

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

LUANA GUEDES BARROS MARTINS

**Análise da capacidade de generalização
de algoritmos de aprendizagem por
reforço no contexto de jogos eletrônicos**

Goiânia
2019

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

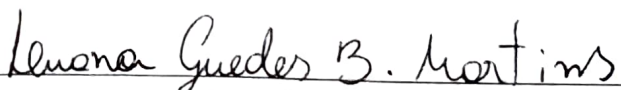
**AUTORIZAÇÃO PARA PUBLICAÇÃO DE TRABALHO DE
CONCLUSÃO DE CURSO EM FORMATO ELETRÔNICO**

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos “reproduzir” e “publicar” conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Análise da capacidade de generalização de algoritmos de aprendizagem por reforço no contexto de jogos eletrônicos

Autor(a): Luana Guedes Barros Martins

Goiânia, 12 de Dezembro de 2019.


Luana Guedes Barros Martins – Autora


Dra. Telma Woerle de Lima Soares – Orientadora

LUANA GUEDES BARROS MARTINS

Análise da capacidade de generalização de algoritmos de aprendizagem por reforço no contexto de jogos eletrônicos

Trabalho de Conclusão apresentado à Coordenação do Curso de Ciência da Computação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração: Ciência da Computação, Inteligência Artificial.

Orientadora: Profa. Dra. Telma Woerle de Lima Soares

Goiânia
2019



MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

LUANA GUEDES BARROS MARTINS

**Análise da capacidade de generalização de algoritmos de aprendizagem por reforço
no contexto de jogos eletrônicos**

Trabalho de conclusão de curso
apresentado à Universidade Federal de
Goiás como parte dos requisitos para a
obtenção do título de Bacharel em Ciências
da Computação.

Orientador: Prof.a Dr.a Telma Woerle de
Lima Soares

Aprovado em 12 de Dezembro de 2019.

BANCA EXAMINADORA

Prof.a Dr.a Telma Woerle de Lima Soares
Universidade Federal de Goiás
Instituto de Informática

Prof. Dr. Anderson da Silva Soares
Universidade Federal de Goiás
Instituto de Informática

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador(a).

Luana Guedes Barros Martins

Graduanda em Ciência da Computação pelo Instituto de Informática na Universidade Federal de Goiás.

Dedico este trabalho a todos aqueles que me incentivaram e me apoiaram durante toda a trajetória acadêmica.

Agradecimentos

Agradeço aos familiares e amigos, por todo o apoio e incentivo que me deram ao longo dos anos, e por toda confiança que sempre depositaram em mim. Em especial, agradeço a minha mãe e psicóloga, que sempre foi minha maior fonte de inspiração e força, por todo conhecimento compartilhado que contribuíram bastante para os estudos deste trabalho. Agradeço ao meu grande companheiro e namorado Bryan, que me fez companhia durante toda esta jornada, aprendendo e crescendo junto comigo. Agradeço ao Instituto de Informática, que ao longo da minha formação ofereceu um ambiente de estudo agradável, motivador e repleto de oportunidades. Agradeço ao laboratório Deep Learning Brasil e aos membros do grupo de pesquisa de Aprendizado por Reforço por toda sabedoria e experiência compartilhados. Agradeço também a todos os professores que contribuíram com a minha trajetória acadêmica, especialmente a Profa. Dra. Telma Woerle de Lima Soares, responsável pela orientação do meu projeto, e ao Prof. Dr. Anderson da Silva Soares. Por fim, agradeço à COPEL e ao laboratório Deep Learning Brasil pelo auxílio financeiro que possibilitou a dedicação e a operacionalização deste estudo.

A todos, minha gratidão eterna por toda sabedoria, tempo e experiência compartilhados.

É verdade quando a filosofia diz que a vida só pode ser compreendida olhando-se para trás. No entanto, esqueceram de outra frase: que ela só pode ser vivida olhando-se para a frente.

Soren Kierkegaard,
Die Tagebücher, 1834 - 1855.

Resumo

Martins, Luana Guedes Barros. **Análise da capacidade de generalização de algoritmos de aprendizagem por reforço no contexto de jogos eletrônicos.** Goiânia, 2019. 42p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

Em grandes problemas, os sistemas de aprendizado por reforço devem utilizar aproximadores de função parametrizados, como redes neurais, para generalizar entre situações e ações semelhantes. Apesar de ter se mostrado um método eficaz em problemas complexos específicos, ainda falham no aspecto de generalização, e os ambientes de avaliação mais comuns de aprendizado por reforço ainda incentivam o treinamento e avaliação no mesmo conjunto de ambientes. Para ser possível avaliar a capacidade de um algoritmo generalizar entre tarefas é necessário ambientes de avaliação que medem o seu desempenho em um conjunto de testes distintos daqueles utilizados no treinamento. Com isso, este trabalho se propõe a avaliar o desempenho do algoritmo *Proximal Policy Optimization* (PPO), no ambiente de avaliação *General Video Game Artificial Intelligence* (GVGAI) que fornece uma subdivisão de um mundo virtual de um jogo em diferentes fases ou níveis. Apesar do PPO em geral reportar ótimos resultados, é possível notar que o algoritmo sofre com sobre-ajuste ao conjunto de treinamento.

Palavras-chave

Inteligência Artificial; Aprendizado de Máquina; Aprendizado por Reforço; Redes Neurais; Agentes Inteligentes; Jogos; GVGAI; Proximal Policy Optimization; PPO

Abstract

Martins, Luana Guedes Barros. **Analysis of the capability of generalization of reinforcement learning algorithms in the context of electronic games.** Goiânia, 2019. 42p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

In major problems, reinforcement learning systems should use parameterized function approximators, such as neural networks, to generalize between similar situations and actions. Although proven to be an effective method for specific complex problems, they still fail in the generalization aspect, and the most common reinforcement learning benchmarks still use the same environments for both training and testing. To be able to evaluate the ability of an algorithm to generalize across tasks requires benchmarks that measure its performance on a set of tests that are distinct from those used in training. Therefore, this work aims to evaluate the performance of the Proximal Policy Optimization (PPO) algorithm in the General Video Game Artificial Intelligence (GVGAI) benchmark that provides a subdivision of a virtual world of a game in different stages or levels. Although PPO generally reports great results, it can be noted that the algorithm suffers from overfitting to the training set.

Keywords

Artificial Intelligence; Machine Learning; Reinforcement Learning; Neural Network; Games; Learned Agents; GVGAI; Proximal Policy Optimization; PPO

Sumário

Lista de Figuras	11
Lista de Tabelas	12
Lista de Algoritmos	13
1 Introdução	14
2 Fundamentos Teóricos	16
2.1 Redes Neurais Artificiais	16
2.2 Aprendizado de Máquina	19
2.3 Aprendizado por Reforço	19
2.4 Métodos de Gradiente de Política	21
2.5 <i>Proximal Policy Optimization</i>	23
3 Metodologia	25
3.1 <i>General Video Game Artificial Intelligence</i>	25
3.2 Algoritmo de Treinamento	28
3.3 Testes	30
4 Resultados	31
5 Conclusão	36
Referências Bibliográficas	38
A Hiperparâmetros	42

Lista de Figuras

2.1	Modelo de um neurônio artificial [31].	16
2.2	Exemplos de funções de ativação.	17
(a)	Degrau	17
(b)	Sigmoidal	17
(c)	Gaussiana	17
2.3	Rede neural com duas camadas intermediárias.	17
2.4	Interação agente-ambiente em um Processo de Decisão de Markov [40].	20
2.5	Gráficos da função L^{CLIP} .	24
3.1	Fluxo de funcionamento da ferramenta GYM.	26
3.2	Exemplos de jogos disponíveis no GVGAI_GYM [41].	26
(a)	<i>Superman</i>	26
(b)	<i>Wait For Breakfast</i>	26
3.3	Captura de tela do jogo <i>Aliens</i> .	27
3.4	Captura de tela do jogo <i>Boulder Dash</i> .	27
3.5	Captura de tela do jogo <i>Missile Command</i> .	27
3.6	Resultado do pré-processamento das imagens.	28
(a)	<i>Aliens</i>	28
(b)	<i>Boulder Dash</i>	28
(c)	<i>Missile Commands</i>	28
3.7	Ilustração esquemática de uma rede neural convolucional [20].	29
3.8	Fluxo de treinamento Ator-Crítico.	29
4.1	Exemplos de execução do jogo <i>Aliens</i> .	32
4.2	Exemplos de execução do jogo <i>Boulder Dash</i> .	33
4.3	Exemplos de execução do jogo <i>Missile Command</i> .	33
4.4	Gráfico de relação treinamento-avaliação.	34
(b)	<i>Aliens</i>	34
(c)	<i>Boulder Dash</i>	34
(d)	<i>Missile Command</i>	34
4.5	Gráfico de recompensa dos modelos treinados.	35
(a)	<i>Aliens</i>	35
(b)	<i>Boulder Dash</i>	35
(c)	<i>Missile Command</i>	35

Lista de Tabelas

4.1	Comparação das recompensas obtidas para cada jogo.	32
A.1	Hiperparâmetros para o algoritmo PPO.	42

Lista de Algoritmos

2.1 *Proximal Policy Optimization* com objetivo limitado

24

Introdução

A capacidade de aprender é um aspecto bastante importante da inteligência artificial. É impossível especificar todo o conhecimento que um agente inteligente precisará em tarefas não triviais do mundo real, portanto o agente precisa ter a capacidade de aprender a lidar com essa falta de conhecimento. Aprendizado por reforço, tendo suas raízes na psicologia comportamental (o behaviorismo), é uma abordagem baseada na ideia de que comportamentos são aprendidos através interação com o ambiente [14]. De acordo com o behaviorismo, o comportamento pode ser estudado de maneira sistemática e observável, independentemente dos estados mentais internos [1].

O aprendizado por reforço é um ramo de aprendizado de máquina, em que um agente aprende como se comportar em um ambiente executando ações e analisando observações e resultados obtidos. É um método de programação de agentes através do oferecimento de recompensas e punições, sem a necessidade de especificar como uma tarefa deve ser realizada, cujo objetivo é maximizar a recompensa esperada. Pensando dessa maneira, jogos podem ser facilmente modelados como um ambiente em uma configuração de aprendizado por reforço, em que agentes (jogadores) têm um conjunto finito de ações que podem ser executadas em cada etapa e sua sequência de movimentos determina seu sucesso.

