ICS 35.240
CCS L 70

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 31007—2025

智能兵棋推演系统算法与模型技术要求

Technical requirements for algorithms and models of intelligent wargaming systems

2025-11-27 发布　　2025-11-27 实施

中国指挥与控制学会　发布

目　　次

前　言

本文件按照 GB/T 1.1－2020《标准化工作导则　第 1 部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由南开大学人工智能学院提出。由中国指挥与控制学会归口。

本文件起草参与单位：

组长单位：南开大学人工智能学院；

副组长单位：中科院自动化所、中国人民解放军网络空间部队信息工程大学、广东美电贝尔科技股份有限公司；

成员单位：中国电子科技集团有限公司电子科学研究院、中国人民解放军国防科技大学试验训练基地、博智安全科技股份有限公司。

本文件主要起草人：张雪波、赵铭慧、赵振杰、郭宪、张世勇、骞晨旭、魏永森、杨传浩、张佳怡、李博文、王萌萌、倪晚成、于海涛、黄凯奇、孙怡峰、张玉臣、廖辉军、郑孙满、曾永健、凌艳香、支烽耀、杨俊强、林晖、郭庆浪、刘建勋、沈松、戎林峰、严鑫鑫（注：排名不分先后）。

智能兵棋推演系统算法与模型技术要求

1　范围

本标准规定了智能兵棋推演系统中算法与模型的技术要求，涵盖算法设计总体规范、算法功能与架构、模型构建总体规范、模型结构与接口规范、模型设计与训练流程、算法模型的评估与性能指标、部署与集成要求等内容，用于指导涵盖国防教育、应急救援、商业博弈等领域的智能兵棋推演系统算法与模型的设计、开发、训练、测试和部署，以确保开发的算法模型在兵棋推演系统中具备高效、准确的决策能力和良好的适应性。可供研发人员和机构参考使用。

2　规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GJB 438C－2021　军用软件开发文档通用要求

GJB 2786A－2009　军用软件开发通用要求

GJB 5235A－2021　军用软件配置管理

T/CICC 31001－2025　兵棋推演名词术语

3　术语和定义

下列术语和定义适用于本文件。

3.1

多模态数据　multi-modal data

指来自不同类型传感器或数据源的信息，如文本（指令日志）、图像（战场遥感）、结构化数据（兵棋规则参数）等，用于构建更全面的战场态势感知。

3.2

时序数据对齐　temporal data alignment

通过插值、重采样等技术，将不同频率或异步采集的传感器数据统一到推演时间坐标系下，确保多源数据的时间一致性。

3.3

态势评估　situation assessment

对当前战场状态进行定量分析的过程，输出威胁等级、关键区域等综合指标，用于支持目标匹配与策略制定。

3.4

对抗性策略生成　adversarial policy generation

预演敌方可能行为并基于此生成具有博弈优势的策略集合。

3.5

态势信息　situational awareness data

在兵棋推演过程中，系统采集和传递的关于战场环境、各方力量、单元状态等多维度数据。用于辅助智能体做出决策，涵盖从宏观态势到微观单元属性的多层级信息。

3.6

动作集合　action set

智能体输出的指令集合，供推演平台解析并执行具体操作。

3.7

模型接口规范　model interface specification

对智能体模型输入输出的数据结构、字段含义及交互协议的详细定义，是实现模块集成和数据交换的基础规范。

3.8

模型目录结构　model directory structure

指智能兵棋模型在文件系统中的组织方式，包括配置文件、策略模块、环境模块、训练模块等子目录的规范划分，以实现模块化管理、便于维护与扩展。

3.9

环境类　env class

封装推演环境的程序类，实现环境初始化、状态重置和推进等核心功能。

3.10

智能体类　agent class

表示智能体的程序类，负责加载策略配置、执行决策逻辑和管理模型参数。

3.11

配置模块　config module

存放智能体策略参数及推演环境配置文件的目录，支持 yaml 或 json 格式，负责定义智能体行为规则、环境参数等关键配置信息。

3.12

感知-判断-决策-行动环　observe-orient-decide-act loop

智能体在环境中进行决策的循环过程，包括感知阶段（获取环境状态信息）、判断阶段（完成环境理解、目标匹配等）、决策阶段（根据状态做出行动选择）和行动阶段（将动作施加于环境并引起状态改变）。该流程不断循环进行，直至推演终止。

3.13

模型检查点　checkpoint

在训练过程中保存模型当前参数和相关状态的快照文件。通过检查点可以在训练中断后恢复模型继续训练，或用于比较不同阶段模型性能。检查点通常包含模型权重以及优化器状态等信息。

3.14

并行训练　parallel training

同时运行多个训练实例来加速模型学习的过程。在并行训练中，多个环境或智能体同时进行感知-判断-决策-行动循环，各自生成数据供模型更新，使总的训练进度大大加快。

4　算法设计技术规范

4.1　总体原则

智能兵棋推演系统算法是指用于实现兵棋推演核心功能的计算方法和逻辑规则。智能兵棋推演系统算法设计应紧密围绕智能兵棋推演的核心目标，即模拟真实环境下的智能决策过程，实现智能体在复杂态势下的高效、合理决策。原则如下：

a） 真实性：模拟真实战场环境，支持多维度对抗；

b） 智能性：智能体需具备自主决策能力（感知-判断-决策-行动循环）；

c） 对抗性：支持红蓝双方动态博弈，包括欺骗、诱骗、反制等策略行为；
d） 鲁棒性：适应战场不确定性（如情报误差、装备故障、环境突变）；
e） 可扩展性：支持新战法、新装备、新规则的快速接入；
f） 模块化设计：决策、推演、评估模块解耦，支持插件式扩展。

4.2 算法功能架构

4.2.1 算法功能架构总述

在智能兵棋推演系统的总体设计中，算法功能架构是决定系统智能水平、信息流转方式与决策效率的核心要素。不同的架构模式体现了对作战过程认知层次、系统耦合度与计算部署方式的不同理解。当前较为成熟的架构形态主要包括基于决策循环的 OODA 分层架构、以深度学习为核心的端到端架构、支持多节点并行的分布式架构以及面向作战体系效能的杀伤网（Kill Web）等组织结构。下面对部分主流架构进行简要阐述。

4.2.2 OODA分层架构

OODA 分层架构（Observe－Orient－Decide－Act）体现了智能兵棋推演系统对作战决策过程的层次化建模思想，其核心组织形式是“感知—判断—决策—行动”的递进式闭环结构。该架构将复杂的博弈决策过程划分为若干功能层，每一层负责处理不同层级的信息抽象与决策逻辑。各层通过智能决策总线实现数据交互，支持多智能体协同推演与异构算法集成。各层间交互时序图如图 1 所示。

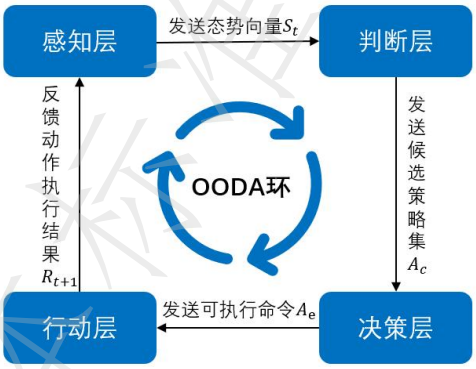

图 1 各层间交互时序图

图 1 展示了基于 OODA 环的四层决策循环，其中观察层首先对当前作战态势进行感知分析，生成态势向量S_t并传递给调整层；判断层依据态势向量结合策略模板和约束条件筛选生成候选策略集A_c并发送至决策层；决策层利用智能体训练模型或规则库对候选策略进行评估、优选与风险校验，输出可执行命令A_e交由行动层执行；行动层根据命令实施动作并将执行结果与反馈信息R_{t+1}返回至观察层，形成观察−调整−决策−行动的闭环时序依赖，实现持续动态优化的作战决策流程。

4.2.3 端到端架构

端到端（End-to-End, E2E）架构代表了智能兵棋推演系统向深度学习驱动与数据自适应演化的方向发展。该架构摒弃传统 OODA 分层中的显式模块划分，而采用从输入到输出的整体映射机制，使模型能够直接学习从原始感知信息到最终决策动作的非线性关系。其组织形式通常表现为单体化或准单体化结构：系统输入包括原始态势图像、传感数据或语义编码向量，经由深度神经网络进行特征提取与表征学习，再通过策略生成网络或价值评估网络输出战术决策结果。整个流程在统一的损失函数驱动下进行端到端优化，从而在数据驱动的方式下自动学习最优策略。

