Papers

Deep RL -

- **DQNs**

1. Playing Atari with Deep Reinforcement Learning [2013] - https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
2. Deep Reinforcement Learning with Double Q-learning [2015] - https://arxiv.org/pdf/1509.06461.pdf
3. Dueling Network Architectures for Deep Reinforcement Learning [2015] - https://arxiv.org/pdf/1511.06581.pdf
4. Noisy Networks for Exploration [2018] - https://arxiv.org/pdf/1706.10295.pdf
5. Rainbow: Combining Improvements in Deep Reinforcement Learning [2018] - https://arxiv.org/pdf/1710.02298.pdf

- **Experience Replay**

1. Prioritized Experience Replay (PER) [2015] - https://arxiv.org/pdf/1511.05952.pdf
2. A Deeper Look at Experience Replay (CER) [2017] - https://arxiv.org/pdf/1712.01275.pdf
3. Reward Backpropagation Prioritized Experience Replay [2017] - https://web.stanford.edu/class/cs234/past_projects/2017/2017_Zhong_Wang_Wang_Reward_Backpropagation_Paper.pdf
4. Not All Samples Are Created Equal: Deep Learning with Importance Sampling [2018] - https://arxiv.org/pdf/1803.00942.pdf
5. Prioritized Sequence Experience Replay [2019] - https://arxiv.org/pdf/1905.12726.pdf
6. Advances in Experience Replay [2018] - https://arxiv.org/pdf/1805.05536.pdf
7. Distributed prioritized experience replay [2018] - https://openreview.net/pdf?id=H1Dy---0Z
8. Experience Replay Optimization [2019] - https://www.ijcai.org/Proceedings/2019/0589.pdf
9. Importance Resampling for Off-policy Prediction [2019] - https://arxiv.org/pdf/1906.04328.pdf
10. Revisiting fundamentals of experience replay - https://arxiv.org/pdf/2007.06700.pdf

- **Continuous Action Space / Policy Gradient Methods**

1. Continuous Control with Deep RL (DDPG) [2015] - https://arxiv.org/pdf/1509.02971.pdf
2. Asynchronous Advantage Actor-Critic method [2016] - https://arxiv.org/pdf/1602.01783.pdf
3. Addressing Function Approximation Error in Actor-Critic Methods (TD3) [2018] - https://arxiv.org/pdf/1802.09477.pdf
4. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor [2018] - https://arxiv.org/pdf/1801.01290.pdf

- **Distribution instead of expectation of reward**

1. A Distributional Perspective on Reinforcement Learning [2018] - https://arxiv.org/pdf/1707.06887.pdf

- **Intrinsic Reward**

1. Unifying Count-Based Exploration and Intrinsic Motivation [2016] - https://arxiv.org/pdf/1606.01868.pdf
2. Curiosity-driven Exploration by Self-supervised Prediction [2017] - https://arxiv.org/pdf/1705.05363.pdf
3. Exploration by Random Network Distillation [2018] - https://arxiv.org/pdf/1810.12894.pdf

- **Hierarchical**

1. Option Critic Architecture [2016] - https://arxiv.org/abs/1609.05140
2. FeUdal Networks for Hierarchical Reinforcement Learning [2017] - https://arxiv.org/pdf/1703.01161.pdf

Survey Paper -

1. A Brief Survey of Deep Reinforcement Learning [2017] - https://arxiv.org/pdf/1708.05866.pdf
2. Deep RL - An Overview [2018] - https://arxiv.org/pdf/1810.06339.pdf