Desde o nascimento da ideia de inteligência artificial jogos tem sido uma maneira eficiente para medir a sua capacidade, tornando a prática de avaliação de algoritmos usando jogos e competições comum entre pesquisadores. Os jogos fornecem referências parametrizáveis que permitem experimentação rápida com várias abordagens, enquanto as competições estabelecem uma estrutura e um conjunto de regras comuns para garantir que esses algoritmos sejam comparados de maneira justa. Treinar um agente para superar os jogadores humanos [38] e otimizar sua pontuação pode nos ensinar como otimizar processos diferentes em uma variedade de subcampos diferentes.

Recentemente, o aprendizado por reforço tem se mostrado muito bem-sucedido em problemas complexos de alta dimensão, em grande parte devido ao aumento da potência computacional e ao uso de redes neurais profundas para aproximação de funções [19, 18, 17]. Embora tenham sido feitos grandes avanços no desenvolvimento de algo-

ritmos que aprendem com eficiência tipos específicos de problemas, agentes inteligentes também deveriam ser capazes de generalizar tarefas, usando a experiência anterior para adquirir novas habilidades mais rapidamente. Entretanto, a generalização entre tarefas permanece difícil para os algoritmos de aprendizado de reforço profundo de última geração, e os ambientes de avaliação mais comuns de aprendizado por reforço ainda incentivam o treinamento e avaliação no mesmo conjunto de ambientes. Buscando avançar os estudos nessa área, foram criados alguns ambientes de jogos eletrônicos com foco em facilitar o desenvolvimento, o teste e a avaliação de algoritmos de aprendizado por reforço com esse tipo de capacidade, separando explicitamente os ambientes de treinamento e teste [22, 12, 41].

Este trabalho tem como objetivo a análise da capacidade de generalização do algoritmo de aprendizado por reforço *Proximal Policy Optimization*, que demonstrou ser aplicável em configurações mais gerais e ter melhor desempenho geral [36]. Com base nisso, este documento está organizado em cinco capítulos. No Capítulo 2 há uma descrição dos conceitos e técnicas utilizadas, incluindo redes neurais, aprendizado de máquina, aprendizado por reforço e por fim a técnica de aprendizado de reforço utilizado neste trabalho. Em seguida, o Capítulo 3 apresenta a explicação dos materiais e métodos utilizados, incluindo os ambientes de jogos eletrônicos e uma descrição dos testes realizados. No Capítulo 4 estão descritos em detalhes os resultados obtidos, e, por fim no Capítulo 5 apresentam-se as conclusões e trabalhos futuros.

Fundamentos Teóricos

Este capítulo aborda conceitos e técnicas que fundamentam este trabalho. Será apresentado uma introdução sobre conceitos de redes neurais artificiais, bem como os fundamentos teóricos de aprendizado de máquina. Nas seções seguintes será abordado com mais detalhes o aprendizado por reforço, assim como a técnica de aprendizado utilizada neste trabalho.

2.1 Redes Neurais Artificiais

Na busca de construção de máquinas inteligentes, um modelo a ser seguido é o do cérebro humano. As Redes Neurais Artificiais (RNA) são modelos computacionais inspirados pelo sistema nervoso central dos seres humanos, capazes de automatizar a aprendizagem por meio da análise de dados. As RNAs são compostas por nós interconectados onde cada nó, também conhecido como unidades lógicas, são os modelos matemáticos de neurônios artificiais [8].

Na Figura 2.1 é representado um modelo simples de neurônio artificial [9]. Cada neurônio recebe valores de entrada x_1, x_2, \dots, x_n , que são ponderados dados os pesos $w_{k1}, w_{k2}, \dots, w_{kn}$, e então combinados em uma soma, gerando um valor de saída v_k . Por fim, uma função de ativação $\varphi(\cdot)$ é utilizada para determinar como o valor gerado irá se propagar para outros neurônios.

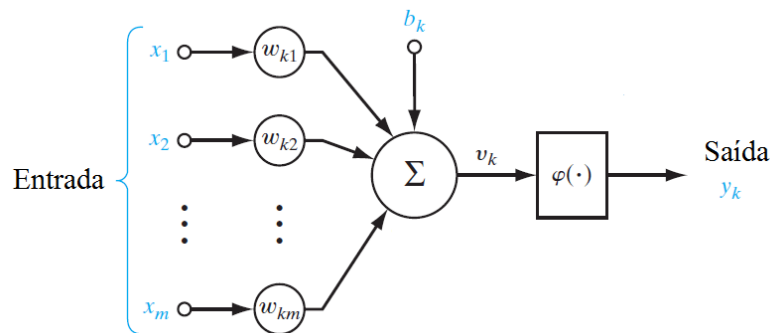


Figura 2.1: Modelo de um neurônio artificial [31].

Os pesos podem assumir valores positivos ou negativos, sendo seus valores ajustados em um processo de otimização, codificando o conhecimento adquirido pela rede. O modelo também inclui um *bias* b_k , utilizado para aumentar o grau de liberdade dos ajustes dos pesos, permitindo uma melhor adaptação, por parte da rede neural, ao conhecimento a ela fornecido.

As funções de ativação introduzem um componente não linear às redes neurais, fazendo com que aprendam mais do que relações lineares entre as variáveis dependentes e independentes. Elas basicamente decidem se um neurônio deve ser ativado ou não, ou seja, se a informação que o neurônio está recebendo é relevante ou deve ser ignorada [7]. Alguns exemplos de funções de ativação são observados na Figura 2.2.

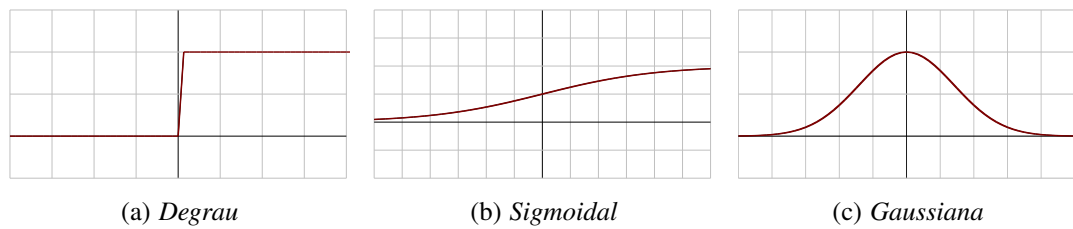


Figura 2.2: Exemplos de funções de ativação.

Os neurônios costumam ser divididos em camadas, onde existe uma camada de neurônios de entrada, podendo ser conectada a uma ou diversas camadas intermediárias que, por sua vez, se conectam aos neurônios da camada de saída, como mostra a Figura 2.3. A camada de entrada é responsável pelo recebimento dos dados a serem analisados, assim como a correspondente associação com os pesos de entrada. As camadas intermediárias tem por finalidade extrair as informações associadas ao sistema inferido, sendo também responsável pela maior parte do processamento destes dados. Já a camada de saída agrega as informações das camadas anteriores e ativa uma resposta adequada.

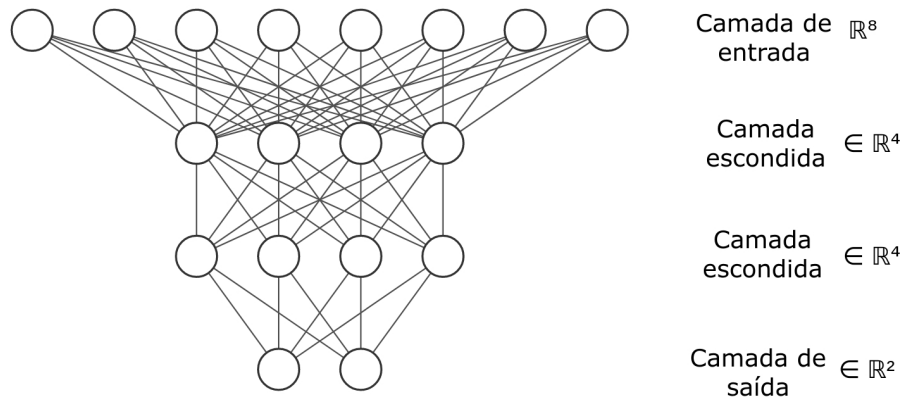


Figura 2.3: Rede neural com duas camadas intermediárias.

Redes neurais com muitas camadas intermediárias são ditas redes neurais profundas. Nessas redes, os dados são submetidos a um processo de várias etapas de reconhecimento de padrões, cada camada sendo responsável por reconhecer um conjunto de características diferente. Quanto mais profundo na rede, mais complexas serão as características reconhecidas, uma vez os dados são agregados e recombinaados da camada anterior.

As conexões de uma RNA define como os neurônios são conectados entre si, podendo ser classificadas em redes diretas (*feedforward*) ou redes recorrentes. Nas redes diretas, as informações fluem da camada de entrada para os neurônios da camada de saída. Já as conexões das redes recorrentes permitem que um neurônio receba a saída de um neurônio da mesma camada, de uma camada posterior, ou até mesmo a sua própria saída. Os sistemas baseados em RNAs, dependem fortemente da topologia destas redes (tamanho, estrutura, conexões), assim como de seus parâmetros. Como resultado, a determinação da arquitetura da rede afeta muito o seu desempenho, isto é, velocidade de aprendizado, exatidão do aprendizado, tolerância a ruídos e capacidade de generalização.

Dentre as topologias existentes, têm-se as Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Network*), que utilizam uma arquitetura especial que é particularmente bem adaptada para classificar imagens [15]. O ponto central de uma CNN são as camadas convolucionais, que dão o nome da rede. Uma convolução, nesse contexto, é uma operação linear que envolve a multiplicação de um conjunto de pesos ao conjunto de entrada. Como a entrada dessa rede é de natureza bi-dimensional, a multiplicação é realizada entre a entrada e vetores bi-dimensionais de pesos, chamados de filtros. Cada um dos filtros é responsável por aprender uma determinada característica nos dados de entrada.

