UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUCIANO HENRIQUE ARENDT RODRIGUES

ANÁLISE COMPARATIVA DE DIFERENTES ALGORITMOS DE APRENDIZADO POR REFORÇO APLICADOS AO MAHJONG

BAURU
Novembro/2025

LUCIANO HENRIQUE ARENDT RODRIGUES

ANÁLISE COMPARATIVA DE DIFERENTES ALGORITMOS DE APRENDIZADO POR REFORÇO APLICADOS AO MAHJONG

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. André Luis Debiaso Rossi

BAURU
Novembro/2025

R696a

Rodrigues, Luciano Henrique Arendt
Análise comparativa de diferentes algoritmos de Aprendizado por Reforço aplicados ao Mahjong / Luciano Henrique Arendt Rodrigues. -- Bauru, 2025
75 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientador: André Luis Debiaso Rossi

1. Aprendizado por Reforço. 2. Inteligência Artificial. 3. Mahjong. 4. PPO. 5. A2C. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Luciano Henrique Arendt Rodrigues

Análise Comparativa de Diferentes Algoritmos de Aprendizado por Reforço Aplicados ao Mahjong

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. André Luis Debiaso Rossi
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profa. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dr. Kelton Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 23 de Novembro de 2025.

Resumo

Este trabalho apresenta uma análise comparativa de diferentes algoritmos de Aprendizado por Reforço Profundo aplicados ao jogo *Riichi Mahjong*, um ambiente caracterizado por múltiplos agentes, grande espaço de estados e informação parcial. Foram implementados e avaliados três agentes baseados em abordagens distintas: *Deep Q-Learning* (DQL), *Advantage Actor–Critic* (A2C) e *Masked Proximal Policy Optimization* (MaskedPPO), todos utilizando uma rede neural convolucional. Os experimentos foram conduzidos em dois cenários: contra agentes do artigo *Variational Oracle Guiding for Reinforcement Learning* (Han et al., 2022) e em partidas entre os próprios agentes, com a inclusão de um agente aleatório. Os resultados mostraram que, em um ambiente de treinamento com número de episódios fixo, os métodos baseados em política apresentaram melhor desempenho e maior capacidade de adaptação, embora exigindo mais tempo e recursos computacionais para o treinamento. O agente *MaskedPPO* obteve a melhor taxa de vitória e estabilidade entre os modelos testados. Apesar de o ambiente reduzido e a ausência de pré-treinamento supervisionado limitarem o desempenho absoluto frente ao modelo de referência, a comparação permitiu identificar diferenças claras entre as abordagens. Os resultados reforçam a adequação dos métodos baseados em política para ambientes complexos e parcialmente observáveis, como o *Mahjong*, e destacam a relevância do Aprendizado por Reforço como ferramenta de pesquisa em inteligência artificial aplicada a jogos.

Palavras-chave: Aprendizado por Reforço, Inteligência Artificial, *Mahjong*, PPO, A2C, DQL.

Abstract

This work presents a comparative analysis of different Deep Reinforcement Learning algorithms applied to the game of *Riichi Mahjong*, an environment characterized by multiple agents, partial information, and a large state space. Three agents based on distinct approaches were implemented and evaluated: Deep Q-Learning (DQL), Advantage Actor–Critic (A2C), and Masked Proximal Policy Optimization (MaskedPPO), all employing a convolutional neural network. The experiments were conducted in two scenarios: against agents from the paper Variational Oracle Guiding for Reinforcement Learning (Han et al., 2022) and in matches between the trained agents themselves, including a random agent. The results showed that, under a fixed number of training episodes, policy-based methods achieved superior performance and greater adaptability, although they required more computational time and resources for training. The MaskedPPO agent achieved the highest win rate and the most stable performance among the tested models. Although the reduced training environment and the absence of a supervised pre-training phase limited the agents absolute performance compared to the reference model, the comparison revealed clear differences between the approaches. The findings highlight the suitability of policy-based methods for complex and partially observable environments such as *Mahjong*, and reinforce the relevance of Deep Reinforcement Learning as a research tool for artificial intelligence applied to strategic games.

Keywords: Reinforcement Learning, Artificial Intelligence, Mahjong, PPO, A2C, DQL.

Lista de figuras

Lista de abreviaturas e siglas

A2C	*Advantage Actor–Critic*
API	*Application Programming Interface*
AR	Aprendizado por Reforço
CNN	*Convolutional Neural Network* (Rede Neural Convolucional)
CPU	*Central Processing Unit*
DL	*Deep Learning* (Aprendizado Profundo)
DQL	*Deep Q-Learning*
DQN	*Deep Q-Network*
EMA	*European Mahjong Association*
FPS	Frames Per Second
GPU	*Graphics Processing Unit*
GRU	*Gated Recurrent Units*
IA	*Inteligência Artifial*
KL	*Kullback-Leibler*
MC	*Monte Carlo*
MCTS	*Monte Carlo Tree Search*
MDP	*Markov Decision Process* (Processo de Decisão de Markov)
ML	*Machine Learning* (Aprendizado de Máquina)
PPO	*Proximal Policy Optimization*
ReLU	*Rectified Linear Unit* (Unidade Linear Retificada)
RN	*Rede Neural*
SB3	*Stable-Baselines3*
TD	*Temporal Difference* (Diferença Temporal)
VLOG	*Variational Latent Oracle Guiding*

Sumário

1 Introdução

O Aprendizado por Reforço (AR) tem se tornado uma das principais abordagens da Inteligência Artificial (IA) para a resolução de problemas complexos de tomada de decisão (HU et al., 2024). Os jogos tem sido um ambiente muito utilizado para experimentação e avanços na área (HU et al., 2024). Exemplos notáveis incluem o *AlphaGo*, que derrotou campeões mundiais de *Go* ao combinar redes neurais profundas com AR (MOZUR, 2017), e o *MuZero*, que demonstrou capacidade de aprendizado eficiente sem conhecimento prévio das regras dos jogos (SCHRITTWIESER et al., 2020). No contexto de jogos com informação incompleta, como o *poker*, o algoritmo *Pluribus* conseguiu superar oponentes humanos lidando com a imprevisibilidade do ambiente (BLAIR; SAFFIDINE, 2019).

Entre os desafios computacionais complexos para o AR está o *Mahjong*, um jogo de estrátegia que envolve múltiplos jogadores, um grande espaço de estados e informações ocultas, tornando a tomada de decisões mais complexa se comparado a jogos de informação perfeita como o xadrez ou o *Go*. Diferente desses jogos, no *Mahjong*, o jogador precisa tomar decisões ótimas sem total acesso às informações do tabuleiro. O *Microsoft Suphx*, um agente baseado em AR, foi um dos primeiros sistemas a atingir nível profissional no *Mahjong* japonês, demonstrando o potencial da IA nesse contexto (LI et al., 2020).

É notável que o uso de Redes Neurais Convolucionais (em inglês, *Convolutional Neural Network*, CNN) (MOZUR, 2017; SCHRITTWIESER et al., 2020) são amplamente utilizadas em agentes treinados por AR no campo de jogos, especialmente no *Mahjong* (ZHAO; HOLDEN, 2022; GAO et al., 2019; LI et al., 2020). Porém, apesar dos avanços recentes, a escolha do algoritmo de AR mais eficaz para jogar *Mahjong* ainda é uma questão em aberto. Diversas abordagens têm sido exploradas na literatura, como *Deep Q-Learning* (DQL) (HAN et al., 2022), *Proximal Policy Optimization* (PPO) (LI et al., 2022) e política de Monte Carlo (MC) (MIZUKAMI; TSURUOKA, 2015). No entanto, a comparação direta entre diferentes algoritmos de AR no contexto do *Mahjong* ainda carece de devidas descrições comparativas.

Dessa forma, este trabalho busca realizar uma análise comparativa entre diferentes algoritmos de AR aplicadas ao *Mahjong*. O objetivo é avaliar o desempenho de agentes treinados com abordagens distintas e identificar quais estratégias apresentam melhor eficácia para lidar com as características específicas do jogo. Ao investigar essas questões, espera-se contribuir para a compreensão do AR e das algoritmos utilizados em ambientes complexos e para o desenvolvimento de agentes mais eficientes para jogos de estratégia com informação incompleta.

1.1 Problema

A aplicação de AR no *Mahjong* apresenta desafios únicos devido à complexidade do jogo e à informação oculta disponível para os jogadores. Estudos recentes propõem adaptações de modelos e algoritmos de aprendizado para lidar com esse problema (LI et al., 2022), além de compararem o desempenho de diferentes estratégias para jogar *Mahjong* (ZHAO; HOLDEN, 2022).

A questão central deste estudo é: "Qual algoritmo de AR apresenta melhor desempenho na tarefa de jogar *Mahjong*?". A resposta a essa questão permitirá entender quais abordagens são mais eficazes na otimização da tomada de decisão do agente inteligente.

Especificamente, o estudo buscará investigar os problemas:

- Diferentes algoritmos de AR podem levar a desempenhos significativamente distintos?
- A escolha do algoritmo impacta diretamente a eficiência do treinamento, o tempo de convergência e a capacidade do agente de tomar decisões estratégicas ao longo da partida?

1.1.1 Delimitações

Este estudo foca exclusivamente no *Riichi Mahjong* com 4 jogadores, uma variante altamente praticada (MILLER, 2015) e compatível com um simulador [1] previamente utilizado no treinamento de IA com AR (HAN et al., 2022). Diferentemente de trabalhos anteriores, que comparam agentes com diferenças estruturais mais amplas, como em Zhao e Holden (2022), no presente estudo, a arquitetura da Rede Neural (RN) será mantida fixa. Essa decisão garante que a comparação entre diferentes algoritmos ocorra de maneira justa, sem interferências advindas de variações na modelagem da representação do estado. Dessa forma, a análise será focada exclusivamente no impacto das algoritmos de AR sobre o desempenho do agente.

1.2 Justificativa

A aplicação de AR em jogos (HU et al., 2024) tem se mostrado uma poderosa ferramenta para o estudo e avanço da IA, pois oferece ambientes controlados para testar agentes em tarefas complexas de tomada de decisão.

A escolha do *Mahjong* como ambiente de estudo se justifica pela sua complexidade e informação incompleta. Com um grande espaço de estados e decisões baseadas em dados parciais, o *Mahjong* representa um excelente modelo para ambientes dinâmicos e incertos, que são comuns em várias áreas da IA, como negociação automatizada e robótica.

[1] Disponível em: <https://github.com/Agony5757/mahjong/>. Acesso em: 20 nov. 2025.

Estudar como diferentes algoritmos de AR se comportam nesse contexto pode trazer contribuições relevantes não só para melhorar as estratégias dentro do jogo, mas também para o desenvolvimento de soluções em problemas reais onde as decisões precisam ser feitas sob condições de incerteza.

1.3 Objetivos

O objetivo geral do projeto é investigar o desempenho de diferentes algoritmos de AR aplicadas ao jogo *Mahjong*.

Neste trabalho são propostos os seguintes objetivos secundários:

- Investigar as redes neurais mais comumente utilizadas para AR e identificar as mais promissoras para aplicação no jogo de *Mahjong* e em jogos de maneira geral.
- Identificar e analisar os algoritmos de AR mais utilizadas em ambientes de jogos;
- Treinar agentes com diferentes algoritmos de AR selecionadas;

1.4 Organização do Trabalho

O restante deste trabalho está estruturado da seguinte forma:

No **Capítulo 2**, é apresentada a fundamentação teórica, abordando os principais conceitos relacionados ao AR, a arquitetura de RN escolhida para o trabalho e sobre o *Mahjong*.

No **Capítulo 3**, é realizada a revisão da literatura, contemplando estudos e pesquisas recentes que aplicam técnicas de AR em jogos, com destaque para o *Mahjong* e outros ambientes de tomada de decisão complexa.

No **Capítulo 4**, é descrito o método de pesquisa, detalhando as etapas do desenvolvimento experimental, incluindo a seleção das políticas de AR, o ambiente de simulação e os procedimentos utilizados para o treinamento dos agentes.

No **Capítulo 5**, são apresentados e discutidos os resultados dos experimentos, com a análise comparativa do desempenho dos agentes treinados segundo as diferentes políticas de AR avaliadas.

Por fim, no **Capítulo 6**, são expostas as conclusões do trabalho, ressaltando as principais contribuições obtidas, as limitações encontradas e possíveis direções para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo tem como objetivo apresentar os principais fundamentos que sustentam o desenvolvimento deste trabalho. São introduzidos os principais conceitos relacionados a AR, com ênfase em suas políticas de aprendizado e nas abordagens que utilizam RN para otimização de agentes inteligentes.

Inicialmente, são abordados os princípios gerais do AR e sua formulação como um problema de decisão sequencial. Em seguida, são apresentadas as principais políticas empregadas nessa área, incluindo métodos baseados em valor, em política e híbridos, destacando suas características e aplicações. Também são discutidos os papéis das RN no contexto do AR e as técnicas de estabilização e treinamento utilizadas em agentes modernos.

Por fim, o capítulo introduz o uso de jogos como ambientes experimentais para pesquisa em AR e descreve brevemente as particularidades do *Riichi Mahjong*, que serve como base para os experimentos conduzidos neste trabalho.

2.1 Aprendizado de Máquina e Aprendizado por Reforço

A IA tem origens que remontam à Antiguidade, com representações de seres artificiais inteligentes na mitologia e, posteriormente, com o pensamento lógico de Aristóteles. Contudo, foi apenas após o surgimento dos computadores digitais, no pós-guerra, que a possibilidade de reproduzir processos mentais humanos por meio de programas passou a ser estudada sistematicamente (BUCHANAN, 2005).

O termo Inteligência Artificial foi definido em 1956, na Conferência de Dartmouth, por John McCarthy, evento que consolidou a IA como um campo científico independente. Nas décadas seguintes, programas como o *Logic Theorist* e o *General Problem Solver* marcaram o início da pesquisa em raciocínio automatizado (BUCHANAN, 2005; RUSSELL; NORVIG, 2021).

Com o avanço computacional e o crescimento dos dados disponíveis, a área passou a focar em métodos de aprendizado, originando o Aprendizado de Máquina (em inglês, *Machine Learning*, ML), voltado à extração de padrões e à adaptação baseada em experiência (MITCHELL, 1997; ALPAYDIN, 2020). Entre suas abordagens, destaca-se o AR, em que um agente aprende por interação com o ambiente, ajustando suas ações com base em recompensas e penalidades (SUTTON; BARTO, 2018; KAELBLING; LITTMAN; MOORE, 1996).

A combinação do AR com RN profundas deu origem ao *Deep Reinforcement Learning*, capaz de lidar com ambientes complexos e de alta dimensionalidade (MNIH et al., 2015; SILVER et al., 2016). Esse avanço representa um marco na trajetória da IA, que evoluiu de

sistemas simbólicos para agentes autônomos capazes de aprender e tomar decisões em cenários dinâmicos e incertos.

2.1.1 Aprendizado de Máquina e suas categorias

O ML é uma área da IA dedicada ao desenvolvimento de algoritmos capazes de melhorar seu desempenho em uma tarefa por meio da experiência (MITCHELL, 1997). Segundo Mitchell (1997), um programa de aprendizado pode ser formalmente definido como aquele que melhora seu desempenho P em uma tarefa T com base em uma medida de experiência E, quando seu desempenho em T, avaliado por P, aumenta em função de E.

De acordo com Alpaydin (2020), o ML pode ser dividido em três categorias principais: aprendizado supervisionado, não supervisionado e por reforço. No aprendizado supervisionado, o modelo é treinado com pares de entrada e saída conhecidos, buscando generalizar padrões para prever a saída de novos exemplos. No aprendizado não supervisionado, o sistema identifica estruturas ou agrupamentos ocultos em dados sem rótulos predefinidos. Já no aprendizado por reforço, o agente aprende a tomar decisões sequenciais interagindo com o ambiente e recebendo recompensas que indicam a qualidade de suas ações (ALPAYDIN, 2020; SUTTON; BARTO, 2018).

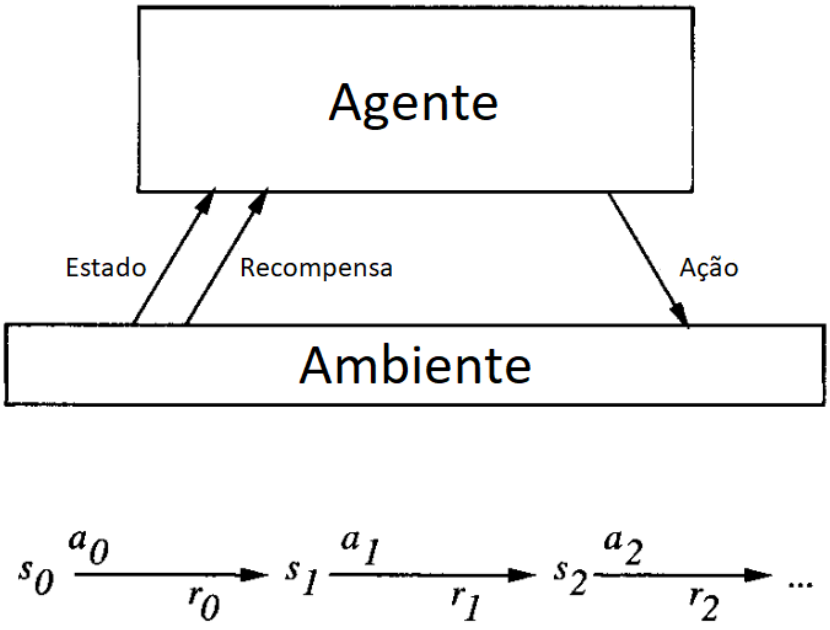
Goodfellow et al. (2016) destacam que os avanços em RN profundas ampliaram as capacidades de generalização dos modelos de ML, permitindo representações hierárquicas complexas de dados e consolidando o *Deep Learning* (DL) como uma base fundamental para o aprendizado moderno. O AR, nesse contexto, ocupa uma posição singular ao integrar esses princípios de representação profunda com a tomada de decisão adaptativa, tornando-se uma das abordagens mais promissoras da IA contemporânea (GOODFELLOW et al., 2016).

2.1.2 Fundamentos do Aprendizado por Reforço

O AR é uma categoria da IA e do ML dedicada ao estudo de agentes que aprendem a tomar decisões por meio da interação com o ambiente. Diferentemente do aprendizado supervisionado, no qual o modelo recebe exemplos rotulados, e do aprendizado não supervisionado, que busca estruturas ocultas nos dados, o AR se baseia em um processo de tentativa e erro mediado por recompensas. O agente observa o estado atual do ambiente, escolhe uma ação, recebe uma recompensa e observa o novo estado resultante, formando um ciclo contínuo de aprendizado (SUTTON; BARTO, 2018; MITCHELL, 1997).

