基于星际争霸的多智能体协同策略优化

一、项目背景

许多实际应用的复杂环境往往需要人工智能体与其他智能体互相协调与竞争。《星际争霸》作为实现此目标的基础对象，在最高难度的专业电竞方面持续占据标志性地位，并且其展现的复杂性与现实生活中的事件息息相关，因此《星际争霸》领域已经成为人工智能研究领域的一项重要挑战。

本项目所使用的SMAC环境基于《星际争霸II》，并专注于微观管理挑战，其中每个单位由一个独立的智能体控制，该智能体必须根据局部观察采取行动，而对手单位由《星际争霸II》内置的AI控制。智能体可以执行的行动包括按四个方向移动（北、南、东、西）、攻击敌人单位、停止或不执行操作。智能体的观察范围限定在一个以单位为中心的圆形区域内，这个范围决定了智能体能够看到的视野，这使得环境对每个智能体来说是部分可观测的。智能体只能攻击其射击范围内的敌人，这通常小于其观察范围，迫使智能体在使用攻击行动之前必须使用移动命令。

SMAC提供了一个复杂的环境。在SMAC中，智能体需要在不完全信息的情况下做出决策，这与现实世界中的许多决策场景相似，如商业策略、军事指挥和紧急响应等；智能体必须学习如何协调他们的行动以实现共同目标，这与现实世界中的许多需要团队合作的现实场景相似，如机器人团队在灾难救援中的协同工作。因此，从SMAC中获得的研究成果和经验教训可以被广泛地应用和借鉴到现实世界的各种领域。

二、实验流程

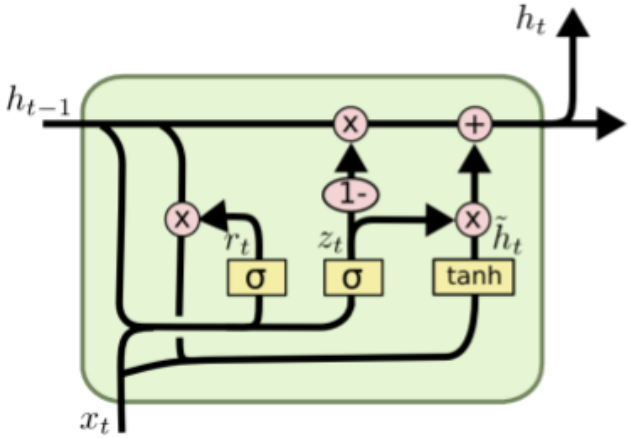
1. 智能体

1.1 网络选择

智能体的网络采用门控循环单元（GRU），它是一种特殊类型的递归神经网络（RNN），被设计用来改善传统RNN在处理长序列数据时遇到的梯度消失或梯度爆炸问题。GRU的核心特性是它的门控机制，这些机制允许网络动态地决定如何更新其隐藏状态。GRU的门控机制使其在处理序列数据时具有灵活性和强大的表达能力，特别是在需要学习长期依赖信息的场景中。

每个智能体都有自己的网络，但这些网络在控制器中共享一些参数。共享参数便于它们从彼此的经验中学习并泛化到新情况。这有助于提高模型在面对不同环境和任务时的适应性。

GRU的逻辑图如下：



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

1.2 优化器

采用RMSProp（Root Mean Square Propagation），它是一种自适应学习率优化算法，在处理非平稳目标时表现出色。

2. 算法选择

我们一共尝试了使用IQL、QMIX、QTRAN三种方法进行训练。

2.1 IQL算法

基本思想

IQL（Independent Q-Learning）基于DQN（Deep Q-Network）方法，将多智能体问题分解为多个单智能体问题，算法将其余智能体直接看作环境的一部分，也就是对于每个智能体都是在解决一个单智能体任务，很显然，因为环境中存在智能体，所以环境是非稳态的，这样就无法保证收敛性，智能体很容易陷入无止境的探索中。

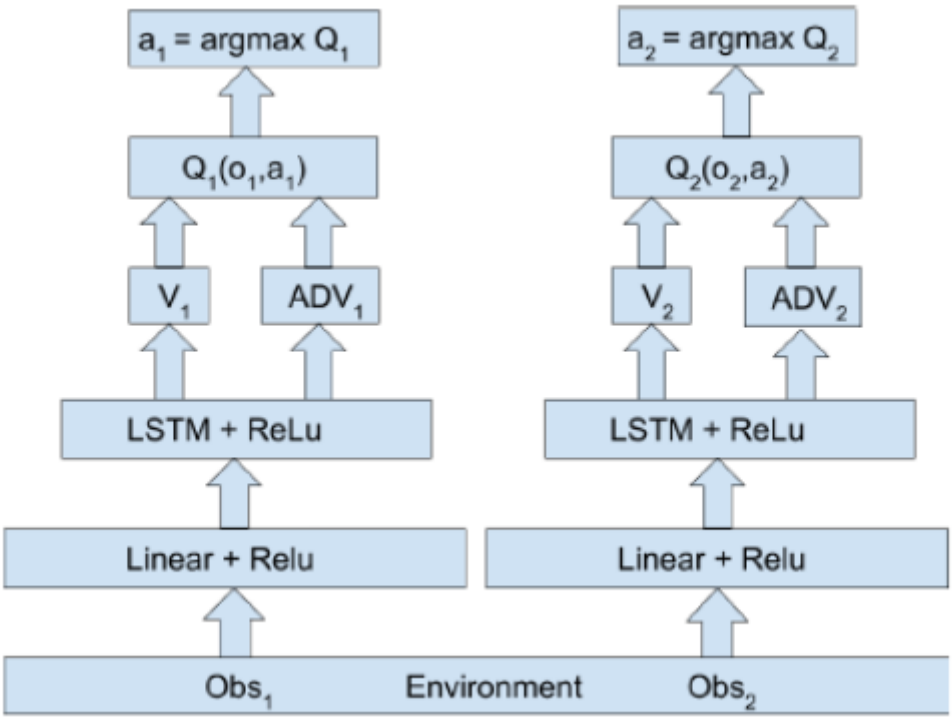
Q值的更新采用时间差分法，公式如下：

$$Q(s,a,w) \leftarrow Q(s,a,w) + \alpha\left[r + \gamma max_{a'} Q(s',a',w) - Q(s,a,w)\right]$$