Um método comum para o treinamento de uma rede neural é o algoritmo de *backpropagation* [27]. Em um processo de otimização, a rede neural transmite o sinal dos dados de entrada até a camada de saída, através de seus parâmetros. O gradiente de uma função de custo é então retro-propagado pela rede para que seus pesos possam ser alterados, visando minimizar essa função.

Nas últimas décadas, com a crescente complexidade dos problemas a serem tratados computacionalmente e do volume de dados gerados por diferentes setores, o uso de RNA se tornou cada vez mais comum. As redes neurais profundas são responsáveis por avanços recentes em uma grande variedade de tarefas difíceis de se resolver utilizando programação baseada em regras comuns, como, por exemplo, áreas de visão computacional [15], de reconhecimento de fala [21] e do processamento de linguagem natural [23].

2.2 Aprendizado de Máquina

O aprendizado de máquina pode ser definido como a área da Inteligência Artificial relacionada à busca de um conjunto de regras/padrões que permitem que as máquinas tomem decisões baseadas em um grande conjunto de dados sem serem especificamente programadas para essa tomada de decisões [30]. Em outras palavras, o aprendizado de máquina é o processo de encontrar um modelo ou hipótese que aproxima uma função alvo usando um conjunto de dados de treinamento. Depois que a máquina é treinada, a hipótese final deve satisfazer os dados de treinamento bem como dados não vistos anteriormente.

Tradicionalmente, essa área é dividida em três diferentes categorias, baseadas na natureza do tipo de aprendizagem: aprendizado supervisionado, aprendizado não supervisionado, e aprendizado por reforço. Além disso, há abordagens de otimização estocástica, como computação evolutiva, para aprendizado [29]. Todas essas abordagens são candidatas viáveis de treinamento para jogos eletrônicos [13].

O Aprendizado Supervisionado consiste na tarefa de inferir uma função a partir de dados de treinamento rotulados (par composto por um objeto de entrada e um valor de saída desejado), com a finalidade de prever rótulos desconhecidos em um conjunto de teste, podendo o rótulo ser um conjunto finito ou um valor real. Em contra-partida, no aprendizado não supervisionado não há nenhum tipo de orientação para verificar se a predição está correta e o objetivo passa a ser encontrar padrões nos dados. Já no aprendizado por reforço, um agente interage com um ambiente e seu objetivo é aprender um comportamento através dessa interação, visando maximizar um valor de retorno, chamado de recompensa.

2.3 Aprendizado por Reforço

Aprendizado por reforço é a área de aprendizado de máquina cujo objetivo reside em criar agentes artificiais que possuem a capacidade de alcançar um nível equivalente de performance e generalização dos seres humanos [39].

O aprendizado se dá pela interação do agente com o ambiente em que ele se encontra. Essa interação é representada pela escolha de ações a serem executadas no ambiente, que levam a mudanças de estados. Dessa interação, o agente obtém sinais de retorno, que irão categorizar se a ação tomada foi uma boa ou má escolha. Os sinais de retorno, também chamados de recompensa, podem ocorrer com frequência, como a alteração na pontuação dentro de um jogo, ou com pouca frequência, como se um agente ganhou ou perdeu um jogo. Todo esse processo de escolhas de ações se dá por tentativa

e erro do agente, tornando esse modelo de aprendizado o mais próximo do processo de aprendizado dos seres humanos e animais.

Um Processo de Decisão de Markov (MDP, do inglês *Markov Decision Process*) fornece uma estrutura matemática para modelagem de tomada de decisão em situações onde os resultados são em parte aleatórios e em parte sob o controle de um tomador de decisão [40]. Se um ambiente pode ser descrito como um MDP, então o agente pode construir uma árvore de probabilidade de estados futuros e suas recompensas, que poderá ser utilizada para calcular o valor do estado atual [16]. A figura 2.4 mostra a interação agente-ambiente em um MDP.

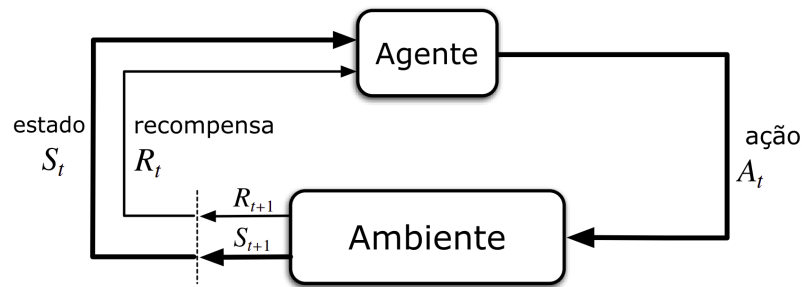


Figura 2.4: Interação agente-ambiente em um Processo de Decisão de Markov [40].

Temos que um MDP consiste de um conjunto finito de estados S , um conjunto de possíveis ações $A(s)$ para cada estado, um valor de retorno R para cada estado, e um modelo de transição $P(s'|s, a)$. Para cada episódio no tempo t , o retorno é definida por:

$$R_t = r_{t+1} + \gamma r_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2-1)$$

O retorno é calculado como uma soma de recompensas subsequentes. Um fator de desconto γ é utilizado para recompensas mais distantes do ponto de início t , definindo a visibilidade do agente. Entretanto, esse valor de retorno a longo prazo não é muito útil na prática: o agente pode tomar diferentes ações resultando em diferentes caminhos para um mesmo estado [16]. Para isso, temos uma função de retorno mais útil, a função de valor (Equação 2-2), que consiste na predição do valor de retorno a longo prazo.

$$V(s) = \mathbb{E} \left[\sum_{t=1}^T \gamma^{t-1} r_t \right] \forall s \in S \quad (2-2)$$

A função de valor representa o quão bom um estado é com base em sua recompensa. O objetivo do agente é maximizar esse valor. Entretanto, para alcançar bons estados é necessário escolher ações que levam para esses estados. Para isso, temos a política, que define o comportamento do agente, ou seja, uma política diz para o agente o que fazer em uma determinada situação [16]. Formalmente, política é definida como

uma distribuição de probabilidades sobre ações a para cada possível estado s em um dado tempo t (Equação 2-3), ou como um simples mapeamento de estados para ações (Equação 2-4).

$$\pi(a|s) = P(A_t = a|S_t = s) \quad (2-3)$$

$$\pi(s) = a \quad (2-4)$$

Há algumas abordagens para resolver um problema de aprendizado por reforço. Dentre elas temos os métodos baseados em valor e métodos baseados em política. Métodos baseados em valor focam em encontrar ou aproximar a função de valor ideal que maximizam a recompensa, enquanto sua política é mantida de tal forma a escolher o par de ação-estado que obtém a maior recompensa. Métodos baseados em política, tentam encontrar a política ideal diretamente, sem consultar a função valor, buscando também maximizar a recompensa. Ambos os métodos baseados em valor e em política são ditos livre de modelo, ou seja, ignoramos o modelo e dependemos apenas da amostragem e simulação para estimar as recompensas, para que não seja preciso conhecer o funcionamento interno do sistema. Já métodos baseados em modelo, se podemos definir uma função de recompensa, podemos calcular as ações ideais usando o modelo diretamente.

Mesmo não precisando levar em consideração a política durante o aprendizado, todo algoritmo de aprendizado por reforço deve seguir alguma política para decidir quais ações executar em cada estado. Algoritmos que levam em consideração a política que gerou decisões passadas de ação-estado são chamados de algoritmos *on-policy*, enquanto aqueles que a ignoram são conhecidos *off-policy*. Algoritmos *off-policy* geralmente empregam uma política de comportamento separada, independente da política que está sendo aprimorada; a política de comportamento é usada para simular trajetórias. Um algoritmo *off-policy* bem conhecido é o *Q-Learning* [42]. Sua regra de atualização usa a ação que produzirá o *Q-Value* mais alto, enquanto que a verdadeira política usada pode restringir essa ação ou escolher outra. O *Q-value* é uma medida da recompensa geral esperada, assumindo que o agente esteja no estado s e execute uma ação a , e continua a jogar até o final do episódio, seguindo alguma política π . A variação *on-policy* do *Q-learning* é o SARSA [28], onde a regra de atualização usa a ação escolhida pela política seguida.

2.4 Métodos de Gradiente de Política

Em métodos baseados em política e, em especial, métodos de gradiente de política, o objetivo é aprender uma política que consegue selecionar ações sem consultar a função de valor diretamente. Nesse método a política pode ser tanto determinística quanto

estocástica, além de ser possível de se trabalhar com um espaço de ações contínua [16]. Quando o agente obtém uma observação e precisa tomar uma ação precisamos da política, ou seja, é da política que precisamos ao resolver um problema de aprendizado de reforço.

Como qualquer configuração de aprendizado de máquina, pode-se definir um conjunto de parâmetros θ (como por exemplo, os pesos e *bias* das redes neurais) para parametrizar uma política - π_θ . O objetivo é buscar os parâmetros θ que maximizam a recompensa obtida, portanto, uma abordagem para resolver esse problema é o método do gradiente (Equação 2-5). No gradiente, continuamente percorremos os parâmetros usando a regra de atualização da Equação 2-6. A recompensa R_t guia a atualização, enquanto que a taxa de aprendizado α define a magnitude do ajuste feito nas atualizações [40].

$$\nabla_\theta J(\theta) = \mathbb{E}_{t \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) R_t \right] \quad (2-5)$$

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\pi_{\theta_t}) \quad (2-6)$$

A Equação 2-5 mede a probabilidade da trajetória sob a política atual. O uso da função de recompensa aumenta a probabilidade de uma política se a trajetória resultar em uma alta recompensa positiva. Por outro lado, diminui a probabilidade de uma política se resultar em uma alta recompensa negativa. Entretanto, a função de recompensa pode impedir a exploração de políticas que poderiam levar a uma recompensa maior, uma vez que um único caminho que leva o agente obter recompensas positivas seria reforçado sempre. O método de gradiente é um método de derivada de primeira ordem, não sendo muito confiável se a função de recompensa tiver uma curvatura muito íngreme [10]. Para resolver isso, é utilizado a função de vantagem (Equação 2-7) em vez da recompensa esperada, por reduzir a variação da estimativa [11].

$$A_t^{(n)}(s, a) = r_t + \gamma V(s_{t+1}) + \dots + \gamma^n V(s_{t+n}) - V(s_t) \quad (2-7)$$

A função de vantagem é uma medida de quanto uma determinada ação é uma decisão boa ou ruim, dado um determinado estado. Em outras palavras, a função informa sobre a recompensa extra que poderia ser obtida pelo agente, executando essa ação específica. Se $A_t(s, a) > 0$ o gradiente é empurrado nessa direção. Se $A_t(s, a) < 0$, ou seja, a ação escolhida é pior que o valor médio desse estado, o gradiente é empurrado na direção oposta.