Classical RL -

1. Policy Gradient Methods for Reinforcement Learning with Function Approximation [Policy Gradient Theorem and Proof of Convergence] - https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf
2. An Analysis of Temporal-Difference Learning with Function Approximation [Proving Convergence of TD methods with linear functional appx] - http://web.mit.edu/jnt/www/Papers/J063-97-bvr-td.pdf
3. A Natural Policy gradient - https://papers.nips.cc/paper/2073-a-natural-policy-gradient.pdf
4. Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time - https://pdfs.semanticscholar.org/6086/ab80f6d2ba77dd7da65c3cadf8cf5fb019ae.pdf
5. Issues in Using Function Approximation for Reinforcement Learning - https://www.ri.cmu.edu/pub_files/pub1/thrun_sebastian_1993_1/thrun_sebastian_1993_1.pdf

Survey Papers -

1. Reinforcement Learning A Survey - https://www.cs.cmu.edu/~tom/10701_sp11/slides/Kaelbling.pdf
2. Reinforcement Learning: A Tutorial Survey and Recent Advances - http://web.mst.edu/~gosavia/joc.pdf
3. Algorithms for Reinforcement Learning - https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf

Some UseFul Links -

1. Some RL Derivations - https://www.alexirpan.com/rl-derivations/
2. https://www.reddit.com/r/reinforcementlearning/comments/aux7a5/question_about_nstep_learning_with_dqn/
3. N Step Bootstrapping - https://lcalem.github.io/blog/2018/11/19/sutton-chap07-nstep

Discussions

1. Work on adding and pruning the sequence
2. Explore a cooperative learning setting where a separate agent learns to insert/prune the replay buffer.
3. Some form of intrinsic motivation based replay buffer management.
4. Adding/deleting entire (sub)-sequences. (have some probability model for these)
5. Use in supervised learning to train on skewed datasets
6. Active Perception
7. **Disentangle novelty and on policyness**
8. What is a good approximation of variance of target $\mathbf{G_t}$ (even in microworld) -
    a. For a given (S_0, A_0, π) we can use $(\sum_{i=1..n} \gamma^{i-1} r_i + \gamma^n Q(S_n, A_n|\pi) - Q(S_0, A_0|\pi^*))^2$. The issues here are -
        i. We only have 1 sample of the target $(\sum_{i=1..n} \gamma^{i-1} r_i + \gamma^n Q(S_n, A_n|\pi))$, the true variance would be computed by enumerating all possible n steps from the initial state (S_0, A_0) and then proceeding with the computation. Can we study whether there is some theoretical soundness to this?
        ii. In the current policy π, $\mathbf{E}(\sum_{i=1..n} \gamma^{i-1} r_i + \gamma^n Q(S_n, A_n|\pi)) \neq Q(S_0, A_0|\pi^*)$, i.e. the expected target as per the current policy may deviate from the true target as per the optimal policy. Thus while $\mathbf{Var}(\mathbf{G_t}|\boldsymbol{\pi})$ may be low, $\mathbf{G_t - Q(S_0, A_0|\pi^*)}$ could be very high and hence an imprecise measure.
        iii. Would using a smoothed average of Q(S,A) be a good proxy? Concretely, at time **t,** the true expectation $\boldsymbol{\mu_t} = (\mathbf{1/k})(\sum_{i=1..k} Q(S_0, A_0, \pi_{\mathbf{t-k}})$
    b. Do multiple (d) runs of the process, and then calculate variance, for a given s, across these d runs, taking the mean as the mean of these d runs. The issue is that the policy might be different in different runs, at this point.
        i. Doesn't make too much sense. Policy at a timesteps may be different across runs and network parameters will almost certainly differ. Makes the whole idea of gaining insight from variance fuzzy.