在该架构中，系统的内部组织以模型层为核心，数据流呈线性或准线性结构，各功能模块以流水线形式紧密耦合。虽然缺乏显式的分层边界，但实际实现中常通过嵌入可解释性中间层（如注意力机

制、潜空间态势编码）来保持一定的结构化信息。端到端架构的优势在于能够减少人工设定的规则约束与中间接口，从而避免信息丢失与误差累积，提高决策效率。然而，这种组织形式对数据质量与模型泛化能力依赖极高，因此通常需要在系统外部建立高保真推演环境与离线训练体系，以支撑其在复杂动态战场中的有效性。

4.2.4 分布式架构

分布式架构是智能兵棋推演系统在多节点协同与大规模仿真条件下的关键组织形式。其核心思想是将系统功能按空间、任务或计算类型进行分布式划分，通过网络通信实现多模块的协同与资源共享。在组织形态上，分布式架构通常由若干自治节点组成，每个节点承担特定职能，如感知处理、态势评估、策略推理、任务分配、推演执行或战果评估等。节点间通过高可靠通信协议（如 DDS、gRPC、HLA 或 ROS2）进行数据与命令交换，形成面向任务的“功能分布—信息汇聚”模式。

该架构可根据需求选择不同的拓扑结构：集中式（由中央协调节点统一调度）、分层式（上层战略决策与下层战术执行分离）或去中心化（各节点依据局部信息进行协商与共识决策）。组织形式上强调自治与协同的平衡：每个节点具有独立计算与决策能力，但在全局上通过任务调度与数据一致性机制保持体系稳定运行。系统的通信层是分布式架构的核心纽带，需确保消息的实时性、一致性与安全性。常采用时间同步机制、状态复制、冗余备份与心跳检测以保障容错性与持续可用性。此外，为支撑不同计算资源的异构性，分布式架构往往结合容器化部署与动态负载调度，实现跨平台的弹性伸缩。总体而言，分布式架构的组织形式体现出模块自治、协同互联与高可靠性特征，是构建大规模智能推演平台的基础。

4.2.5 杀伤网架构

杀伤网（亦称作战效应网或 Kill Web）是面向体系作战效能的组织形式，其核心目标是通过信息流与能量流的多节点耦合，形成跨域协同的闭环打击体系。在智能兵棋推演系统中，杀伤网的组织形态通常以“感知-识别-决策-打击-评估”的连续链条为基本结构，不同功能节点（传感器、情报处理单元、指挥决策中心、武器平台、效果评估模块）通过网络互联形成多层、多通道的作战体系。

与传统线性 Kill Chain 不同，现代杀伤网的组织形式呈现网络化、动态化与去中心化特征。系统中不存在单一中心节点，而是多个功能单元依据信息优先级与任务需求动态组成“任务子网”，并通过协同协议实现资源的快速汇聚与任务的自组织执行。在算法层面，杀伤网组织形式常采用基于拍卖、博弈或强化学习的协同分配机制，以在多平台、多目标场景下实现火力资源的最优匹配。系统内部的任务链条既包括纵向的信息传递（从情报获取到武器释放），也包括横向的节点协同（同类型或异类型平台间的协调与支援）。战果评估单元则在链路末端对任务效果进行量化反馈，形成完整的闭环优化机制。该组织形式强调“体系作战、节点互联、效能最大化”，是面向未来智能化作战推演与决策支撑的关键架构模式。

4.3 智能兵棋推演系统算法

4.3.1 智能兵棋推演算法总述

智能兵棋推演系统算法通过对战场态势进行处理和分析，来选择最佳行动以最大化己方收益。智能算法的设计是实现高精度战场模拟和高效智能决策支持的关键，应针对不同场景和具体环境要求灵活选择。在兵器推演的算法设计中，主流的智能决策算法可分为知识驱动算法、数据驱动算法以及知识数据混合驱动算法三个范式。

本标准不对具体算法的设计做任何硬性约束，在算法设计时应根据推演任务场景的不同，选择合适的算法，需确保算法的稳定性、可扩展性与实时性。

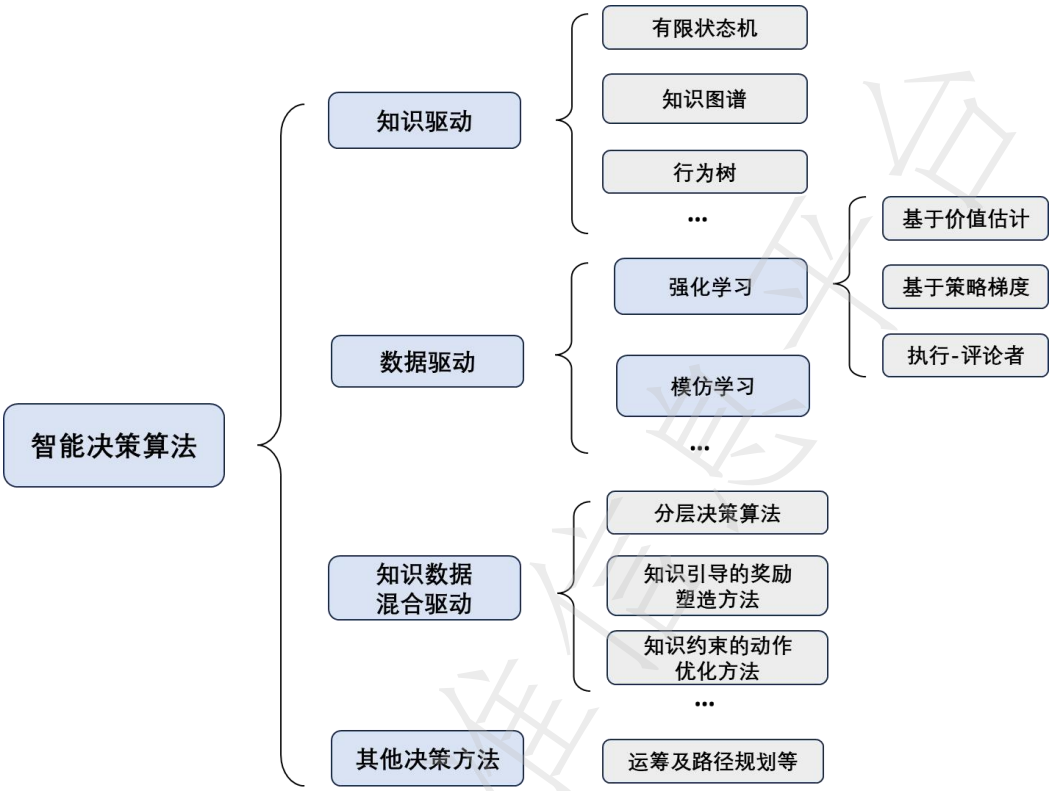

图 2　智能决策算法

4.3.2　知识驱动决策算法

基于领域知识驱动的决策算法是兵棋推演中的基础算法，通过预定义的逻辑、数学模型或启发式方法（如运筹及路径规划）生成决策。知识驱动决策算法的设计应保证推演过程符合军事规则和战术约束，同时具备良好的计算效率与扩展性。在具体实践中，应做到：

a） 采用模块化设计，支持动态加载和更新，适应战术规则的演变；

b） 具备良好的适应性，能够应对战场态势的动态变化；

c） 满足执行效率要求，支持大规模实体的实时计算。

4.3.3　数据驱动决策算法

4.3.3.1　强化学习

强化学习(ReinforcementLearning,RL)的核心是研究智能体与环境的相互作用，通过不断学习来寻找最优策略，以获得最大累积回报。强化学习可以建模为马尔科夫决策过程，其参数空间定义为一个五元组⟨S,A,P,R,γ⟩，这五个变量分别表示环境状态空间、动作空间、状态转移函数、奖励函数和折扣因子。依据智能体动作选取方式，可以将主流强化学习算法分为基于价值（value-based）、基于策略（policy-based）、及结合价值与策略的执行-评论者（actor-critic）方法。在具有多个智能体的场景中，可以采用多智能体强化学习算法，这是一种高效且通用的马尔科夫博弈求解方法，可解决多个智能体在共享随机环境中的序贯决策问题。

