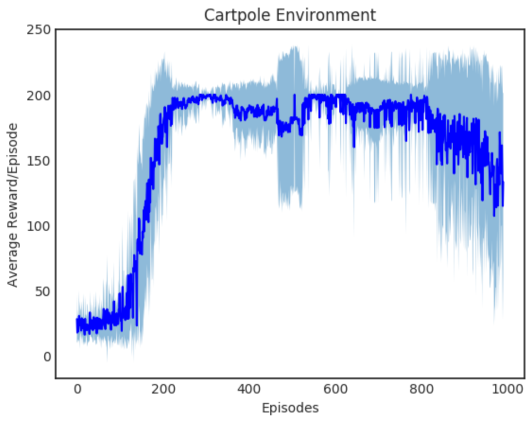
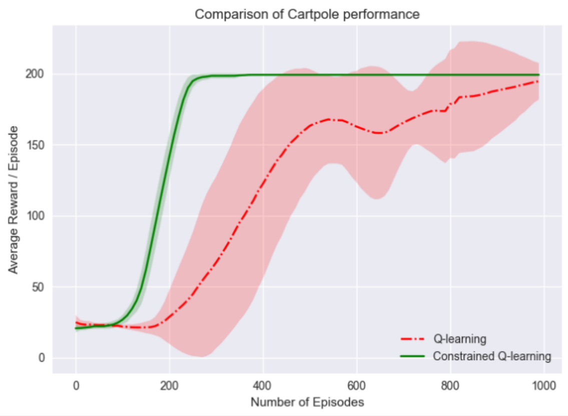


Fig. 3. CartPole baselines

However it is interesting to note that a network architecture with 2 hidden units of size $[5, 32]$ is not an optimal one, and the model under performs. In an effort to reproduce the results, we next changed the network architecture to a single hidden unit with size 64 and took the average of 10 independent runs keeping all other parameters the same as that mentioned in the paper. We ended up getting baseline results that is as good as the results claimed by the authors with an approach of constraining the gradients in terms of the convergence of the mean.



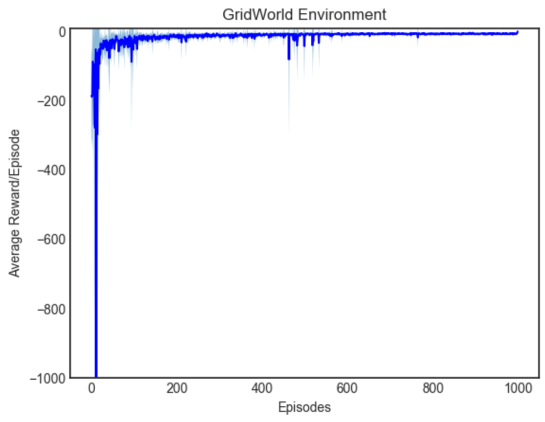
(a) Reproduced Baseline



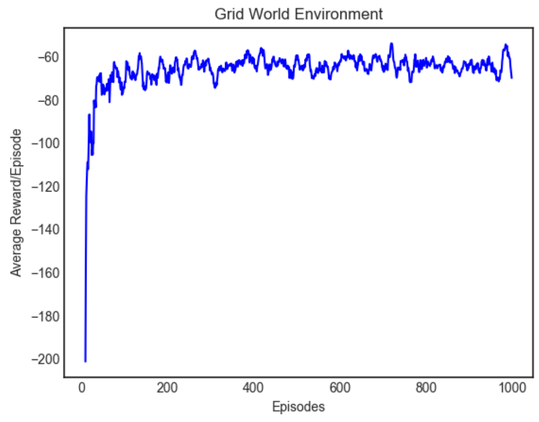
(b) Baseline in Paper

Fig. 4. Comparison of Reproduced Baseline with 1 hidden unit and Author's Claimed Baseline Results

2) **GridWorld** The Grid World environment has been coded and replicated as mentioned in the paper.We effectively ran DQN in the 10×10 Grid World environment as proposed by the authors.For DQN we used a multilayer perceptron with two hidden units of size 32 for approximating the Q functions which was mentioned in the paper. We execute a soft max policy and feed the (x, y) coordinates of the agent in the network. Since the author did not mention the total episodes for which they ran DQN and constrained DQN in Grid World environment, we therefore ran them over 1000 episodes and took an average over 10 independent runs as before.