Experiments to perform

1. Recreate the results of the "revisiting …" paper.

2. Study impact of
    a. Eligibility traces (by varying lambda) vs size of replay buffer
    b. Variance reduced bootstraps like Average DQN in n step reward setting vs buffer size
3. Validate the hypothesis that large replay buffers help reduce the variance of the target. We should observe this for different forms of multi step returns - uncorrected n step, eligibility trace, n step + avg DQN bootstrap and possibly tree backup as well.
    a. First run this on a finite state space such as a grid world. In such an environment we have a finite SxA space which can allow us to track the mean of the target for each such tuple. We can also control the sparsity of rewards as well. Two such experiments can be performed -
        i. Mean of target is true target. Precompute optimal policy and Q value and use this as mean. Compute MSE wrt this for variance.
        ii. Mean is the running average of targets from the last **k** replays of the given (s,a) tuple. Compute MSE wrt this for variance.
        iii. Refer to Point 8 of discussions
    b. Compare the variance achieved in the experiments performed in both settings. If running average is a good proxy for the true target, then this will help in subsequent experiments.
    c. Next we need to run this on continuous state and action spaces. The issue is that since SxA is infinite we cannot keep track of the running mean/ true target. To get around this problem -
        i. The first approach would be to bucket the SxA space and store the means for each such bucket.This however has a possible drawback. It could be possible that states very similar in SxA space could have vastly different expected targets. An example would brickbreaker where the difference between hitting and just missing the ball while small could result in very different target values.
        ii. To solve this would require devising something that maps SxA to a space where spatial locality implies similar targets for learning. An appropriate siamese network could be designed for this task. Sub problems to this -
            1. Can we have such a siamese network which generalizes across environments, or do we need to pretrain this for each new environment that we use?

Playing Atari with Deep Reinforcement Learning (DQN) [2013]

Challenges -

1. The RL agent needs to learn from a reward that is sparse, noisy and delayed. Delays can be spanning 1000s of timesteps as compared to a direct association between input and output during supervised learning.
2. In RL contiguous states may be highly correlated. This poses a problem as other paradigms of learning assume independence of input data.
3. As the policy trains, the data distribution observed by the agent alters (change in exploration strategy by updated policy). This breaks the assumption of a fixed input distribution employed by other learning paradigms.

Contributions -

1. Alleviate problem 1 using CNNs with a variant of Q Learning.Weights updated through SGD.
2. Problems 2 and 3 are handled using an experience replay buffer. Randomly samples prior transitions for training. This breaks correlations as well as smooths the data distribution.

Results -

1. The NN agent beat prior RL agents on 6/7 games on Atari 2600 and surpassed expert human performance on 3.
2. Trained using video game footage, rewards and an action space as used by humans. No game specific information is provided.

Limitations -

1. Finite replay buffer which discards oldest transition when full. No notion of discarding transitions least effective for training.
2. Uniform random sampling of transitions from replay buffer. No notion of using more important transitions frequently.

Further Thoughts -

1. Whether randomly selecting states whose max Q values are chosen to be tracked is a good choice for evaluating training progress?
2. Is fixing rewards to {-1,0,1} to make it scale invariant across games the best approach?

Prioritized Experience Replay (PER) [2015]

Challenges -

1. Uniform sampling of experiences from replay buffer. Consequently no notion of using important transitions more frequently. They use a blind cliff walk example with $½^n$ chance of a non zero reward to show the exponential reduction in updates needed when using optimal transition selection (through an oracle) as compared to uniform random selection.

Contributions -

1. Demonstrate the effectiveness of TD Error as a reasonable proxy to the learning capability offered by a transition. (Though this could be a poor choice if rewards are noisy - leads to high prioritization of inconsequential transitions or low prioritization of

important ones.). Some possible alternatives proposed (none surpassing the chosen one in preliminary tests) -
    a. Norm of weight-change through replay of transition - effective if optimizer uses adaptive step size to limit gradients in noisy directions
    b. Change in TD error b/w replays
    c. Asymmetric treatment by prioritizing positive TD error over same magnitude negative error
2. Stochastic Prioritization overcomes issues posed by greedy TD error prioritization -
    a. The error of high priority transitions shrink slowly when using functional approximators, thus a greedy scheme would pick only a small subset of transitions, making the network prone to overfitting. Transitions with small TD errors may never be replayed.
    b. Stochastic prioritizations improves the chance of old transitions being replayed before being discarded, as well the hastened replay of a new transition. Both being issues in uniform sampling.
    c. Greedy scheme is sensitive to stochastic/noisy rewards.
    d. Repeated update of entire replay memory not needed at each iteration.
3. Propose 2 stochastic sampling techniques and correct for bias - rank based (power law distribution) and proportional to TD error based.