4.3.3.2　模仿学习

模仿学习（Imitation Learning, IL）是一类以专家示范为引导的机器学习方法，其核心思想是从专家提供的决策样本中归纳出有效的决策策略。在兵棋推演动态决策环境中，模仿学习能够有效利用高质量的行为数据，引导智能体在复杂对抗态势下实现接近专家水平的判断与决策。专家行为可以是来

源于历史战例中经过验证的真实指挥决策，也可以是从高水平指挥员在模拟推演中所积累的操作记录，甚至是为训练智能体而采集的人工干预数据。通过模仿这些具备专业军事经验的决策行为，智能体能够快速掌握基本战术原则、兵力运用方式和典型应对策略，从而显著缩短训练周期、降低探索成本。

4.3.4 知识与数据混合驱动决策算法

4.3.4.1 方法总述

知识与数据混合驱动已成为智能博弈决策算法发展的主流范式。融合知识驱动范式的高效性、可靠性和强泛化能力，能够利用先验规则约束搜索空间、提升学习效率，并在样本稀缺情境下保持稳定决策；又兼具数据驱动范式的适应性与发现能力，可从海量对抗数据中挖掘人类专家未能察觉的新模式与创新策略。

4.3.4.2 分层决策算法

将决策过程分层，不同层级采用不同的驱动范式。上层战略规划采用专家知识驱动，根据知识图谱、专家系统生成宏观任务指令。底层动作控制采用数据驱动算法，执行具体的、精细化的动作，如单个飞行器的机动控制。

随着大语言模型（Large Language Models, LLMs）和多模态基础模型的发展，大模型决策逐渐成为复杂对抗环境中智能决策的新范式。在兵棋推演系统中，基于领域知识驱动的专用大模型可以作为高层指挥智能体，负责任务分解、目标设定、资源调配与战术意图生成，从而实现更高层次的作战意图理解与更灵活的战略决策。

4.3.4.3 知识引导的奖励塑造方法

在强化学习中。在基础的环境奖励之上，根据知识库中的战术规则增加额外的奖励信号，将专家知识嵌入奖励函数。这可以提升强化学习算法训练效率，引导智能体学习符合战术原则的行为，避免探索危险或无效的动作。

4.3.4.4 知识约束的动作优化方法

利用专家先验知识对数据驱动算法的动作选择进行过滤或约束，确保决策的合理性。具体来说，智能体每步输出的原始动作须通过一个基于规则的“行为校验器”，以根据当前态势屏蔽无效、非法得到动作，缩减动作空间，同时保证决策符合规则约束。

5 模型构建技术规范

5.1 总体原则

智能兵棋推演系统智能模型是指对具有自主决策能力的作战实体或指挥单元进行数学建模和算法实现的计算机模型，该模型通过模拟真实作战环境中各类智能体的感知、决策、行动和学习能力，实现高度拟真的兵棋推演过程。智能兵棋推演智能体模型设计与开发应以模拟真实战场决策为核心，构建具备自主决策能力、战场适应性和军事合理性的智能体模型。总体要求包括：

a） 采用多源信息融合技术实现战场态势精准感知，为智能决策提供基础；

b） 基于各类智能架构实现智能决策，确保决策过程兼具合理性和适应性；

c） 支持多层级指挥协同和跨域联合作战；

d） 具备对抗环境下的强鲁棒性，能够应对战场不确定性；

e） 满足实时性要求，并保持决策过程的可解释性。

本章规范智能兵棋推演系统中智能模型的模块结构、接口规范及设计与训练流程，旨在确保各类模型均可在系统中统一加载、调用与复用，提升通用性与工程集成效率。

5.2 模型结构与接口

5.2.1 模型目录结构

为了实现兵棋智能体模型的高可移植性、强扩展性及良好维护性，开发者应统一采用模块化、分层化的目录结构。规范的模型根目录组织明确划分各模块职责、层级关系及文件分类标准，为后续模型的集成、调用和部署奠定坚实基础。模型根目录组织方式如图 3 所示，完整介绍见附录 A《空战算法与模型目录结构与必要模块文件》。

```
/model_root/
├── config/                          # 配置文件
│   ├── agent.yaml                   # 智能体配置
│   └── env.yaml                     # 环境配置
├── models/                          # 模型定义
│   ├── agent.py                     # 智能体类定义
│   ├── strategy_core.py             # 策略实现（规则/网络）
├── envs/                             # 环境实现
│   ├── base_env.py                  # 环境接口抽象类
│   └── env.py                       # 环境实现
├── utils/                           # 工具函数
├── train/                           # 训练相关文件
│   ├── train.py                     # 训练主流程
│   └── evaluate.py                  # 性能评估
├── saved_models/                    # 模型保存文件
├── docs/                            # 文档
│   ├── README.md                    # 项目说明
│   └── interface_spec.md            # 接口说明
└── run.py                           # 启动入口脚本
```

图 3　模型根目录组织方式

5.2.2　模型核心模块体系结构

模型的必要模块文件包括配置模块、策略模块、环境模块等，其中策略模块作为智能体的核心模块，采用“接口层-核心层”双层架构，实现从原始态势输入到标准化动作输出的全流程决策。具体架构如下。

a） 接口层

1） 实现方法：该层通过 agent.py 中的 Agent 类模块化实现，采用动态加载机制解析 agent.yaml 配置文件，利用反射技术实例化指定的策略核心（如规则引擎或深度学习模型），并通过标准化接口封装数据流转全流程，确保环境调用与策略执行的解耦；

2） 核心功能：接口层承担战场感知整合与决策动作控制双重职责。一方面接收原始环境态势数据，提取关键战场要素并输出维度标准化的状态向量；另一方面动态调度策略核心，将抽象决策指令转换为符合规范的标准化动作，同步执行多重安全校验，最终生成可执行指令完成战场交互闭环。

b） 核心层

1） 实现方法：该层通过 strategy_core.py 中的抽象基类统一策略实现范式，支持三类技术路径：基于专家经验构建的规则引擎采用逻辑链解析，深度强化学习模型通过神经网络映射状态-动作关系，混合策略则动态融合规则与学习模型的决策输出。所有策略均需实现动作接口，确保算法组件可插拔替换；

2） 核心功能：决策层专注于战场态势到动作策略的生成。规则引擎依赖预定义逻辑库处理高确定性场景，深度强化学习模型适应复杂态势的自主决策，混合策略通过安全规则约

束与学习模型协同实现智能性与鲁棒性平衡。最终输出抽象动作指令，由接口层转换为可执行指令，完成从认知到行动的闭环。

5.2.3 模型接口规范

标准化的输入输出格式是保障不同模块互通、模型部署落地以及后续集成的基础。应涵盖环境输入、智能体输出的统一格式。

a） 态势数据格式原则

1） 结构化设计：采用分层嵌套字典结构传递战场多维信息，覆盖作战实体状态（如飞机、舰艇、潜艇动态属性）、传感器数据、任务执行态势（巡逻、打击任务状态）、战场环境（气象、地理）及后勤保障（弹药、挂载状态）五大核心维度；