其中，s, a 代表当前状态及基于当前状态采取的行为，s' 为基于当前状态采取行为后的下一个状态，a' 为基于下一个状态采取的行为，r 为基于当前状态采取行为后的奖励，α 为学习率，γ 为折扣率。

网络结构

下图展示了同一个环境中两个智能体的网络结构，但与下图不同的是，本项目中每一个智能体的网络结构都是基于GRU的。



2.2 QMIX算法

前置工作

IGM——一种条件

对于联合动作价值函数 Q_{jt} ，如果存在 $[Q_i]$ ，使得 Q_{jt} 的最优联合动作和对于 $[Q_i]$ 的最优动作联合相同，则可以认为 $[Q_i]$ 对于 Q_{jt} 满足IGM。通俗来讲就是，在联合最优时让每个智能体采取的动作可以使得每个智能体自身的价值最优

$$\arg\max_{\boldsymbol{u}} Q_{\mathrm{jt}}(\boldsymbol{\tau}, \boldsymbol{u}) = \begin{pmatrix} \arg\max_{u_1} Q_1(\tau_1, u_1) \\ \vdots \\ \arg\max_{u_N} Q_N(\tau_n, u_N) \end{pmatrix}, \quad (1)$$

then, we say that $[Q_i]$ *satisfy* **IGM** *for* Q_{jt} *under* $\boldsymbol{\tau}$*. In this case, we also say that* $Q_{\mathrm{jt}}(\boldsymbol{\tau}, \boldsymbol{u})$ *is factorized by* $[Q_i(\tau_i, u_i)]$*, or that* $[Q_i]$ *are factors of* Q_{jt}*.*

VDN——一种简单的实现IGM的方法(充分条件)

$$Q_{\mathrm{jt}}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{N} Q_i(\tau_i, u_i),$$

主要将系统的联合价值函数近似成为多个单智能体的价值函数的和因为VDN的联合函数的求和形式表现力有限，在有复杂组合的环境中，表现的很差。例如非线性环境。

基本思想

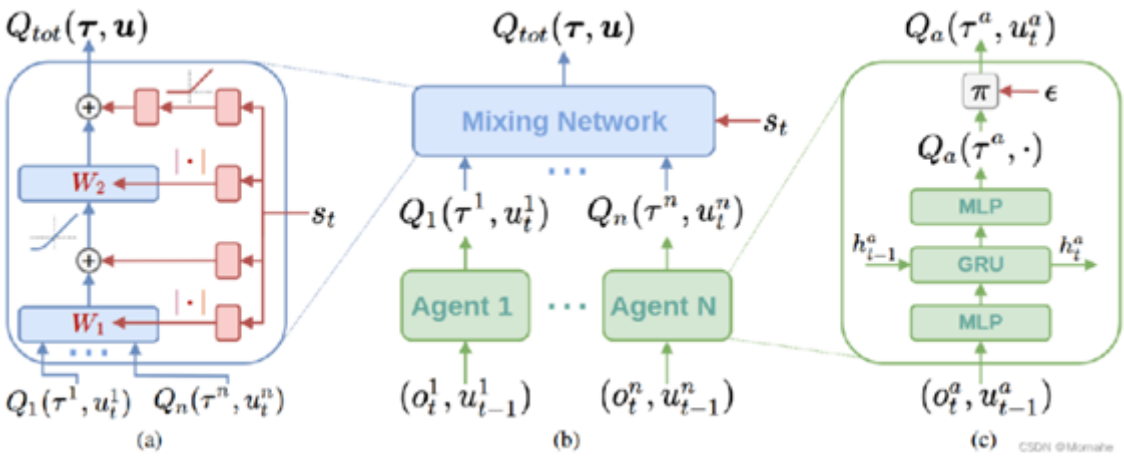
在VDN上进行拓展：求和相加 -> 非线性性函数（混合网络） + 全局状态信息辅助，仍然满足对“联合动作值函数取argmax <=> 对每个局部动作值函数取argmax”这一条件此外还需要满足：

$$\frac{\partial Q_{tot}}{\partial Q^i} \geq 0, \forall i \in \{1, 2, ..., n\}$$

不难看出，当 $\frac{\partial Q_{tot}}{\partial Q^{\mathrm{i}}} = 1$ 的时候就是VDN，VDN是QMIX的一种特殊情况

模型结构

QMIX 引入混合网络来学习联合动作值函数与局部动作值函数之间的非线性关系，并在训练过程中加入了全局状态信息。相比于VDN的线性求和，QMIX使用神经网络来整合各个局部值函数。



这个网络的含义是：中间时模型的主题架构，右侧是每个智能体的价值计算，左侧是Mixing Network的结构，是要以每个智能体的价值作为输入，输出一个联合的价值。因为我们要保证上面提到的单调性，所以mixing network的网络参数（权重和偏置）是通过一个超网络(hypernetwork)计算得出的。超网络确保输出的权重都是大于0的。中红色部分表示超网络，它产生蓝色部分所示的mixing network层的权重和偏置。