A Equação 2-7 assume a forma de estimadores de diferença temporal, onde primeiro é estimado a soma das recompensas com desconto e então é descontado a estimativa da função de valor. Para $A_t^{(n)}$ com valores baixos de n a estimativa terá baixa variação, mas alto viés, enquanto para valores altos de n a estimativa terá viés baixo, mas alta variação [37]. Dado esse dilema de viés e variância, foi proposto uma abordagem

mais geral para calcular a vantagem, a Estimativa de Vantagem Generalizada (GAE, do inglês *Generalized Advantage Estimation*) [35].

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (2-8)$$

$$A_t^{GAE(\gamma, \lambda)} = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_t^V \quad (2-9)$$

A Equação 2-9 é um estimador de vantagem com dois parâmetros separados γ e λ , os quais contribuem para o balanceamento entre a variação e o viés. No entanto, ambos os parâmetros servem a propósitos diferentes e funcionam melhor com diferentes faixas de valores [35]. O fator γ ainda definirá o desconto de recompensas futuras, enquanto que o fator λ é responsável pelo balanceamento entre o viés e a variação.

2.5 Proximal Policy Optimization

Alguns problemas dos métodos de gradiente de política aparecem da taxa de aprendizado: se for muito pequena o processo de treinamento será muito lento, porém se for muito alta terá muita variabilidade no treinamento. A ideia central do algoritmo *Proximal Policy Optimization* (PPO) [36] é evitar grandes atualizações de políticas. A taxa de aprendizado restringe as atualizações relacionadas aos parâmetros, enquanto que o PPO leva em consideração as probabilidades das ações, utilizando uma proporção que indica a diferença entre uma nova política e uma política antiga (Equação 2-10). Como as probabilidades das ações que definem o comportamento da política, estabilizá-los entre as atualizações garante uma maior estabilidade [34].

$$r_t = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \quad (2-10)$$

Se $r_t(\theta) > 1$ então a ação é mais provável na política atual do que na política antiga, enquanto que se $r_t(\theta)$ está entre 0 e 1 a ação é menos provável na nova política do que na antiga. Entretanto, a ausência de uma restrição pode levar a grandes atualizações [36]. O que o PPO faz é tentar garantir pequenas alterações restringindo a razão de probabilidade das políticas, penalizando grandes passos que levam a políticas muito diferentes. Dessa maneira teremos duas proporções, uma sem restrição e uma restringida a um intervalo (entre $[1 - \epsilon, 1 + \epsilon]$, onde ϵ é um hiper-parâmetro). Então, pega-se o mínimo entre esses valores, sendo o objetivo final um limite inferior para a atualização da política, evitando grandes atualizações. A nova função objetivo fica definida como:

$$L^{CLIP}(\theta) = \mathbb{E}_t [\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)] \quad (2-11)$$

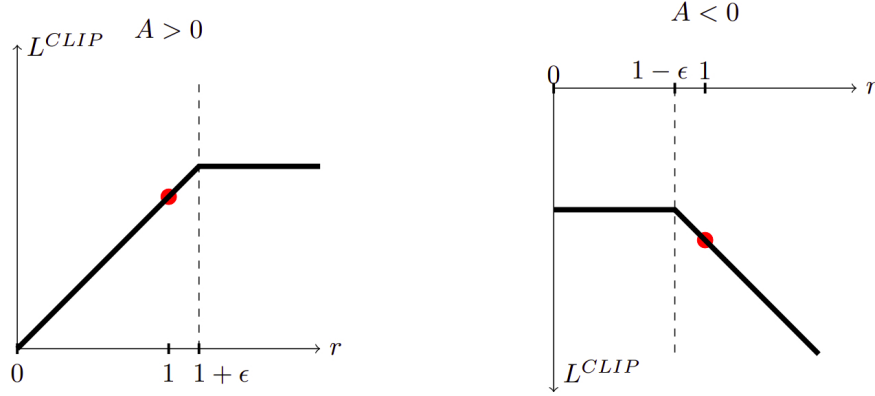


Figura 2.5: Gráficos da função L^{CLIP} . Valores em função da razão de probabilidade r , para vantagens positivas (esquerda) e vantagens negativas (direita) [36].

A nova função objetivo (Equação 2-11) reduz a vantagem estimada se a nova política estiver longe da antiga. Se uma ação é muito mais provável sob a nova política do que a antiga, não queremos exagerar na atualização da ação, então restringimos a função objetivo. Se a nova política for muito menos provável do que a antiga, novamente deve-se impedir atualizações exageradas.

O PPO tenta delimitar uma pequena região para atualizações, mas trata o problema como um método iterativo de otimização, onde é possível utilizar gradiente ascendente. O Algoritmo 2.1 representa o funcionamento do PPO com objetivo limitado [2]. Primeiro, avalia-se a política sobre um conjunto de dados, e então realiza-se melhorias aplicando atualizações de otimização da Equação 2-11. Durante o período de avaliação, a função de vantagem A^π da política atual π será estimada. Em seguida, é possível modificar a política atual executando atualizações com base na função de vantagem. Depois de gerar essa nova política, voltamos à avaliação. Esse ciclo ocorre até encontrarmos a função de vantagem e política ideais.

Algoritmo 2.1: *Proximal Policy Optimization* com objetivo limitado

Entrada: parâmetros da política inicial θ_0 , limiar de corte ϵ

para $k = 0, 1, 2, \dots$ **faça**

 Colete um conjunto de trajetórias D_k com política $\pi_k = \pi(\theta_k)$

 Estime a função de vantagem $A_t^{GAE(\gamma, \lambda)}$

 Calcule a atualização da política

$$\theta_{k+1} = \arg \max_{\theta} L_{\theta_k}^{CLIP}(\theta)$$

 executando N etapas do gradiente ascendente, onde

$$L_{\theta_k}^{CLIP}(\theta) = \mathbb{E}_t [\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)]$$

Metodologia

Este trabalho tem como proposta utilizar os conceitos antecedentes e as ferramentas apresentadas na seção seguinte para construção e treinamento de um agente de aprendizado por reforço, a fim de que se possa analisar sua capacidade de abstrair informações, generalizando o conhecimento obtido de uma fase de treinamento e sendo capaz de atuar em um ambiente com elementos não vistos anteriormente. Para isso, neste Capítulo será apresentada uma descrição do projeto realizado, fazendo uso das técnicas descritas no Capítulo 2 aplicadas ao ambiente simulado *General Video Game Artificial Intelligence*, para a análise da capacidade de generalização do algoritmo *Proximal Policy Optimization*. Ao final, são descritos os experimentos realizados para validação da proposta.

3.1 *General Video Game Artificial Intelligence*

Recentemente, pesquisadores começaram a explorar sistematicamente a generalização em aprendizado por reforço, desenvolvendo novos ambientes simulados que permitem criar uma distribuição de Processo de Decisão de Markov e dividir instâncias únicas de treinamento e teste, buscando fugir das limitações de ambientes baseados em jogos existentes. Tais limitações são decorrentes da ausência de disponibilidade de avaliação do agente em jogos nos quais não foi treinado, limitações essas que impedem a avaliação do aspecto de generalidade do algoritmo utilizado. Com base nisso foi desenvolvida uma *framework* e competição de mesmo nome: o *General Video Game Artificial Intelligence* (GVGAI) [41, 25].

A *framework* trabalha com a ideia de medir o desempenho do agente em um grande número de ambientes, que poderiam ser jogos amostrados de um espaço de jogo [33]. Os ambientes, nesse caso, são definidos por meio de uma Linguagem de Descrição de *Video Games* (VGDL, do inglês *Video Game Description Language*) [32], que permite facilmente customizar jogos e níveis, possibilitando a criação de variantes dos jogos em pouco tempo [26]. Na competição de aprendizado do GVGAI, os competidores possuem a sua disposição um conjunto de três fases de um jogo a ser utilizado no treinamento de um

agente inteligente. A validação do agente é realizada sobre outras duas fases do mesmo jogo, e os resultados obtidos nesses níveis de validação são os utilizados na competição para classificar as entradas.

A estrutura do GVGAI é conectada através da interface do Gym (GVGAI_GYM¹), que consiste em um conjunto de ferramentas que permite desenvolver e comparar algoritmos de aprendizado por reforço [3]. Essa ferramenta permite flexibilidade, visto que não faz suposições sobre a estrutura do agente. Os ambientes providos retornam os valores de observação e recompensa ao informar a ação selecionada, além de informar também sobre o término de um episódio, seja porque o agente cumpriu o seu objetivo ou está incapacitado de cumpri-lo. Na Figura 3.1 está ilustrado o fluxo de funcionamento da interface GYM em relação ao ambiente e ao agente.