2） 字段约束：字段类型严格限定（如字典、列表、字符串），禁止未声明字段扩展；关键字段强制非空校验；空值字段需显式标注 None，避免隐性数据缺失导致逻辑异常。

b） 动作数据格式原则

1） 指令标准化设计：采用键值对结构组织指令，核心动作通过预定义键名标识；指令参数值需严格匹配语义规则，未声明字段或类型错误视为非法指令；

2） 安全双校验机制：进行资源审查，执行指令前动态校验关联资源状态：进行条令合规拦截，依据专家规则拦截危险操作。

c） 开发者交互要求

1） 态势解析自主性：开发者需根据业务逻辑设计态势解析器，提取关键信息生成状态向量；

2） 动作转换责任：模型输出的内部动作须映射至规范格式；

3） 调试支持：logged_messages 字段记录原始决策日志，供行为回溯分析。

态势和动作数据的完整字段定义见附录 B《空战算法与模型接口数据格式规范》（含表 B. 1 态势数据格式和表 B. 2 动作数据格式）。

5.2.4 模型必备函数定义

模型必备函数函为算法的各个特定环节提供必要的支持，下面给出必备函数的功能描述和基本形式，详细的输入输出参数见附录 C《空战算法与模型必备函数定义》。

a） 环境类（Env）必备函数

1） __init__() 方法

方法功能：构建环境实例并完成基本初始化操作；

方法形式：__init__() -> Env。

2） reset() 方法

方法功能：重置环境至初始状态，开始新一轮推演；

方法形式：reset() -> Dict。

3） step(action_dict) 方法

方法功能：执行所有智能体的动作，并推进环境至下一状态；

方法形式：step(action_dict: Dict) -> Tuple[obs, reward, done, info]。

4） get_env_info() 方法

方法功能：返回环境的基础配置信息（如动作空间、状态维度等）；

方法形式：get_env_info() -> Dict。

b） 智能体类（Agent）必备函数

1） __init__() 方法

方法功能：构建智能体对象，初始化策略结构与参数；

方法形式：__init__(agent_id: str, config: Dict) -> Agent。

2） save(path)方法

方法功能：保存当前策略或模型参数到指定路径；

方法形式：save(path: str) -> None。

3） load(path)方法

方法功能：从指定路径加载策略或模型参数；

方法形式：load(path: str) -> None。

5.3 模型设计与训练流程

5.3.1 训练初始化要求

在开始训练智能兵棋推演模型之前，应进行充分的初始化准备工作，以确保训练过程能够顺利开展并具有可控性。训练初始化要求包括以下几个方面：

a） 环境与场景配置：训练开始前，根据任务需求配置模拟环境的初始状态和参数（战场环境、兵力兵器部署、规则约束等），并明确这些要素。若存在多个训练场景，可制定场景选择策略或设置随机种子，确保模型在多样化情形下训练，避免过度拟合单一场景。

b） 模型结构与参数初始化：选择适当的算法和模型结构（如规则规划、机器学习模型或强化学习智能体），并合理初始化模型参数，避免梯度消失或发散。使用预训练模型时应正确加载预训练权重；从零训练时采用行业最佳实践的随机初始化。模型输入输出接口应与环境状态表示和动作空间对接，并经验证确保一致。

c） 训练超参数与配置：初始化时确定训练所需的超参数和配置（如学习率、折扣因子、训练步长、批次大小等）。本规范不限定具体取值，但要求训练前明确这些参数，并通过统一配置接口或文件管理。所有配置应设合理默认值，可根据需要灵活调整。

d） 随机种子与可重复初始化：为增强实验可重复性，训练初始化应允许设置随机数种子。在环境模拟、模型初始化、数据混洗等环节使用固定种子，可重复相同训练过程验证结果；更改种子则确保训练多样性。系统应提供设置随机种子的接口，并确保环境、模型、算法等不同模块的随机源统一可控。

e） 资源与设备准备：训练前检查硬件（如 CPU/GPU/TPU）和必要的软件库，确保资源就绪且与模型、环境兼容。长时间训练还应有容错续训机制，在中断时可从检查点恢复训练（参见 5.3.6 节）。

f） 初始化验证：完成以上准备后，应进行简要测试，如让模型运行少量步骤（可使用随机动作），验证训练闭环是否正常（接口匹配、观测和奖励反馈正确，日志记录和数据管道正常）。初始验证通过后方可开始大规模训练。

通过全面的训练初始化，能够为后续的模型训练奠定可靠基础，减少训练过程中不必要的中断和调试成本。

5.3.2 感知-判断-决策-行动流程标准

智能兵棋推演系统中的模型训练应遵循标准的感知-判断-决策-行动循环流程，以模拟真实决策主体（智能体）的行为闭环。在每一个推演时间步，模型都应完成以下阶段：

a） 感知阶段：模型通过环境接口获取当前战场态势，包括自身与敌方状态、环境变化及其他必要情报。需定义清晰的观察空间，确保模型获取决策所需的完整、无歧义的数据。环境应提供标准化的状态表示（如张量、符号特征或结构化数据）。模型应能接收并处理这些观测数据，可对原始信息进行滤波、归一化或特征提取，但不得遗漏关键态势信息。

b） 判断阶段：模型对感知阶段获取的战场态势数据进行深度解析与逻辑推理，构建局势认知框架。需完成信息关联性分析、态势趋势预测及威胁等级评估。判断过程应基于兵棋推演的核

心规则与战场常识，可引入知识图谱或规则库辅助推理，确保输出的局势认知结果准确、全面，为后续决策阶段提供明确的判断依据。同时，需建立判断结果的校验机制，若发现感知数据存在矛盾或缺失，应反馈至感知阶段触发二次信息获取，避免因误判导致决策偏差。

c） 决策阶段：模型基于内部策略或算法评估当前局势并给出决策，包括分析观测信息、预测局势发展和选择下一步行动方案。模型可通过规则引擎、博弈搜索算法或训练的策略网络等实现决策。本规范不限定具体算法，但要求决策模块能在有限时间内输出决策，且动作格式与取值范围符合环境动作空间要求。多智能体情况下，每个智能体应独立执行自身的感知和决策；若为多智能体对抗训练，则需定义各智能体的决策顺序或并发机制，以确保协调。

d） 行动阶段：模型通过标准接口将决策动作发送给环境，引发环境状态更新。动作接口应定义明确（动作类别如移动、攻击、侦察等及其参数格式），环境接收后按兵棋规则裁决并模拟结果。系统应校验动作有效性，违规或无效动作应被环境拒绝或纠正并给予反馈。环境完成状态更新后，进入下一个时间步，继续感知-判断-决策-行动循环。

上述感知、判断、决策、行动四个阶段应以固定的时序反复进行，直至达到训练终止条件。整个流程需要保证时序的一致性和数据传递的可靠性，避免阶段间的数据耦合不清或延迟累积。标准的感知-判断-决策-行动流程为模型提供了模拟真实指挥决策过程的框架，使训练得到的模型能够更好地适应实际应用场景。

5.3.3 奖励函数设计指导

在智能兵棋推演系统的模型训练过程中，奖励函数（Reward Function）的设计对模型行为的引导至关重要，应确保模型围绕预期作战目标优化策略。以下是奖励函数设计的指导原则：

a） 任务目标对齐：奖励函数应与兵棋推演的作战任务目标相对应。模型获得的奖励应反映决策行为对最终作战效果的贡献。例如，在攻防对抗场景中，歼灭对方兵力、达成占领目标等正面成果应赋予正向奖励，而己方损失、违背战术要求等结果应给予负向奖励。

b） 及时性与长期性平衡：奖励设计需平衡即时反馈与长期目标。适当的即时奖励有助于加快收敛，例如对阶段性成果（如一次成功防御或进攻）给予小额奖励，鼓励正确的短期行为。同时应以最终结果的长期奖励为主，避免模型只顾短期利益而忽视全局。可通过加权将最终胜负奖励与过程关键行为奖励相结合，但须避免权重失衡导致模型行为偏差。

c） 惩罚不当行为：奖励函数应针对违规或不当行为设置惩罚。例如，对超出规则的动作、消极拖延或友军误伤给予负奖励。惩罚机制可防止模型利用漏洞获取不当优势（避免“奖励函数漏洞利用”），确保模型策略在合理约束内优化。

d） 数值尺度与平滑性：奖励值的规模和分布应精心设计，避免数值过大导致训练不稳定或梯度爆炸。可对奖励值适当缩放或归一化，保持数值平滑。如采用强化学习，可裁剪累计回报或设定奖励上下限以提高训练稳定性。还应确保不同情境下奖励差异明显，便于模型区分成功/失败和优劣策略。

e） 可解释性与可调试性：奖励函数应尽量可解释，便于开发者理解模型行为。如必要，可将总奖励拆分为多项（如作战效能奖励、资源消耗惩罚等）并在日志中记录各项。这有助于分析模型策略偏好来源，便于在策略偏离预期时根据各分量贡献调整奖励参数。奖励参数调整应小步渐进，每次修改后观察影响，避免频繁大幅调整导致训练振荡。