2.3 QTRAN算法

基本思想

不管是VDN还是Qmix, 他们都分别对 Q_i 和 Q_{tot} 之间的关系进行了限制，即认为它们之间的关系式加和或者单调的，因此对于一些多智能体合作任务，如果二者之间的关系并不是这种，这两种方法就未必会应用的很好了，对此，有人提出了提供了一种新的方法QTRAN，既满足IGM，且没有提出额外的假设。

我们知道全局奖励函数分布由联合动作决定。从某个智能体看来，奖励分布会随其他智能体动作取值不同而变化然而在强化学习中，我们最关注的永远是最大值点!至于其他点的分布形式我们并不关心。

QTRAN算法的全称是Q-transformation，该算法认为VDN的加和约束严重限制了值函数的函数表达能力，无法解决非单调协作任务。 QTRAN算法指出，更加通用的方法应该要满足独立-全局最大化（Independent-Global-Max, IGM）条件， 为了满足该条件，QTRAN算法将原始的整体值函数映射至新的值函数，使得它们在IGM条件下是等价的。

作者提出了IGM的充分必要条件：

Theorem 1. *A factorizable joint action-value function* $Q_{\text{jt}}(\boldsymbol{\tau}, \boldsymbol{u})$ *is factorized by* $[Q_i(\tau_i, u_i)]$, *if*

$$\sum_{i=1}^{N} Q_i(\tau_i, u_i) - Q_{\text{jt}}(\boldsymbol{\tau}, \boldsymbol{u}) + V_{\text{jt}}(\boldsymbol{\tau}) = \begin{cases} 0 & \boldsymbol{u} = \bar{\boldsymbol{u}}, \quad (4a) \\ \geq 0 & \boldsymbol{u} \neq \bar{\boldsymbol{u}}, \quad (4b) \end{cases}$$

where

$$V_{\text{jt}}(\boldsymbol{\tau}) = \max_{\boldsymbol{u}} Q_{\text{jt}}(\boldsymbol{\tau}, \boldsymbol{u}) - \sum_{i=1}^{N} Q_i(\tau_i, \bar{u}_i).$$

作者首先定义了一个 $Q_{tot}' = \sum_{i=1}^{N} Q_i(\tau_i, u_i)$,其实就是上述条件中的求和项，也是VDN中的 Q_{tot} ,称之为transformed joint-action value function,这里由于真实的 Q_{tot} , $[Q_i]$ 是满足IGM的，而根据上述定义， Q_{tot}' ， $[Q_i]$ 本身也是符合IGM的，所以 Q_{tot} Q_{tot}' 的optimal joint action是一样的，即 $\arg\max_u Q_{tot}(\tau, u) = \arg\max_u Q_{tot}'(\tau, u)$

模型结构

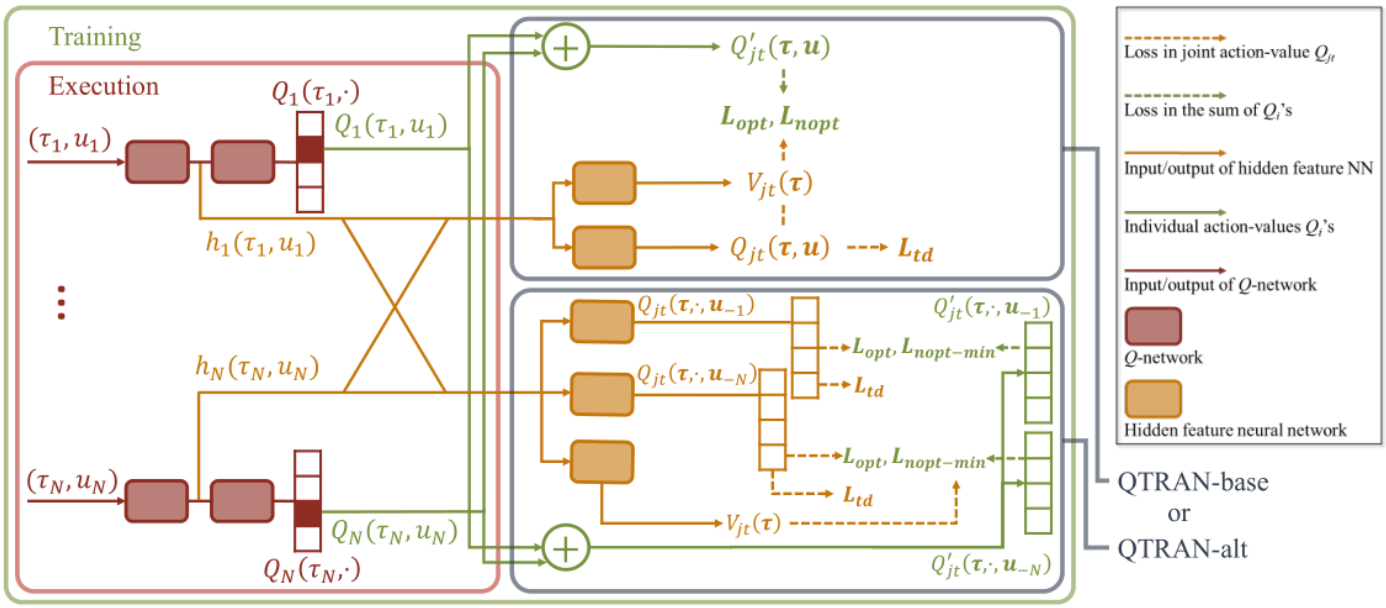


Figure 1. QTRAN-base and QTRAN-alt Architecture

QTRAN包含三个独立的估计器: 每个智能体i的独立Q值网络, 联合Q值网络(可以分解为 Q_i), 状态价值网络

联合Q网络: 逼近 Q_{jt} , 输入为选择的动作, 输出为该动作得Q值.