Esse processo é ilustrado na Figura 1, retirada de Mitchell (1997). Nela, o agente atua sobre o ambiente, recebe um sinal de reforço (recompensa) e, com base nisso, ajusta sua política — o conjunto de regras que define qual ação tomar em cada situação.

Figura 1 – Ciclo de aprendizado por reforço.



Fonte: Mitchell (1997).

O sistema de AR é composto por cinco elementos principais:

- **Agente:** a entidade que aprende e toma decisões com base nas observações do ambiente;
- **Ambiente:** tudo o que interage com o agente e responde às suas ações, alterando o estado e fornecendo recompensas;
- **Política (π):** a estratégia que define o comportamento do agente, mapeando estados para probabilidades de ações;
- **Recompensa (R):** o sinal numérico que informa ao agente o valor imediato de sua ação, servindo como guia para o aprendizado;
- **Função de valor (V ou Q):** uma estimativa do retorno esperado, indicando quão bom é um estado ou uma ação sob determinada política.

Esses componentes interagem continuamente durante o aprendizado, permitindo que o agente ajuste sua política com base na experiência acumulada.

Formalmente, o problema de aprendizado é descrito por um Processo de Decisão de Markov (em inglês, *Markov Decision Process*, MDP), composto por um conjunto de estados S, um conjunto de ações A, uma função de transição $P(s'|s,a)$ e uma função de recompensa

$R(s, a)$. O objetivo do agente é maximizar o retorno total esperado, definido de acordo com a equação 2.1:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

onde γ é o fator de desconto que controla a importância relativa das recompensas futuras (SUTTON; BARTO, 2018). Essa formulação permite que o agente aprenda estratégias de longo prazo, mesmo quando o *feedback* direto ocorre apenas após várias ações — um desafio conhecido como atribuição temporal de crédito.

2.1.2.1 Exploração e aproveitamento

Um dos desafios fundamentais do AR é o equilíbrio entre exploração e aproveitamento (em inglês, *exploration–exploitation trade-off*). O agente deve decidir entre explorar novas ações, que podem levar a recompensas melhores, ou explorar o conhecimento já adquirido, repetindo ações que se mostraram eficazes (aproveitamento). Estratégias como o método ϵ-greedy ou a amostragem de Thompson são frequentemente empregadas para equilibrar essas duas abordagens, garantindo tanto a descoberta de novas soluções quanto a estabilidade do aprendizado (SUTTON; BARTO, 2018).

2.1.2.2 Ambientes episódicos e continuados

Os ambientes de aprendizado por reforço podem ser classificados como episódicos ou continuados. Nos ambientes episódicos, o aprendizado ocorre em episódios independentes, que começam em um estado inicial e terminam em um estado terminal; o desempenho do agente é avaliado ao final de cada episódio. Já nos ambientes continuados, a interação entre agente e ambiente não possui um ponto final definido, sendo o aprendizado contínuo ao longo do tempo (SUTTON; BARTO, 2018).

O jogo de *Mahjong*, utilizado neste trabalho, é um exemplo de ambiente episódico, pois cada partida possui um início e um fim bem definidos. O agente deve aprender a otimizar suas ações dentro dos limites de um episódio, reiniciando o aprendizado prático a cada nova partida, o que facilita a avaliação de desempenho e a análise de convergência da política.

2.1.2.3 Aprendizado *Model-Free* e *Model-Based*

Os métodos de AR também se diferenciam quanto à presença de um modelo explícito do ambiente. Nos métodos *model-free*, o agente aprende diretamente a partir das interações, sem tentar prever as transições de estado ou recompensas futuras. Exemplos desses métodos incluem o *Q-Learning* e o SARSA. Já os métodos *model-based* tentam aprender uma representação aproximada das dinâmicas do ambiente (P e R) e utilizam essa informação para planejar ações de forma mais eficiente (SUTTON; BARTO, 2018). Embora os métodos baseados em modelo

tendam a exigir maior capacidade computacional, eles podem aprender com menos amostras, tornando-se úteis quando as interações com o ambiente são custosas.

2.1.2.4 Métodos de aprendizado: Monte Carlo e Diferença Temporal

Os algoritmos de AR podem ser classificados de acordo com a forma como estimam o retorno esperado. Os métodos de Monte Carlo (MC) aprendem a partir de episódios completos, atualizando as estimativas apenas após o término do episódio, o que garante precisão, mas torna o aprendizado mais lento. Em contraste, os métodos de Diferença Temporal (TD) atualizam as estimativas de valor a cada passo de tempo, utilizando previsões parciais das recompensas futuras. O TD combina as vantagens do aprendizado incremental com a eficiência do uso de experiências parciais, sendo a base de algoritmos amplamente utilizados, como o *Q-Learning* (SUTTON; BARTO, 2018).

2.1.2.5 Categorias de algoritmos de Aprendizado por Reforço

De forma geral, os algoritmos de AR podem ser divididos em três grandes categorias:

- ***Value-Based:*** aprendem uma função de valor (como $V(s)$ ou $Q(s, a)$) que representa a utilidade de estados ou ações. A política é derivada implicitamente dessa função, escolhendo as ações que maximizam o valor estimado;
- ***Policy-Based:*** aprendem diretamente uma política parametrizada, sem estimar explicitamente uma função de valor. Esses métodos são adequados para espaços de ação contínuos e problemas de alta dimensionalidade;
- ***Actor–Critic:*** combinam as duas abordagens anteriores, com um componente ator responsável pela política e um componente crítico que avalia as ações tomadas, promovendo aprendizado mais estável e eficiente.

Cada uma dessas categorias será discutida em maior profundidade nas seções seguintes, junto à análise comparativa dos métodos utilizados neste trabalho.

2.2 *Advantage Actor–Critic*

O método *Advantage Actor–Critic* (A2C) é uma variação dos algoritmos de gradiente de política que combina duas redes neurais complementares: o ator (*actor*), responsável por selecionar ações, e o crítico (*critic*), encarregado de avaliar a qualidade dessas ações. Essa arquitetura busca reduzir a variância do gradiente de política e melhorar a estabilidade do aprendizado (LAPAN, 2018).

2.2.1 Arquitetura e princípio de funcionamento

Em vez de otimizar diretamente a política com base nas recompensas obtidas, o A2C introduz um estimador de valor $V(s)$, que serve como uma base para calcular o quão boa foi uma ação em relação ao esperado para aquele estado. O gradiente de política é ajustado de modo proporcional à vantagem da ação, definida pela equação 2.2:

$$A(s,a) = Q(s,a) - V(s) \tag{2.2}$$

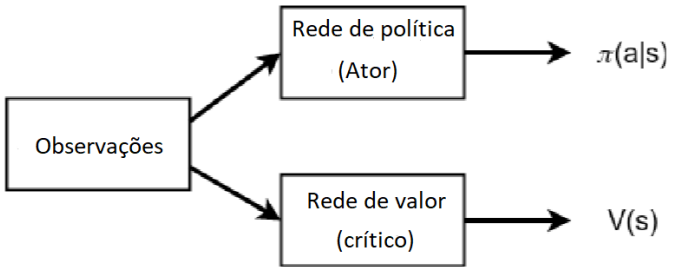
em que $Q(s,a)$ representa o valor esperado da ação e $V(s)$ o valor esperado do estado. Assim, o aprendizado busca aumentar a probabilidade de ações com vantagem positiva e reduzir a de ações com vantagem negativa. O termo $A(s,a)$ reduz a variância das atualizações e acelera a convergência do treinamento, mantendo o algoritmo sem viés (LAPAN, 2018).

A arquitetura do A2C, ilustrada na Figura 2, utiliza duas redes neurais:

- A **rede do ator (rede de política)** gera uma distribuição de probabilidade $\pi_\theta(a|s)$ sobre as ações possíveis, representando a política do agente.
- A **rede do crítico (rede de valor)** estima o valor do estado $V_\theta(s)$, aproximando o retorno esperado a partir daquele ponto.

Na prática, essas redes costumam compartilhar uma mesma base convolucional, responsável por extrair características de alto nível das observações do ambiente, divergindo apenas nas camadas finais (as chamadas cabeças da rede), que produzem as saídas de política e valor, respectivamente (LAPAN, 2018).

Figura 2 – Arquitetura do A2C



Fonte: Lapan (2018).

2.2.2 Procedimento de treinamento

O treinamento do A2C é conduzido em múltiplas etapas intercaladas entre coleta de experiências e atualização de parâmetros. O procedimento geral pode ser descrito como:

1. Inicializar os parâmetros da rede θ aleatoriamente.
2. Interagir com o ambiente por N passos, registrando os estados s_t, ações a_t e recompensas r_t.
3. Calcular o retorno acumulado R_t com base nas recompensas futuras e na estimativa de valor do último estado, segundo a equação 2.3:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n V(s_{t+n}) \tag{2.3}$$

4. Calcular as vantagens, segundo a equação 2.4

$$A_t = R_t - V(s_t) \tag{2.4}$$

5. Atualizar a rede do ator com o gradiente de política ponderado pela vantagem, segundo a equação 2.5:

$$\nabla_\theta J(\theta) = E_t[\nabla_\theta \log \pi_\theta(a_t|s_t) A_t] \tag{2.5}$$

6. Atualizar a rede do crítico minimizando o erro quadrático médio entre o retorno observado e o valor estimado, conforme a equação 2.6:

$$L_v = (R_t - V_\theta(s_t))^2 \tag{2.6}$$

7. Adicionar um termo de entropia $\mathcal{L}_H = -\beta \sum_i \pi_\theta(a_i|s) \log \pi_\theta(a_i|s)$ à função de perda total, estimulando a exploração.

A perda final do A2C combina esses três termos:

$$L_{total} = L_\pi + c_v L_v - \beta \mathcal{L}_H \tag{2.7}$$

onde L_π é a perda de política, c_v é o coeficiente de ponderação do crítico e β controla a intensidade do termo de entropia. Essa formulação garante que o agente aprenda a escolher boas ações, estimar corretamente o valor dos estados e manter a diversidade de comportamento durante o aprendizado (LAPAN, 2018).

2.3 *Proximal Policy Optimization* e *MaskedPPO*

2.3.1 *Proximal Policy Optimization*

O algoritmo PPO, proposto por Schulman et al. (2017), é uma das abordagens mais populares e eficazes para aprendizado por reforço profundo. Ele pertence à classe dos métodos ator-crítico, combinando os benefícios de aprendizado direto de políticas com a estabilidade de estimativas baseadas em valor. Seu objetivo é otimizar a política de forma gradual, evitando

atualizações abruptas que possam degradar o desempenho do agente (SCHULMAN et al., 2017).

Em algoritmos de gradiente de política tradicionais as atualizações podem ser instáveis quando a nova política diverge significativamente da antiga. O PPO propõe uma solução a esse problema por meio da maximização de uma função de objetivo limitada, denominada *clipped surrogate objective*, que restringe o tamanho das atualizações de política. Essa função é dada segundo a equação 2.8:

$$L^{CLIP}(\theta) = E_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2.8}$$

onde $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ é a razão entre a probabilidade da ação sob a nova política e a antiga, $\hat{A}_t$ representa a vantagem estimada (obtida via *Generalized Advantage Estimation* (SCHULMAN et al., 2017)), e ϵ define o limite de variação permitido (tipicamente $0.1 \leq \epsilon \leq 0.3$).

A função *clip* impede que o gradiente da política sofra grandes oscilações ao limitar a razão $r_t(\theta)$ a um intervalo seguro, garantindo que as atualizações mantenham a política próxima à anterior. Isso melhora substancialmente a estabilidade e a eficiência amostral do treinamento.

O PPO utiliza ainda várias técnicas complementares para estabilizar o aprendizado, como:

- Normalização das vantagens e observações;
- *Clipping* da função de valor;
- Uso do otimizador Adam com *learning rate annealing*;
- Atualização em mini-lotes e *gradient clipping global*.

Essas estratégias, discutidas em Engstrom et al. (2020), têm impacto significativo no desempenho prático do algoritmo, garantindo convergência estável mesmo em ambientes de alta dimensionalidade. Em termos gerais, o PPO equilibra desempenho, estabilidade e simplicidade de implementação, tornando-se uma das escolhas padrão em aplicações de aprendizado por reforço profundo (LAPAN, 2018).

2.3.2 *Masked Proximal Policy Optimization*

O *Masked Proximal Policy Optimization* MaskedPPO é uma extensão do PPO voltada para ambientes com grandes espaços de ação e restrições de validade, um cenário comum em jogos complexos e tarefas estratégicas. Nesses ambientes, muitas ações possíveis em um determinado estado são inválidas segundo as regras do sistema, o que torna o processo de

aprendizado ineficiente quando o agente tenta executar ações proibidas (HUANG; ONTAñóN, 2022).

Para mitigar esse problema, o MaskedPPO utiliza a técnica de *invalid action masking*, que consiste em remover da distribuição de probabilidade todas as ações inválidas antes da amostragem. Essa remoção é realizada substituindo os *logits* das ações inválidas por um valor negativo extremo (por exemplo, $M = -10^8$), de modo que sua probabilidade após a operação *softmax* seja praticamente zero, conforme a equação 2.10:

$$\pi'_\theta(\cdot|s) = \text{softmax}(\text{mask}(l(s))) \tag{2.9}$$

$$\text{mask}(l(s))_i = \begin{cases} l_i, & \text{se } a_i \text{ é válida em } s, \\ M, & \text{caso contrário.} \end{cases} \tag{2.10}$$

Esse processo garante que apenas ações válidas sejam amostradas, sem introduzir viés na atualização dos gradientes. De acordo com Huang e Ontañón (2022), a técnica de mascaramento ainda preserva a validade teórica do gradiente de política, uma vez que o mascaramento pode ser interpretado como uma função diferenciável dependente do estado aplicada aos *logits*. Assim, o gradiente obtido permanece consistente com o teorema do gradiente de política de Sutton et al. (1999).

Os experimentos conduzidos por Huang e Ontañón (2022) demonstram que o uso de *invalid action masking* melhora substancialmente a eficiência de exploração e a taxa de convergência em ambientes com grandes espaços de ação, como o jogo de estratégia µRTS. O método mostrou escalabilidade superior em comparação com abordagens que penalizam ações inválidas com recompensas negativas, especialmente quando o número de ações possíveis cresce exponencialmente.

Outra vantagem do MaskedPPO é sua robustez: mesmo quando o mascaramento é removido após o treinamento, os agentes continuam apresentando comportamento coerente, evidenciando que o aprendizado adquirido é generalizável e não dependente da presença do filtro de máscara.

Em síntese, o MaskedPPO mantém as propriedades de estabilidade do PPO e acrescenta um mecanismo eficaz para lidar com restrições de ação em ambientes complexos. Essa combinação torna o algoritmo particularmente adequado para jogos de tabuleiro e outros domínios nos quais o espaço de ação é extenso, mas parcialmente inválido em cada estado — como o *Mahjong*, estudado neste trabalho.

2.4 *Q-Learning*

O *Q-Learning*, introduzido por Watkins e Dayan (1992), é um dos algoritmos mais influentes do aprendizado por reforço, sendo a base conceitual para diversas variantes modernas,

incluindo o *Deep Q-Network* (DQN). Ele é um método *model-free*, ou seja, não requer o conhecimento prévio das probabilidades de transição do ambiente, e busca estimar diretamente a função de ação-valor (*Q-function*) associada à política ótima (CLIFTON; LABER, 2020).

A função $Q(s, a)$ representa o retorno esperado ao executar uma ação a em um estado s e seguir, a partir daí, a política ótima. O algoritmo procura encontrar uma aproximação para a função $Q^*(s, a)$ que satisfaça a equação de Bellman de otimalidade 2.11:

$$Q^*(s, a) = E\left[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \tag{2.11}$$

onde γ é o fator de desconto que pondera recompensas futuras (BELLMAN, 2010; SUTTON; BARTO, 2018). Essa equação define um ponto fixo da transformação de Bellman, e o *Q-Learning* busca aproximá-lo iterativamente via atualizações sucessivas.

2.4.1 Atualização incremental

Em sua forma original, o *Q-Learning* realiza atualizações incrementais com base na experiência do agente. Dado um par de transição $(S_t, A_t, R_{t+1}, S_{t+1})$, a atualização da função Q é feita segundo a equação 2.12:

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_{a'} Q_t(S_{t+1}, a') - Q_t(S_t, A_t)\right] \tag{2.12}$$

onde α é a taxa de aprendizado e o termo entre colchetes representa o erro de previsão temporal (*temporal-difference error*). Com repetições suficientes e uma política de exploração apropriada, a sequência $\{Q_t\}$ converge para Q^*, garantindo a obtenção da política ótima $\pi^*(s) = \arg\max_a Q^*(s, a)$ (CLIFTON; LABER, 2020).

2.4.2 *Q-Learning* e programação dinâmica

O *Q-Learning* pode ser interpretado como uma forma de programação dinâmica estocástica que dispensa o conhecimento explícito do modelo de transição do ambiente. Enquanto os métodos de programação dinâmica clássicos, como o Método de Iteração de Valor, requerem a modelagem da função de transição $P(s'|s, a)$, o *Q-Learning* utiliza amostras reais de interação para estimar o mesmo ponto fixo da equação de Bellman (SUTTON; BARTO, 2018). Assim, o algoritmo combina as vantagens da inferência empírica com a teoria de controle ótima, tornando-se especialmente útil em contextos de simulação ou em ambientes complexos, como jogos e robótica (SILVER et al., 2016).

2.4.3 Abordagens finito e infinito-horizonte

Conforme discutido por Clifton e Laber (2020), o *Q-Learning* pode ser formulado tanto em horizontes finitos quanto infinitos. No caso de horizonte finito, a estimação segue um

processo de indução regressiva (ou *backward induction*), resolvendo sequencialmente as funções Q_t a partir da etapa final, segundo a equação 2.13:

$$\hat{Q}_t = \arg\min_{Q_t} E\left[\left(Q_t(H_t, A_t) - (R_t + \max_{a_{t+1}} \hat{Q}_{t+1}(H_{t+1}, a_{t+1}))\right)^2\right] \tag{2.13}$$

onde H_t representa o histórico até o tempo t (CLIFTON; LABER, 2020). Já no horizonte infinito, a função Q é estacionária e o problema é resolvido iterativamente até convergência, geralmente utilizando aproximação de função (*function approximation*) para lidar com espaços de estado grandes ou contínuos.