奖励函数设计没有固定范式，应结合具体兵棋推演任务不断试验和改进，但所有修改均应记录在案（参考 5.3.5 节和 5.3.7 节），以保证训练过程的可追溯性。

5.3.4 训练终止与重置流程

训练终止条件和环境重置机制是训练过程中必不可少的环节，它们确保每轮训练（Episode）正确结束并为下一轮训练做好准备。训练终止与重置流程应满足以下要求：

a） 终止条件定义：训练开始前应明确每个训练回合的终止条件和整个训练过程的终止条件。前者可根据规则和目标设定，如达到预定时长、一方胜利、全军覆没、僵局超限或违规等；后者如模型性能达到预期或训练迭代次数/回合数达到上限等。满足终止条件时，应立即结束当前训练回合并进行终止处理；训练过程的终止可由人工指令或自动条件触发，停止训练前应保存模型状态（参见 5.3.6 节）并记录终止日志。

b） 终止处理：回合结束时，系统应执行终止处理：计算汇总该回合的绩效指标（累计奖励、损失情况、达成目标数等），输出最终结果和统计数据。应通知模型当前回合结束，以便重置内部状态。同时触发日志记录，将本回合关键数据写入日志（见 5.3.5 节）。

c） 环境重置：新一轮训练开始前应将环境重置到初始状态或生成新的初始场景。重置包括重置各作战单元位置和状态，并清空上轮遗留的事件、计数等。环境重置接口应是幂等的，即多次调用均能得到一致的起始状态（若初始状态有随机性，则每次重置重新采样随机种子）。重置后应确认环境状态有效，再开始下一回合训练。

d） 资源和状态清理：训练终止或每回合结束后，系统应释放或重置相关资源和状态，以防止影响后续训练。这包括清理缓存的数据、关闭不再使用的线程或进程、归零相关的计数变量等。对于长时间运行的训练，应定期清理资源，防止内存泄漏或资源占用累积。良好的终止与重置流程能够保证训练循环在多个回合间稳定运行，也为并行训练场景下的多环境重置提供了规范（见 5.3.8 节并行训练部分）。

通过严格定义训练终止条件和规范化重置流程，保证模型训练能够有序地分批进行，并为每一次独立的训练回合提供可重复的起点，从而提高训练过程的稳定性和实验对比分析的可靠性。

5.3.5 训练记录与日志要求

完整、详实的训练日志记录对于分析模型行为、诊断训练问题以及确保结果可追溯至关重要。智能兵棋推演系统应建立统一的训练记录与日志机制，包括以下要求：

a） 日志内容：日志应记录训练中的关键事件和数据。至少包括每个训练回合的开始结束时间、回合编号/标识、各时间步主要决策事件、即时和累计奖励，以及回合结束时的胜负结果或绩效指标。重要中间状态变化（如据点占领、部队重大损失等）也应记录。模型的内部评估值或不确定性评估信息可选择记录以辅助分析。

b） 日志格式：日志应采用结构化格式（如 JSON、CSV），便于解析和阅读。每条日志至少包含时间戳/时间步索引、事件类型、相关实体、动作或状态摘要、奖励值变动等字段。规范的日志格式便于汇总分析训练数据，并支持跨团队的理解交流。

c） 日志系统和接口：系统应提供统一的日志管理模块和接口，方便开发人员插入日志记录。日志记录应分级：概要日志（仅记录回合级信息）和详细日志（记录每个时间步细节），便于调试阶段获取详尽信息，批量训练阶段仅保留必要信息。避免过度记录造成性能问题，应支持按需启停不同详度的记录。

d） 实时监控与可视化：日志机制应支持实时监控，将关键训练指标（当前回合数、平均每回合奖励、最近 N 回合胜率等）实时输出到控制台或可视化仪表板，便于监控人员了解训练进展，及时发现奖励异常或训练停滞等情况并采取措施。实时监控输出应与日志数据源一致，确保事后分析与实时观察结论一致。

e） 日志保存与归档：训练完成后，应妥善保存和归档日志数据。重要训练实验的日志需长期保存，文件命名应包含实验版本或日期以便查阅。日志可与模型检查点一同归档，确保恢复历

史模型时有对应日志了解训练详情。对于大量日志数据，建议定期备份并清理旧日志释放存储空间，但应确保在实验报告周期内日志数据完整可用。

健全的训练日志记录机制，使得训练的每一步都有据可查。无论是在模型最终部署前进行性能验证，还是在出现异常行为时追溯训练源头，日志都是不可或缺的依据。符合要求的日志系统将大大提升训练流程的透明度和可诊断性。

5.3.6 模型检查点策略

在长周期的模型训练过程中，建立有效的模型检查点策略十分必要。模型检查点机制可以定期保存模型的状态，既用于容错续训，也便于后期分析和比较模型版本。模型检查点策略应满足以下要求：

a） 定期保存：训练系统应按预定频率定期保存模型参数及相关状态（如每 1000 步或每 10 个回合）。定期保存确保训练中断（如程序崩溃、断电等）时可从最近的检查点恢复，避免全部进度丢失。保存频率应权衡可靠性与性能：太频繁会影响训练速度，太少则增加数据丢失风险。

b） 触发式保存：除定期保存外，系统还应支持根据特定事件或条件触发检查点保存。例如，当模型达到新的性能高点时自动保存该版本；检测到训练出现异常时，保存当前模型供调试分析。触发式检查点确保关键节点的模型状态被保留，便于后续不同阶段性能对比。

c） 保存内容：检查点至少应包含模型完整参数（权重、偏置等）的快照，以及恢复训练所需的优化器状态、学习率等元数据。多智能体训练时，应保存每个智能体的模型参数并注明其角色。所有检查点文件应附带元数据文件，记录保存时的训练步数/回合数、时间戳及主要性能指标，便于后续查找。

d） 命名规范与版本管理：检查点文件命名应规范化，包含模型标识和保存序号，必要时附加实验标识或日期以避免歧义。对于多个检查点，系统应提供版本管理功能，可列出现有 检查点并允许按编号或日期加载指定文件继续训练或评估。旧的检查点可按策略定期清理（如仅保留最近 N 个或重要里程碑版本），但清理前应确认相关版本不再需要。

e） 恢复与验证：从检查点恢复训练应简便可靠。系统需提供加载检查点的接口，使用户可指定文件并重启训练。恢复后应验证模型参数和优化器状态正确载入，训练可继续且无明显性能突变或异常。系统还应支持从检查点加载模型用于推演模拟或部署测试。

合理的检查点策略既保障训练连续性，又方便模型性能比较和回溯分析，应严格遵循以上规范确保每个模型版本可靠保存和再利用。

5.3.7 实验版本管理与可复现机制

在智能兵棋推演模型的研发过程中，往往需要进行大量实验以调整模型和算法。为保证不同实验结果的可比性和可信度，应建立完善的实验版本管理与可复现机制：

a） 实验配置与追溯： 每次训练实验应指定明确的版本号或标识，并记录其全部配置（模型算法、超参数、训练环境、奖励函数等版本信息）。同时保存训练日志（见 5.3.5 节）、检查点（见 5.3.6 节）和评估报告等，并将这些产出与实验版本关联，以便追溯训练过程和结果。建议建立实验记录文档或数据库，记录版本 ID、训练时长、最终性能指标、主要结论等，方便查询历史实验。

b） 可重复试验：在相同版本的代码、配置和随机种子下，多次独立运行应产生等效的训练结果。为此需确保实验环境确定性：固定随机种子、使用相同推演场景和初始化，确保并行执行顺序一致。对于不可避免的非确定性因素，应在记录中注明，并通过多次重复实验以均

值、方差等统计指标描述结果稳定性。只有当实验结果可独立重复验证时，其结论和模型才可靠。

c） 模型基线和对比：版本管理中可指定某一模型版本作为基线进行对比。当尝试新的算法改进或参数调整时，应在记录中注明基线版本，并采用统一评估标准比较新旧版本性能差异。此方法确保改进有据，避免将环境变化等因素导致的性能变化误判为算法改进。所有对比实验的过程和结果也应纳入版本管理，确保日后可重现相同对比。

d） 团队协作与审批：多人协作开发时，不同人员的实验应有各自的分支或标识，避免相互覆盖成果。拟合并入主版本的改动需经充分验证和讨论，确认优于当前基线后再合并主版本库。重要里程碑版本应及时标记保存，防止后续修改影响其可复现性。