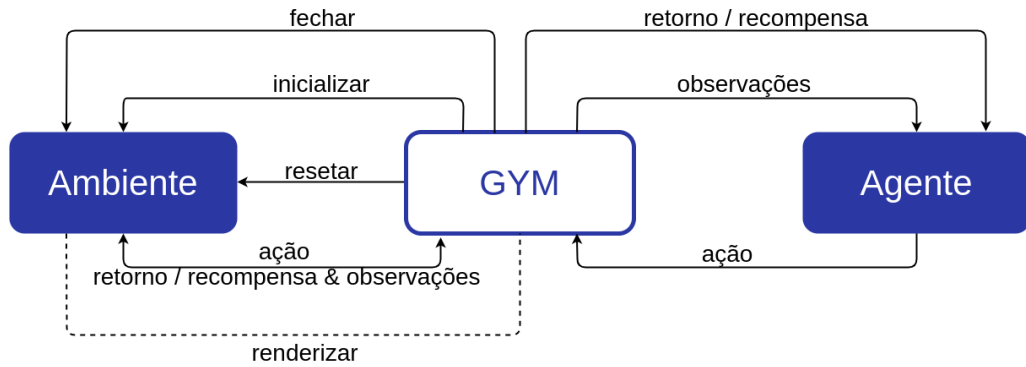
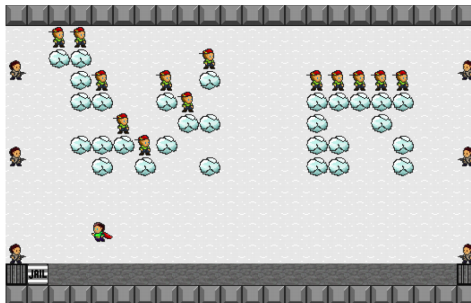


Figura 3.1: Fluxo de funcionamento da ferramenta GYM.

Os jogos disponíveis no GVGAI_GYM são jogos bidimensionais do tipo arcade. Jogos de arcade, geralmente, possuem níveis de curta duração, controles simples e uma dificuldade crescente, exigindo bons reflexos e pensamento estratégico dos seus jogadores. A Figura 3.2(a) é referente ao jogo *Superman* e a 3.2(a) ao jogo *Wait For Breakfast*, ambos disponíveis no GVGAI_GYM.



(a) *Superman*



(b) *Wait For Breakfast*

Figura 3.2: Exemplos de jogos disponíveis no GVGAI_GYM [41].

¹<https://github.com/rubenrtorradogvgaigym>

Para os testes da proposta desenvolvida neste trabalho, serão utilizados os jogos *Aliens*, *Boulder Dash* e *Missile Commands*. No primeiro jogo, *Aliens* (Figura 3.3), as principais tarefas são evitar projéteis inimigos recebidos e disparar no momento certo para atingir o inimigo. Todos os personagens não jogáveis e projéteis se comportam de maneira determinística (projéteis inimigos são disparados estocasticamente, mas leva algum tempo para alcançar o jogador).

Já no jogo *Boulder Dash* (Figura 3.4), o jogador deve explorar cavernas, coletando diamantes e chegando a uma saída dentro de um prazo, evitando criaturas perigosas e obstáculos. *Boulder Dash* exige que seu jogador possua reações rápidas para evitar quedas de pedras, e planejamento estratégico a longo prazo para escavar terra e coletar diamantes não ficando preso entre as pedras.

Por fim, no jogo *Missile Command* (Figura 3.5), cidades estão sendo atacadas por mísseis e o jogador deve destruí-los utilizando os canhões disponíveis. Os mísseis disparados pelo jogador explodem ao atingir a mira, deixando uma bola de fogo que persiste por alguns segundos e destruindo quaisquer mísseis inimigos que entrem nela. Um recompensa positiva é recebida para cada míssil inimigo destruído, e uma recompensa negativa para cada canhão destruído.

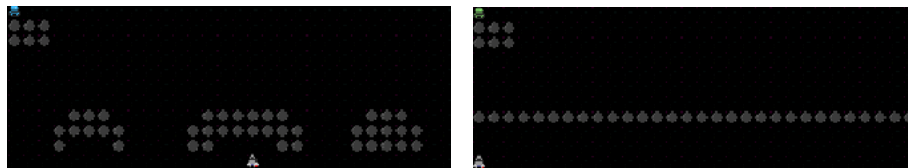


Figura 3.3: Captura de tela do jogo *Aliens*.

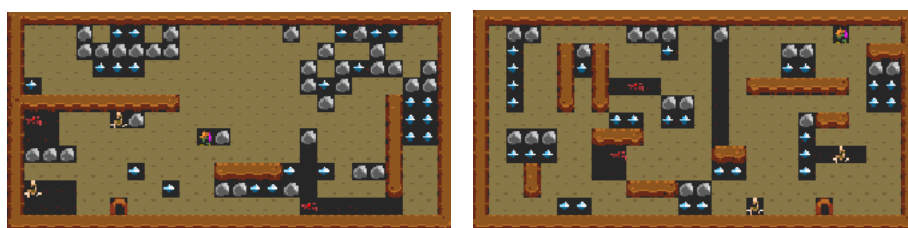


Figura 3.4: Captura de tela do jogo *Boulder Dash*.

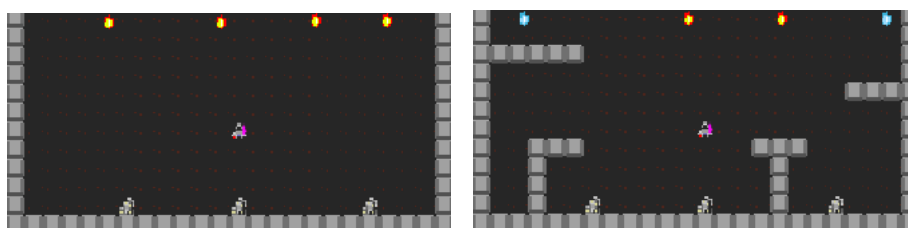


Figura 3.5: Captura de tela do jogo *Missile Command*.

3.2 Algoritmo de Treinamento

A arquitetura do agente de aprendizado por reforço consiste em duas partes: o algoritmo *Proximal Policy Optimization* e o ambiente de simulação fornecido pelo GVGAI_GYM. Os ambientes do GVGAI_GYM se diferenciam em dois aspectos importantes para o agente: o tamanho do espaço de ações disponíveis e o tamanho da imagem do jogo. As imagens são referentes aos estados observáveis do jogo que são passadas para o agente. Para obter dados mais relevantes das observações e tornar o algoritmo mais computacionalmente eficiente, é realizado um pré-processamento nas imagens de entrada. O processo consiste em eliminar as cores presentes na imagem, transformando-a em uma escala de cinza, e redimensionando o tamanho das imagens para o tamanho padrão de 84×84 pixels. A Figura 3.6 apresenta a saída do pré-processamento feito sobre uma imagem do jogo. Por fim, é combinado várias imagens em uma única entrada, fazendo com que o modelo escolha ações com base em uma série de quadros em vez de apenas um único quadro. Isso permite ao modelo discernir coisas como a direção do personagem ou sua velocidade.

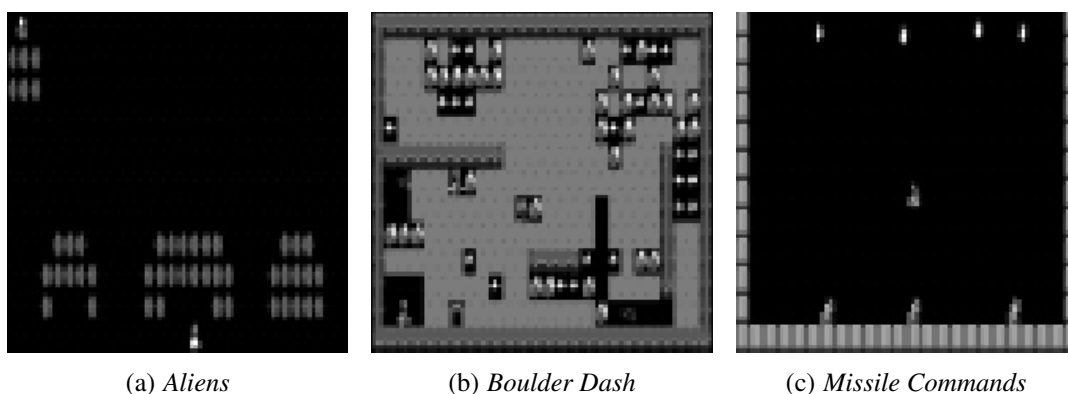


Figura 3.6: Resultado do pré-processamento das imagens.

As imagens pré-processadas alimentam duas redes neurais, uma chamada de Ator e outra chamada de Crítico. O modelo Ator é responsável pelo aprendizado de qual ação selecionar a partir de um estado observado do ambiente. Aqui, as imagens do jogo, já pré-processadas, são passadas para uma Rede Neural Convolucional (CNN do inglês *Convolutional Neural Network*) [15], que gera como saída uma ação, como andar ou atirar.

As ações previstas pelo Ator são passadas para o ambiente do GVGAI_GYM, que é responsável pela execução da ação no jogo. A ação executada gera uma recompensa que é calculada e retornada juntamente com o próximo estado do jogo. O modelo Crítico deve aprender a avaliar o estado atual, calculando o quão bom o estado é com base em sua recompensa através da função de valor (Função 2-2). Todo esse processo é repetido por uma quantidade finita de passos, em um processo de coleta de trajetórias. Ao final

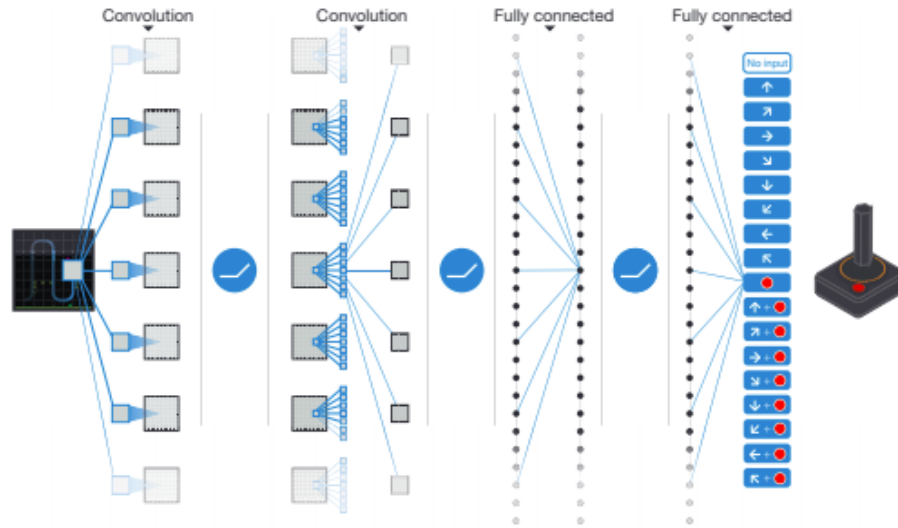


Figura 3.7: Ilustração esquemática de uma rede neural convolucional [20].

da coleta de trajetórias o valor da função de vantagem (Função 2-9) é calculado, e os parâmetros do Ator e do Crítico são otimizados com base no Algoritmo 2.1, *Proximal Policy Optimization*. O fluxo de treinamento Ator-Crítico é mostrado na Figura 3.8. As configuração de hiperparâmetros do algoritmo utilizado nos experimentos pode ser conferida no Apêndice A, e a implementação do algoritmo pode ser encontrada no Github².