(a) DQN with Grid World Environment



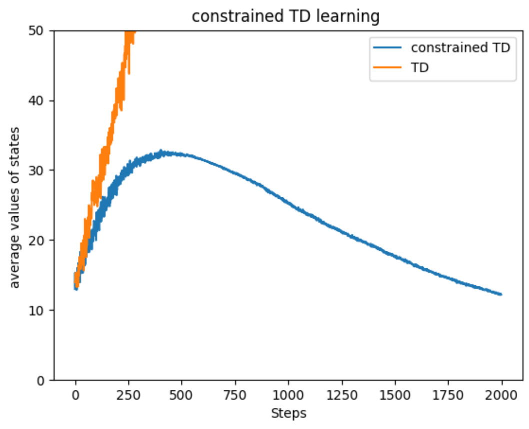
(b) Q learning without function approximation

Fig. 5. Comparison of DQN with Q learning in 10×10Grid World

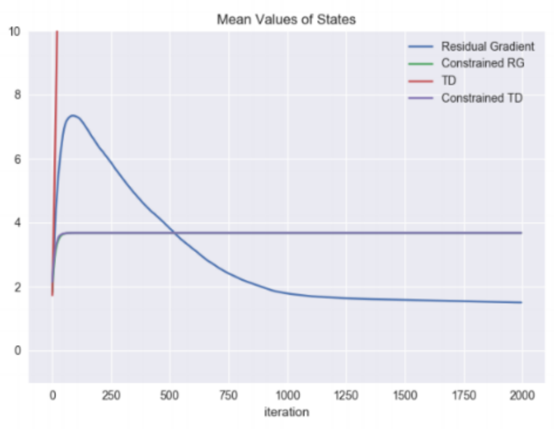
We computed the Q values for DQN with that of the value function obtained by running policy evaluation in this domain, and obtained a mean squared error of 0.38 ± 0.0032 which is not exact to what the authors mentioned for the baseline but reasonably close. We also included the result of regular Q learning. The comparison of the results suggests that DQN performs better than regular Q learning in this problem thus justifies the choice of using DQN.

3) **Baird's Counterexample** We did observe the diverging behavior when running regular TD learning on Baird's counterexample. We obtained similar baselines to that mentioned in the paper. We also observed that the constrained TD does not converge to the true value of 0. Nevertheless, we found that the shape of our result is quite different compared to the ones mentioned in the paper. More specifically,they reported a converging straight line since iteration 100. However, we observed a decreasing curve. There are many plausible interpretations behind this since there was a lack of clarity of the expressions that are actually involved in the gradient update. Hence duplicating their results was quite difficult. The results for out experiments is

shown in the following figure



(a) Reproduced TD and TD constrained



(b) Reported Results in the Paper

Fig. 6. Comparison of Reproduced Baseline with 1 hidden unit and Author's Claimed Baseline Results

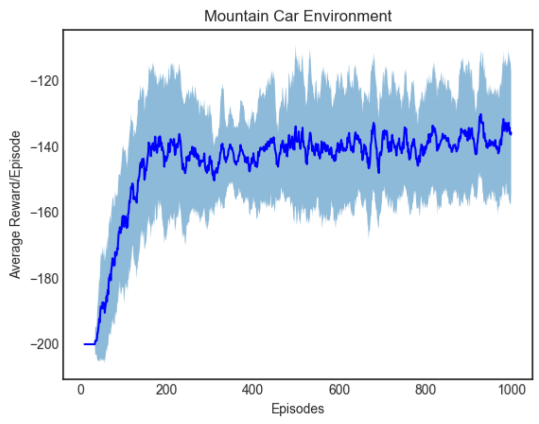
4) **Linear Function Approximation** With the given theory, we also tried some experiments with linear function approximation in mountain car environment. The following results were obtained with an average over 10 independent runs. We ran both Q learning and SARSA and tried to implement the Constrained SARSA and Constrained Q learning. However for constrained SARSA and constrained Q learning we did not get any satisfactory results with linear function approximation. In the following sections we outline in detail the difficulties we faced while implementing the constrained updates as mentioned by the authors.