实验版本管理与可复现机制不仅提高了研发流程的规范性，还为后续的成果报告和产品化部署提供了可信的依据，这是保障智能兵棋推演算法可靠性的基石。

5.3.8 并行训练与加速执行支持

由于兵棋推演的复杂性和模型训练过程的长期性，为提高训练效率，系统应支持并行训练和加速执行机制，以充分利用计算资源并缩短模型收敛时间：

a） 并行环境实例：系统应支持多个兵棋推演环境实例并行运行，以同时从多个独立场景收集训练数据，加快经验积累。需确保各环境实例独立，使用各自的随机种子和日志记录，防止数据混淆。并行环境数量应可根据硬件资源动态调整，训练框架应提供参数配置并行度。

b） 分布式框架兼容性： 系统应兼容主流分布式训练框架（如 Ray、Horovod、DeepSpeed），支持多机多卡横向扩展。接口应标准化，确保训练任务在单机和分布式环境间平滑迁移，训练逻辑不因环境变化而改变。

c） 硬件加速：系统应充分利用硬件加速提升训练速度。例如在决策阶段用 GPU 加速神经网络计算；使用向量化运算或 SIMD 指令优化兵棋环境计算逻辑；在分布式环境中利用高速通信减少节点间同步开销。系统应支持专用 AI 加速芯片（如 TPU）或本地集群进行大规模并行训练。使用硬件加速时需注意训练结果一致性，确保单机、多机、多设备环境下模型结果一致。

d） 加速模拟：除了并行和硬件，加速执行还包括优化兵棋推演的运行效率。应调优环境执行引擎性能，如减少不必要计算、采用高效的数据结构和算法、降低非必要渲染开销，以便在相同时间内执行更多训练步。如果模拟允许，可提供快进模式或简化物理模式，在不影响决策正确性的前提下加快推演进程。所有加速手段应在不改变环境逻辑正确性的前提下实施，确保模型仍在真实可信的环境下训练。

并行训练和各种加速执行技术，可有效缩短智能兵棋推演模型的训练时间，尤其适用于大规模的深度学习模型的训练中。

6 模型验证评估技术规范

6.1 评估方法与流程

6.1.1 评估方法

智能兵棋推演系统的算法与模型评估需构建多维度验证体系，以系统化方法论确保技术方案的可靠性、鲁棒性与可扩展性。本标准采用仿真环境验证-实际推演校验-横向对比分析的三层递进框架，覆盖从理论假设到实践落地的全链路验证需求。

仿真测试：在仿真环境中，设置不同的想定和对抗场景，对算法与模型进行全面测试，评估其在各种情况下的性能表现。可采用自动化测试工具，批量运行测试用例，提高测试效率。

实际推演验证：组织实际的兵棋推演活动，邀请专业的推演人员参与，对算法与模型的决策能力进行实际验证。通过观察推演过程和分析推演结果，评估算法与模型的实用性和有效性。

对比评估：将开发的算法与模型与已有的同类算法和模型进行对比，分析其优势和不足，为算法与模型的优化提供参考。对比可从决策速度、准确性、适应性、可扩展性等多个方面进行。

6.1.2 评估流程

a） 基础环境构建阶段

根据算法与模型特性，从核心场景（城市巷战，空域攻防，海空联合，电子对抗，后勤压力或个性化设计的其他想定场景）中挑选合适场景，分别面向不同对抗强度生成想定，每种对抗强度下生成超过 100 种标准化测试用例，覆盖典型决策边界，并用于算法泛化性验证。

b） 仿真环境验证阶段

在可控仿真环境中实施分层压力测试，依次验证算法的基础性能与鲁棒性。从常规对抗开始，检验算法在标准参数下的决策逻辑完备性；逐步提升至高强度对抗，可以注入战场迷雾、通信丢包等扰动变量，观测决策延迟波动与策略稳定性；最终在极限施压场景中模拟多域复合打击（如电子干扰叠加后勤断链），探测系统崩溃阈值与失效模式。通过蒙特卡洛模拟批量执行测试用例，生成涵盖决策质量、资源效率与响应速度的三维评估图谱。

c） 实际推演校验阶段

将算法导入真实人机协同推演环境，渐进式提升对抗强度以验证实战效能。初期采用简单对抗，观察人类决策与机器建议的交互模式；进阶至更高强度的对抗，以评估复杂突发场景的处置能力。同步构建动态评估体系，量化人机认知对齐度、复合问题解决率等指标，并引入军事专家对关键决策节点进行合规性与创新性双轨评分。

d） 横向对比分析阶段

基于仿真与推演数据，开展多维度方法学对比。在相同的想定设计下，与传统规则引擎、经典强化学习算法及人类专家基准进行交叉验证，聚焦决策速度、场景迁移衰减率、对抗强度适应性等核心维度。

构建消融实验，分解算法各个模块，验证算法各个模块各自的贡献度，揭示所提算法与模型中技术路径的差异化竞争力与理论瓶颈。

6.2 性能指标要求

6.2.1 概述

智能兵棋推演系统的算法与模型性能指标需构建多维度量化评估体系，通过基础性能、决策质量与系统效能三类核心指标，客观反映系统在动态对抗环境中的技术成熟度与边界条件。基础性能指标聚焦计算资源效率（如响应延迟、并行规模），决策质量指标衡量策略的战术合理性与创新性（如目标达成率、风险规避系数），系统效能指标则评估人机协同的认知对齐度与复杂场景泛化能力（如跨想定策略迁移率、突发扰动容错阈值）。三类指标通过仿真推演数据联动分析，形成从微观算力消耗到宏观战役效能的完整观测链，为技术验证提供可量化、可追溯的优化锚点，确保智能决策能力与真实军事需求的精准适配。

6.2.2 基础性能指标评估

a） 实时性

衡量模型在推理过程中的计算效率，以每秒推理的样本数（IPS）或推理延迟（ms）为指标，需满足系统对模型部署的性能要求。

战术/战役/战略层分别设定差异化阈值，用于衡量算法与模型的计算效率与实时性。

b） 资源占用率

监控算法与模型在运行过程中对硬件资源（如 CPU、GPU、内存、磁盘等）的占用情况，以确保系统的稳定性和可靠性，避免资源过度占用导致系统崩溃。

在指定计算资源的服务器上，统计 99%分位硬件资源（如 CPU、GPU、内存、磁盘等）占用率，战术/战役/战略层分别设定差异化阈值，用于衡量算法与模型的资源占用率。

c） 可扩展性

系统支持并行推演的作战实体数量级与异构智能体类型兼容性，通过逐步增加推演实体密度，观测决策延迟与资源占用率的增长斜率。进一步，记录推演实体数量与种类增加时，算法决策质量的衰减率。通过记录资源增长的斜率与决策质量的衰减率，作为可扩展性量化指标的体现。

d） 训练稳定性

训练稳定性指算法在迭代优化过程中表现出的收敛一致性，包括损失函数波动幅度、策略震荡范围与对抗训练噪声的鲁棒性。其反映模型在数据分布偏移或超参数微调时的抗干扰能力，确保不同训练周期产出的决策模型具备可预期的性能基线，可通过以下量化性指标进行衡量：

1） 收敛一致性：统计 10 次独立训练的收敛步长方差；

2） 策略偏移度：对比相邻训练周期策略库的 Jensen-Shannon 散度；

3） 噪声容忍性：注入 20%标签噪声后，评估模型性能衰减。

6.2.3 决策质量指标评估

a） 胜率与任务完成率

胜率和任务完成率是评估智能兵棋推演算法绩效的基本指标，它们直接反映算法在模拟对抗中的效果。

算法的胜率是衡量其决策有效性的首要指标。在仿真对抗试验中，算法应达到预定的胜率目标，胜率指标应通过足够次数的重复推演获得，以确保具有统计显著性和稳定性。

对于多目标任务，除总体胜负外，还应统计任务完成率以综合评价算法表现。通过足够次数的重复推演，统计算法达到的任务完成率，以评估算法与模型的决策质量。

b） 战术合理性

决策方案符合军事原则与战场物理约束的程度（如火力覆盖密度不超过目标价值阈值）。可以通过专家评分（0-10 分制）与规则冲突检测（如违反战争法条款次数）双轨评估作为战术合理性的量化性指标。

c） 策略鲁棒性

鲁棒性测试旨在检验智能兵棋推演算法在各种非理想条件下的表现，确保其决策能力对噪声、异常和对抗性行为具有足够的耐受性。可通过对比完整信息场景与干扰场景的核心 KPI（如关键节点占领成功率）差值作为策略鲁棒性的量化指标。