- 首先使用所有个体Q网络采样的动作向量来更新联合Q网络. 因为联合动作空间 U^N , 找到最优联合动作复杂度很高, 而每个个体的最优动作取argmax就行, 是线性的.
- 第二, 联合Q网络共享个体网络低层的参数, 联合Q网络把个体网络隐层特征加和整合 $\sum_i h_{Q,i}(\tau_i, u_i)$, 其来自于 $h_i(\tau_i, u_i) = [h_{Q,i}(\tau_i, u_i), h_{V,i}(\tau_i)]$ (**使用此参数共享样本效率高, 可进行可扩展的训练, 但会牺牲表达能力.**)

状态值网络: 计算标量状态价值, 类似于dueling网络的⊠(⊠).

- 用来在计算argmax时匹配 Q_{jt} 和 $Q_{jt}{}' + V_{jt}$. 没有V, 部分观察可能限制 $Q_{jt}{}'$ 的表达复杂性;
- 输入也是个体网络隐特征的组合 $h_{V,i}(\tau_i)$

3. 探索策略

采用 ε-greedy探索策略，ε 是一个在 0 和 1 之间的数值，以 1 - ε 的概率选择当前的最优动作，以 ε 的概率随机选择一个动作。这样做的原因是，如果智能体总是选择当前评估为最优的动作，它可能会陷入局部最优解而无法找到全局最优解；通过随机探索，智能体有机会发现新的状态-动作对，这可能引导智能体学习到更好的策略或发现高回报的区域。

4. 改进方法

由于qmix算法的效果是三种方法中最佳的，为了进一步提升模型的性能，我们决定针对qmix算法中的Qmixer进行改进。

4.1 改进方向1：添加Dropout层并使用Relu函数

1. 正则化项：

由于我们只在单一环境下进行训练，为了增强模型在其他环境下的泛化能力，添加了 Dropout 层来防止过拟合。

2. 更高级的激活函数：

在Dropout后使用 ELU 激活函数来增强模型的非线性表达能力。与ReLU相比，ELU有负值，这会使激活的平均值接近零。均值激活接近于零可以使学习更快，因为它们使梯度更接近自然梯度。这也有助于减少学习过程中的内部协变量偏移，即网络层输入的分布随着网络参数更新而发生的偏移，进而使梯度下降更加稳定。

4.2 改进方向2：将超网络层替换成注意力层

将超网络层替换成注意力层，用于更好地捕捉不同智能体之间的交互关系。具体来说，使用了多头注意力机制计算注意力，这种机制允许模型从不同的角度和抽象层次捕捉信息，从而获得更全面的交互特征表示。每个头专注于输入数据的不同方面，然后将这些局部特征整合起来，形成对智能体交互的全面理解。接下来用softmax 函数计算出归一化权重，并使用这些注意力权重对每个智能体的 Q 值进行加权求和，最后通过一个全连接网络计算优势值，并加上状态依赖的偏置，得到最终的 Q 值。