Results -

1. Propose a technique to replay important experiences more frequently by prioritizing transitions.
2. Outperform DQN with uniform sampling on 41/49 benchmarks to achieve new SOTA.

Limitations -

1. Does not optimize which transitions to remove from a full replay memory. Another extension could be to reduce the replay memory in case a lot of transitions are inconsequential.

Further thoughts -

1. Prioritized replay can be used to address class imbalance in supervised training without having to explicitly tune the network for the same.
2. Prioritization based on episodic returns?
3. Correlation between td errors of contiguous experiences. Could this be used to make priorities less noisy

A Deeper Look at Experience Replay (CER) [2017]

Challenges -

1. The effect of replay buffer size has been thus far overlooked. The effect of varying buffer size needs to be studied in terms of optimality and speed of convergence.
2. On increasing buffer size, assuming uniform sampling from a buffer size m, the chance of a newly added transition being played before k ($k<m$) timesteps is $1 - (1-1/m)^k$. This is monotonically decreasing in m. Thus an important transition may be delayed a long time

before it can affect the agent if the buffer size is large. Similar issues can affect PER as new transitions are assigned highest priority, but this doesn't guarantee immediate usage for training.

Contributions -

1. Demonstrate that the buffer size is an important task dependent hyper parameter. Its effect has been empirically studied using tabular, linear and non-linear functional representations.
2. Propose Combined Experience Replay (CER) that in O(1) adds the latest transition to the random uniform/priority wise sampled batch. This remedies the negative influence of large buffer sizes on the training process.

Results -

1. Highlights the importance of tuning the buffer size for a task.
2. Proposes CER as an O(1) augmentation to existing sampling techniques which enable immediate replay of important transitions, thereby contracting the negative effects of large buffers (which are need to produce nearly iid samples)

Limitations and further thoughts -

1. The question of how dynamically adjusting buffer size while training remains unresolved
2. Which transitions to remove from a full buffer and how many (related to pt 1.)
3. Sampling based on episodes?

Prioritized Sequence Experience Replay (PSER) [2019] {TODO - Proof}

Challenges -

1. Existing sampling techniques assign a high priority to a single important transition, however this in turn should make transitions leading up to this of higher priority as well. No technique thus far handles this and there are potential efficiency gains to be had since rewards are generally sparse.
2. In a blind cliff walk with 2 actions, n chained states and 1 reward state at the opposite end from the starting state, the chance of a random sequence of actions leading to reward is 2^{-n}. Starting with a small constant priority for both all states will lead to uniform sampling of all states until the reward state is encountered yielding a high TD error. Decaying the priority to preceding states will help identify the important transitions faster leading to faster convergence.

Contributions -

1. Propose a sampling scheme (PSER) that decays the priority of high TD error (important) transitions backwards, so that the importance of transitions leading up to this may also be updated.
2. Prove that PSER is guaranteed to converge faster than PER on the blind cliff walk environment.
3. Identify the issue of priority collapse in this approach and rectify it by introducing a new hyperparameter to slow the rate of priority decay. Assuming all Q vals to be 0, a sequence of transitions from T_{i-2} to T_i (high TD error) would lead to the priority of T_{i-2}

reducing to near zero constant epsilon. Sufficient such collapses render the technique useless.

Results -

1. Outperms PER on 40/60 atari games while also showing substantial improvement in the Blind Cliffwalk environment .
2. Theoretically show that PSER converges faster than PER in the later environment.
3. Empirically show better performance and faster convergence.