6.2.4 系统效能指标评估

a） 认知对齐度

机器建议与人类指挥员决策逻辑的一致性水平，包含显性行动吻合度与隐性意图推理匹配度。首先，计算人机协同模式下机器建议采纳率，其次，分析被修正建议的语义偏差距离，将两个指标作为衡量算法认知对齐度的量化指标。

b） 跨域泛化能力

用于衡量单一场景训练模型在未见过场景（如从城市巷战迁移至丛林战）的性能保持率。通过计算同一个算法与模型在不同想定设计下的 KPI（如关键节点占领成功率）方差，作为衡量算法跨域泛化能力的量化指标。

c） 抗扰容错阈值

系统在连续扰动（如通信中断+传感器故障叠加）下维持最小有效决策能力的时间窗口。注入多模态故障（从单点失效到级联崩溃），记录系统进入不可逆失效状态的临界点。通过记录该临界点对应的扰动噪声方差与输入的故障数量，作为衡量算法抗扰容错阈值的量化指标。

d） 决策可解释性

算法决策逻辑的可追溯性与人类认知兼容性，包括行动建议的因果链推导、关键决策因子的显性化表达以及对抗性样本的归因解析能力。其确保人类操作者对机器决策的信任建立与风险预判。可通过因果链完整性，语义可理解度与反事实解释力三方面对其进行衡量。通过计算模型决策解释与人类专家对决策解释的匹配度，作为衡量算法决策可解释性的量化指标。

e） 决策安全约束

算法决策必须遵守预先设定的各项安全约束条件，确保其行为在军事和技术安全范围内。首先，算法不得违反作战规则和交战规则，例如不得攻击被明令禁止的非作战目标，不得进行超出任务授权范围的行动，不得有任何“误伤”己方单位的决策。其次，算法的输出不得对仿真系统安全造成威胁，例如发送非法指令导致系统崩溃或资源耗尽。这要求算法在设计时考虑安全边界，对输出的动作进行约束筛查（例如对攻击范围、武器使用等限定条件）。最后，通过记录在博弈推演过程中违背安全约束的决策次数，计算违背安全约束的决策比例，作为衡量算法决策安全约束的量化指标。

7 模型部署与推演系统集成技术规范

7.1 部署环境

a） 硬件环境：根据算法与模型的计算需求，结合实际情况，选择合适的硬件设备，如：

1） 计算设备：建议使用多核高性能处理器，满足基础模型推理与多线程任务处理，确保计算资源的高效利用。若使用深度学习或其他计算密集型算法，可结合实际任务负载采用适当规模的硬件加速方案（如 GPU、TPU 或 FPGA 等），以优化计算能力并提升模型推理速度。

2） 存储设备：建议采用高性能存储，具备高速缓存，确保推演数据、模型参数以及系列日志文件的快速存取，并建议配置冗余机制以增强数据可靠性。

3） 网络带宽：考虑频繁的数据交互，特别是在云端部署时，需提供稳定的高带宽网络，以确保数据传输的高效与安全。

4） 综合性能：为应对高负载的计算环境，应配备高效的散热系统及空调设备，以维持设备与环境的适宜温度。硬件设施应具有高度可靠性，并对核心部件实施冗余设计，定期进行检查和维护。

b） 软件环境：搭建稳定的软件运行环境，包括操作系统、编程语言、框架库、依赖包等，确保算法与模型能够在目标环境中正常运行。需注意软件版本的兼容性和安全性，及时更新软件补丁和漏洞修复。

1） 操作系统：根据硬件平台选择选择合适的操作系统，推荐使用 Linux 类操作系统作为开发和运行环境，以保证在高性能计算环境中稳定良好的表现。

2） 编程语言与框架：对于以数据处理和强化学习为主的智能兵棋推演算法，Python 为首选编程语言。建议结合主流的科学数据计算库及强化学习框架进行模型的开发。建议进行容器化部署，确保部署环境的一致性，支持进行容器编排和管理。

3） 依赖包管理：通过虚拟环境管理工具进行隔离管理，确保所使用的各版本依赖包之间的兼容性，避免依赖冲突并及时进行更新与漏洞修复。

7.2 集成规范

a） 与智能兵棋推演系统的集成：将算法与模型集成到智能兵棋推演系统中，实现与系统的各个模块之间的无缝对接和信息交互。具体包括：模型应能够输出预测结果，为推演引擎的决策提供参考；模型可以通过智能体框架获取所需态势信息，并根据兵棋推演的环境做出响应；同时，模型应具备通过决策总线与系统中的其他模块交换信息，进行多方协同的能力。

b） 接口规范：算法与模型应遵循系统所定义的各类接口规范，如兵棋智能体接口、推演环境接口等，以保证能够通过标准接口接收战场态势信息、发送动作指令，并与其他模块进行协同工作。接口应符合系统整体设计要求，采用标准协议，确保高效简洁。接口的输入输出数据的格式需要符合系统要求，对于战场态势信息等结构化数据，建议采用 JSON 等通用数据格式，保证数据的可解析性与高效传输。此外，模型应能够与系统其他模块实现数据同步，可借助消息队列实现模块间的异步消息传递，保证系统的响应能力。

c） 兼容性测试：在集成过程中，应进行兼容性测试，验证算法与模型在不同系统版本及硬件环境下的运行情况，确保系统的稳定性和可靠性。兼容性测试需覆盖多维度场景，如测试算法模型在不同类型硬件加速器（根据实际情况考虑如 GPU、FPGA 等）环境下的运行稳定性，以及在低带宽、高延迟等网络条件下的传输完整性与系统容错能力，以确保系统在全场景下的可靠性。

d） 维护与升级要求：为实现环境一致性并便于快速部署与更新，建议将算法与模型封装为 Docker 容器。通过其支持的滚动升级机制，可以实现在不影响系统正常运行的情况下平滑更新模型。建议为每个模型和算法版本创建标签，确保可追溯和版本控制，以便后期审查和追踪。此外，建议详细设计并实施版本回滚策略，如在出现问题时，通过备份快速恢复到稳定版本。加强系统的监控和日志记录，及时发现潜在问题并进行修复。

附 录 A
空战算法与模型目录结构与必要模块文件

A.1 概述

开发者应统一采用模块化、分层化的目录结构。规范的模型根目录组织明确划分各模块职责、层级关系及文件分类标准，为后续模型的集成、调用和部署奠定坚实基础。模型根目录组织方式如图A.1所示。

```
/model_root/
├── config/                          # 配置文件
│   ├── agent.yaml                   # 智能体配置
│   └── env.yaml                     # 环境配置
├── models/                          # 模型定义
│   ├── agent.py                     # 智能体类定义
│   ├── strategy_core.py             # 策略实现（规则/网络）
├── envs/                             # 环境实现
│   ├── base_env.py                  # 环境接口抽象类
│   └── env.py                       # 环境实现
├── utils/                           # 工具函数
├── train/                           # 训练相关文件
│   ├── train.py                     # 训练主流程
│   └── evaluate.py                  # 性能评估
├── saved_models/                    # 模型保存文件
├── docs/                            # 文档
│   ├── README.md                    # 项目说明
│   └── interface_spec.md            # 接口说明
└── run.py                           # 启动入口脚本
```