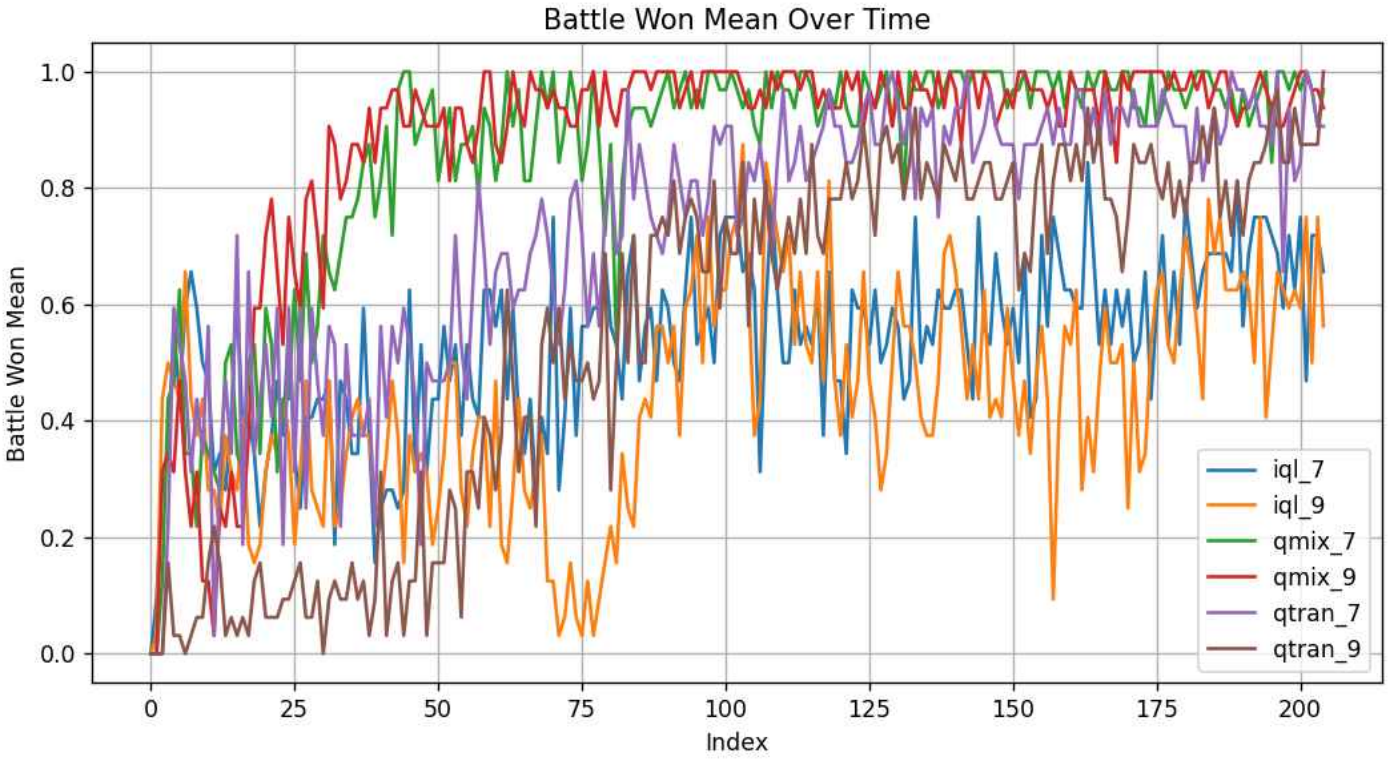
三、实验结果和分析

1. 训练结果

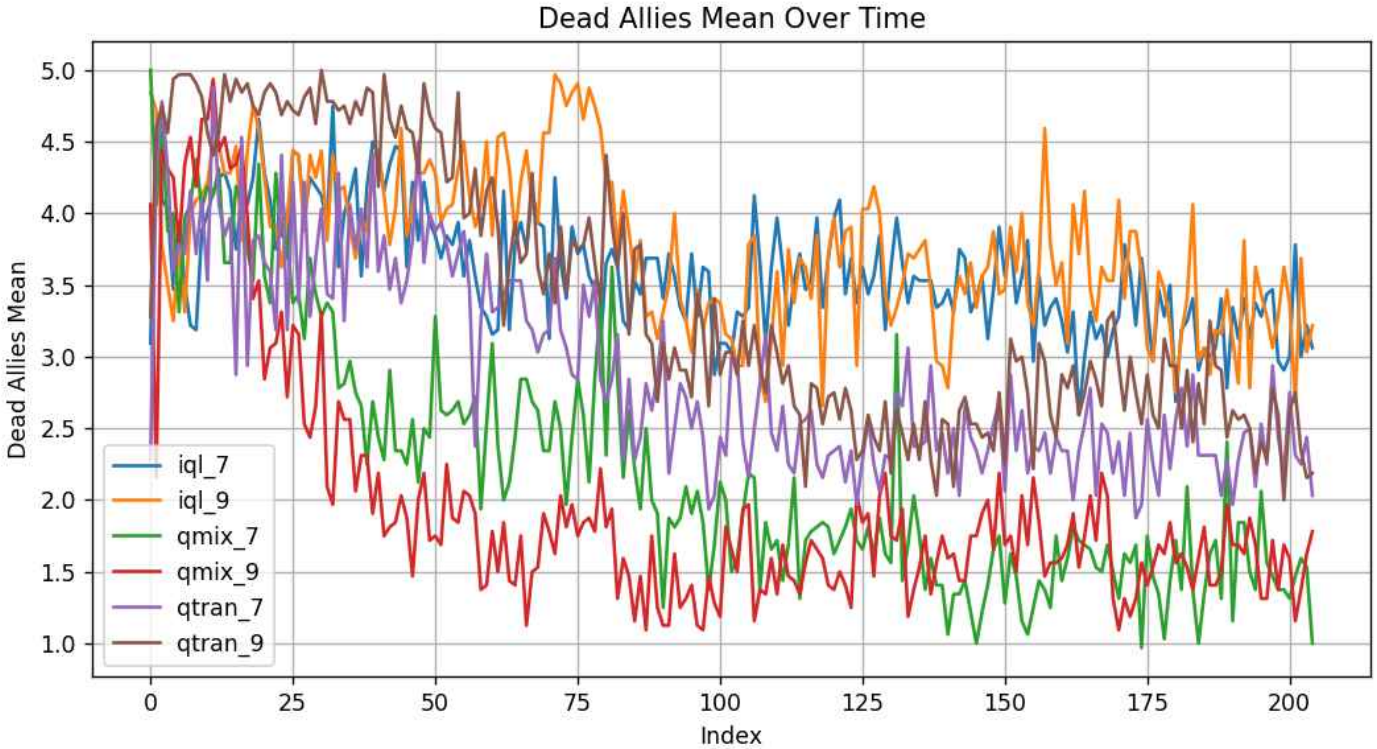
在难度7和难度9的情况下（难度代表敌方的战斗水平，9是最高难度），分别使用iql，qmix，qtran算法训练约2000000步后保存模型（保存模型的具体步数因为完成一局游戏花费步数不同而有细微的差别），但我们是每隔10000步计算一次模型的各数值，因此下面的表格展示的均为与2000000步最接近的一次计算时得到的数据。由于不同算法损失函数不同，这里不对比损失函数的值，而是使用测试阶段的胜率、我方平均每局游戏牺牲数、敌方平均每局游戏牺牲数来评价模型的性能。表格中，标红的数字表示效果最好的数据。可以看出，qmix算法的效果比另外两种算法要好。

难度	网络	胜率	我方平均牺牲	敌方平均牺牲
7	iql	0.6687	3.2125	4.3750
	qmix	0.9625	1.5688	4.9000
	qtran	0.8562	2.4688	4.8312
9	iql	0.6062	3.3375	4.1375
	qmix	0.9437	1.5375	4.9125
	qtran	0.9062	2.4125	4.8438