Limitations and further thoughts -

1. Possibility of generalizing the results to any environment / coming up with a counter example .

Experience Replay Optimization [2019]

Challenges -

1. Rule based heuristics used for sampling from RM. These may be suboptimal.

Contributions -

1. Presents a framework to learn a replay policy that chooses what to sample from the RM to best aid the agent policy.
2. Maintains a learnt priority vector to sample replays. These priorities are updated batchwise to avoid computational overhead of large RMs.
    a. Uses difference of cumulative rewards before and after a policy update based on a selected priority vector to optimize the replay policy. (Essentially tell the policy how its decision improved/deteriorated the cumulative reward of the agent policy.

Results -

1. Evaluate vs uniform and PER on continuous control tasks. Demonstrates consistent improvement.
2. Shown to pick low TD-Error transitions instead of high TD-error ones. In direct contradiction to what PER proposes. Possible explanation may be the on-policyness of the low TD transitions. They would better align with current agent policy and contain more useful signals

Limitations and further thoughts -

1. Framework could allow the addition of some form of intrinsic motivation to make the difference in reward less noisy.
2. Can be used to delete transitions as well.
3. Perhaps rethink PSER based on insights from this approach. (Given the fact that High TD error may not be a good heuristic for prioritization)

Remember and Forget For Experience Replay [2019]

Challenges -

1. As the state distribution of agent policy deviates from the RM distribution, performance of the learnt policy reduces.

2. Currently, to mitigate this hyperparameter tuning like annealing LR is used. The intuition is to make smaller changes in policy as training progresses.

Contributions -

1. Classifies transitions as near policy and far-policy based on importance sampling ratio. Clips gradients of far-policy samples to zero (Rule 1)
2. Penalizes policy on far policy samples using KL divergence to bring the learnt policy closer to the RM distribution. When there are many far-policy samples it increases the ratio of KL divergence in the overall gradient, preventing the target policy from straying too far from the RM distribution. And thus reducing variation.

Results -

1. Results on continuous action space tasks outperform methods like PER.
2. Average KL divergence is seen to continuously and smoothly fall across tasks.
3. Results validated on partially observable flow problem (**To Read**)

Limitations and further thoughts -

1. Persistence of far policy samples can be used as a heuristic for deletion.

Summary -

This paper aims to smoothen the change in policy of an agent in the later stages of learning. To do so it does 2 things - 1. Backpropagate the TD error of only those transitions who's IS lies within $[1/C_{max}, C_{max}]$. As the network becomes smarter this interval becomes narrower ultimately converging to the immediate neighbourhood of unity. 2. To explicitly make the network not diverge too much from the transitions in the replay buffer an additional KL Divergence loss is backpropagated for all transitions to align the current target policy ($\boldsymbol{\pi}$) with an earlier target policy($\boldsymbol{\mu}$). The contribution of the KL divergence gradient can be increased if the number of transitions having IS outside $[1/C_{max}, C_{max}]$ are beyond a threshold.This makes sense as the replay buffer is at least two orders of magnitude less in size than the total training transitions, and all its transitions are sampled from a relatively recent policy. In the later stages of training when transitions are allowed from a narrow IS range, both policies $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ are well trained. All that is being done is smoothening the change from a good policy to a slightly better policy.

Importance Resampling for off policy predictions [2019]

Uses IS while sampling from RM rather than applying this at the time of update. Reduces variance of update.

Challenges -

1. Importance Sampling (IS) used for off-policy learning is prone to high variance in the updates to the parameters.
2. Weighted IS (WIS) algorithms which normalize each update by the sample average of the ratios, improve learning over standard IS strategies, but are not straightforward to extend to nonlinear function approximation. Obtaining an efficient WIS update is not straightforward.

Contributions -

1. Proposes an importance resampling strategy (IR) rather than a reweighting strategy. Concretely, it samples transitions from the replay buffer with probabilities proportional to the importance sampling ratio. This enables on policy updates to the agent.
2. Compare IR with weighted importance sampling, and show that IR leads to lower variance (both theoretically and experimentally). Additionally it is shown that IR can be corrected to eliminate bias.