2.4.4 *Q-Learning* e inferência estatística

Uma das principais contribuições do trabalho de Clifton e Laber (2020) é a formalização do *Q-Learning* dentro do arcabouço de inferência causal e aprendizado sequencial. O algoritmo é interpretado como um estimador recursivo da função Q^* em um MDP, com implicações diretas na estimação de estratégias ótimas em problemas de decisão dinâmica, como regimes de tratamento em medicina personalizada e estratégias de controle adaptativo (CLIFTON; LABER, 2020). Essa perspectiva permite estender o *Q-Learning* para cenários com incerteza causal, viés de confusão e observações dependentes no tempo.

2.4.5 Exploração, estabilidade e convergência

O *Q-Learning* é, por natureza, um algoritmo *off-policy*, isto é, ele pode aprender a política ótima independentemente da política utilizada para coletar os dados. Entretanto, o equilíbrio entre exploração e aproveitamento é essencial: sem uma política de exploração adequada, o agente pode convergir para políticas subótimas. Estratégias como ϵ-greedy, *Boltzmann exploration* ou *Thompson sampling* são frequentemente utilizadas para garantir diversidade na experiência (SUTTON; BARTO, 2018).

A estabilidade do aprendizado também depende da escolha apropriada da taxa α e do uso de aproximações de função compatíveis. Com aproximações não lineares, como redes neurais profundas, o *Q-Learning* evolui para o chamado DQN, que combina aprendizado de representação com otimização incremental da função Q, alcançando resultados notáveis em tarefas de alta complexidade, como jogos de *Atari* e *Go* (SILVER et al., 2016).

2.5 Redes Neurais Convolucionais

As CNNs são uma classe especializada de redes neurais artificiais projetadas para processar dados com estrutura em grade, como imagens (2D), sinais de áudio (1D) ou vídeos (3D). Introduzidas por LeCun, Bengio e Hinton (2015), as CNNs se tornaram um dos pilares do aprendizado profundo moderno, especialmente em tarefas de visão computacional, reconhecimento de fala e análise de séries temporais (GOODFELLOW et al., 2016).

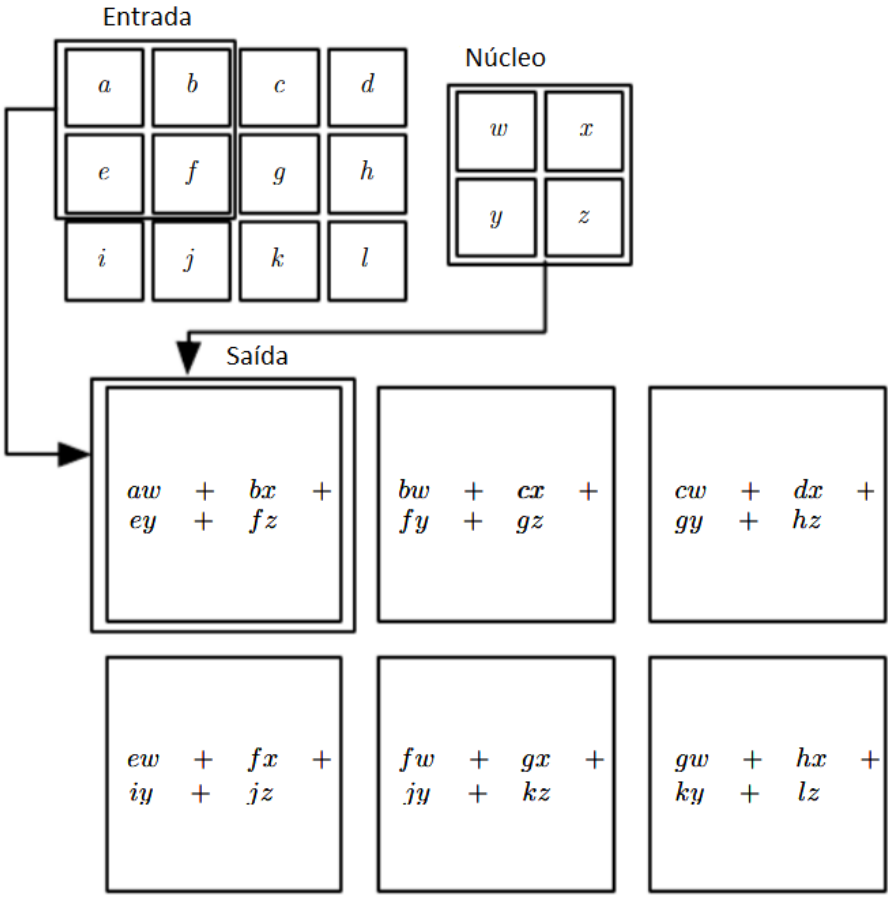
2.5.1 Operação de Convolução

O elemento central de uma CNN é a operação de convolução, que substitui a multiplicação matricial tradicional das redes densas. Em sua forma discreta, a convolução entre uma entrada I e um núcleo K é definida pela equação 2.14:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.14}$$

onde $S(i,j)$ representa o *feature map* resultante. Na prática, muitas bibliotecas de aprendizado de máquina implementam a operação de *cross-correlation* (sem inversão do núcleo), mantendo o mesmo princípio funcional (GOODFELLOW et al., 2016). A figura 3 representa um exemplo de convolução 2D sem inversão do núcleo. Nesse caso, o resultado é restrito apenas às posições onde o núcleo se encontra completamente dentro da imagem, operação conhecida como *valid convolution* em alguns contextos. As setas indicam como o elemento no canto superior esquerdo do tensor de saída é formado pela aplicação do núcleo sobre a região correspondente do tensor de entrada.

Figura 3 – Exemplo de convolução 2D sem inversão do núcleo.



Fonte: Goodfellow et al. (2016).

A convolução pode ser compreendida como um mecanismo de detecção de padrões locais (bordas, texturas, cantos) que são relevantes independentemente de sua posição no

espaço. Cada núcleo atua como um detector de uma característica específica, e diferentes filtros aprendem a responder a diferentes padrões da entrada.

2.5.2 Princípios Fundamentais das CNNs

Goodfellow et al. (2016) destacam três princípios fundamentais que tornam as CNNs mais eficientes do que as redes densas tradicionais:

1. **Conectividade esparsa (*sparse interactions*):** Em uma rede densa, cada neurônio da camada de saída está conectado a todos os neurônios da camada anterior. Nas CNNs, cada neurônio se conecta apenas a uma pequena vizinhança local da entrada, chamada de *campo receptivo* (*receptive field*). Essa restrição reduz drasticamente o número de parâmetros e operações computacionais, sem perda significativa de capacidade representacional.

2. **Compartilhamento de parâmetros (*parameter sharing*):** O mesmo conjunto de pesos (núcleo) é utilizado em todas as posições espaciais da entrada. Isso significa que um mesmo filtro é responsável por detectar um padrão específico em qualquer região da imagem, reforçando a eficiência estatística e reduzindo o número de parâmetros de forma exponencial.

3. **Equivariância à translação (*translation equivariance*):** A convolução preserva o padrão de deslocamento espacial, se a entrada é transladada, o mapa de ativação resultante também se desloca na mesma direção. Essa propriedade é fundamental para tarefas em que a posição do objeto não altera sua identidade, como reconhecimento de imagens.

Esses princípios tornam as CNNs altamente eficientes tanto em termos de memória quanto de generalização, e permitem que camadas mais profundas aprendam representações hierárquicas, de bordas simples a estruturas complexas de alto nível.

2.5.3 Camadas e Estrutura de uma CNN

Uma camada convolucional típica é composta por três estágios principais (GOODFELLOW et al., 2016):

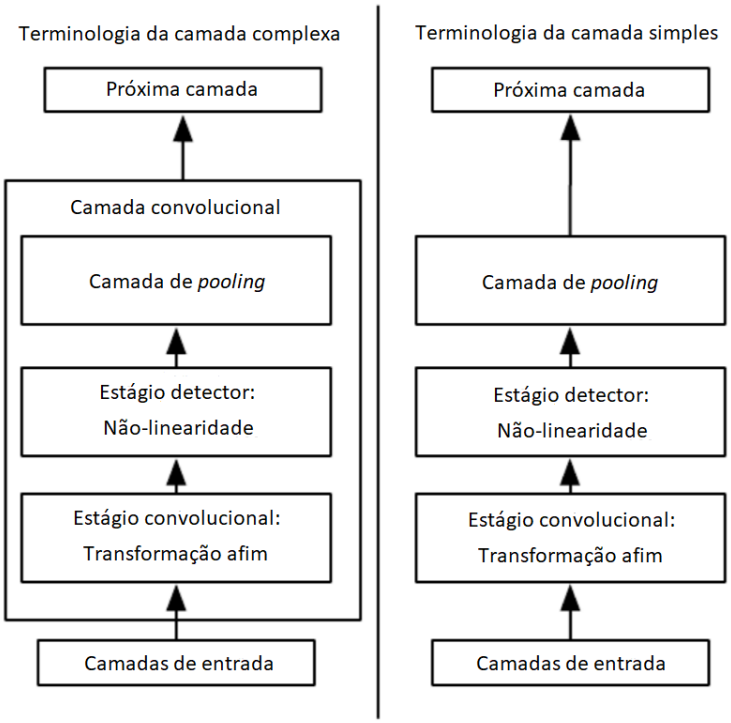
1. **Convolução linear:** aplicação de múltiplos filtros convolucionais à entrada, gerando mapas de ativação lineares;

2. **Função de ativação:** aplicação de uma não linearidade (por exemplo, *Rectified Linear Unit*, ReLU) aos resultados da convolução, introduzindo capacidade de modelar funções não lineares;

3. ***Pooling:*** operação de agregação local que reduz a dimensionalidade e fornece invariância a pequenas translações.

A figura 4 ilustra esses estágios e sua interação.

Figura 4 – Estrutura típica de uma camada convolucional: convolução, não linearidade e *pooling*.



Fonte: Goodfellow et al. (2016).

2.5.4 *Pooling* e Invariância Local

O *pooling* resume as ativações em regiões locais, normalmente utilizando operações de máximo (*max pooling*) ou média (*average pooling*). Por exemplo, o *max pooling* com janelas 2×2 e *stride 2* reduz a resolução espacial pela metade em cada dimensão. Essa operação introduz invariância a pequenas variações de posição e reduz o custo computacional, ao mesmo tempo em que mantém as informações mais salientes (GOODFELLOW et al., 2016).

O *pooling* pode ser interpretado como a introdução de um *prior* forte de invariância local: o modelo aprende a focar na presença de características, e não em sua localização exata. Essa propriedade é particularmente útil em tarefas de classificação de imagens, onde a posição exata do objeto é irrelevante.

2.5.5 Convolução e *Pooling* como um Prior Forte

Goodfellow et al. (2016) descrevem as CNNs como redes densas com um prior infinitamente forte sobre seus pesos. Esse prior impõe duas restrições fundamentais:

1. os pesos são compartilhados espacialmente (tradução equivariância);
2. a conectividade é local (campos receptivos limitados).

Essas restrições reduzem a variância e melhoram a generalização em dados estruturados. Entretanto, quando o problema requer alta precisão espacial (por exemplo, segmentação de imagens), o *pooling* excessivo pode levar ao *underfitting*, pois remove detalhes relevantes da localização (GOODFELLOW et al., 2016).

2.5.6 Variações da Convolução

Na prática, as CNNs modernas utilizam diversas variações da convolução básica:

- ***Stride:*** define o espaçamento entre as posições da janela de convolução, controlando o tamanho da saída e a resolução espacial.
- ***Zero-padding:*** adiciona bordas de zeros ao redor da entrada para controlar o tamanho da saída e preservar as bordas da imagem.
- ***Convolução dilatada:*** insere espaçamentos entre elementos do núcleo, permitindo ampliar o campo receptivo sem aumentar o número de parâmetros.
- **Convoluções localmente conectadas:** semelhantes às convoluções, mas sem compartilhamento de pesos — úteis quando diferentes regiões da entrada possuem significados distintos.
- **Convoluções agrupadas e separáveis:** reduzem a complexidade ao aplicar filtros a subconjuntos de canais, comuns em arquiteturas como *MobileNet* e *Xception*.

2.5.7 Gradientes e Retropropagação

Durante o treinamento, os gradientes das convoluções são calculados via retropropagação. (GOODFELLOW et al., 2016) apresentam três operações centrais:

1. Convolução direta (em inglês, *forward pass*);
2. Convolução transposta (para retropropagar gradientes);
3. Gradiente em relação aos pesos (cálculo de ∇_K).

Essas operações garantem que as CNNs possam ser treinadas de forma eficiente via *backpropagation*, mesmo com múltiplas camadas e parâmetros compartilhados.

2.5.8 Aplicações e Arquiteturas

As CNNs se tornaram o paradigma dominante em visão computacional, com arquiteturas como *LeNet-5*, *AlexNet*, VGG, *ResNet* e *Inception*. Esses modelos exploram o empilhamento de múltiplas camadas convolucionais e *pooling*, formando hierarquias de abstração progressivamente mais complexas, de bordas e texturas nas primeiras camadas até conceitos semânticos nas últimas (GOODFELLOW et al., 2016).

Além da visão, CNNs também são amplamente utilizadas em:

- Processamento de áudio e fala;
- Modelagem de séries temporais;
- Análise de dados volumétricos (como tomografia e vídeo);
- Representações espaciais em jogos e agentes de aprendizado por reforço.

2.5.9 Síntese

Em síntese, as CNNs representam uma das aplicações mais bem-sucedidas dos princípios de aprendizado profundo, unindo eficiência estatística, robustez e interpretabilidade estrutural. Seu impacto extrapola a área de visão computacional, servindo como base para arquiteturas híbridas em aprendizado por reforço, geração de imagens e modelagem sequencial.

2.6 Principais Regras do *Riichi Mahjong*

Esta seção resume os elementos essenciais do *Riichi Mahjong* segundo as regras da edição de 2016 da *European Mahjong Association* (EMA) (European Mahjong Association, 2016).

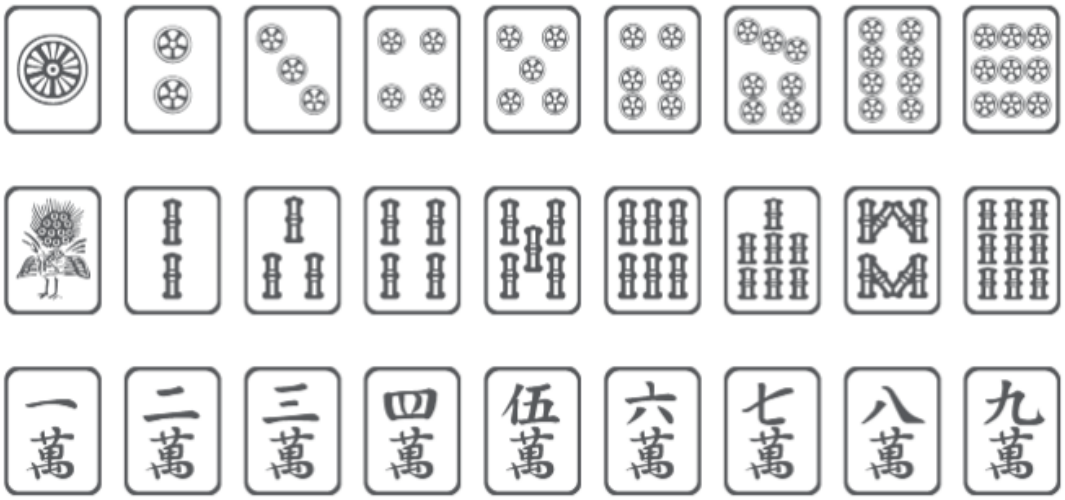
2.6.1 Conjunto de Peças e Equipamentos

O conjunto básico possui **136 peças**, quatro cópias de cada um dos **34 tipos**:

- **Três naipes (1–9):** *Círculos*, *Bambus* e *Caracteres*. Os **1** e **9** são *terminais*. (Figura 5)
- ***Honors*:** *Quatro Ventos* (Leste, Sul, Oeste, Norte) e *Três Dragões* (Branco, Verde, Vermelho). (Figura 6)

- **Peças adicionais (não usadas no EMA):** flores/estações e *red fives* não são utilizadas nas regras EMA 2016.

Figura 5 – Naipe de Círculos, Bambus e Caracteres (1–9).



Fonte: European Mahjong Association (2016).

Figura 6 – *Honors*: Ventos (E, S, W, N) e Dragões (Branco, Verde, Vermelho).

Fonte: European Mahjong Association (2016).

Equipamentos auxiliares

Marcador de *vento prevalente*, *sticks* de pontuação (100, 1 000, 5 000, 10 000; às vezes 500), presente na figura 7 e *dados*. Pontuação inicial típica: 25 000 pontos por jogador (European Mahjong Association, 2016).

Figura 7 – *Sticks* de pontuação e marcador de vento prevalente.

Fonte: European Mahjong Association (2016).

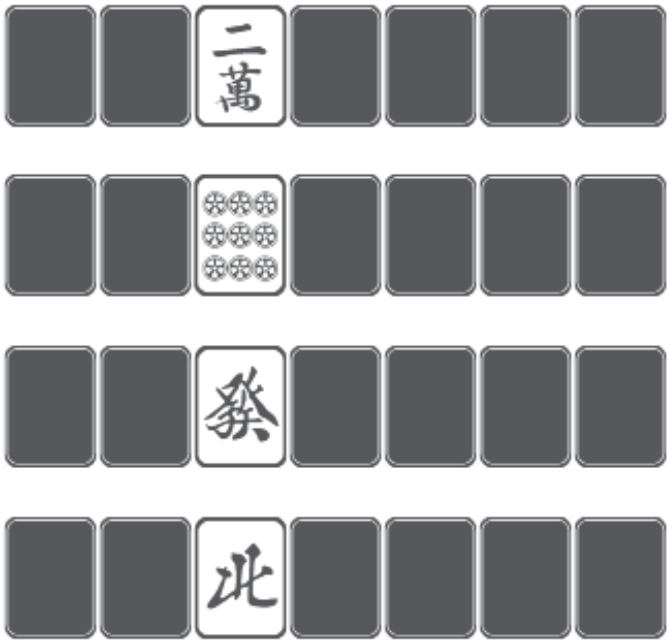
2.6.2 Preparação e *Dead Wall*

Os quatro jogadores constroem *paredes* de 17 pilhas duplas formando um quadrado; quebra-se a parede por rolagem de dados. À direita da quebra, separam-se 7 pilhas para a ***dead wall*** (14 peças) usada para reposições de *kong* e indicadores de *dora* (European Mahjong Association, 2016).

2.6.3 Dora e Indicadores

Vira-se a peça superior da *dead wall* (contando três pilhas a partir da quebra) para indicar o **dora indicator**; a peça imediatamente seguinte no ciclo do mesmo naipe/vento/dragão é o **dora**. Para ventos: E→S→W→N→E; para dragões: Vermelho→Branco→Verde→Vermelho; para 1–9, o sucessor cíclico dentro do naipe. *Kan dora* e *ura dora* (após *riichi*) também podem se aplicar (European Mahjong Association, 2016). Exemplos de peças viradas conforme a figura 8.

