1. DQN中的replay buffer是为了防止网络只学习当前的经验，比如马里奥游戏，防止agent打到第二关就忘记第一关怎么玩了
2. 传统的策略梯度算法是在每一个episode结束后根据整体轨迹的回报进行更新(MC)，而AC方法可以通过Critic对每个step的回报进行更新
3. A2C的优势函数可以解决传统值函数方法具有大的训练波动的问题，因为优势函数的公式里减去了当前状态的基准值，因此可以降低因为状态基准值发生变化而引起的高方差
4. RL里不能使用BN，效果远不如soft target update和generalization advantage estimate
5. 减少网络层+合理初始化参数+降低学习率+限制梯度幅值可以有效的提升DRL性能
6. DQN处理连续动作的方法：离散化，梯度上升，miu P和V线性代数，感觉miu这个方法和确定性策略网络一样输出的a=miu
7. 分享一个经验吧，ddpg里，可以先调q的lr，尽量调大，需保证q的loss在降。再调pi的lr，尽量调大，出现负几百几千就不能调了。
8. DuelingDQN保证更新V值的方法是约束每次A值的平均期望为0
9. NLP需要看的模型：LSTM，Transform，attention，Bert
10. Onpolicy算法如果不采用多线程，效果并不是很理想，因为前后训练之间相关性很强
11. 需要分清优势函数和TD误差的区别：优势函数是Q-V，TD误差是r+γV’ - V
12. 蒙特卡洛方法的一个trick是对采用试验性动作的幕进行打折或删除(因为他们影响了对V的估算)
13. Sarsa和Q-learning的区别不是同轨和离轨，而是自举的时候采用什么策略，Sarsa自举的时候Q’采用行动策略，Q-learning自举的时候Q’采用目标策略(贪婪)
14. Reward function的设计最好用-1 0 而不是0 1，因为0会导致稀疏奖励，而且0并不能促进学习，总之就是没有好好利用样本。对于以优势函数做reward也适用
15. 对于类别的输入要用one-hot，而不能用一个连续值，因为(1+3)/2 = 2，容易产生错误，用二进制表示也并不能优化，因为二进制本质上也是欧氏空间上的距离度量，但是可以用word embedding，使得近似词在同一个维度上的距离相近
16. 遇到稀疏奖励问题一般用优先级replay buffer和多目标强化学习效果比较好
17. Vf\_loss上升一般是因为探索到了新的环境，获得了新的奖励。而且RL的loss一般不会太收敛，因为在监督学习中数据是不变的，而在RL中数据会随着策略不断变化，如果loss收敛了说明agent充分探索了环境 或者 策略自信地选择了最优轨迹(没办法探索其他位置的loss了)。使用target网络并设定较长的更新周期有助于稳定策略，减小loss的波动。
18. DRL是天生自带过拟合属性的算法。而且泛化能力很差，因为真正的人类智能具有高效学习(只需极少样本)和终身学习(举一反三)的特点，是对DRL人工智能的降维碾压
19. 如何实现DRL的泛化能力：元强化学习，愿景很美好，实用有点早
20. DRL最适合解决的问题：场景固定，目标明确，数据廉价，过程复杂，自由度高