Results -

1. Show reduced variance as compared to other importance sampling based approaches in multiple settings using tabular methods, tile coded features and CNNs on a car simulation.
2. Also show lower sensitivity to some hyperparameters like LR.
3. Empirically show faster convergence vs IS and V-Trace(IS with clipping).

Limitations and Further Thoughts -

1. Transitions are sampled with probabilities proportional to IS ratio. Is there some other prioritization which may yield better results.
    a. Like the on-policyness idea of using transitions with IS ~ 1,
    b. Or PER based approach
2. Results not shown on standard benchmarks like Atari.
3. Authors propose learning many value functions in parallel, because many fewer updates can be made for each value function.

Attention Experience Replay [2020]

Prioritize transitions that contain states frequently visited by current policy. Introduces Attentive Experience Replay (AER), a novel experience replay algorithm that samples transitions according to the similarities between their states and the agent's state.

Revisiting Fundamentals of Experience Replay [2020]

Challenges -

1. The interplay between RM, algorithm, optimizer and reward is not well understood.
2. What properties of the remaining pipeline (if at all) enable performance improvements through larger replay buffers

Contributions -

1. Isolate the replay capacity and the replay ratio (gradient updates/environment step) and consequently age of oldest transition as the key factors to characterize the performance of RM.
2. Identify uncorrected n-step returns as the primary reason for improved performance on larger RM capacities.

Results -

1. Using Rainbow, performance improves consistently when replay capacity increases or when oldest policy age reduces (Replay ratio decreases).

    a. Larger capacity - More state action coverage, leading to possibly less overfitting
    b. Reducing oldest policy - Heuristic on/off-policyness. More on-policy implies the network trains on regions yielding higher returns and thus yields higher cumulative rewards.
        i. There is however an **anomaly**. In hard exploration, sparse reward games, reduces the age of the oldest policy harms performance.
    c. Increasing capacity at fixed replay ratio - No general trend as performance gains are manipulated by competing effects of increasing capacity and increasing age of oldest policy.
2. Through ablation experiments it is found that adding uncorrected n-step returns to DQN enables large gains when RM capacity increases. Similarly on removing uncorrected n-step returns from RAINBOW causes least improvement on increasing capacity.
3. To test whether uncorrected n-steps are beneficial in highly off-policy environments at massive replay capacities, the authors turn to offline DeepRL. Even in this setting the n-step returns with a 200M replay buffer far exceed the DQN on which the data was collected.
4. Where do the gains of n-step come from?
    a. Contracting to 1 step and reducing the discount factor for bootstrapping has no effect on the performance with increase in capacity.
    b. Variance Reduction - Increasing RM capacity may help mitigate added variance of n-step returns (TD - low variance,high bias (imperfect bootstrapping) ; MC - low bias, high variance(stochasticity of reward)). By comparing with and without sticky keys (high and low variance environment) it is found that low variance env, reduce benefits on increasing capacity with n-step rewards, yet there is still substantial improvement and does not fully explain the efficacy of n-step returns.

Limitations and further thoughts -

1. Study how other multi-step algorithms (Q(lambda),TreeBackUp,Retrace) interact with experience replay.
2. Disentangling on-policyness, state-space coverage (novelty?), correlation between transitions, sequences.

W

Understanding Multi-Step Deep Reinforcement Learning: A Systematic Study of the DQN Target [2019] - https://arxiv.org/pdf/1901.07510.pdf

Challenges -

1. While DeepRL has begun to use multi step returns, it is not understood how various details of these returns affect learning.