Figura 8 – Exemplo de *dora indicator* e mapeamento para o *dora* efetivo.



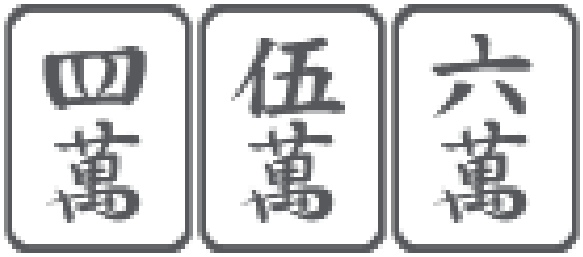
Fonte: European Mahjong Association (2016).

2.6.4 Estrutura da Mão e *Yaku*

Uma mão vencedora (*agari*) é composta por **quatro conjuntos** e **um par** (salvo exceções como *Seven Pairs* e *Thirteen Orphans*) e deve conter ao menos um *yaku* (European Mahjong Association, 2016). Conjuntos:

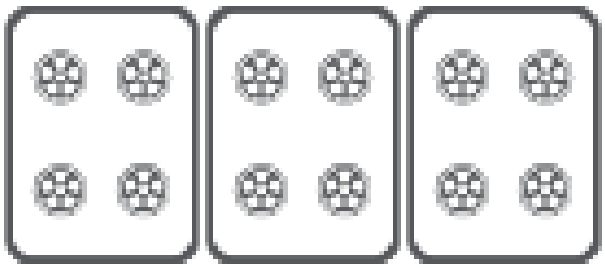
- **Chow**: sequência de três do mesmo naipe (não existe com honors) (Figura 9);
- **Pung**: trinca idêntica (Figura 10);
- **Kong**: quadra idêntica (com regras de reposição) (Figura 11).

Figura 9 – *Chow*: sequência (ex.: 3–4–5 do mesmo naipe).



Fonte: European Mahjong Association (2016).

Figura 10 – *Pung*: trinca idêntica.



Fonte: European Mahjong Association (2016).

Figura 11 – *Kong*: quadra idêntica; pode ser *melded*, *concealed* ou extensão de *pung*.

Fonte: European Mahjong Association (2016).

2.6.5 Fluxo do Jogo e Declarações

A rodada segue em sentido anti-horário; um turno consiste em *comprar* e *descartar*. Discardes podem ser reivindicados para formar *pung*/*kong* por qualquer jogador antes do próximo compra, e *chow* apenas pelo jogador à esquerda do descartante; vitórias por descarte têm precedência. *Riichi* pode ser declarado com mão completamente fechada em *tenpai* mediante aposta de 1 000 pontos; *tsumo* é vitória por auto-compra, *ron* por descarte alheio. Regras de *furiten* impedem *ron* em certas condições (European Mahjong Association, 2016).

2.6.6 Término da Mão e Continuidade

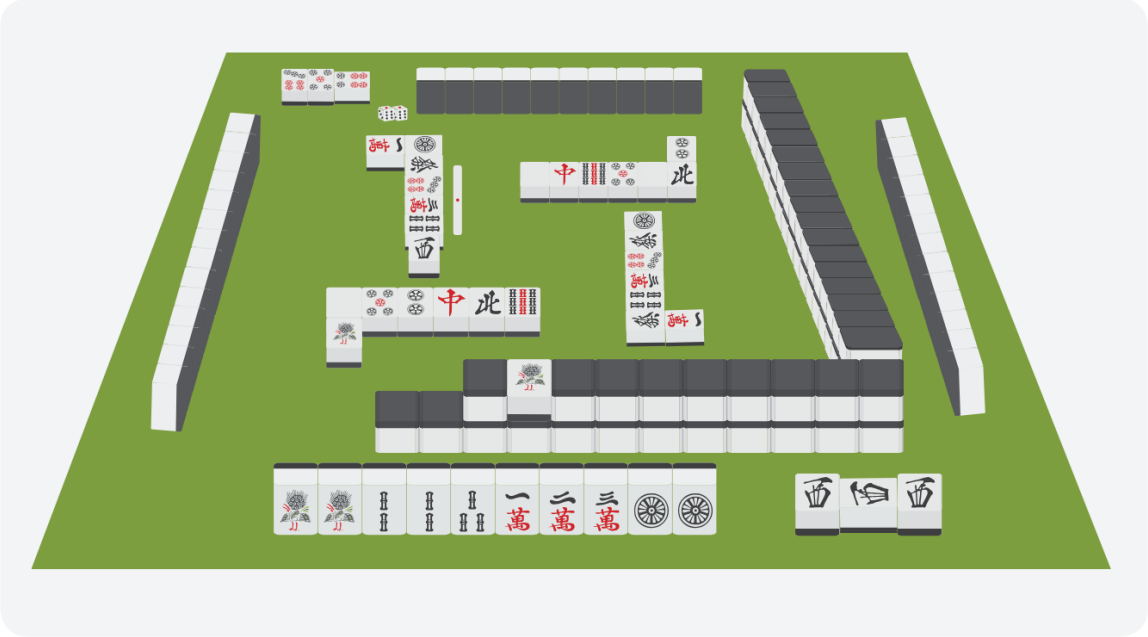
Uma mão termina por vitória, *draw* exaustivo (sem vitória após a última compra) ou penalidades específicas. Em *draw* exaustivo, aplica-se penalidade *noten*/*tenpai* (3 000 pontos distribuídos). O dealer (*East*) permanece se vencer ou estiver em *tenpai*; caso contrário, rotação de ventos. O jogo padrão cobre *East round* e *South round* (European Mahjong Association, 2016).

2.6.7 Pontuação Essencial

A pontuação combina **han** (dobras) de *yaku*/*dora* com **fu** (minipontos), arredondando *fu* para a dezena. Para mãos de ≥ 5 *han*, aplica-se limite: *Mangan*, *Haneman*, *Baiman*, *Sanbaiman*, *Yakuman*. Pagamentos diferem para *ron*/*tsumo* e se o vencedor é *East* ou não; contadores (*honba*) adicionam 300/100 por unidade (European Mahjong Association, 2016).

A Figura 12 representa um exemplo de mesa de *mahjong*.

Figura 12 – Exemplo de mesa: paredes, *dead wall*, descarte organizado e marcador de vento.



Fonte: European Mahjong Association (2016).

3 Revisão de Literatura

O presente capítulo tem como objetivo situar o desenvolvimento deste trabalho dentro do contexto das pesquisas existentes, apresentando uma visão geral sobre os principais avanços relacionados ao aprendizado por reforço, suas aplicações em jogos e o uso de RN profundas nesse domínio. A revisão busca identificar as contribuições mais relevantes da literatura, as lacunas ainda existentes e as motivações que fundamentam a proposta deste estudo.

Inicialmente, são abordados os trabalhos clássicos e contemporâneos que consolidaram uso de IA como uma abordagem eficaz para problemas relacionados a jogos. Em seguida, discute-se a aplicação dessa técnica em jogos de diferentes naturezas — desde ambientes totalmente observáveis, como o xadrez e o *Go*, até aqueles de informação incompleta, como o pôquer e o *Riichi Mahjong*.

Serão então apresentados os principais estudos que aplicam aprendizado por reforço ao jogo de *Mahjong*, com destaque para o sistema **Suphx**, considerado o estado da arte na área. Desenvolvido pela *Microsoft Research Asia*, o *Suphx* alcançou desempenho de nível profissional na plataforma *Tenhou*, sendo reconhecido como o primeiro agente de *Riichi Mahjong* capaz de competir em igualdade com jogadores humanos de alto nível. Além desse trabalho, serão analisadas outras abordagens relevantes que contribuíram para o avanço da área, explorando técnicas de aprendizado profundo, modelagem de decisão em ambientes parcialmente observáveis e estratégias de representação de estados.

Por fim, o capítulo apresenta uma síntese crítica da literatura, destacando as oportunidades de investigação identificadas e justificando a relevância do presente estudo, que propõe a aplicação e comparação de diferentes algoritmos de aprendizado por reforço profundo no jogo *Riichi Mahjong*, um ambiente de informação parcial, múltiplos agentes e grande espaço de ações possíveis.

3.1 Uso de Inteligência Artificial em Jogos

O desenvolvimento de técnicas de IA tem encontrado nos jogos um campo fértil para experimentação e avanço científico. Por representarem ambientes controlados, dinâmicos e desafiadores, os jogos fornecem um cenário ideal para testar algoritmos de tomada de decisão, aprendizado e planejamento em contextos que refletem problemas do mundo real, como incerteza, competição e necessidade de adaptação (HU et al., 2024).

3.1.1 Jogos como ambientes de pesquisa em IA

Segundo Hu et al. (2024), os jogos funcionam como simuladores realistas e economicamente viáveis para o desenvolvimento e avaliação de algoritmos de IA, permitindo a exploração de habilidades como raciocínio estratégico, coordenação multiagente, aprendizado em ambientes parciais e planejamento de longo prazo. O estudo de Hu et al. (2024) apresenta uma revisão abrangente dos principais ambientes de jogos aplicados à pesquisa em IA, incluindo desde jogos de tabuleiro e cartas até plataformas de simulação em três dimensões. Os autores destacam que a evolução das técnicas de IA em jogos acompanha três eixos principais: (i) o aumento da complexidade dos ambientes; (ii) a transição de jogos determinísticos e totalmente observáveis para jogos de informação incompleta; e (iii) a incorporação de aprendizado profundo (*deep learning*) como meio de generalizar políticas de decisão Hu et al. (2024).

Entre os marcos históricos do uso de IA em jogos estão os sucessos alcançados em ambientes como xadrez, *Go* e *poker*; exemplos clássicos de problemas de decisão sequencial. Esses jogos permitiram o desenvolvimento de técnicas que combinam busca heurística, aprendizado supervisionado e aprendizado por reforço, culminando em resultados que superaram o desempenho humano.

3.1.2 *AlphaGo* e o aprendizado por reforço profundo

O projeto **AlphaGo**, desenvolvido pela *DeepMind*, representou um ponto de inflexão na aplicação de aprendizado profundo a jogos de tabuleiro complexos. De acordo com Silver et al. (2016), o sistema combinou CNNs com o método de busca *Monte Carlo Tree Search* (MCTS), criando um agente capaz de aprender tanto a partir de partidas humanas quanto de autojogos. Duas redes principais foram utilizadas: uma *policy network*, responsável por prever a probabilidade de cada movimento, e uma *value network*, encarregada de estimar a probabilidade de vitória a partir de um estado específico (SILVER et al., 2016).

A fase inicial do treinamento foi supervisionada, utilizando milhões de jogadas humanas para ensinar a política básica do jogo. Em seguida, o agente evoluiu por meio de aprendizado por reforço, refinando sua política ao jogar contra versões anteriores de si mesmo e maximizando a probabilidade de vitória. Essa abordagem de autoaperfeiçoamento resultou em um desempenho que superou todos os programas existentes, culminando na vitória histórica sobre o campeão europeu e, posteriormente, sobre o campeão mundial de *Go*. O êxito do *AlphaGo* demonstrou a capacidade das CNNs de extrair padrões espaciais complexos de ambientes simbólicos e a eficácia da integração entre redes neurais e algoritmos de busca probabilística (SILVER et al., 2016).

3.1.3 *MuZero* e a integração de modelos aprendidos

A sequência natural da evolução do *AlphaGo* levou à criação do *MuZero*, também pela *DeepMind*, que unificou o aprendizado de políticas, valores e recompensas em um modelo único, sem depender do conhecimento explícito das regras do jogo. Diferentemente de abordagens anteriores, o *MuZero* aprendeu internamente uma representação abstrata das dinâmicas do ambiente, permitindo o planejamento mesmo em situações onde as regras ou transições de estado não são conhecidas (SCHRITTWIESER et al., 2020).

O algoritmo combina uma função de representação, uma função de dinâmica e uma função de previsão, treinadas de forma conjunta para estimar política, valor e recompensa esperada. A cada passo, o modelo prevê ações futuras hipotéticas e avalia suas consequências por meio de busca em árvore, semelhante ao MCTS, mas alimentada por um modelo interno aprendido. Essa estratégia tornou o *MuZero* capaz de alcançar desempenho super-humano não apenas em jogos de tabuleiro como *Go*, xadrez e *shogi*, mas também em 57 jogos da plataforma *Atari*, caracterizados por observações visuais complexas e dinâmicas contínuas (SCHRITTWIESER et al., 2020).

A principal contribuição do *MuZero* reside na eliminação da necessidade de modelar explicitamente as regras do ambiente. O agente "inventa" internamente uma representação que preserve apenas as informações relevantes para o planejamento e a predição de recompensas, resultando em um sistema mais geral e aplicável a uma ampla gama de domínios.

3.1.4 Tendências e implicações

Os avanços proporcionados por *AlphaGo* e *MuZero* exemplificam a transição de uma IA especializada para uma IA mais generalista e autônoma, capaz de aprender representações internas do mundo a partir da experiência. Segundo Hu et al. (2024), essa trajetória reflete uma convergência entre áreas antes separadas; aprendizado profundo, busca heurística e raciocínio simbólico; o que torna os jogos ambientes ideais para o estudo de generalização e raciocínio em múltiplos níveis. Além disso, o uso de jogos de múltiplos agentes e informação parcial (como, por exemplo, *Mahjong* e *Hanabi*) amplia o escopo de pesquisa, introduzindo desafios relacionados à inferência, comunicação e adaptação estratégica, aspectos diretamente relevantes para o desenvolvimento de sistemas de IA mais próximos do comportamento humano.

Dessa forma, os jogos continuam a representar não apenas um campo de validação para novas arquiteturas de aprendizado, mas também um microcosmo de problemas reais de decisão sob incerteza. Os resultados obtidos por *AlphaGo* e *MuZero* consolidam o aprendizado por reforço profundo como paradigma central na busca por agentes inteligentes, e estabelecem as bases conceituais sobre as quais este trabalho se apoia.

3.2 Aplicações de Aprendizado por Reforço em *Mahjong*

O jogo *Riichi Mahjong* tem se mostrado um ambiente de grande interesse para a pesquisa em IA, devido à sua natureza de informação incompleta, múltiplos agentes e grande espaço de estados e ações. Diferentemente de jogos totalmente observáveis, como o xadrez ou o *Go*, o *Mahjong* impõe desafios adicionais relacionados à inferência de informações ocultas, à modelagem de comportamento de oponentes e à necessidade de tomada de decisão sob incerteza. Esses fatores tornam o domínio um excelente campo de testes para algoritmos avançados de aprendizado por reforço profundo.

3.2.1 *Suphx*: o estado da arte em *Mahjong*

O trabalho mais proeminente na área é o **Suphx**, desenvolvido pela *Microsoft Research Asia*, amplamente reconhecido como o estado da arte em agentes de *Mahjong*. O sistema combina aprendizado por reforço profundo com técnicas de aprendizado supervisionado e arquitetura distribuída de treinamento. O agente é composto por múltiplas redes neurais convolucionais responsáveis por prever ações e estimar recompensas, utilizando um modelo de aprendizado baseado em autojogo (*self-play*) e refinamento contínuo (LI et al., 2020).

O *Suphx* introduziu dois elementos-chave que impulsionaram seu desempenho: (i) o uso de um modelo de predição global de recompensas baseado em unidades recorrentes (em inglês, *Gated Recurrent Units, GRU*) para capturar dependências temporais ao longo de rodadas; e (ii) o conceito de *oracle guiding*, no qual informações ocultas — como as mãos dos oponentes — são utilizadas apenas durante o treinamento, servindo como uma forma de observação privilegiada para acelerar o aprendizado. Durante a execução, o agente toma decisões apenas com base em informações observáveis, garantindo validade e generalização. Graças a essas inovações, o *Suphx* alcançou um desempenho de **8,74 dan** na plataforma *Tenhou*, superando 99,99% dos jogadores humanos registrados, e se consolidou como o primeiro sistema de IA a atingir nível profissional em *Riichi Mahjong* (LI et al., 2020).

3.2.2 Aprendizado orientado por oráculo

O conceito de *oracle guiding* foi posteriormente formalizado por Han et al. (2022) no método ***Variational Latent Oracle Guiding*** (VLOG). Esse *framework* teórico define um processo de aprendizado baseado em inferência bayesiana variacional, no qual o modelo aprende simultaneamente uma estimativa *a priori* (baseada em observações executoras) e uma estimativa *a posteriori* (baseada em observações de oráculo). Durante o treinamento, ambas são comparadas por meio de uma divergência de Kullback–Leibler (KL), de forma que o modelo executor aprenda representações mais próximas das distribuições ideais obtidas com o oráculo (HAN et al., 2022).

No contexto do *Mahjong*, o VLOG permite que o agente aprenda a inferir melhor as mãos ocultas dos oponentes, analogamente ao processo humano de assistir *replays* de partidas, reduzindo a incerteza e acelerando a convergência do aprendizado. Os autores demonstraram empiricamente ganhos de desempenho significativos em relação a abordagens padrão de AR profundo, consolidando o *Mahjong* como ambiente de referência para pesquisa em aprendizado com observações parciais.

3.2.3 Otimização do aprendizado e aceleração do treinamento

O trabalho de Li et al. (2022) abordou o problema do tempo de convergência em ambientes de alta dimensionalidade, propondo técnicas de **aceleração do aprendizado por reforço em *Mahjong***. Entre as estratégias apresentadas estão a reutilização de experiências por meio de memória de *replay* distribuída e o uso de aprendizado assíncrono em múltiplas instâncias de autojogo. Essas abordagens reduziram significativamente o tempo de treinamento de agentes em comparação a métodos tradicionais, demonstrando a viabilidade do treinamento em larga escala de sistemas de *Mahjong* baseados em aprendizado profundo (LI et al., 2022).

3.2.4 Abordagens alternativas e variações do jogo

Pesquisas mais recentes ampliaram o escopo das aplicações para outras variantes do jogo. Zhao e Holden (2022) desenvolveram o ***Meowjong***, o primeiro agente para *Sanma* — a versão de três jogadores do *Riichi Mahjong*. O sistema emprega cinco redes neurais convolucionais independentes, cada uma treinada para uma ação específica (descartar, *Pon*, *Kan*, *Kita* e *Riichi*), e posteriormente aprimoradas via aprendizado por reforço com gradiente de política MC. Apesar da simplificação estrutural do jogo, o ambiente de *Sanma* manteve desafios de ocultamento de informação e adaptação de estratégia, e o agente alcançou desempenho comparável aos sistemas de quatro jogadores, validando a aplicabilidade de abordagens baseadas em CNNs e aprendizado por reforço a múltiplas variações do *Mahjong* (ZHAO; HOLDEN, 2022).