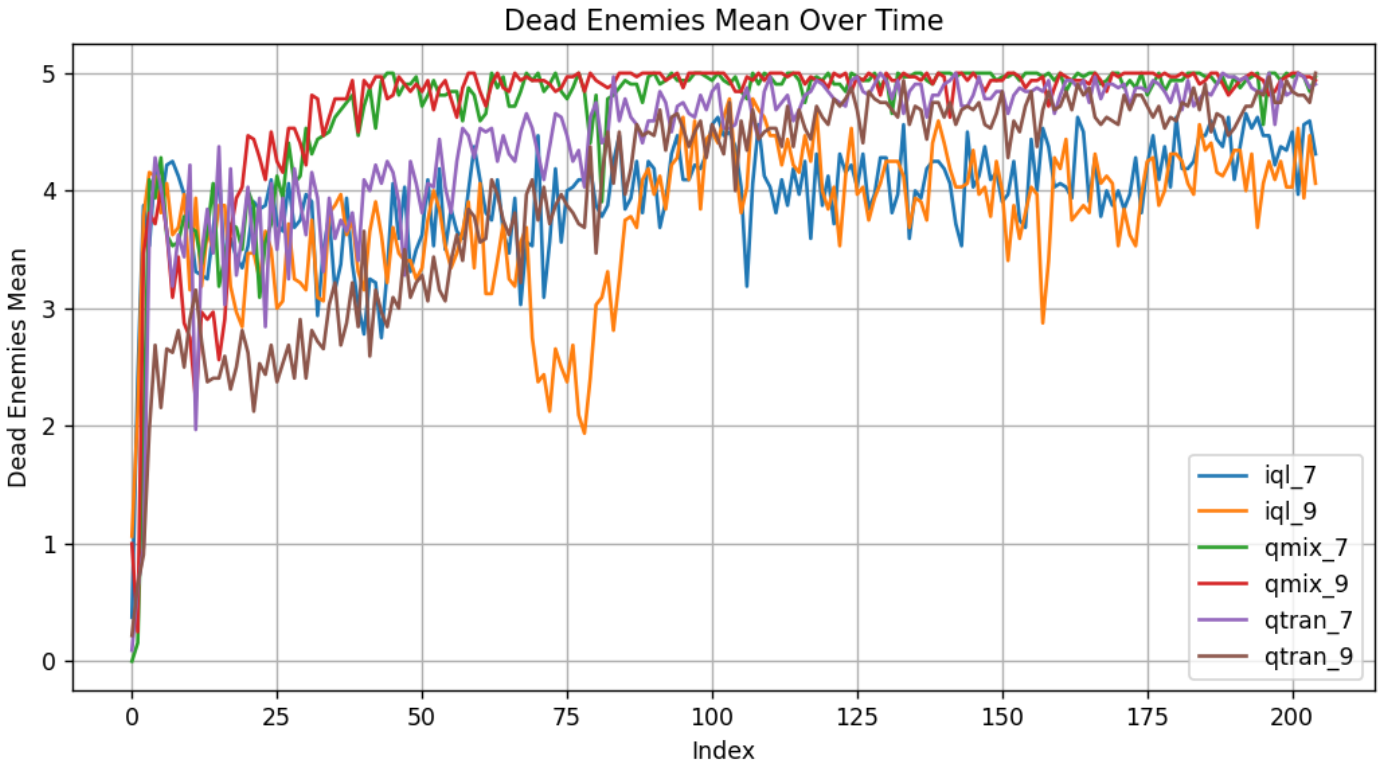
下表中展示了不同算法的测试胜率变化曲线图，可以看到，在iql、qmix、qtran三种算法中，qmix算法训练的效果是最佳的，在难度9下的测试胜率多次达到0.99甚至1.00，在难度7下的测试胜率经常徘徊在0.9以上；qtran算法虽然是qmix算法的改进，表现却稍微逊色于qmix算法；iql算法作为最基本的多智能体强化学习算法之一，表现远远逊色于其他两种算法。