Contributions -

1. Analyze the effects of off-policy correction, length *n* and update frequency of the target network.

2. Study these effects on Q-Learning, Tree BackUp, SARSA, Retrace and Q(sigma) {linear combination of n-step SARSA and TreeBackUp.}

Off-Policy N-Step -

Theoretically correct n-step TD -

$$\hat{G}^S_{t:t+n} = R_{t+1} + \gamma\rho_{t+1}\hat{G}^S_{t+1:t+n}, \qquad \rho_{t+1} \doteq \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})}.$$

N-step Q-learning (used in algo like Rainbow) -

$$\hat{G}^{QL}_{t:t+n} = R_{t+1} + \gamma\hat{G}^{QL}_{t+1:t+n}, \qquad \hat{G}^{QL}_{t+n:t+n} = \max_a Q_t(S_{t+n}, a).$$

Results -

1. Off-policy correction is not always necessary. It is possible to ignore off-policy correction without seeing an adverse effect in the overall performance of Sarsa and Q(σ). Nonetheless, off-policy correction seems to benefit performance during early training.
2. Parameter n can result in significant improvement in the performance of all of these algorithms. Though this diminishes for n>3.
3. Some algorithms are more robust to the update frequency of the target network. Our results seem to indicate that algorithms that retrieve more information from the target network are also more sensitive to changes in this hyper-parameter. One interesting possibility is that these types of algorithms could greatly benefit from not using a target network at all, as long as it does not affect the stability of the network.

Deep Reinforcement Learning with Double Q learning (DDQN) [2015]

Challenges -

1. It was unknown whether overestimation in Q learning affected the algorithm in practice and whether this effect harmed performance.

Contributions -

1. Establish the existence of overestimation bias in deep Q learning
2. Propose DDQN which keeps overestimation bias in check. Demonstrate this by approximating various functions with approximators of varying capacity while also depicting the gains on the Atari benchmark as compared to DQN
3. Show that DDQN learns better policies and outperforms DQN on various benchmarks.

Results -

1. Demonstrate lower overestimation on the Atari benchmark as compared to DQN as well as higher optimal value function scores.
2. Outperforms DQN on the Atari benchmark as well as obtains higher scores on human starts.

Limitations and further thoughts -

1. The theorem proving a lower bound for overestimation implies reducing lower bound as the number of actions increase. However the opposite is observed in practice. The

explanation for this is insufficient and there appears to be no significance of this theorem.
2. **CHECK** - Theorem proof - the last line causing the contradiction appears to be incorrect.

Dueling Network Architectures for Deep RL [2015]

Challenges -

1. Most advancements in RL have focussed on deriving better algorithms or better sampling techniques. No work has been done on building architectures suited for model free RL.
2. Existing architectures are single stream and compute the Q value. Thus for a good state value estimate all state action value estimates are needed.

Contributions -

1. Propose an architecture with two streams - one to compute the advantage function A(s,a) = Q(s,a) + V(s) and the other to compute the state value function. Helps learn good state value estimates of important states without needing to be trained on every action from that state.
2. Enables learning good policies in environments with large actions spaces quickly. Can quickly learn good actions even when redundant actions are added. Is easily usable with any RL algorithm and sampling techniques (prioritized replay buffer).

Results -

1. Beats SOTA set by DDQN by surpassing it on various benchmark evaluations of the arcade learning environment of 57 Atari games.
2. Decouples V() and A(), and demonstrates through saliency maps that the value function focuses on image regions which would affect future gains even when no actions prejudice any perceivable gains. The advantage function is shown to focus on regions which affect the immediate reward.
3. The frequent update of the V function stream helps form better state estimates.
4. Often, the differences between Q-values for a state are small relative to the magnitude of Q. This can make the greedy action susceptible to abrupt changes in the presence of small noise. This problem does not arise with the dueling architecture.

Limitations -

1. Don’t see any limitations per say except the scope for better architectures.

Issues -

1. Doesn't provide an adequate reasoning, for why they used q(s, a) = v(s) + (A(a, s) - mean(A(a', s))) instead of the technically correct eqn, q(s, a) = v(s) + A(a, s)

Curiosity Driven Exploration by Self Supervised Prediction [2017]

Challenges -

1. When extrinsic rewards are very sparse (like in many real world scenarios) the exploration tends to be very random only encountering rewarding/penalizing states by chance. The exploration in this case is very inefficient.
2. Shortcomings with current intrinsic motivation techniques -
    a. Encourage exploring novel states - Require building statistical models of states' distribution. This is difficult where input is continuous image data of high dimensionality.
    b. Encourages the agent to predict the consequence of its action (predicting next state) - disentangling changes due to stochasticity in the environment (leaves moving in the wind) from those affected by the agent's action is challenging.