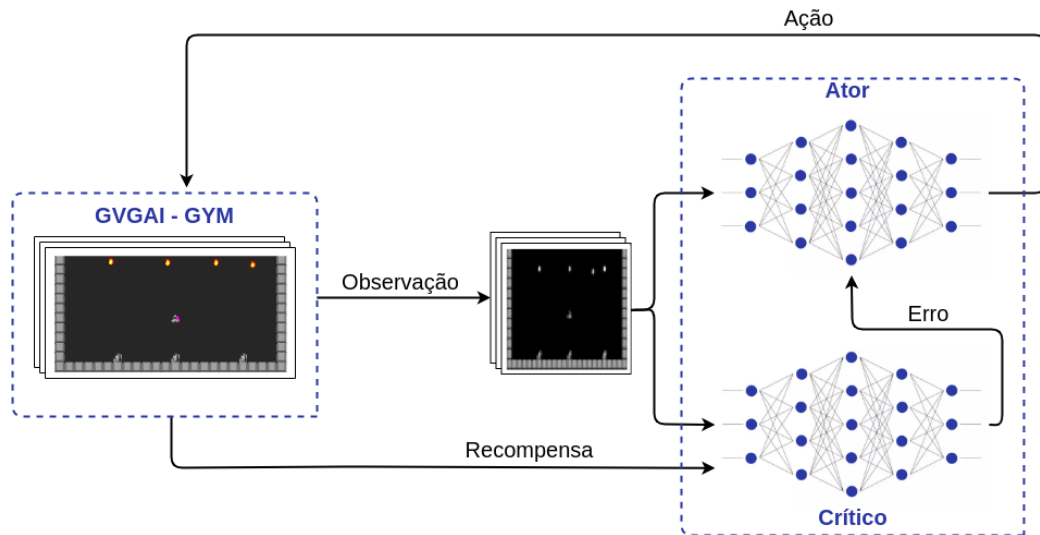


Figura 3.8: Fluxo de treinamento Ator-Crítico.

²<https://github.com/luanagbmartins/general-game-playing>

3.3 Testes

Todos os três jogos apresentados na seção anterior são divididos em cinco fases, ou níveis, que serão por sua vez divididos entre conjuntos de treino e teste. Todos os níveis de cada jogo são obtidos da mesma distribuição, portanto a diferença entre o desempenho do conjunto de treinamento e de teste determina o quão super ajustado o agente ficou ao conjunto de treinamento. A medida que o número de níveis de treinamento disponíveis aumenta, espera-se que o desempenho no conjunto de testes melhore, mesmo quando os agentes são treinados para um número fixo de iterações. Para todos os jogos, o treinamento foi limitado a 3 milhões de iterações distribuídas paralelamente entre 8 processos. Cada processo foi responsável por um ambiente do conjunto de treinamento. Os testes de cada jogo foram divididos entre conjuntos de treinamento com uma, duas e três fases. As fases não utilizadas em treinamento irão compor o conjunto de testes utilizados na fase de avaliação do agente.

Resultados

Neste capítulo serão apresentados e analisados todos os resultados obtidos do treinamento do *Proximal Policy Optimization* (PPO) para jogar os jogos disponibilizados pela ferramenta GVGAI_GYM, *Aliens*, *Boulder Dash* e *Missile Command*. Os jogos selecionados variam em termos de recompensas que eles oferecem, e as recompensas disponíveis possuem um grande impacto no sucesso do aprendizado por reforço. Para cada um dos jogos, foi modificado o tamanho do conjunto de treinamento, ou seja, a quantidade de fases disponíveis para treinamento, sendo divididos em conjuntos com um nível de treinamento (PPO1), dois níveis de treinamento (PPO2) e três níveis de treinamento (PPO3).

Cada fase de um jogo difere em sua dificuldade, definindo o quão desafiante é o jogo. O nível de dificuldade varia na adição ou remoção de eventos desafiadores ou obstáculos para alcançar o objetivo final. A medida que um jogador avança nas fases disponíveis, ele esbarra em desafios cada vez mais complexos, exigindo mais planejamento estratégico e reações rápidas para ser capaz de lidar com as adversidades.

Durante o treinamento do agente uma política é selecionada para interagir com o ambiente em um número fixo de etapas, com o objetivo de coletar um conjunto de experiências. Essas experiências são então utilizadas para atualizar o modelo. Para cada lote de experiências é calculado uma média das recompensas totais dos últimos dez episódios concluídos. A cada dez atualizações do modelo o agente é submetido a uma avaliação de desempenho nos ambientes de testes. A Figura 4.4 contém as informações suavizadas¹ de média das recompensas obtidas pelo agente durante o treinamento (cores mais escuras) e uma média das recompensas de dez episódios da fase de avaliação (cores mais claras).

A medida que o número de interações com o ambiente aumenta é esperado que as recompensas durante o treinamento também aumentem: cada atualização deve levar a políticas que maximizem cada vez mais a recompensa obtida. Se o treinamento chegar

¹O filtro de Savitzky-Golay pela biblioteca Scipy foi utilizado como uma janela de tamanho 53 e ordem polinomial 3.

a tal ponto em que a cada atualização não é observado uma mudança significativa dos valores de retorno, como visto na Figura 4.4(d) para o PPO1, então diz-se que foi atingido um máximo local do problema de otimização da política.

Jogos	Agente Aleatório	PPO 1	PPO 2	PPO 3	DQN	A2C
Aliens	38	62.7	62.7	63.5	75	77
Boulder Dash	1.69	19.8	19.8	20	2.5	15.5
Missile Command	-1.46	5	11.6	10.3	5	5

Tabela 4.1: Comparação das recompensas obtidas para cada jogo.

Na Tabela 4.1 é possível comparar os melhores resultados de treinamento de cada jogo com os melhores resultados dos algoritmos de aprendizado por reforço *Deep Q-Network* (DQN) [19] e *Advantage Actor Critic* (A2C) [43] obtidos de Torrado et.al. [41], e de um agente com política aleatória. Em ambos *Boulder Dash* e *Missile Command* o algoritmo PPO se sobressaiu aos demais, obtendo pior desempenho apenas no jogo *Aliens*.

Apesar de não ter ultrapassado os outros algoritmos de aprendizado por reforço (Tabela 4.1), o agente no jogo *Aliens* conseguiu aprender a jogar o jogo razoavelmente bem no conjunto de treinamento. Todos os personagens não-jogáveis e projéteis deste jogo se comportam de maneira determinística e o jogo pode ser jogado bem com muito pouco planejamento, o que pode levar o agente a explorar brechas de posicionamento dos jogos de treinamento. Uma vez que o cenário sofre mudanças entre as fases, o pouco planejamento pode resultar em um superajustamento do conjunto de treinamento. Apesar dos agentes aprenderem a jogar o jogo de maneira satisfatória, falham em conseguir generalizar seu objetivo, não conseguindo obter desempenho satisfatório no conjunto de teste, como pode ser observado na Figura 4.4(b).

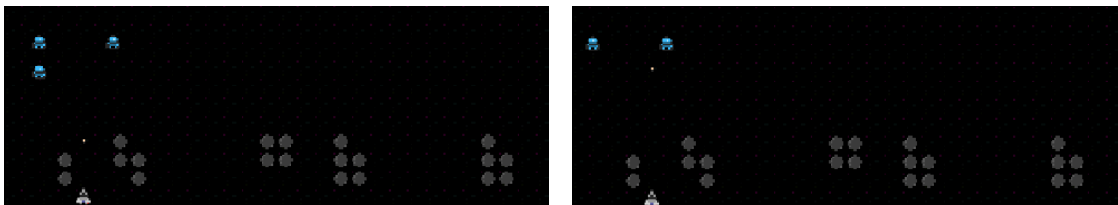


Figura 4.1: Exemplos de execução do jogo *Aliens*.

No *Boulder Dash*, quando o jogador coleta um diamante por pontos, uma pedra pode cair em sua na direção. Isso significa que existe um retorno negativo se um agente coleta um diamante e não se mover. Não coletar nenhum diamante e sobreviver parece

uma ótima solução local de que os agentes têm dificuldade em escapar. Entretanto, todos os três agentes conseguiram boas pontuações durante o treinamento (Tabela 4.1), indicando sucesso em conseguir contornar essa dificuldade. Apesar de conseguir aprender bem essa mecânica do jogo, observando a Figura 4.4(c) o agente falha em conseguir abstrair o objetivo geral, ficando limitado no aprendizado de um conjunto de ações específicas para o ambiente em que foi treinado.

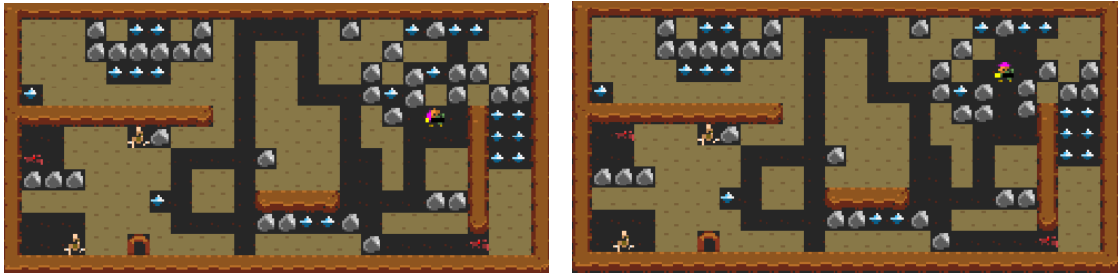


Figura 4.2: Exemplos de execução do jogo *Boulder Dash*.