3.2.5 Síntese dos avanços

Os estudos revisados evidenciam que o *Mahjong* evoluiu de um problema de nicho para um importante campo experimental em aprendizado por reforço profundo. Os trabalhos analisados introduziram contribuições que vão desde representações estruturadas e mecanismos de explicabilidade até técnicas avançadas de orientação variacional e aceleração de aprendizado. O *Suphx* e o VLOG, em particular, estabeleceram paradigmas metodológicos que unem aprendizado supervisionado, AR e inferência bayesiana sob incerteza, constituindo o estado da arte na área e fornecendo base conceitual para o desenvolvimento deste trabalho.

3.3 Síntese Crítica da Literatura

A literatura revisada evidencia um avanço notável na aplicação de técnicas de aprendizado por reforço e aprendizado profundo em jogos, consolidando-os como ambientes experimentais essenciais para o desenvolvimento de sistemas inteligentes. Trabalhos como *AlphaGo* e *MuZero* demonstraram a eficácia de combinar redes neurais com algoritmos de busca e planejamento, alcançando resultados de nível super-humano em jogos de informação completa. No entanto, esses mesmos estudos também ressaltam que a transposição dessas abordagens para ambientes de informação parcial, como o *Riichi Mahjong*, impõe novos desafios relacionados à inferência, ambiguidade e variabilidade estratégica.

As pesquisas recentes aplicadas ao *Mahjong* reforçam a relevância desse jogo como um domínio de teste sofisticado para aprendizado por reforço profundo. Iniciativas como o *Suphx* e o *Variational Oracle Guiding (VLOG)* estabeleceram novas fronteiras metodológicas ao introduzir mecanismos de aprendizado com observações privilegiadas e guias variacionais, capazes de reduzir a incerteza e acelerar a convergência em ambientes parcialmente observáveis. Outros estudos abordaram aspectos complementares, como explicabilidade, otimização do treinamento e variações estruturais do jogo, ampliando a compreensão sobre as dinâmicas de tomada de decisão sob incerteza.

Apesar desses avanços, observa-se que a maioria das abordagens permanece concentrada em sistemas de larga escala e de alto custo computacional, muitas vezes inacessíveis para experimentação acadêmica ou reprodutibilidade aberta. Além disso, há uma lacuna significativa na comparação sistemática entre diferentes classes de algoritmos, baseados em valor, em política e híbridos, aplicados ao contexto do *Mahjong*. Também se nota uma escassez de estudos que investiguem o impacto das representações convolucionais sobre o desempenho de agentes em jogos de informação parcial.

Diante desse panorama, o presente trabalho propõe-se a contribuir com uma análise comparativa detalhada entre três métodos representativos de aprendizado por reforço profundo, *Q-Learning*, *A2C* e *MaskedPPO*, aplicados ao jogo *Riichi Mahjong*. A abordagem emprega redes neurais convolucionais para a representação de estados e explora o uso de técnicas inspiradas no conceito de *oracle guiding*, visando aprimorar a eficiência e a consistência do aprendizado em cenários de observação limitada. Assim, o estudo busca não apenas reproduzir resultados práticos de alto desempenho, mas também oferecer subsídios conceituais para o entendimento dos mecanismos de aprendizado em ambientes complexos e parcialmente observáveis.

4 Metodologia Experimental

Este capítulo descreve a metodologia adotada para o desenvolvimento deste trabalho, detalhando as etapas que orientaram a investigação, o delineamento experimental, o ambiente computacional, as técnicas de aprendizado utilizadas e os critérios de análise empregados. O objetivo é apresentar de forma clara e reprodutível o processo seguido para a comparação entre diferentes políticas de AR aplicadas ao jogo *Riichi Mahjong*.

A pesquisa tem natureza exploratória e descritiva, uma vez que busca investigar e caracterizar o desempenho de agentes inteligentes submetidos a distintas políticas de aprendizado, sem a pretensão inicial de formular um modelo preditivo generalizável. O caráter exploratório se justifica pela escassez de estudos comparativos diretos entre abordagens de AR no domínio do *Mahjong*, enquanto o aspecto descritivo reflete a análise quantitativa do comportamento dos agentes sob diferentes condições de treinamento.

O objeto de estudo deste trabalho consiste em agentes treinados com três políticas representativas de AR Profundo, *Q-Learning*, A2C e *MaskedPPO*, implementadas sobre uma mesma arquitetura de CNN. A comparação entre essas políticas visa identificar diferenças de desempenho, estabilidade e eficiência no processo de aprendizado em um ambiente complexo e de informação parcial.

O desenvolvimento experimental foi realizado no simulador *PyMahjong* [1] Han et al. (2022), descrito em detalhes na Seção 4.1.3, que implementa a dinâmica do *Riichi Mahjong* com quatro jogadores. Esse ambiente permite a interação entre agentes e fornece as observações, recompensas e restrições necessárias para a aplicação das técnicas de AR. O uso do *PyMahjong* garante compatibilidade com abordagens anteriores encontradas na literatura e facilita a reprodutibilidade dos experimentos.

As etapas do método incluem a implementação dos agentes, o treinamento supervisionado e por reforço em ambiente simulado, a coleta e o tratamento dos dados gerados, e a análise dos resultados com base em métricas de desempenho específicas, como taxa de vitória, *payoff* médio e tempo de treinamento. A partir dessas etapas, busca-se avaliar a eficácia de cada política e discutir suas implicações teóricas e práticas no contexto de jogos de informação incompleta.

[1] Disponível em: <https://github.com/Agony5757/mahjong/>. Acesso em: 20 nov. 2025.

4.1 Ambiente de Desenvolvimento

4.1.1 Linguagem e Bibliotecas

O desenvolvimento dos agentes e experimentos foi realizado em **Python 3.9**, devido à sua ampla adoção na área de IA e à disponibilidade de bibliotecas específicas para AR e RN. A linguagem oferece suporte abrangente para integração entre módulos científicos, *frameworks* de aprendizado profundo e ferramentas de visualização, facilitando o desenvolvimento e a análise dos modelos implementados.

Para a implementação dos agentes de AR, utilizou-se o ***Stable-Baselines3* (SB3)** e sua extensão ***SB3-Contrib*** [2] , que fornecem implementações consolidadas e estáveis de diversos algoritmos clássicos e modernos. Entre eles, foram empregados:

- **DQN:** utilizado como base para o agente de *Q-Learning*, que aproxima a função-valor $Q(s, a)$ por meio de uma rede neural e atualiza seus parâmetros via aprendizado por diferença temporal;
- **A2C:** um método *on-policy* que combina redes de política e valor em um mesmo modelo, reduzindo a variância do gradiente de política e melhorando a estabilidade do aprendizado;
- ***MaskedPPO*:** disponibilizado no *SB3-Contrib*, que estende o PPO tradicional com mascaramento de ações inválidas, evitando penalizações desnecessárias em ambientes com restrições complexas de ação, como o *Mahjong*.

O módulo *ActionMasker* foi utilizado para aplicar dinamicamente as máscaras de ação, garantindo que o agente apenas selecionasse movimentos válidos em cada estado do jogo, o que reduziu o espaço de busca e acelerou a convergência. (HUANG; ONTAñóN, 2022)

A biblioteca ***PyTorch*** [3] foi adotada como *backend* de AR, responsável pelo treinamento das CNNs que constituem a base de todos os agentes. O *PyTorch* possibilitou a definição explícita de arquiteturas, funções de ativação e otimizadores, além de permitir a execução dos modelos em *Graphics Processing Unit* (GPU), reduzindo significativamente o tempo de treinamento.

Para a construção e integração do ambiente de simulação, foi utilizada a biblioteca **Gymnasium** [4] (sucessora do *OpenAI Gym*) em conjunto com o ambiente **PyMahjong** (HAN et al., 2022), descrito em detalhes na Seção 4.1.3, que implementa as regras e dinâmicas do

[2] Documentação oficial do *Stable-Baselines3* e *Stable-Baselines3 Contrib*: <https://stable-baselines3.readthedocs.io/en/master/> e <https://stable-baselines3.readthedocs.io/en/master/guide/sb3_contrib.html>. Acesso em: 20 nov. 2025.

[3] Documentação oficial do *PyTorch*: <https://docs.pytorch.org/docs/stable/index.html>. Acesso em: 20 nov. 2025.

[4] Documentação oficial do ambiente *Gymnasium*: <https://gymnasium.farama.org/index.html>. Acesso em: 20 nov. 2025.

jogo *Riichi Mahjong*. Um *wrapper* foi desenvolvido para compatibilizar o *PyMahjong* com a *Application Programming Interface* (API) moderna do *Gymnasium*, corrigindo formatos de observação, normalização de tipos e integração com a função de máscara de ações.

A arquitetura convolucional empregada pelos agentes foi definida no módulo *mahjong_cnn.py*, utilizando camadas Convolucionais Bidimencionais e *ReLU* para extrair padrões espaciais das observações matriciais do jogo. O extrator de características gera vetores de 512 dimensões que alimentam as redes de política e valor dos algoritmos.

Além dessas bibliotecas principais, foram utilizadas ferramentas auxiliares como ***NumPy*** para operações vetoriais e ***Matplotlib*** para visualização dos resultados. Todos os módulos foram estruturados de forma modular, permitindo a reutilização de componentes de treinamento, avaliação e análise. O ***TensorBoard*** foi utilizado para monitorar métricas de aprendizado, como perda, recompensas médias e tempo de atualização dos modelos, possibilitando a análise comparativa e o diagnóstico visual do processo de convergência.

Esse conjunto de bibliotecas e *frameworks* formou um ecossistema robusto e reprodutível para experimentação em AR profundo, garantindo consistência entre os agentes e conformidade com as práticas modernas da área.

4.1.2 Infraestrutura de Execução

Os experimentos foram conduzidos em ambiente de execução local, utilizando uma **GPU NVIDIA GeForce RTX 3060** com **6 GB de VRAM** e **16 GB de memória RAM**. A aceleração por GPU foi fundamental para o treinamento das redes neurais convolucionais, permitindo a execução paralela das operações matriciais e a atualização eficiente dos parâmetros durante o aprendizado.

Para garantir a **replicabilidade dos experimentos**, todas as sementes de geração aleatória foram fixadas em **42**. Essa escolha foi realizada de forma arbitrária, servindo como referência única para assegurar que todos os resultados pudessem ser reproduzidos sob as mesmas condições experimentais, eliminando variações decorrentes de inicializações estocásticas.

O treinamento de cada agente foi realizado ao longo de **500 000 *timesteps*** de interação entre o agente e o ambiente, número definido empiricamente para permitir a convergência das políticas de aprendizado. Durante o processo, foram gerados ***checkpoints* a cada 100 000 episódios**, possibilitando a retomada de sessões de treinamento e a análise progressiva do desempenho.

A execução dos agentes foi paralelizada por meio do módulo *SubprocVecEnv* do **SB3**, configurado com **seis ambientes simultâneos**. Essa abordagem multiprocessada aumentou a eficiência na coleta de experiências, diversificou os estados observados e acelerou a aprendizagem, contribuindo também para a estabilidade da função de valor.

A combinação desses recursos garantiu uma infraestrutura experimental robusta, eficiente e reprodutível, adequada ao estudo comparativo de políticas de Aprendizado por Reforço Profundo em um ambiente complexo como o *Riichi Mahjong*.

4.1.3 Ambiente *PyMahjong*

O ambiente de simulação utilizado neste trabalho é o ***PyMahjong***, uma implementação completa do jogo *Riichi Mahjong* compatível com a interface da biblioteca *Gymnasium*. Ele reproduz com fidelidade as regras do jogo japonês, permitindo a interação entre quatro agentes de forma totalmente observável e controlada. Essa estrutura possibilita a aplicação direta de técnicas de AR, fornecendo observações estruturadas, ações válidas e recompensas numéricas associadas ao desempenho dos jogadores.

4.1.3.1 Observações

Cada estado do ambiente é representado por um tensor tridimensional contendo 93 canais (para o executor) e dimensões $(93, 34)$, correspondendo às possíveis características de cada uma das 34 espécies de peças do jogo. As observações incluem informações sobre:

- **Mão atual do jogador**: número de cópias de cada peça, presença de peças *Dora* e descartes anteriores;
- **Chamadas (melds)** dos quatro jogadores: como *Chi*, *Pon* e *Kan*, distinguindo se vieram de descartes de outros jogadores;
- **Descarte de cada jogador**: histórico de peças descartadas, indicações de *Riichi* e uso de peças *Red Dora*;
- **Informações públicas**: indicadores de *Dora*, ventos do jogador e da rodada;
- **Ações possíveis**: codificação binária das ações válidas em cada estado, usada pelo agente para mascarar ações ilegais.

4.1.3.2 Espaço de Ações

O espaço de ações é discreto e contém 47 possíveis ações, numeradas de 0 a 46. As ações incluem:

- **Descarte de peças** (índices 0–33);
- **Chamadas** como *Chi*, *Pon* e diferentes tipos de *Kan* (34–40);
- **Ações especiais**, como *Riichi*, *Ron* e *Tsumo* (41–43);
- **Ações de controle**, como *reiniciar com Kyūshūkyūhai* ou recusar chamadas (44–46).

O uso do algoritmo *MaskedPPO* foi essencial para lidar com esse espaço de ações parcialmente válido, pois apenas um subconjunto das ações é permitido em cada estado.

4.1.3.3 Recompensa e Sistema de Pontuação

A função de recompensa do ambiente é diretamente baseada na **pontuação oficial do Riichi Mahjong**, em que ganhos resultam em recompensas positivas e perdas em valores negativos. A pontuação é determinada a partir dos valores de **Han** (multiplicadores de valor base obtidos por combinações e dora) e **Fu** (pontos base calculados a partir da estrutura da mão) (HAN et al., 2022). O valor base é calculado segundo a equação 4.1:

$$\text{Base Points} = Fu \times 2^{2+Han} \tag{4.1}$$

Entretanto, o sistema do *Riichi Mahjong* utiliza categorias de pontuação padronizadas para simplificar os cálculos e evitar resultados desproporcionais. Assim, as mãos são classificadas de acordo com seu valor de *Han* e *Fu*, conforme a seguinte hierarquia:

- ***Mangan***: alcançada com 5 *Han*, ou com 4 *Han* e *Fu* $\geq$ 40, fixando o valor base em 8000 pontos;
- ***Haneman***: 6–7 *Han*, valor base de 12000 pontos;
- ***Baiman***: 8–10 *Han*, valor base de 16000 pontos;
- ***Sanbaiman***: 11–12 *Han*, base fixo de 24000 pontos;
- ***Yakuman***: 13 ou mais *Han*, valor base de 32000 pontos.

As pontuações efetivas variam conforme a forma de vitória:

- ***Tsumo*** (vitória ao comprar): o jogador vencedor recebe pontos de todos os oponentes;
- ***Ron*** (vitória sobre descarte): o jogador vencedor recebe a pontuação total apenas do jogador que descartou a peça vencedora.

Esses valores são então ajustados conforme a posição do jogador e o bônus de dealer (*oya*), que multiplica o valor total por 1,5. No ***PyMahjong***, o valor final da pontuação é retornado como a **recompensa do episódio**, de modo que ganhos de pontos resultam em recompensas positivas e perdas em negativas. Essa formulação fornece um sinal de aprendizado diretamente proporcional ao sucesso estratégico do agente, incentivando políticas que maximizem ganhos e minimizem perdas em longo prazo.

Assim, o ambiente *PyMahjong* provê uma simulação completa e realista do *Riichi Mahjong*, equilibrando fidelidade às regras do jogo e compatibilidade com o paradigma de AR profundo.

4.2 Seleção dos Algoritmos de Aprendizado e da Arquitetura da Rede Neural

4.2.1 Justificativa da Seleção dos Algoritmos de Aprendizado

A escolha dos algoritmos de AR empregados neste trabalho, ***Q-Learning***, **A2C** e **PPO**, foi orientada pelo objetivo de comparar diferentes paradigmas de aprendizado dentro do mesmo ambiente e arquitetura de rede, avaliando suas capacidades de adaptação, estabilidade e eficiência em um cenário de informação parcial como o *Riichi Mahjong*.

O ***Q-Learning*** foi selecionado por representar a classe de algoritmos *value-based*, nos quais o aprendizado é guiado pela estimativa direta da função-valor de ação $Q(s, a)$. Sua escolha se justifica por ser um método conceitualmente simples, bem estabelecido na literatura e amplamente utilizado como ponto de partida para o desenvolvimento de técnicas mais complexas de aprendizado por reforço (SUTTON; BARTO, 2018). Além disso, sua natureza *off-policy* permite o reaproveitamento de experiências passadas, o que o torna uma boa referência de comparação quanto à eficiência amostral em ambientes com elevada dimensionalidade de estado.

O **A2C** foi incluído como representante da classe *actor–critic*, que combina as vantagens dos métodos baseados em valor e em política. O A2C mantém duas redes separadas: uma para estimar a função de valor (crítico) e outra para representar a política de decisão (ator). Essa arquitetura reduz a variância do gradiente de política e melhora a estabilidade do treinamento em comparação aos métodos puramente baseados em política (MNIH et al., 2016). Além disso, por ser um algoritmo *on-policy*, o A2C permite observar como políticas otimizadas com base em experiências recentes se comportam em um ambiente competitivo e estocástico.

Por fim, o **PPO** foi selecionado por representar uma evolução direta dos métodos de política pura, incorporando mecanismos de estabilidade por meio de uma função de perda com restrições de atualização (*clipped objective*). Essa característica evita grandes variações entre políticas consecutivas, reduzindo a oscilação do aprendizado e tornando o treinamento mais confiável (SCHULMAN et al., 2017). A versão *MaskedPPO*, utilizada neste trabalho, estende o algoritmo original para lidar com ações inválidas, aplicando máscaras sobre o espaço de ação, um requisito essencial para o jogo *Riichi Mahjong*, no qual nem todas as jogadas são permitidas em todos os estados.

A seleção desses três algoritmos foi, portanto, motivada por sua representatividade nas principais categorias do AR:

- ***Q-Learning***: abordagem baseada em valor (*value-based*);
- **A2C**: abordagem híbrida (*actor–critic*);
- **MaskedPPO**: abordagem baseada em política (*policy-based*).

Essa diversidade metodológica permite uma análise comparativa abrangente sobre o impacto da formulação do aprendizado na performance dos agentes, considerando aspectos como estabilidade, eficiência e capacidade de generalização em um ambiente de informação parcial e alta complexidade combinatória.