图 A.1 模型根目录组织方式

A.2 config（配置模块）

该目录存放智能体策略与推演环境的配置文件，支持YAML或JSON格式。通常包含：

a） agent.yaml：定义智能体相关参数，包括策略类型、超参数设置、行为规则 参数等。支持多种策略实现的参数配置；

b） env.yaml：定义兵棋环境参数，包括地图信息、作战单元属性、推演时间控制等。

A.3 models（策略模块）

该目录负责智能体的核心逻辑实现，是系统的核心组成部分，作为决策引擎，实现了从状态到动作的映射关系。通常包含：

a） agent.py：智能体类的定义文件，负责整合策略模块，处理感知输入，输出 动作指令。作为智能体的接口层，协调策略实现与环境交互；

b） strategy_core.py：策略核心实现文件。该模块不限定具体实现方式，既可采 用深度神经网络，也可实现规则逻辑、启发式算法或混合策略。

A.4 envs（环境模块）

该目录用于实现兵棋推演环境，包括封装平台接口、处理状态转移逻辑等，支持模拟真实推演过程并反馈训练样本。通常包含：

a） base_env.py：平台提供的统一环境接口抽象类，开发者继承此类；

b） env.py：实际推演环境类，实现 reset()、step()、render()等核心函数。

A.5 utils（工具模块）

该目录为空，留给开发者根据项目需求自行添加通用工具函数，如随机数种子控制、数据转换、评价指标等。

A.6 train（训练模块）

该目录实现智能体模型的训练、验证与性能评估。通常包含：

a） train.py：主训练流程脚本，包括数据采样、策略优化、模型保存；

b） evaluate.py：评估脚本，支持模型性能测试、可视化输出、推演结果统计等。

A.7 saved_models（存储模块）

该目录用于存放训练过程中保存的模型权重、优化器状态及相关快照文件，支持模型的断点续训、调用部署和行为复现。还可作为模型推理或逆行测试的输入来源，便于开展策略效果验证与重复实验。

A.8 docs（文档模块）

该目录用于存放项目说明、模型使用文档与接口协议定义等。通常包含：

a） README.md：项目总体说明，对整体模型结构与使用方式的简要介绍；

b） interface_spec.md：模型接口规范文档，对外接口字段说明与输入输出数据格式定义。

A.9 run.py（启动模块）

是智能兵棋推演系统的统一启动入口，支持训练、评估和推理等多种运行模式。开发者可在该脚本中灵活配置对抗场景的关键参数，如：推演轮数与回合数；对抗智能体组合与策略类型；推理模式；模型加载路径；日志与结果保存设置。开发者可根据任务需要自由扩展推演逻辑，使系统适配不同对抗策略、作战任务或评估指标。

附 录 B

空战算法与模型接口数据格式规范

B.1 态势数据格式

B.1.1 格式规范

态势数据格式规范见表 B.1。

表 B.1 态势数据格式

编号	参数名	类型	含义说明	备注
1	side_dic	dict	当前推演中各方的信息	包含己方与敌方 ID、立场等
2	aircraft_dic	dict	己方与敌方飞机对象信息	位置、速度、航向、状态等
3	ship_dic	dict	己方与敌方舰艇信息	同上
4	submarine_dic	dict	己方与敌方潜艇信息	同上
5	sensor_dic	dict	所有单位传感器状态与性能信息	包括雷达、红外、声纳等
6	contact_dic	dict	当前探测到的目标列表及其属性	识别等级、位置、威胁等
7	doctrine_dic	dict	各作战单元适用的条令配置	武器使用、交战规则等
8	mssnpatrol_dic	dict	巡逻任务信息	任务分配、执行状态等
9	mssnstrike_dic	dict	打击任务信息	同上
10	loadout_dic	dict	挂载信息	武器类型、剩余弹药、状态等
11	mount_dic	dict	挂架信息	对应挂载编号与可用性
12	magazine_dic	dict	弹药库状态	储量、补给能力等
13	weapon_dic	dict	武器列表	武器种类、射程、命中率等
14	waypoint_dic	dict	飞行或航行路径点信息	路径规划与航线目标
15	weather	str	天气信息	会影响探测与武器使用
16	response_dic	dict	系统返回信息	包括执行状态、警告、错误等
17	logged_messages	dict	推演日志	可选调试信息，不直接决策

B.1.2 示例

aircraft_dic 包含己方和敌方飞机的相关信息。每个飞机的信息包括其位置、速度、航向等。

```
aircraft_dic = {"blue": {
    "id": 101,
    "position": [40.7128, -74.0060],
    "speed": 250,
    "heading": 180},
  "red": {"id": 201,
    "position": [40.7129, -74.0061],
    "speed": 240,
```

"heading": 180}}

B.2 动作数据格式

B.2.1 格式规范

动作数据格式规范见表 B.2。

表 B.2 动作数据格式

编号	参数名	类型	含义说明	备注
1	add_mission	dict	添加一个新任务	包括类型、名称等参数
2	set_mission_status	str	启用或禁用任务	控制任务是否激活
3	set_mission_doctrine	dict	设置任务的作战条令	
4	add_area	list	添加一个区域	由多个经纬度点组成
5	add_wayline	list	添加一条预设航线	包括航点、速度、高度等
6	assign_area_to_mission	str	将区域绑定到任务	如巡逻区、防御区等
7	assign_wayline_to_mission	str	将航线绑定到任务	
8	assign_target_to_mission	str	给任务分配攻击目标	
9	assign_unit_to_mission	str	将平台加入指定任务	飞机、舰艇等
10	set_unit_status	dict	启动/关闭平台系统	如雷达、武器等

B.2.2 示例

add_mission 是一个字典，用于添加一个新任务。它包含任务的类型、名称、目标等参数。

add_mission = {
"mission_type": "strike",
"mission_name": "AirStrike-001",
"target": "enemy_airbase"}

附 录 C
空战算法与模型必备函数定义

C.1 环境类（Env）必备函数

C.1.1 __init__() 方法

方法功能：构建环境实例并完成基本初始化操作；
方法形式：__init__() -> Env；
输入参数：无；
输出参数：见表 C.1。

表 C.1 __init__() 方法输出参数

参数名	数据类型	说明
env	Env	环境实例对象

C.1.2 reset() 方法

方法功能：重置环境至初始状态，开始新一轮推演；
方法形式：reset() -> Dict；
输入参数：无；
输出参数：见表 C.2。

表 C.2 reset() 方法输出参数

参数名	数据类型	说明
obs_dict	Dict[str, Any]	初始状态字典，包含所有智能体的观测信息

C.1.3 step(action_dict) 方法

方法功能：执行所有智能体的动作，并推进环境至下一状态；
方法形式：step(action_dict: Dict) -> Tuple[obs, reward, done, info]；
输入参数：见表 C.3；

表 C.3 step(action_dict) 方法输入参数

参数名	数据类型	说明
action_dict	Dict[str, Any]	智能体动作字典，键为智能体 ID，值为动作

输出参数：见表 C.4。

表 C.4 step(action_dict) 方法输出参数

参数名	数据类型	说明
obs	Dict[str, Any]	所有智能体新的状态观测
reward	Dict[str, float]	各智能体的奖励值
done	Dict[str, bool]	是否结束标志，含 "__all__"全局结束标志
info	Dict[str, Any]	诊断信息（可选）

C.1.4 get_env_info() 方法

方法功能：返回环境的基础配置信息（如动作空间、状态维度等）；
方法形式：get_env_info() -> Dict；
输入参数：无；
输出参数：见表 C.5。

表 C.5　get_env_info() 方法输出参数

参数名	数据类型	说明
env_info	Dict[str, Any]	包含状态空间、动作空间、最大步数等信息

C.2　智能体类（Agent）必备函数

C.2.1　__init__() 方法

方法功能：构建智能体对象，初始化策略结构与参数；
方法形式：__init__(agent_id: str, config: Dict) -> Agent；
输入参数：见表 C.6；

表 C.6　__init__() 方法输入参数

参数名	数据类型	说明
agent_id	str	智能体唯一标识符
config	Dict[str, Any]	策略配置参数

输出参数：见表 C.7。

表 C.7　__init__() 方法输出参数

参数名	数据类型	说明
agent	Agent	智能体实例

C.2.2　save(path) 方法

方法功能：保存当前策略或模型参数到指定路径；
方法形式：save(path: str) -> None；
输入参数：见表 C.8；

表 C.8　save(path) 方法输入参数

参数名	数据类型	说明
path	str	保存路径

输出参数：无。

C.2.3　load(path) 方法

方法功能：从指定路径加载策略或模型参数；
方法形式：load(path: str) -> None；
输入参数：见表 C.9；

输入参数：表 C.9　load(path) 方法输入参数

参数名	数据类型	说明
path	str	模型文件路径

输出参数：无。