Por fim, o *Missile Command* parece ser possível de se jogar simplesmente aproximando-se dos mísseis mais próximos e os atacando. Entretanto é necessário um tempo de reação hábil para conseguir defender todos os canhões a tempo. As recompensas totais de cada fase variam conforme o número de canhões e mísseis inimigos também mudam, gerando uma distribuição de recompensas não linear. Na primeira fase do jogo, por exemplo, há quatro bolas de fogo e três canhões, levando a um total de oito pontos de recompensa. No PPO1 o agente não parecia ser capaz de manter uma pontuação perfeita pois alguns erros levaram a cinco pontos. Entretanto, em ambos PPO2 e PPO3 (Tabela 4.1), os agentes foram capazes de melhorar sua performance, atingindo pontuação máxima na primeira fase do jogo. Apesar da variedade de exemplos no conjunto de treinamento ter levado a uma melhor exploração do conjunto de ação-estado, assim como nos outros jogos, é possível observar na Figura 4.4(d) que os agentes ficaram restritos a um conjunto de sequência de ações que levaram ao sucesso nos jogos de treinamento.

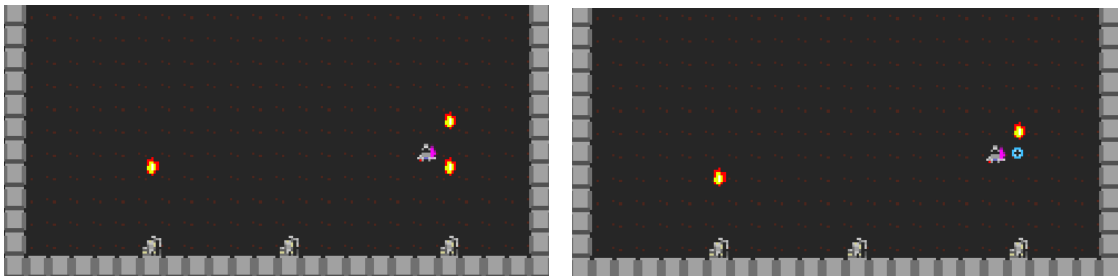


Figura 4.3: Exemplos de execução do jogo *Missile Command*.

Na Figura 4.5 é possível observar o desempenho médio de cada agente treinado, ao longo das cinco fases dos três jogos, em comparação a um agente com política aleatória. Assim como no jogo *Missile Command*, todos os agentes foram direcionados a uma melhor exploração do espaço de ações a medida que aumentava a variedade da amostragem do conjunto de treinamento, levando a um aumento de desempenho geral ao longo das cinco fases de cada jogo. No entanto, nas fases que não foram utilizadas para compor o conjunto de treinamento, o desempenho se manteve muito próximo do agente com política aleatória.

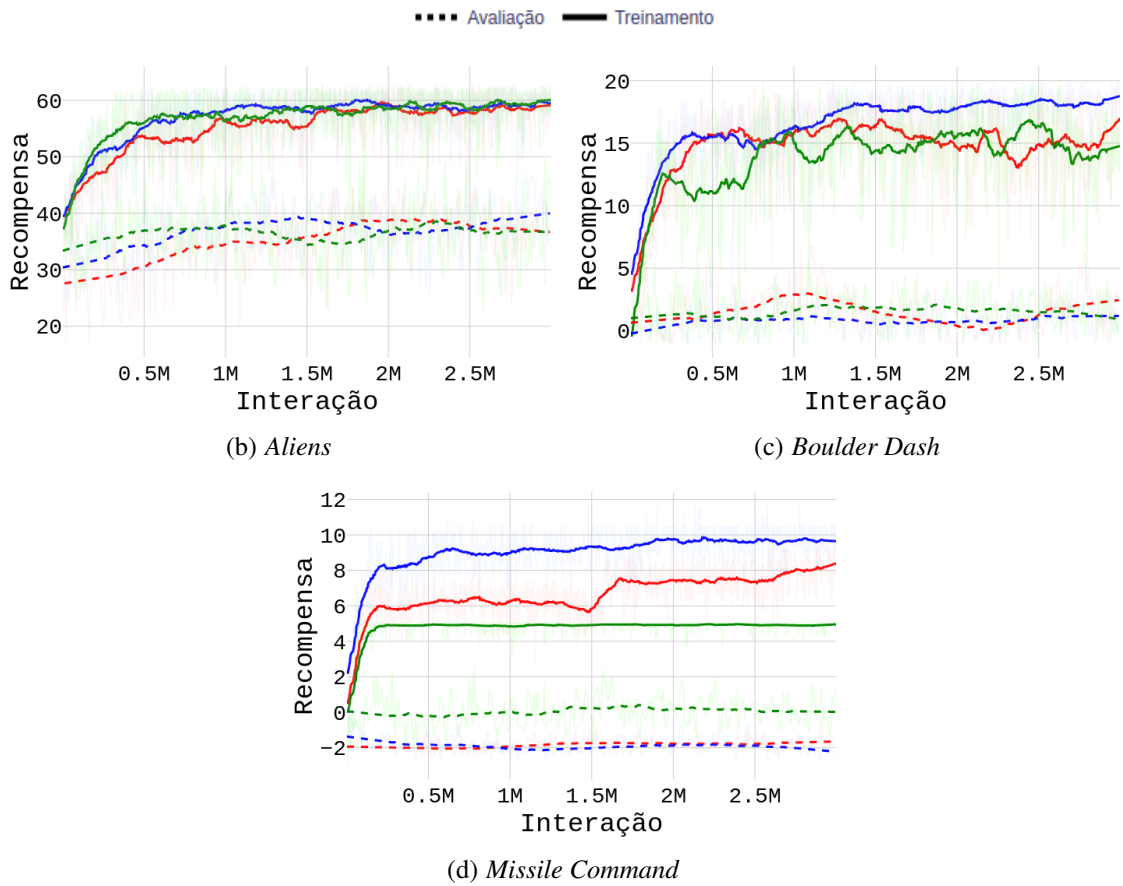


Figura 4.4: Gráfico de relação treinamento-avaliação. Em vermelho, conjunto de treinamento com três fases (PPO3); em azul, conjunto de treinamento com duas fases (PPO2); em verde, conjunto de treinamento com uma fase (PPO1).

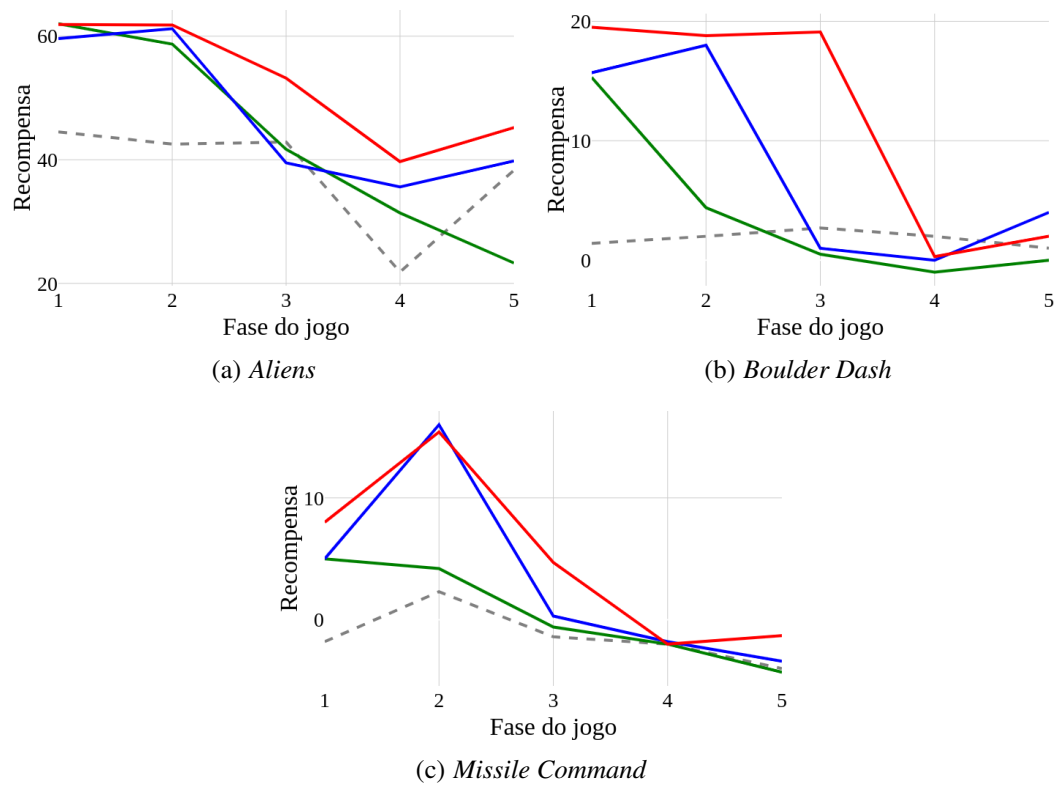


Figura 4.5: Gráfico de recompensa dos modelos treinados. Em vermelho, conjunto de treinamento com três fases (PPO3); em azul, conjunto de treinamento com duas fases (PPO2); em verde, conjunto de treinamento com uma fase (PPO1); em linhas tracejadas, agente com política aleatória.

Conclusão

Neste trabalho foram estudados os conceitos de aprendizado de máquina, em especial o aprendizado por reforço. Dentre as diferentes técnicas que podem ser utilizadas para realizar a tarefa de aprendizado por reforço, as abordagens baseadas em redes neurais profundas têm apresentado resultados satisfatórios no aprendizado de problemas complexos de alta dimensão. Dentre essas técnicas, destaca-se o algoritmo *Proximal Policy Optimization* (PPO), que obteve melhor desempenho geral em comparação a outros algoritmos de aprendizado por reforço.