4.2.2 Justificativa da Seleção da Arquitetura da Rede Neural

A escolha por utilizar **CNNs** como base arquitetural dos agentes desenvolvidos neste trabalho fundamenta-se em sua comprovada capacidade de extrair padrões espaciais complexos de dados estruturados, característica essencial para lidar com a representação matricial dos estados do jogo de *Riichi Mahjong*. As CNNs permitem que o modelo aprenda automaticamente relações locais e globais entre as peças e os descartes, reduzindo a necessidade de engenharia manual de atributos e aumentando a generalização do aprendizado.

O uso de CNNs em jogos de informação perfeita e imperfeita já se mostrou altamente eficaz em diferentes contextos. No caso do ***AlphaGo***, desenvolvido pela *DeepMind*, as CNNs foram responsáveis por capturar padrões espaciais no tabuleiro de *Go*, permitindo que o agente alcançasse desempenho superior ao de campeões humanos (SILVER et al., 2016). Essa mesma estratégia inspirou pesquisas subsequentes em jogos de natureza mais complexa, como o *Mahjong*.

Entre essas pesquisas, destaca-se o sistema ***Suphx*** (LI et al., 2020), considerado o estado da arte em IA para *Mahjong*. O *Suphx* emprega redes convolucionais profundas como base para seus modelos de política e decisão, codificando o estado do jogo em múltiplos canais bidimensionais de 34 colunas, cada uma representando um tipo de peça, e várias linhas representando camadas de informação distintas (mão do jogador, descartes, dora, etc.). Essa abordagem demonstrou ser altamente eficaz para capturar as interdependências espaciais e temporais do jogo, permitindo que o agente atingisse desempenho superior ao de 99,99% dos jogadores humanos ranqueados na plataforma *Tenhou* [5] .

De forma semelhante, o trabalho de Gao et al. (2019), propôs uma arquitetura convolucional para prever as jogadas humanas em *logs* de partidas, alcançando precisão superior a abordagens tradicionais baseadas em árvores de decisão ou redes densas. O estudo mostrou que as CNNs conseguem identificar padrões sutis nas distribuições das peças e nos históricos de descartes, elementos que influenciam diretamente a qualidade das decisões tomadas pelo agente.

Esses resultados reforçam que as CNNs são especialmente adequadas para ambientes com estados complexos, representáveis em múltiplas dimensões e camadas de contexto. No presente trabalho, a observação do ambiente foi estruturada em um tensor tridimensional $(93, 34)$, onde os canais codificam informações distintas (como mão do jogador, descartes e

[5] Site oficial do servidor japonês de *Mahjong Tenhou*: <https://tenhou.net/>. Acesso em: 20 nov. 2025.

dora). Essa configuração é análoga à representação empregada em trabalhos anteriores de sucesso, permitindo que a rede extraia padrões de dependência entre peças e regiões do estado de jogo, de maneira comparável a um reconhecimento de padrões visuais.

Portanto, a adoção de CNNs neste projeto é justificada tanto pela literatura consolidada quanto pela adequação prática à estrutura do problema. Elas oferecem um meio eficiente de transformar a representação matricial do *Mahjong* em um espaço de características discriminativas, viabilizando o aprendizado eficaz das políticas de decisão empregadas pelos algoritmos *Q-Learning*, A2C e PPO.

4.2.3 Arquitetura da Rede Neural Convolucional

A rede convolucional utilizada neste trabalho foi inspirada na arquitetura proposta por Gao et al. (2019), que representa o estado do jogo de *Riichi Mahjong* como um conjunto de planos bidimensionais (*planes*) contendo as informações visíveis na mesa. Assim como no trabalho original, a arquitetura aqui empregada visa extrair padrões espaciais das distribuições de peças e jogadas, permitindo ao agente identificar relações complexas entre elementos do estado.

4.2.3.1 Arquitetura Original

No modelo proposto por Gao et al. (2019), a entrada da rede é composta por um tensor tridimensional de tamanho $(86, 34, 4)$, correspondendo a 86 planos de informação de dimensões 34×4, que codificam diferentes aspectos do jogo, como mão do jogador, descartes, indicadores de *dora*, ventos, *riichi*, entre outros. A rede é composta por três blocos convolucionais, cada um contendo uma camada convolucional, normalização em lote e *dropout*, seguidos por uma camada totalmente conectada de 300 neurônios e uma camada de saída adaptada ao tipo de ação (por exemplo, 34 classes para o descarte de peças). Os filtros utilizados possuem tamanho 5×2, sem *padding*, com 100 mapas de características em cada camada convolucional. A rede é relativamente profunda e pesada, adequada a treinamentos em GPUs de alto desempenho.

4.2.3.2 Arquitetura Utilizada neste Trabalho

A rede desenvolvida neste projeto, denominada *MahjongCNN*, foi concebida como uma versão mais enxuta e eficiente do modelo original, visando reduzir o custo computacional e o tempo de treinamento, uma vez que os experimentos foram realizados em um hardware de médio porte, descrito em detalhes na Seção 4.1.2.

A arquitetura empregada é composta por três camadas convolucionais sucessivas com filtros 3×3 e *padding* igual a 1, seguidas por funções de ativação *ReLU*. Após as camadas convolucionais, os mapas de características são achatados e passados a uma camada densa final com 512 neurônios, também seguida por uma ativação *ReLU*. A rede não utiliza *Batch*

Normalization nem *Dropout*, o que reduz a complexidade do modelo e acelera o treinamento, mantendo boa estabilidade devido ao tamanho moderado dos dados e à natureza do problema.

Formalmente, o modelo é definido da conforme a figura 13:

onde n_{flat} é calculado dinamicamente com base nas dimensões da entrada $(93, 34)$ fornecida pelo ambiente *PyMahjong*. A camada linear final produz uma representação vetorial compacta de 512 dimensões, utilizada pelos algoritmos de AR (*Q-Learning*, *A2C* e *PPO*) para a estimativa de políticas e funções de valor.

Semelhanças e Diferenças

A principal semelhança com a arquitetura proposta por Gao et al. (2019) está na **quantidade de camadas convolucionais**, sendo três em ambos os modelos. Essa profundidade foi mantida por representar um equilíbrio entre capacidade de extração de características e estabilidade de treinamento. Em ambas as arquiteturas, o estado do jogo de *Mahjong* é representado como uma matriz bidimensional, tratada de maneira análoga a uma imagem. As convoluções, portanto, são responsáveis por capturar dependências locais e globais entre as peças, identificando padrões estratégicos e estruturais que influenciam as decisões do agente.

As diferenças entre as arquiteturas concentram-se principalmente na profundidade interna das camadas, nos tamanhos de filtros e na presença de mecanismos de regularização:

- A rede deste trabalho omite camadas de *Batch Normalization* e *Dropout*, enquanto o modelo original as emprega para mitigar o sobreajuste em conjuntos de dados maiores;
- O tamanho dos filtros foi reduzido de 5×2 para 3×3, otimizando o custo computacional e adaptando-se melhor às dimensões $(93, 34)$ da entrada fornecida pelo ambiente *PyMahjong*;
- O número total de parâmetros foi significativamente reduzido, resultando em menor tempo de treinamento e consumo de memória GPU, sem comprometer a estabilidade do aprendizado;
- A camada totalmente conectada foi ampliada para 512 neurônios (em vez de 300), compensando a simplificação das camadas convolucionais e preservando a expressividade da rede;
- O modelo deste trabalho foi implementado com foco em eficiência, visando possibilitar o treinamento de múltiplos agentes (PPO, A2C e *Q-Learning*) em hardware de médio porte (GPU RTX 3060).

Essas modificações resultaram em uma rede mais leve e eficiente, adequada às limitações de hardware disponíveis, sem comprometer sua capacidade de extração hierárquica de

padrões. A `MahjongCNN` preserva o princípio fundamental da arquitetura original, três camadas convolucionais sucessivas, mantendo a coerência conceitual com trabalhos anteriores que utilizam redes convolucionais em jogos, como o *AlphaGo* (SILVER et al., 2016), o *Suphx* (LI et al., 2020) e o próprio modelo proposto por Gao et al. (2019), adaptando-o a um contexto de pesquisa mais acessível e reprodutível.

4.3 Construção dos Agentes de Aprendizado por Reforço

4.3.1 Agente *MaskedPPO*

O agente *MaskedPPO* é baseado na implementação disponível no pacote *sb3_contrib* do *Stable-Baselines3*. Ele utiliza o algoritmo PPO, adaptado para ambientes com ações inválidas por meio do uso de máscaras booleanas que impedem a escolha de ações não permitidas em cada estado do jogo (SUTTON; BARTO, 2018).

4.3.1.1 Tratamento de ações inválidas:

O *MaskedPPO* aplica a função de máscara (*ActionMasker*) ao espaço de ação antes de cada decisão, assegurando que apenas ações válidas tenham probabilidade não nula. Essa abordagem elimina penalidades artificiais e acelera a convergência do aprendizado, conforme proposto por Huang e Ontañón (2022).

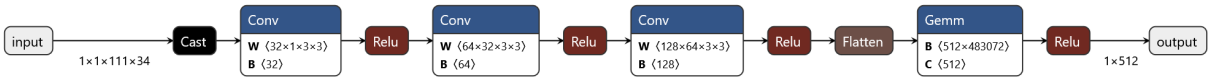
4.3.1.2 Parâmetros principais:

Os parâmetros utilizados no treinamento do agente estão apresentados na Tabela 1. O modelo foi treinado em GPU (*CUDA*) com seis ambientes paralelos e semente fixa para reprodutibilidade.

Tabela 1 – Parâmetros principais do agente *MaskedPPO*.

Parâmetro	**Valor**
Algoritmo	*MaskedPPO* (*sb3_contrib*)
`n_envs`	6
`total_timesteps`	500 000
`seed`	42
Extrator de características	*MahjongCNN (features_dim = 512)*
Arquitetura da política	*pi = [256, 128]*
Arquitetura da função de valor	*vf = [256, 128]*
`device`	GPU (CUDA)

Figura 13 – Arquitetura da Rede Neural utilizada



Fonte: Elaborado pelo autor

4.3.2 Agente A2C

O agente A2C foi implementado com base na biblioteca SB3. O mesmo extrator de características *MahjongCNN* utilizado nos demais agentes foi empregado para processar o estado do jogo, convertendo observações espaciais em representações compactas que alimentam a rede (SUTTON; BARTO, 2018).

4.3.2.1 Tratamento de ações inválidas:

Para garantir que o agente nunca execute ações ilegais, foi desenvolvido o *ValidActionWrapper*, uma classe derivada de *gym.Wrapper* que intercepta as chamadas a *env.step()*. Caso o agente proponha uma ação inválida, o *wrapper* aplica até duas tentativas de repetição (*max_invalid_retries* = *6*) sem avançar o ambiente, penalizando a decisão com uma recompensa negativa (*invalid_penalty* = *-10*). Se o erro persistir, a ação é substituída por uma válida escolhida aleatoriamente (*replacement* = *random*), garantindo continuidade do episódio. Esse mecanismo assegura estabilidade no aprendizado e coleta de experiências, mantendo rastreamento das falhas por meio dos contadores *invalid_counter* e *total_counter*.

4.3.2.2 Parâmetros do Agente A2C

Os principais parâmetros utilizados estão apresentados na Tabela 2.

Tabela 2 – Parâmetros principais do agente A2C.

Parâmetro	**Valor**
Algoritmo	A2C (SB3)
n_envs	6
total_timesteps	500 000
seed	42
Extrator de características	*MahjongCNN (features_dim = 512)*
Arquitetura da política	*pi = [256, 128]*
Arquitetura da função de valor	*vf = [256, 128]*
n_steps	6
gamma	0.95
learning_rate	3e-4
device	GPU (CUDA)

Fonte: Elaborado pelo autor.

4.3.3 Agente DQN

O agente DQN foi implementado com base na biblioteca SB3, seguindo a formulação clássica de aprendizado por diferença temporal proposta por Watkins e Dayan (1992). Assim como nos demais agentes, a rede *MahjongCNN* foi utilizada como extratora de características do estado do jogo.

4.3.3.1 Tratamento de ações inválidas:

O agente DQN utiliza o mesmo mecanismo de tratamento de ações inválidas descrito na Seção 4.3.2.1, baseado no *ValidActionWrapper*, que evita que nenhuma ação ilegal seja executada durante o treinamento.

4.3.3.2 Parâmetros do Agente DQN

Os principais parâmetros utilizados no treinamento do agente estão apresentados na Tabela 3.

Tabela 3 – Parâmetros principais do agente *DQN*.

Parâmetro	**Valor**
Algoritmo	DQN (SB3)
`n_envs`	6
`total_timesteps`	500 000
`seed`	42
Extrator de características	*MahjongCNN (features_dim = 512)*
Arquitetura da rede	`net_arch = [256]`
learning_rate	1e-4
buffer_size	100 000
learning_starts	10 000
batch_size	128
gamma	0.95
train_freq	16
gradient_steps	1
target_update_interval	20 000
exploration_fraction	0.2
exploration_initial_eps	1.0
exploration_final_eps	0.05
n_steps	1
device	GPU (CUDA)

Fonte: Elaborado pelo autor.

4.4 Síntese do Método

O método adotado neste trabalho combinou princípios experimentais e comparativos, estruturados para investigar o desempenho de diferentes algoritmos de AR profundo aplicados ao jogo *Riichi Mahjong*. A pesquisa foi de natureza exploratória e descritiva, com abordagem quantitativa, buscando compreender o comportamento dos agentes sob distintas políticas de aprendizado em um ambiente de informação parcial.

A implementação dos agentes foi realizada em linguagem Python 3.9, utilizando as bibliotecas SB3 e *SB3-Contrib* para os algoritmos *Q-Learning*, A2C e *MaskedPPO*, e o *PyTorch*

como *backend* de aprendizado profundo. A arquitetura convolucional adotada, denominada `MahjongCNN`, foi inspirada no modelo de Gao et al. (2019), mas simplificada para reduzir o custo computacional e permitir o treinamento em hardware de médio porte (descrito em detalhes na Seção 4.1.2). Essa estrutura foi integrada ao ambiente *PyMahjong*, compatível com a API do *Gymnasium*, que reproduz as regras oficiais do *Riichi Mahjong* e fornece observações, recompensas e restrições adequadas à aplicação de técnicas de AR.

Cada agente foi treinado por 500 000 episódios, com geração de *checkpoints* a cada 100 000 interações, utilizando seis ambientes paralelos (*SubprocVecEnv*) para aumentar a eficiência e a diversidade das experiências coletadas. As sementes de geração aleatória foram fixadas em 42, assegurando a replicabilidade dos resultados. As recompensas foram diretamente baseadas no sistema oficial de pontuação do *Mahjong*, permitindo que o desempenho do agente refletisse fielmente seu sucesso estratégico no jogo.

A análise dos resultados considerou métricas quantitativas, como taxa de vitória, *payoff* médio e tempo de treinamento, possibilitando comparar a estabilidade, a eficiência e a capacidade de aprendizado de cada algoritmo. Esse delineamento experimental buscou, portanto, não apenas verificar o desempenho individual de cada agente, mas também identificar vantagens e limitações de cada abordagem de aprendizado no contexto de jogos de informação incompleta.

De forma geral, o método proposto garante transparência, reprodutibilidade e consistência com as boas práticas de pesquisa em AR profundo. A integração entre um ambiente simulado realista, algoritmos consolidados na literatura e uma arquitetura neural adaptada ao contexto computacional disponível constitui a base para a análise comparativa apresentada nos capítulos seguintes.

5 Resultados

Esta seção apresenta os resultados obtidos nos **experimentos de avaliação dos agentes**, conduzidos com o objetivo de analisar o desempenho das três abordagens de AR profundo desenvolvidas neste trabalho: *Q-Learning*, A2C e *MaskedPPO*. Os experimentos foram realizados em dois cenários distintos, a fim de observar o comportamento dos agentes em situações competitivas controladas e de interação mútua.

No primeiro cenário, cada agente foi avaliado individualmente em um ambiente composto por três instâncias do agente *mahjong_VLOG_CQL*, conforme descrito por Han et al. (2022). Esse cenário teve como objetivo analisar a capacidade de cada modelo em competir contra oponentes otimizados, permitindo avaliar sua estabilidade, capacidade de decisão e desempenho em relação a um agente de referência reconhecido na literatura.

No segundo cenário, foi criado um ambiente misto contendo simultaneamente os três agentes treinados e um agente aleatório, cuja função era realizar ações de forma puramente estocástica. Esse arranjo experimental teve como finalidade observar a dinâmica competitiva entre os algoritmos, analisando como cada um se comporta diante de adversários com diferentes estratégias e níveis de racionalidade.

Os resultados obtidos nos experimentos foram organizados em tabelas que apresentam as principais métricas quantitativas de desempenho, incluindo a **taxa de vitória**, a **pontuação média** e a **variância da pontuação** dos agentes avaliados. Além disso, foram elaborados gráficos que ilustram a evolução de alguns parâmetros de treinamento ao longo do tempo, como o número de *frames per second* (FPS), a *perda de entropia* (*entropy loss*) e a *perda de valor* (*value loss*). Essas representações visuais complementam a análise quantitativa, permitindo uma avaliação mais clara do comportamento e da estabilidade dos algoritmos durante o processo de aprendizado.

5.1 Comparação com Han et al. (2022)

Os experimentos de avaliação contra os agentes de Han et al. (2022) foram realizados com o objetivo de comparar o desempenho dos três agentes desenvolvidos neste trabalho em relação ao modelo de referência *mahjong_VLOG_CQL*. Cada agente foi testado individualmente em um ambiente contendo três instâncias do agente do artigo, totalizando 100 partidas por modelo.