下表中展示了不同算法的测试阶段的我方平均每局游戏牺牲数，三种算法的效果排行和胜率差不多。



下表中展示了不同算法的测试阶段的敌方平均每局游戏牺牲数，三种算法的效果排行和胜率差不多。

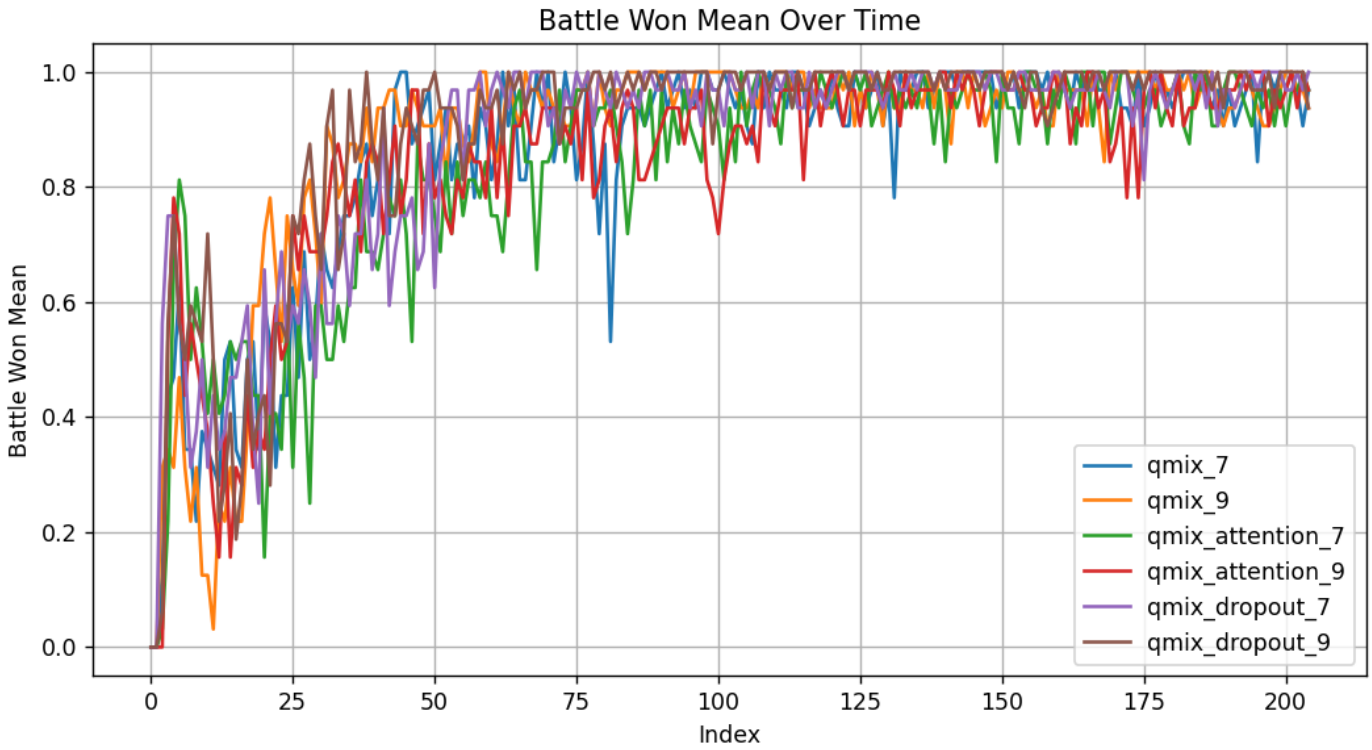


2. 改进qmix算法前后效果对比

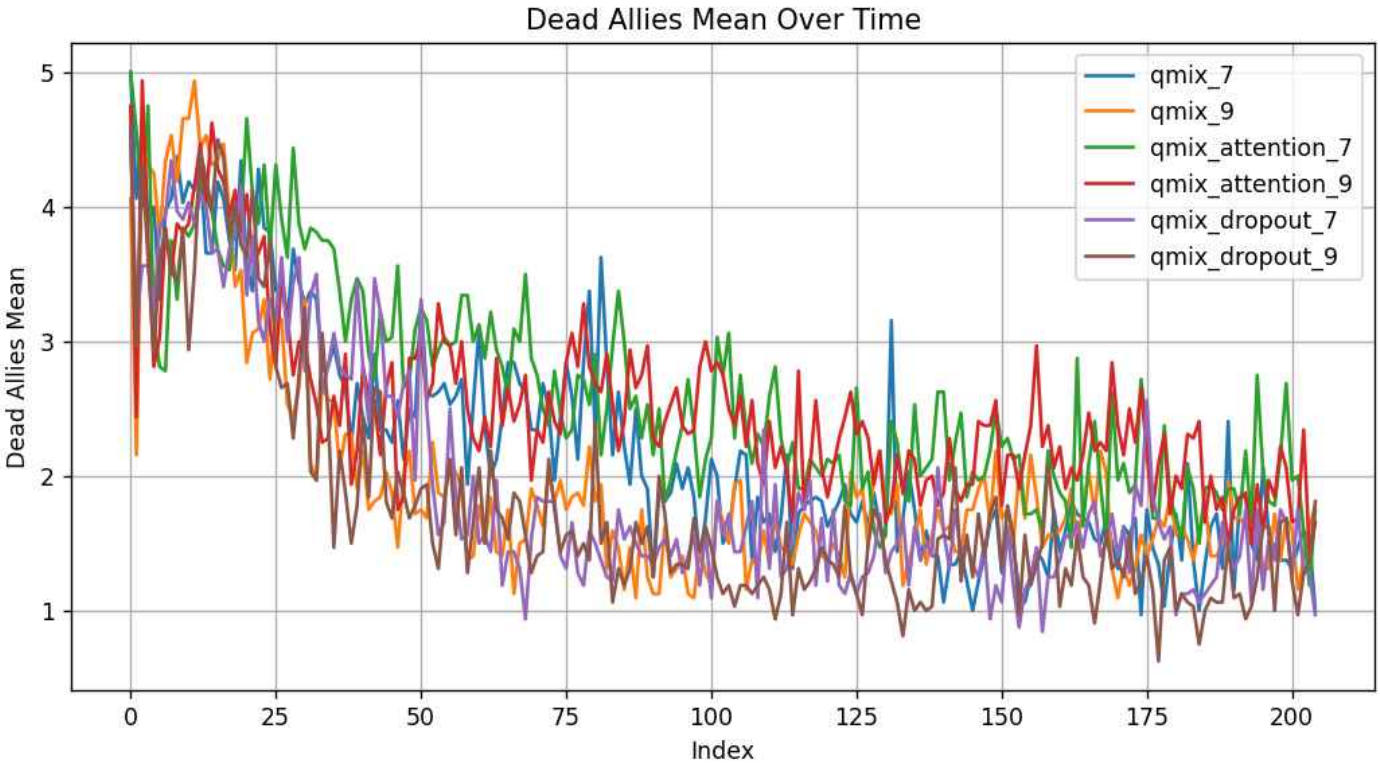
在难度7和难度9的情况下（难度代表敌方的战斗水平，9是最高难度），分别使用原始qmix，对mixer添加dropout并使用ELU激活函数的qmix，将超网络层替换成注意力层的qmix训练约2000000步后保存模型（保存模型的具体步数因为完成一局游戏花费步数不同而有细微的差别），但我们是每隔10000步计算一次模型的各数值，因此下面的表格展示的均为与2000000步最接近的一次计算时得到的数据。由于不同算法损失函数不同，这里不对比损失函数的值，而是使用测试阶段的胜率、我方平均每局游戏牺牲数、敌方平均每局游戏牺牲数来评价模型的性能。表格中，标红的数字表示效果最好的数据。可以看出，对mixer添加dropout并使用ELU激活函数的qmix算法效果比其他两种方案效果略好。

难度	网络	胜率	我方平均牺牲	敌方平均牺牲
7	原始qmix	0.9625	1.5688	4.9000
	添加dropout	0.9938	1.5188	4.9875
	attention	0.9500	2.0938	4.9250
9	原始qmix	0.9437	1.5375	4.9125
	添加dropout	0.9812	1.4375	4.9750
	attention	0.9625	1.9375	4.9562