Contributions -

1. Propose a curiosity driven intrinsic reward mechanism which is based on the agent predicting the consequences of its own actions while avoiding the pitfalls of large state space and disentangling environmental stochasticity from changes due to agent action.
2. Make predictions using the feature space learnt in a self supervised manner by jointly learning a forward dynamic model (predicting next state embedding from current state embedding and current action) along with an inverse dynamics module (predicting action given current and next state). The inverse model ends up learning state embeddings that model aspects of the state that can be influenced by the agent.

Results -

1. The intrinsic curiosity module (ICM) when paired with algorithms like A3C demonstrate growing gains with the increasing sparseness of the extrinsic reward.
2. The A3C + ICM shows large improvements over VIME + TRPO notwithstanding the fact that TRPO is more sample efficient than A3C.
3. When trained without any extrinsic rewards the agent still learns to meaningfully explore the 3D VizDoom as well as crossing 30% of level 1 on Super Mario
4. On evaluating the learnt policy in a different test setting (generalization), the learnt policy performs as is or with some fine tuning surpassing policies learnt from scratch in the new environment.

Limitations and further thoughts -

1. How to extend such a set up to more complex setups where agents possess multiple actuation joints (like a humanoid/quadruped). Would a single such policy work or would multiple policies in a hierarchical or distributed manner be needed
2. As the authors mention this approach does not extend to situations where opportunities to interact are rare. In such situations integrating a replay memory can be explored
3. Large scale randomness within the observed environment can cause the agent to be stuck without exploration. For example if a specific action does move an agent rather changes the entire environment (traveling between worlds in super mario) and each such consecutive action yields a new unseen environment, then the agent will likely be stuck in position just changing environments without doing any exploration until (if at all) all environments are seen a sufficient number of times to be able to predict the forward dynamic with sufficient certainty.

Noisy Networks for Exploration [2017]

Challenges -

1. There exist shortcomings in existing exploration mechanisms -
    a. Intrinsic Motivation - The weighting of the intrinsic reward wrt the env. Reward needs to be hand picked rather than learned. Poor selection could lead to a policy optimized towards a different objective.
    b. Evolutionary or black box algos **(?)** while applicable to any parametric policy are not efficient.
    c. Random perturbations like epsilon greedy and entropy regularization (UCB?) cause local 'dithering' and aren't likely to create substantially different behavioural patterns.

Contributions -

1. Learnable random perturbations
2. TODO - Read A3C and Distributed Reward first

Policy Gradient Methods for Reinforcement Learning with Function Approximation

Challenges -

1. Most existing techniques use greedy action policy, while focusing function estimation entirely on the value function
    a. Only finds deterministic policies, but best might be stochastic
    b. Small change in value function may lead to large change in policy
    c. prevent convergence guarantees

Contribution-

1. Use a seperate function approximator, for policy, instead of simple greedy.
2. Proved that the gradient of performance wrt to policy function parameters is independent of the effect of policy changes on the distribution of states.
3. Provide a formulation for updating the policy parameters, so it converges to locally optimum policy. Proved theoretically that this works for any differentiable function approximation.
    a. Formulation works with unbiased approximation for the Q-values
4.

Limitations-

1. Proof assumes that, the function approximation for Q-values, can only be linear in the same features as the policy.
2. Before update step of policy approximation, we need to reach local optimal for the Q-value approximation.
3. No results mentioned, or given.

Unifying Count-Based Exploration and Intrinsic Motivation [2016]