5.1.1 Agente *MaskedPPO*

A Tabela 4 mostra o último registro de treinamento do agente A2C, evidenciando os principais parâmetros relacionados ao processo de aprendizado e convergência do modelo

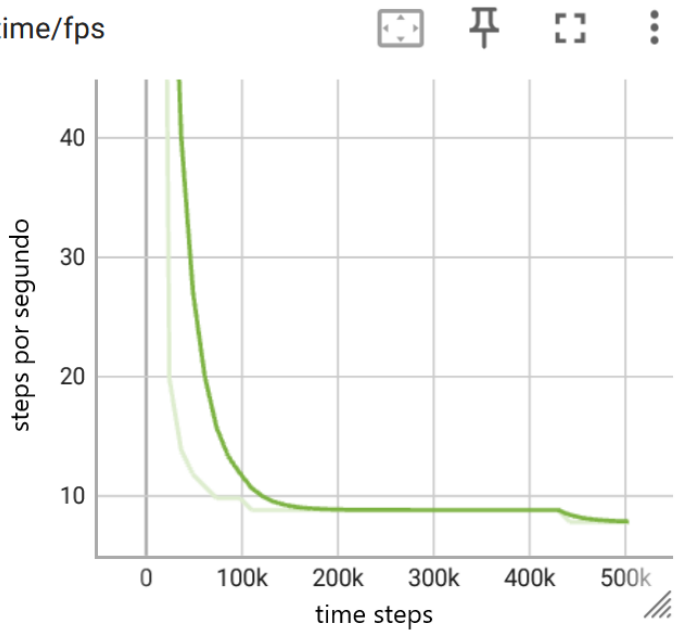
Tabela 4 – Último *log* de treinamento do agente *MaskedPPO*

Parâmetro	**Valor**
Quadros por segundo	8
Iterações	41
Tempo decorrido	56223
Total de *timesteps*	500000
Divergência KL aproximada	0.04077525
Fração de *clipping*	0.243
Faixa de *clipping*	0.2
Perda de entropia	-1.32
Variância explicada	0
Taxa de aprendizado	0.0003
Perda	1.97e+06
Número de atualizações	400
Perda do gradiente da política	0.0137
Perda da função de valor	3.17e+06

Fonte: Elaborado pelo autor.

A Figura 14 mostra a variação da métrica FPS ao longo do treinamento do agente *MaskedPPO*. Observa-se estabilidade após as iterações iniciais, indicando desempenho consistente do ambiente e boa eficiência computacional do processo de aprendizado.

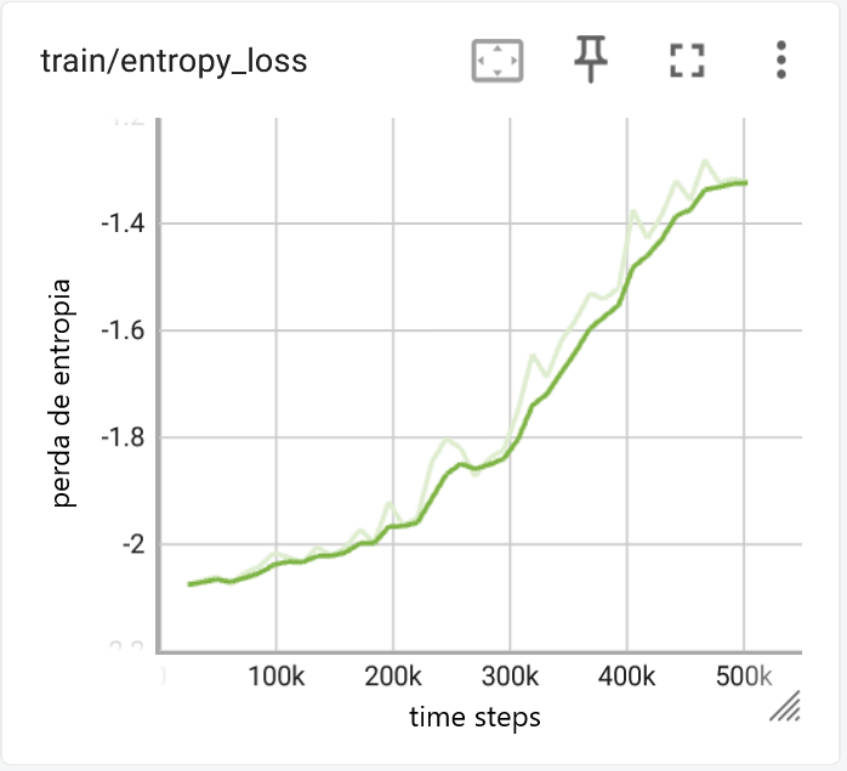
Figura 14 – Evolução da métrica FPS ao longo do tempo de treinamento do agente *MaskedPPO*.



Fonte: Elaborado pelo autor.

A Figura 15 apresenta a variação da *perda de entropia* (*entropy loss*) durante o treinamento do agente *MaskedPPO*. Observa-se uma tendência crescente ao longo do tempo, indicando redução da aleatoriedade nas ações e maior convergência da política de decisão.

Figura 15 – Evolução da *perda de entropia* (*entropy loss*) ao longo do tempo de treinamento do agente *MaskedPPO*.



Fonte: Elaborado pelo autor.

A Figura 16 mostra a evolução da *perda de valor* (*value loss*) durante o treinamento do agente *MaskedPPO*. Nota-se uma redução gradual dessa métrica, indicando melhoria na estimativa da função de valor e maior estabilidade na convergência do aprendizado.

Figura 16 – Evolução da *perda de valor* (*value loss*) ao longo do tempo de treinamento do agente *MaskedPPO*.

Fonte: Elaborado pelo autor.

A Tabela 5 apresenta os resultados obtidos pelo agente *MaskedPPO* durante os experimentos de avaliação contra o *mahjong_VLOG_CQL*, incluindo taxa de vitória e pontuação média ao longo das 100 partidas simuladas.

Tabela 5 – Resultados do agente *MaskedPPO* contra o *mahjong_VLOG_CQL*

Métrica	**Valor**	**Partidas**
Taxa de vitória (%)	2.00	100
Payoff médio	-1000.000	100
Variância	± 800.000	100

Fonte: Elaborado pelo autor.

Os resultados presentes na tabela 5, indicam que o agente performou pior que os agentes de Han et al. (2022), porém manteve um *payoff* relativamente bom, visto que representa a derrota por 1 mão simples de *Riichi Mahjong* (European Mahjong Association, 2016), indicando um bom comportamento defensivo do agente treinado com *MaskedPPO*.

5.1.2 Agente A2C

A Tabela 6 mostra o último registro de treinamento do agente A2C, evidenciando os principais parâmetros relacionados ao processo de aprendizado e convergência do modelo.

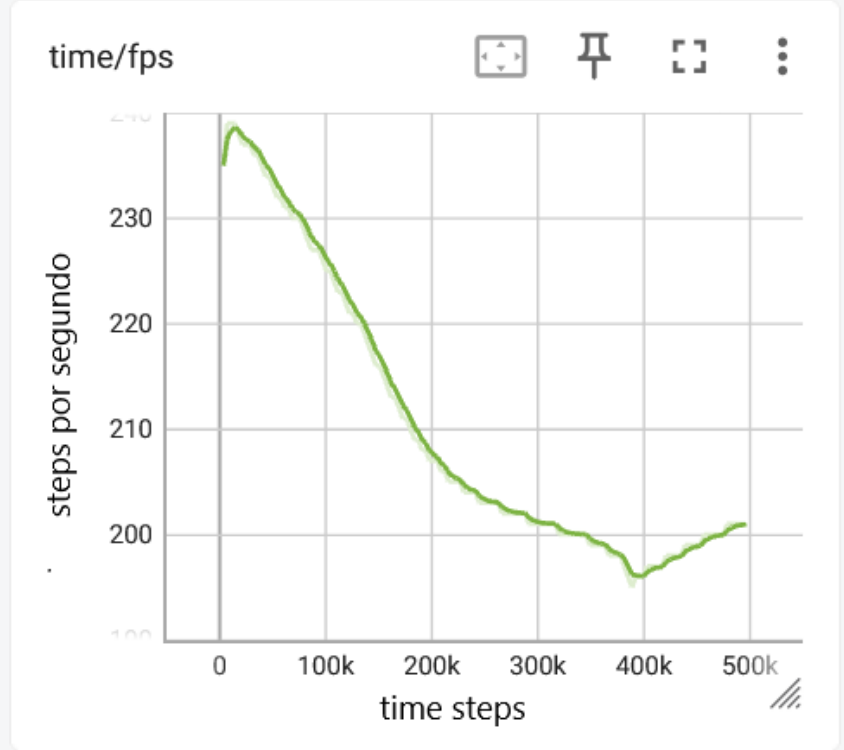
Tabela 6 – Último *log* de treinamento do agente A2C

Parâmetro	**Valor**
Ações inválidas por *rollout*	21
Proporção de ações válidas	0.417
Quadros por segundo	201
Iterações	13800
Tempo decorrido	2460
Total de *timesteps*	496800
Perda de entropia	-3.16
Variância explicada	0.817
Taxa de aprendizado	0.0003
Número de atualizações	13799
Perda da política	-4.89
Perda da função de valor	36.9

Fonte: Elaborado pelo autor.

A Figura 17 apresenta a variação da métrica FPS durante o treinamento do agente A2C.

Figura 17 – Evolução da métrica FPS ao longo do tempo de treinamento do agente A2C.

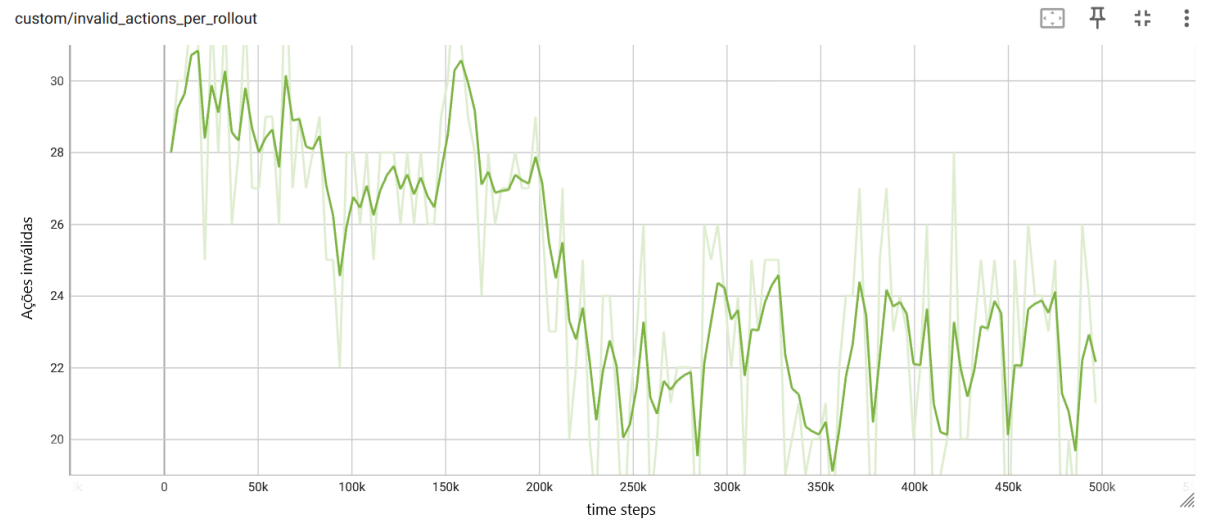


Fonte: Elaborado pelo autor.

A Figura 18 apresenta a quantidade média de ações inválidas por *rollout* durante o treinamento do agente A2C. Observa-se uma redução gradual desse valor ao longo do tempo, indicando que o agente aprendeu a reconhecer e evitar ações ilegais, refletindo melhora em sua

política de decisão.

Figura 18 – Evolução da média de ações inválidas por *rollout* durante o treinamento do agente A2C.



Fonte: Elaborado pelo autor.

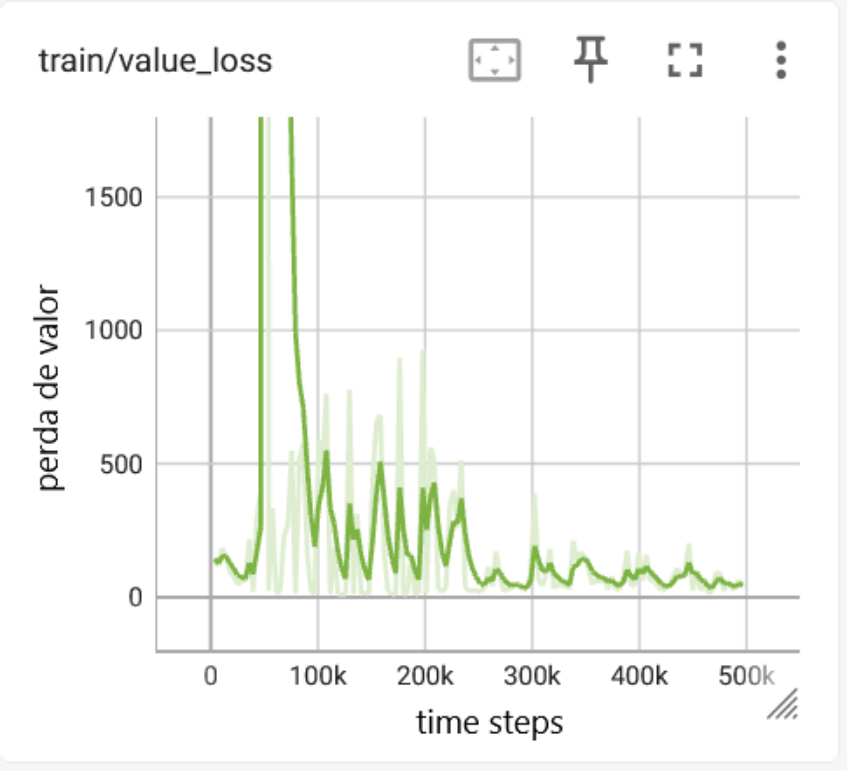
A Figura 19 mostra a evolução da perda de entropia (em inglês, *entropy loss*) do agente A2C. Embora o gráfico apresente maior instabilidade (consequência da possibilidade de seleção de ações inválidas durante o treinamento). Observa-se uma tendência decrescente, indicando redução da aleatoriedade nas decisões e maior consistência na política aprendida.

Figura 19 – Evolução da perda de entropia (em inglês, *entropy loss*) ao longo do tempo de treinamento do agente A2C.

Fonte: Elaborado pelo autor.

A Figura 20 apresenta a *perda de valor* (em inglês, *value loss*) ao longo do treinamento do agente A2C. Apesar da maior instabilidade observada (também decorrente de ações inválidas). Nota-se uma tendência geral de redução dessa métrica, o que indica melhora gradual na estimativa da função de valor e, consequentemente, no desempenho do agente.

Figura 20 – Evolução da *perda de valor* (em inglês, *value loss*) ao longo do tempo de treinamento do agente A2C.



Fonte: Elaborado pelo autor.

Os resultados de desempenho do agente A2C contra o *mahjong_VLOG_CQL* estão resumidos na Tabela 7, com destaque para a taxa de vitória e o valor médio de pontuação obtido nas 100 partidas realizadas.

Tabela 7 – Resultados do agente A2C contra o *mahjong_VLOG_CQL*

Métrica	**Valor**	**Partidas**
Taxa de vitória (%)	1.00	100
Payoff médio	-1800.000	100
Variância	± 1700.000	100

Fonte: Elaborado pelo autor.

Os resultados presentes na tabela 7, indicam que o agente performou pior que os agentes de Han et al. (2022), porém manteve um *payoff* relativamente bom, visto que representa a derrota de aproximadamente 2 mãos simples de *Riichi Mahjong* (European Mahjong Association, 2016), indicando um bom comportamento defensivo do agente treinado com A2C, porém inferior ao agente treinado com *MaskedPPO*.

5.1.3 Agente DQL

A Tabela 8 apresenta o último *log* de treinamento do agente DQL, contendo os principais parâmetros observados ao longo do processo de aprendizado e das atualizações de valor.

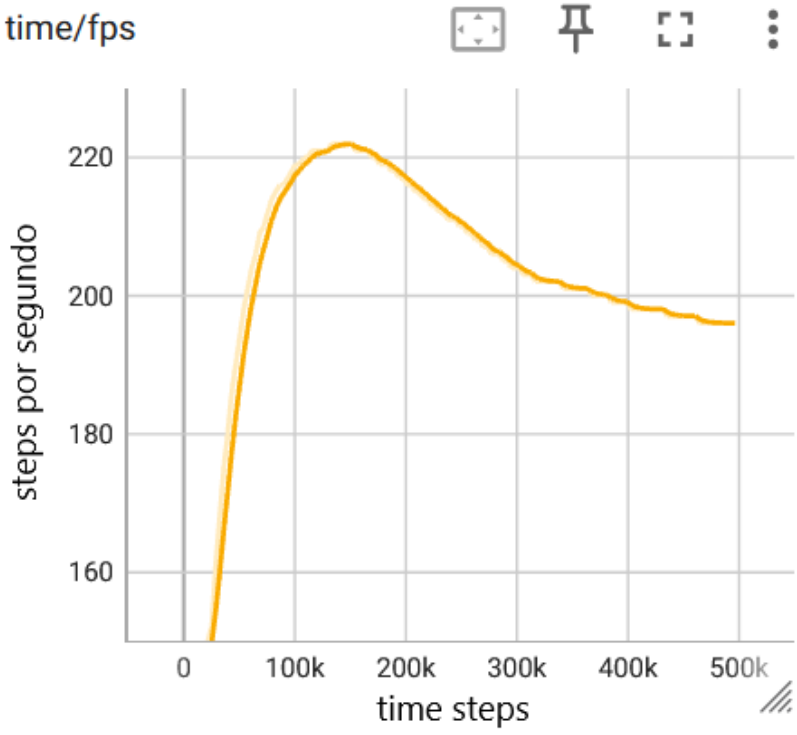
Tabela 8 – Último *log* de treinamento do agente DQL

Parâmetro	**Valor**
Ações inválidas por *rollout*	49
Proporção de ações válidas	0.119
Taxa de exploração	0.05
Episódios	3632
Quadros por segundo	877
Tempo decorrido	569
Total de *timesteps*	499968
Taxa de aprendizado	0.0001
Perda	0.331
Número de atualizações	1913

Fonte: Elaborado pelo autor.

A Figura 21 apresenta a variação da métrica FPS durante o treinamento do agente DQN.

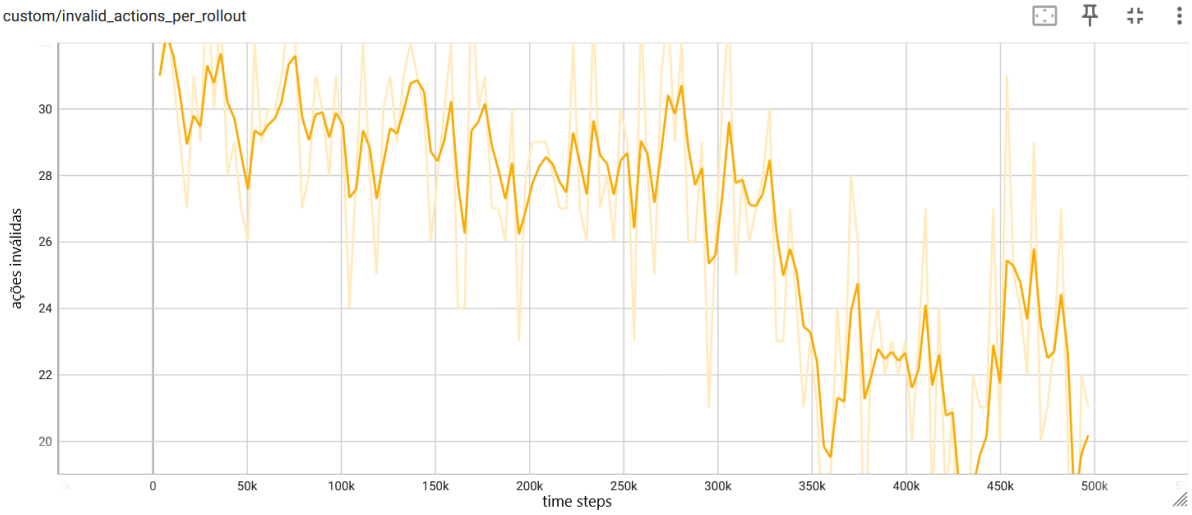
Figura 21 – Evolução da métrica FPS ao longo do tempo de treinamento do agente DQN.