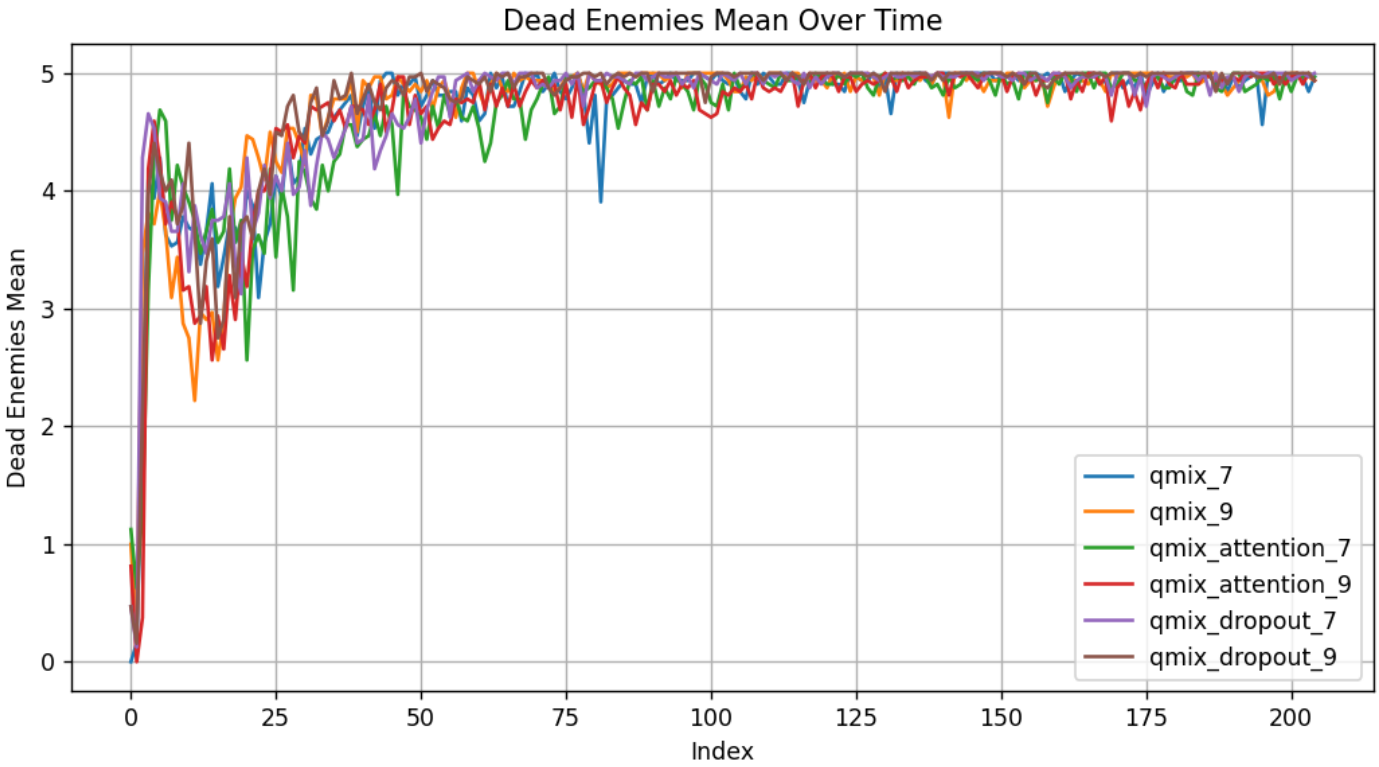
下表中展示了不同模型的测试胜率变化曲线图，可以看到，在原始方法和两种改进方法中，三种方法都可以达到很高的胜率，但是原始方法和注意力层方法的曲线稳定性较差，常常出现大幅度的振荡，而dropout方法的曲线稳定性较好。



下表中展示了不同方法的测试阶段的我方平均每局游戏牺牲数，可以看出加深网络方法的我方牺牲数稳定在较低水平，而另外两种方法的曲线逊色一些。



下表中展示了不同方法的测试阶段的敌方平均每局游戏牺牲数，三种方法的敌方牺牲数都可以达到很高的水平，但是原始方法和注意力层方法的曲线稳定性较差，常常出现大幅度的振荡，而加深网络方法的曲线稳定性较好。



3. 可视化展示

采用3m地图，即地图上有3个出生点。参数采用2s3z，即我方和敌方均有两个追猎者（Stalkers），三个狂热者（Zealots）。在星际争霸II中，追猎者和狂热者是星灵（Protoss）种族的两种不同的战斗单位。追猎者是一种快速、远程的星灵地面单位，擅长中距离战斗，它们装备有可进行快速射击的粒子碎裂枪，能够对敌方单位造成持续伤害；狂热者是星灵的前线战士，以其高生命值和强大的近战攻击力而著称，它们装备有能量刀，能够在近战中造成显著伤害，特别适合对抗敌方的轻型单位。狂热者通常用于前线作战，而追猎者则更多地用于侧翼攻击或在战场上进行快速打击。

狂热者和追猎者的外形如下：

在游戏中，我方穿红色盔甲，敌方穿蓝色盔甲。

在训练初期，我方战斗单位只会盲目地来回移动，也极少进行攻击，任凭敌方围攻自己。

在训练后期，以qmix网络在难度为7时训练出的模型为例，我方追猎者学会了向敌方某一个追猎者集中火力，而我方狂热者学会了先围攻敌方某一个狂热者，这样做的好处是能够尽快减少敌方的战斗单位数量，进而降低敌方的伤害能力。在敌方狂热者都被杀死之后，我方追猎者会转而围攻敌方某一个追猎者。