A. *Difficulties in Reproducing the results*

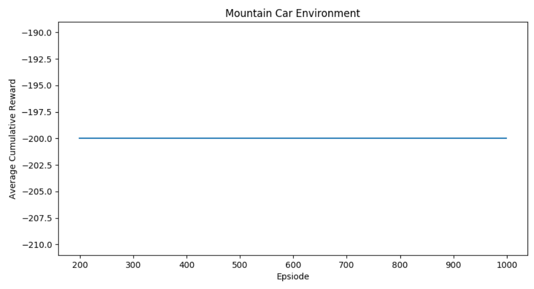
The paper outlines the equation to the constrained gradient updates. However in Equation (2) they have incorporated an additional term $g_v(s_{t+1})$ the paper however only mentions that this term is the gradient of the value of the next state that will change value the most. In order to get further clarification we contacted the authors and they assured that this term is calculated by taking the gradient of the TD error with respect to target.

However, in terms of Q learning the target is simply a max function over the next state action pair, and in such cases this max function is not differentiable. Further more we tried a different approach and used SARSA in which case the target is differentiable. For our implementation we use RBF kernel for approximating value functions. We have incorporated a learning rate of $\alpha = 0.01$ and a discount factor of 0.99 for our implementation.

For SARSA with Mountain Car and Constrained SARSA with Mountain Car we get the following results.



(a) SARSA with Linear Function Approximation in Mountain Car



(b) Constrained SARSA with Linear Function Approximation in Mountain Car

Fig. 7. Comparison of SARSA and Constrained SARSA with Linear Function Approximation

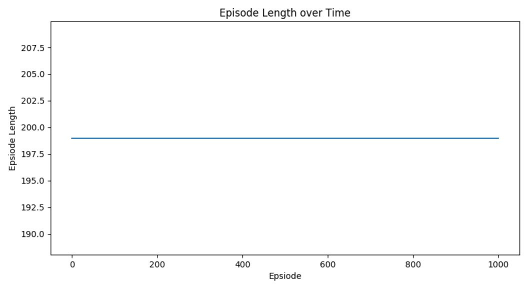


Fig. 8. Episode Length for Constrained SARSA in Mountain Car Environment

The figure above for the constrained case shows that no learning seems to be occurring for the mountain car environment. We tracked the results and realized that the length of the episode is always constant to 200 steps. For the open-AI gym environments the episodes are terminated internally when a maximum step size of 200 is reached. Therefore it might be the case that due to the hard constraints on the gradients, no learning is actually taking place with in the length of an episode defined in open-AI gym.

V. Discussion

We tried to reproduce the results as mentioned in the paper. Since there were no code available for the experiments. We discussed with the authors regarding the availability of the code base and was assured that it is going to be released soon. The paper only mentions that the experiments were performed with open-AI baselines and did not specify the versions that they were using. For our work we have used the version up to date and conducted the experiments. The experiments with DQN did not mention the random seed or the hyper parameters except for the network architecture and the optimization technique that they have used. We did not have clear information regarding the learning rate, batch size which are essential for reproducing the experiments. In terms of the clarity of the expression, the paper does not clearly mention how they are calculating the $g_v(s_{t+1})$ they simply mention that it is the gradient at s_{t+1} that will change the value most.

Also they did not mention clearly about their method of calculating $g_{TD}(s_{t+1})$ which seems to be the gradient of the TD error with respect to the next state. However, after communicating with the authors we found that $g_{TD}(s_{t+1})$ is the gradient of the same TD error with respect to the target and all the experiments are done by taking a single step within the environment. We had difficulties in figuring out how this translates to Q learning because the target is a max operator applied to the next state action pair. Under such circumstances we will not be able to differentiate with respect to the target. However the experiments for the constrained update in case of Baird Counter example was done by taking two steps with in the environment and calculating $g_{TD}(s_{t+1})$ with respect to the next state.

Regarding the Computation time and memory requirements and in terms of convergence guarantees for the constrained methods, the paper does not outline any computation time requirements or the memory requirements.

Appendix

Openreview executive summary In efforts to ensure that published results are reliable and reproducible in Machine Learning research, we investigated the reproducibility of empirical results this paper. We tried to reproduce the experimental results shown in the paper. However, there were some difficulties we faced.