Fonte: Elaborado pelo autor.

A Figura 22 apresenta a quantidade média de ações inválidas por *rollout* durante o treinamento do agente DQN. Assim como observado no agente A2C, nota-se uma redução gradual desse valor ao longo do tempo, indicando que o agente aprendeu progressivamente a evitar ações ilegais, o que reflete uma melhora em sua política de decisão.

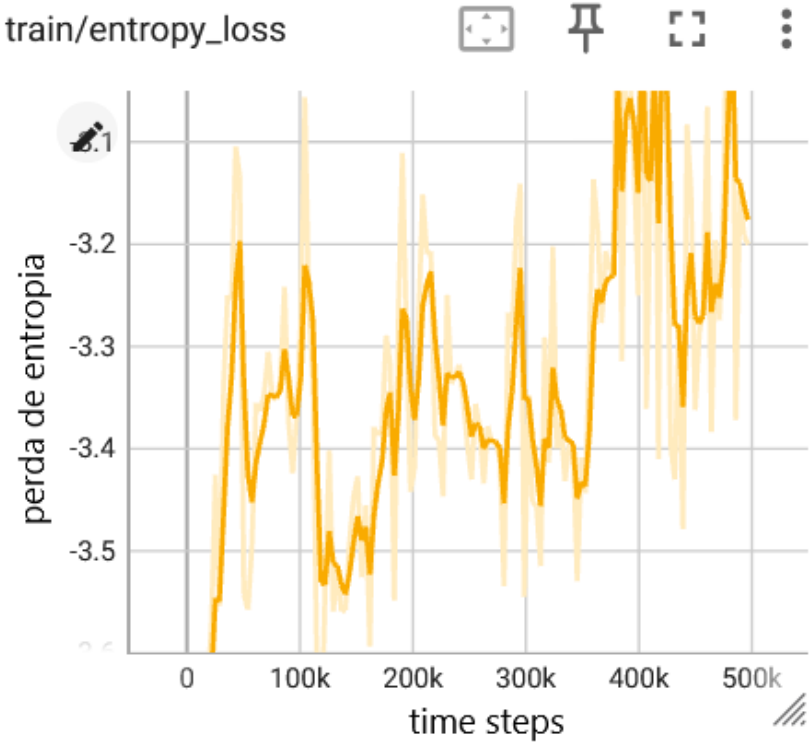
Figura 22 – Evolução da média de ações inválidas por *rollout* durante o treinamento do agente DQN.



Fonte: Elaborado pelo autor

A Figura 23 mostra a evolução da *perda de entropia* (*entropy loss*) do agente DQN. Embora o gráfico apresente maior instabilidade — resultado da possibilidade de seleção de ações inválidas durante o treinamento — observa-se uma tendência decrescente, indicando redução da aleatoriedade nas decisões e maior consistência na política aprendida.

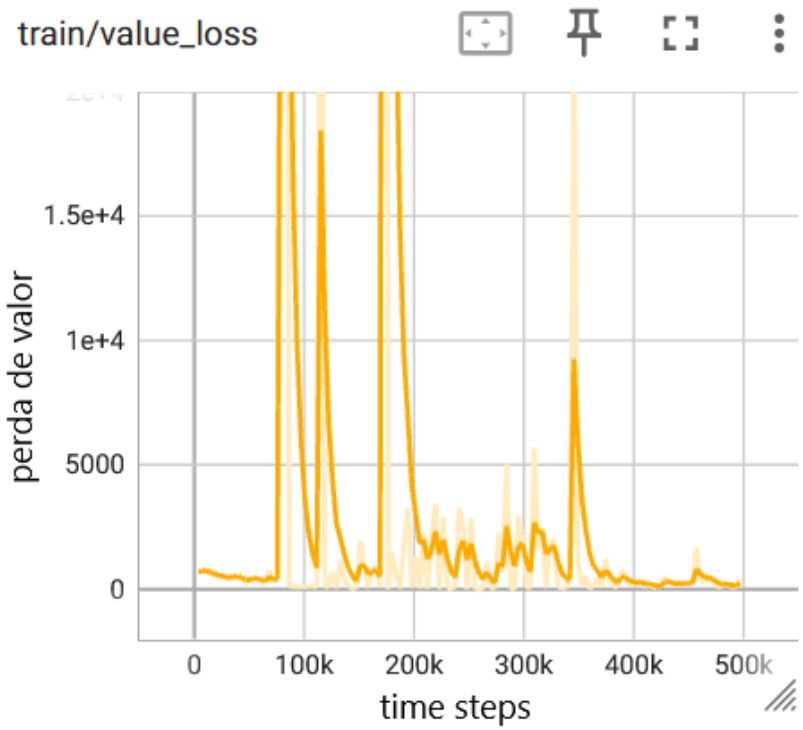
Figura 23 – Evolução da *perda de entropia* (*entropy loss*) ao longo do tempo de treinamento do agente DQN.



Fonte: Elaborado pelo autor.

A Figura 24 apresenta a *perda de valor* (*value loss*) ao longo do treinamento do agente DQN. Apesar da maior instabilidade observada — também decorrente das ações inválidas — nota-se uma tendência geral de redução dessa métrica, o que indica melhora gradual na estimativa da função de valor e, consequentemente, no desempenho do agente.

Figura 24 – Evolução da *perda de valor* (*value loss*) ao longo do tempo de treinamento do agente DQN.



Fonte: Elaborado pelo autor.

A Tabela 9 sintetiza o desempenho do agente DQL nos experimentos de avaliação, mostrando a taxa de vitória e o *payoff* médio obtido frente ao *mahjong_VLOG_CQL*.

Tabela 9 – Resultados do agente DQL contra o *mahjong_VLOG_CQL*

Métrica	**Valor**	**Partidas**
Taxa de vitória (%)	0.00	100
Payoff médio	-2500.000	100
Variância	± 2300.000	100

Fonte: Elaborado pelo autor.

Os resultados presentes na tabela 7, indicam que o agente performou pior que os agentes de Han et al. (2022), mantendo um *payoff* relativamente ruim, visto que representa a derrota de aproximadamente 3 mãos simples de *Riichi Mahjong* (European Mahjong Association, 2016).

5.1.4 Síntese

A Tabela 10 resume os resultados obtidos nos experimentos de avaliação contra os agentes do artigo.

De modo geral, os três agentes apresentaram desempenho inferior ao modelo de referência. Esse resultado pode ser atribuído ao ambiente de treinamento mais reduzido

Tabela 10 – Resumo dos resultados dos agentes treinados contra o *mahjong_VLOG_CQL*

Agente	**Taxa de Vitória (%)**	***Payoff Médio***	**Variância**	**Partidas**
MaskedPPO	2.00	-1000.000	± 800	100
A2C	1.00	-1800.000	± 1700	100
DQL	0.00	-2500.000	± 2300	100

Fonte: Elaborado pelo autor.

utilizado neste trabalho, que contou com menor capacidade computacional e menor número de instâncias de treino em comparação ao ambiente do artigo original. Além disso, o presente estudo teve como foco a comparação da eficiência entre algoritmos de AR puro, enquanto o trabalho de (HAN et al., 2022) inclui uma etapa prévia de Aprendizado Supervisionado antes do AR, o que fornece aos agentes de referência uma vantagem inicial significativa em termos de conhecimento do ambiente.

5.2 Resultados entre agentes e agente aleatório

Após a avaliação individual dos agentes contra o modelo de referência, foi conduzido um segundo experimento de avaliação para observar as interações diretas entre os três agentes desenvolvidos e um agente de comportamento aleatório. O objetivo foi analisar o desempenho relativo e o equilíbrio competitivo entre as abordagens de Aprendizado por Reforço, bem como identificar se algum dos agentes apresentaria dominância estratégica em um ambiente misto.

Nesse experimento, o ambiente foi composto pelos agentes *MaskedPPO*, A2C, DQL e um agente aleatório que selecionava suas ações de forma totalmente estocástica. Foram realizadas 100 partidas completas, e as métricas observadas incluíram o número total de vitórias, o *payoff* médio e a variância de cada agente, conforme apresentado na Tabela 11.

Tabela 11 – Resultados dos agentes em ambiente competitivo com agente aleatório

Agente	**Vitórias (em 100)**	***Payoff* médio**	***Payoff* médio**
MaskedPPO	**55**	**+4110**	**± 1810**
A2C	30	$+1090$	± 2120
DQL	14	-1240	± 2370
Aleatório	1	-3960	± 3290

Fonte: Elaborado pelo autor./

Os resultados indicam que o agente *MaskedPPO* apresentou desempenho superior em relação aos demais, vencendo mais da metade das partidas e mantendo um *payoff* médio positivo e elevado. O A2C obteve desempenho intermediário, demonstrando estabilidade e menor variabilidade, enquanto o agente DQL apresentou ganhos pontuais, porém com média negativa. Como esperado, o agente aleatório apresentou o pior desempenho geral, servindo como limite inferior de comparação.

Esses resultados sugerem que abordagens baseadas em política, como o *MaskedPPO* e o A2C, são mais eficazes em ambientes com múltiplos agentes e ações dependentes do contexto, enquanto o DQL, por ser um método puramente baseado em valor, mostrou-se menos adaptável às dinâmicas complexas do jogo *Riichi Mahjong*. Assim, o experimento reforça a vantagem das arquiteturas *actor–critic* e das políticas parametrizadas em cenários de informação parcial e alta variabilidade estratégica.

6 Conclusão

O presente trabalho teve como objetivo investigar o desempenho de diferentes algoritmos de AR profundo aplicados ao jogo *Riichi Mahjong*, um ambiente caracterizado por informação parcial, alto grau de incerteza e complexidade combinatória. Foram implementados e avaliados três agentes baseados em abordagens distintas: *Q-Learning* (método *value-based*), A2C (método *actor–critic*) e *MaskedPPO* (método *policy-based*), todos utilizando uma CNN.

Os experimentos de avaliação mostraram que, embora o desempenho absoluto dos agentes frente ao modelo de referência *mahjong_VLOG_CQL* (HAN et al., 2022) tenha sido limitado, o *MaskedPPO* apresentou o melhor resultado entre os três, seguido do A2C e do *Q-Learning*. Quando inseridos em um ambiente competitivo contendo os três agentes e um agente aleatório, observou-se que os métodos baseados em política foram mais eficazes na exploração do espaço de ações e na adaptação a adversários com diferentes estratégias.

Em ambientes de treinamento com número de episódios fixo, verificou-se que os agentes treinados com métodos baseados em política tendem a apresentar desempenho superior em relação aos métodos baseados em valor, evidenciando melhor capacidade de generalização e estabilidade. No entanto, esses métodos demandam maior tempo de treinamento e maior capacidade computacional, o que se refletiu diretamente na duração dos experimentos e no uso intensivo da GPU. Já o *Q-Learning*, apesar de mais leve e rápido de treinar, mostrou dificuldade em lidar com o grande espaço de estados e com a natureza estocástica do jogo.

A diferença de desempenho em relação ao trabalho de (HAN et al., 2022) é explicada pelas condições experimentais deste estudo, que utilizou um ambiente de treinamento mais reduzido e não incluiu uma etapa de aprendizado supervisionado anterior ao AR. Essa decisão metodológica teve como propósito isolar o comportamento intrínseco dos algoritmos de Aprendizado por Reforço puro, permitindo comparações mais diretas entre suas dinâmicas de aprendizado.

De forma geral, os resultados obtidos indicam que abordagens baseadas em política, especialmente as que incorporam otimização proximal e mascaramento de ações inválidas, são mais promissoras para ambientes complexos e parcialmente observáveis como o *Mahjong*. Entretanto, o elevado custo computacional associado a essas abordagens representa um desafio relevante, principalmente em contextos com restrições de hardware.

Como perspectivas futuras, recomenda-se expandir o número de episódios de treinamento, explorar arquiteturas neurais mais profundas e investigar o uso de técnicas híbridas, como o pré-treinamento supervisionado ou o aprendizado hierárquico. Além disso, a integração de mecanismos de atenção e memória de longo prazo pode aprimorar a capacidade dos agentes de lidar com a sequência temporal de decisões característica do *Mahjong*.

Em síntese, este trabalho contribuiu para a análise comparativa de algoritmos de AR Profundo em um ambiente de alta complexidade, evidenciando suas vantagens, limitações e o impacto do custo computacional no processo de aprendizado.

Referências

ALPAYDIN, E. *Introduction to Machine Learning*. 4. ed. [S.l.]: MIT Press, 2020.

BELLMAN, R. *Dynamic Programming*. USA: Princeton University Press, 2010. ISBN 0691146683.

BLAIR, A.; SAFFIDINE, A. Ai surpasses humans at six-player poker. *Science*, American Association for the Advancement of Science, v. 365, n. 6456, p. 864–865, 2019.

BUCHANAN, B. A (very) brief history of artificial intelligence. *AI Magazine*, v. 26, p. 53–60, 12 2005.

CLIFTON, J.; LABER, E. Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, Annual Reviews, v. 7, n. Volume 7, 2020, p. 279–301, 2020. ISSN 2326-831X. Disponível em: <https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-031219-041220>.

ENGSTROM, L.; ILYAS, A.; SANTURKAR, S.; TSIPRAS, D.; JANOOS, F.; RUDOLPH, L.; MADRY, A. *Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO*. 2020. Disponível em: <https://arxiv.org/abs/2005.12729>.

European Mahjong Association. *Riichi – Rules for Japanese Mahjong (2016 edition)*. 2016. ed. [S.l.], 2016. Revision 1.0, last updated April 1, 2016. Disponível em: <https://mahjong-europe.org/portal/images/docs/Riichi-rules-2016-EN.pdf>.

GAO, S.; OKUYA, F.; KAWAHARA, Y.; TSURUOKA, Y. *Building a Computer Mahjong Player via Deep Convolutional Neural Networks*. 2019. Disponível em: <https://arxiv.org/abs/1906.02146>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.; BENGIO, Y. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1.

HAN, D.; KOZUNO, T.; LUO, X.; CHEN, Z.-Y.; DOYA, K.; YANG, Y.; LI, D. Variational oracle guiding for reinforcement learning. In: *International Conference on Learning Representations*. [s.n.], 2022. Disponível em: <https://openreview.net/forum?id=pjqqxepwoMy>.

HU, C.; ZHAO, Y.; WANG, Z.; DU, H.; LIU, J. Games for artificial intelligence research: A review and perspectives. *IEEE Transactions on Artificial Intelligence*, v. 5, n. 12, p. 5949–5968, 2024.

HUANG, S.; ONTAñóN, S. A closer look at invalid action masking in policy gradient algorithms. *The International FLAIRS Conference Proceedings*, University of Florida George A Smathers Libraries, v. 35, maio 2022. ISSN 2334-0762. Disponível em: <http://dx.doi.org/10.32473/flairs.v35i.130584>.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, v. 4, p. 237–285, 1996.

LAPAN, M. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. [S.l.]: Packt Publishing Ltd, 2018.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. Disponível em: <https://doi.org/10.1038/nature14539>.

LI, J.; KOYAMADA, S.; YE, Q.; LIU, G.; WANG, C.; YANG, R.; ZHAO, L.; QIN, T.; LIU, T.-Y.; HON, H.-W. *Suphx: Mastering Mahjong with Deep Reinforcement Learning*. 2020. Disponível em: <https://arxiv.org/abs/2003.13590>.

LI, J.; WU, S.; FU, H.; FU, Q.; ZHAO, E.; XING, J. Speedup training artificial intelligence for mahjong via reward variance reduction. In: *2022 IEEE Conference on Games (CoG)*. [S.l.: s.n.], 2022. p. 345–352.

MILLER, S. D. *Riichi Mahjong: the ultimate guide to the Japanese game taking the world by storm*. [S.l.]: Lulu. com, 2015.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997.

MIZUKAMI, N.; TSURUOKA, Y. Building a computer mahjong player based on monte carlo simulation and opponent models. In: *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. [S.l.: s.n.], 2015. p. 275–283.

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T. P.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. *Asynchronous Methods for Deep Reinforcement Learning*. 2016. Disponível em: <https://arxiv.org/abs/1602.01783>.

MNIH, V. et al. Human-level control through deep reinforcement learning. *Nature*, v. 518, p. 529–533, 2015.

MOZUR, P. Google's alphago defeats chinese go master in win for ai. *International New York Times*, International Herald Tribune, p. NA–NA, 2017.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4. ed. [S.l.]: Pearson, 2021.

SCHRITTWIESER, J.; ANTONOGLOU, I.; HUBERT, T.; SIMONYAN, K.; SIFRE, L.; SCHMITT, S.; GUEZ, A.; LOCKHART, E.; HASSABIS, D.; GRAEPEL, T. et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, Nature Publishing Group UK London, v. 588, n. 7839, p. 604–609, 2020.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. *Proximal Policy Optimization Algorithms*. 2017. Disponível em: <https://arxiv.org/abs/1707.06347>.

SILVER, D. et al. Mastering the game of go with deep neural networks and tree search. *Nature*, v. 529, p. 484–489, 2016.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. 2. ed. [S.l.]: MIT Press, 2018.

SUTTON, R. S.; MCALLESTER, D.; SINGH, S.; MANSOUR, Y. Policy gradient methods for reinforcement learning with function approximation. In: SOLLA, S.; LEEN, T.; MüLLER, K. (Ed.). *Advances in Neural Information Processing Systems*. MIT Press, 1999. v. 12. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf>.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. *Machine Learning*, Kluwer Academic Publishers, v. 8, n. 3-4, p. 279–292, May 1992. Disponível em: <https://doi.org/10.1007/BF00992698>.

ZHAO, X.; HOLDEN, S. B. Towards a competitive 3-player mahjong ai using deep reinforcement learning. In: IEEE. *2022 IEEE Conference on Games (CoG)*. [S.l.], 2022. p. 524–527.