1) The notation and equations shown in the paper lack of clarity. For example, the authors did not mathematically define the variable $g_v(s_{t+1})$, they described it as the gradient at s_{t+1} that will change the value most. After some research, we found out that the authors have replaced the $g_v(s_{t+1})$ with $gTD(s_{t+1})$ in their revised version of this paper that was submitted to the Deep Reinforcement Learning Symposium at NIPS. Only after this discovery were we able to proceed with the implementation.
2) There are no clear mention of how they are calculating $g_{TD}(s_{t+1})$ which seems to be the gradient of the TD error with respect to the next state. However, after communicating with the authors we found that $g_{TD}(s_{t+1})$ is the gradient of the same TD error with respect to the target and all the experiments are done by taking a single step within the environment. We had difficulties in figuring out how this translates to Q learning because the target is a max operator applied to the next state action pair. Under such circumstances we will not be able to differentiate with respect to the target.
3) There were no code available for the experiments by the time we finished this report. We discussed with the authors regarding the availability of the code base and was assured that it is going to be released soon.
4) The authors did not report all the hyper-parameters they used in their experiments.

Although we were not able to fully duplicate the experiments conducted by the authors due to the above reasons, we would like to share what we did and our findings. You can check out the full report at https://www.overleaf.com/read/tdzyfmjzkhyj

1) Cartpole : we used the exact set of hyper-parameters reported by the authors. In addition, we used the default open-AI batch size as it is not mentioned by the authors. The baseline we got is quite different than the one of the authors'. However, interestingly, a model with single hidden layer of 64 units got us a baseline result that is as good as the results claimed by the authors.
2) **GridWorld** The Grid World environment has been coded and replicated as mentioned in the paper. We have run DQN in the 10×10 Grid World environment as proposed by the authors. Since the authors did not mention the starting point they used, we set it to be $(0, 0)$ For DQN we used a multilayer perceptron with two hidden units of size 32 for approximating the Q functions which was mentioned in the paper. We executed a soft max policy and feed the (x, y) coordinates of the agent in the network. Since the authors did not mention the total episodes for which they ran DQN and constrained DQN in Grid World environment, we therefore ran them over 1000 episodes and took an average over 10 independent runs as before.
   We computed the Q values for DQN with that of the value function obtained by running policy evaluation

in this domain, and obtained a mean squared error of 0.38 ± 0.0032 which is not exact to what the authors mentioned for the baseline but reasonably close. Note that we only verified the DQN baseline, we did not verify the proposed algorithm in the DQN setup. We also ran the environment with regular Q learning. The comparison of the results suggests that DQN performs better than regular Q learning in this problem thus justifies the choice of using DQN.

3) **Baird's counterexample** We ran both TD and constrained TD proposed by the authors on the Baird-6-state setup for 2000 steps each run and we made 10 independent runs.Specifically we ran Baird's counterexample using TD off-policy learning with linear function approximation. We set the discount factor $\gamma = 0.99$, the learning rate $\alpha = 0.01$. In addition we extracted the feature values from the state graph shown in the paper and we initialized the weights the same way Sutton did in his book for the Baird's counterexample section. We did observe the diverging behavior when running TD learning on Baird's counterexample. We obtained a similar baseline to that mentioned in the paper. We also observed that the constrained TD does not converge to the true value of 0. Nevertheless, we found that our constrained TD produced a quite different shape than the one shown in the paper. More specifically, they reported a converging straight line after iteration 100. However, we observed a bell shape, heavy tail curve.
4) **linear function approximation** The authors claimed that the constraint can be applied to the gradients of any TD objective. Thus we also tried some experiments with linear function approximation in open-AI mountain car environment (max size of step =200). We made 10 independent runs and took the average. We ran both Q learning and SARSA and tried to implement the Constrained SARSA and Constrained Q learning. The author claimed $g_v(s_{t+1})$ is calculated by taking the gradient of the TD error with respect to target. However, in terms of Q learning the target is simply a max function over the next state action pair, and in such cases this max function is not differentiable. Further more we tried a different approach and used SARSA in which case the target is differentiable as desired. For our implementation we use RBF kernel for approximating value functions. We have incorporated a learning rate of $\alpha = 0.01$ and a discount factor of 0.99 for our implementation. It is important to note that we observed no signs of learning for Constrained SARSA in this environment.

In conclusion, we hope that our above findings are helpful to the authors as well as people who are interested in this paper. We encourage the authors to publish their code and provide more details about hyper-parameters for their work.

References

[1] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.

[2] Henderson, Peter, et al. "Deep reinforcement learning that matters." arXiv preprint arXiv:1709.06560 (2017).

[3] `http://github.com/openai/baselines`

[4] `https://github.com/ShangtongZhang/reinforcement-learning-an-introduction/commit/2b2badc35d5401d6ba1ad7fb3294dc57ffa1b170`

[5] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.