Apesar dos grandes avanços realizados durante os últimos anos no desenvolvimento de algoritmos que aprendem tipos específicos de problemas, os agentes falham em generalizar entre tarefas, o que se mostrou não ser diferente para o algoritmo PPO. Contudo, os ambientes de avaliação de aprendizado por reforço insistem em avaliar o desempenho dos algoritmos nos mesmos ambientes de treinamento. Quando é divulgado que um algoritmo foi capaz de aprender uma política capaz de jogar bem um jogo, pode significar, na verdade, que a política encontrou ações ideais para o espaço de observações que o jogo fornece.

Nos testes realizados foram observados que os agentes tendem a explorar o determinismo do ambiente, ignorando os estados e memorizando sequências de ação efetivas que os levam a atingir pontuações cada vez maiores, sem aprender conceitos gerais do jogo. Esses agentes sofrem de sobre-ajuste ao conjunto de treinamento, sendo sensíveis a pequenas perturbações nos ambientes e incapazes de generalizar para estados não vistos anteriormente. Entretanto, foi possível notar que conjuntos de treinamento com uma maior variedade contribuem para uma melhor exploração da política, levando a resultados melhores em comparação a conjuntos de treinamento com baixa variedade. Acredita-se que esse aumento no desempenho do treinamento provém de um currículo implícito fornecido por um conjunto diversificado de fases.

Em um ambiente de aprendizado supervisionado, a generalização é obtida treinando um modelo em um grande conjunto de dados, geralmente com milhares de exemplos. A introdução de fases geradas proceduralmente, encorajadas pelo GVGAI_GYM, permite que os comportamentos treinados generalizem para fases não vistas na distribui-

ção do treinamento. Entretanto, os agentes sofrem para generalizar a partir de poucas fases de treinamento disponíveis.

De maneira equivalente ao aprendizado supervisionado, os algoritmos de aprendizado por reforço devem alcançar uma boa generalização se muitas variações dos ambientes forem usadas durante o treinamento. Trabalhos recentes exploram essa preocupação gerando proceduralmente enormes conjuntos de treinamento e teste para avaliar melhor a capacidade de generalização dos algoritmos [5, 6], reforçando que algoritmos de aprendizado por reforço ainda são ineficientes em termos de amostragem.

Como trabalhos futuros são propostas algumas direções com foco em uma melhor eficiência de amostragem, buscando diminuir a necessidade de grandes conjuntos de treinamento. Maiores investigações devem ser realizadas do impacto de diferentes arquiteturas e formas de regularização, como sugerido em [6]. Além disso, [4, 24] fazem uso de modelos de dinâmica inversa afim de melhorar a extração de características da observação, de forma que características mais importantes para a decisão do agente tenham uma maior importância. Isso pode ser explorado no contexto de generalização por poder desencorajar a memorização de sequência de ações efetivas.

Referências Bibliográficas

- [1] ABRAMSON, C. **Problems of teaching the behaviorist perspective in the cognitive revolution.** *Behavioral sciences (Basel, Switzerland)*, 3:55–71, 03 2013.
- [2] ACHIAM, J. **Advanced policy gradient methods.** Disponível para visualização em: http://rail.eecs.berkeley.edu/deeprlcourse-fal7/f17docs/lecture_13_advanced_pg.pdf . Acessado em 15 nov. 2019.
- [3] BROCKMAN, G.; CHEUNG, V.; PETTERSSON, L.; SCHNEIDER, J.; SCHULMAN, J.; TANG, J.; ZAREMBA, W. **Openai gym**, 2016.
- [4] BURDA, Y.; EDWARDS, H.; PATHAK, D.; STORKEY, A.; DARRELL, T.; EFROS, A. A. **Large-scale study of curiosity-driven learning**, 2018.
- [5] COBBE, K.; HESSE, C.; HILTON, J.; SCHULMAN, J. **Leveraging procedural generation to benchmark reinforcement learning**, 2019.
- [6] COBBE, K.; KLIMOV, O.; HESSE, C.; KIM, T.; SCHULMAN, J. **Quantifying generalization in reinforcement learning.** *CoRR*, abs/1812.02341, 2018.
- [7] **Deep learning book.** Disponível para visualização em: <http://www.deeplearningbook.com.br> . Acessado em 10 nov. 2019.
- [8] GAMA, J.; FACELI, K.; LORENA, A.; DE CARVALHO, A. **Inteligência artificial: uma abordagem de aprendizado de máquina.** Grupo Gen - LTC, 2011.
- [9] HAYKIN, S. **Neural Networks: A Comprehensive Introduction.** Prentice Hall, 1999.
- [10] HUI, J. **RI: Introduction to deep reinforcement learning.** Disponível para visualização em: https://medium.com/@jonathan_hui/rl-introduction-to-deep-reinforcement-learning-35c25e04c199 . Acessado em 15 nov. 2019.
- [11] HUI, J. **RI: Proximal policy optimization (ppo) explained.** Disponível para visualização em: https://medium.com/@jonathan_hui/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12 . Acessado em 15 nov. 2019.

- [12] JULIANI, A.; KHALIFA, A.; BERGES, V.-P.; HARPER, J.; TENG, E.; HENRY, H.; CRESPI, A.; TOGELIUS, J.; LANGE, D. **Obstacle tower: A generalization challenge in vision, control, and planning**. In: Kraus, S., editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, p. 2684–2691. ijcai.org, 2019.
- [13] JUSTESEN, N.; BONTRAGER, P.; TOGELIUS, J.; RISI, S. **Deep learning for video game playing**. *CoRR*, abs/1708.07902, 2017.
- [14] KRAPFL, J. **Behaviorism and society**. *The Behavior Analyst*, 39, 04 2016.
- [15] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **Imagenet classification with deep convolutional neural networks**. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12*, p. 1097–1105, USA, 2012. Curran Associates Inc.
- [16] LAPAN, M. **Deep Reinforcement Learning Hands-On: Apply Modern RL Methods, with Deep Q-networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More**. Packt Publishing, 2018.
- [17] LILLICRAP, T. P.; HUNT, J. J.; PRITZEL, A.; HEESS, N.; EREZ, T.; TASSA, Y.; SILVER, D.; WIERSTRA, D. **Continuous control with deep reinforcement learning**, 2015.
- [18] 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.
- [19] MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLU, I.; WIERSTRA, D.; RIEDMILLER, M. A. **Playing atari with deep reinforcement learning**. *CoRR*, abs/1312.5602, 2013.
- [20] MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A.; VENESS, J.; BELLEMARE, M.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A.; OSTROVSKI, G.; PETERSEN, S.; BEATTIE, C.; SADIK, A.; ANTONOGLU, I.; KING, H.; KUMARAN, D.; WIERSTRA, D.; LEGG, S.; HASSABIS, D. **Human-level control through deep reinforcement learning**. *Nature*, 518:529–33, 02 2015.
- [21] NASSIF, A. B.; SHAHIN, I.; ATTILI, I.; AZZEH, M.; SHAALAN, K. **Speech recognition using deep neural networks: A systematic review**. *IEEE Access*, 7:19143–19165, 2019.
- [22] NICHOL, A.; PFAU, V.; HESSE, C.; KLIMOV, O.; SCHULMAN, J. **Gotta learn fast: A new benchmark for generalization in rl**. *CoRR*, abs/1804.03720, 2018.

- [23] OTTER, D. W.; MEDINA, J. R.; KALITA, J. K. **A survey of the usages of deep learning in natural language processing.** *CoRR*, abs/1807.10854, 2018.
- [24] PATHAK, D.; AGRAWAL, P.; EFROS, A. A.; DARRELL, T. **Curiosity-driven exploration by self-supervised prediction.** *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul 2017.
- [25] PÉREZ-LIÉBANA, D.; LIU, J.; KHALIFA, A.; GAINA, R. D.; TOGELIUS, J.; LUCAS, S. M. **General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms.** *CoRR*, abs/1802.10363, 2018.
- [26] PEREZ-LIEBANA, D.; LUCAS, S. M.; GAINA, R. D.; TOGELIUS, J.; KHALIFA, A.; LIU, J. **General Video Game Artificial Intelligence**, volume 3. Morgan & Claypool Publishers, 2019. <https://gaigresearch.github.io/gvgaibook/> .
- [27] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning Representations by Back-propagating Errors.** *Nature*, 323(6088):533–536, 1986.
- [28] RUMMERY, G.; NIRANJAN, M. **On-line q-learning using connectionist systems.** *Technical Report CUED/F-INFENG/TR 166*, 11 1994.
- [29] SALIMANS, T.; HO, J.; 0022, X. C.; SUTSKEVER, I. **Evolution strategies as a scalable alternative to reinforcement learning.** *CoRR*, abs/1703.03864, 2017.
- [30] SAMUEL, A. L. **Some studies in machine learning using the game of checkers.** *IBM Journal of Research and Development*, 3, 1959.
- [31] SANTA MARIA, T.; DA SILVA CASTRO, T. **Controle e Simulação de Robôs Utilizando Topologia de Redes Neuroevolutivas - NEAT.** PhD thesis, Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais, 07 2018.
- [32] SCHAUL, T. **A video game description language for model-based or interactive learning.** In: *Proceedings of the IEEE Conference on Computational Intelligence in Games*, Niagara Falls, 2013. IEEE Press.
- [33] SCHAUL, T.; TOGELIUS, J.; SCHMIDHUBER, J. **Measuring intelligence through games.** *CoRR*, abs/1109.1314, 2011.
- [34] SCHULMAN, J.; LEVINE, S.; MORITZ, P.; JORDAN, M. I.; ABBEEL, P. **Trust region policy optimization.** *CoRR*, abs/1502.05477, 2015.
- [35] SCHULMAN, J.; MORITZ, P.; LEVINE, S.; JORDAN, M. I.; ABBEEL, P. **High-dimensional continuous control using generalized advantage estimation.** In:

- Bengio, Y.; LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- [36] SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. **Proximal policy optimization algorithms**. *CoRR*, abs/1707.06347, 2017.
- [37] SEITA, D. **Notes on the generalized advantage estimation paper**. Disponível para visualização em: <https://danieltakeshi.github.io/2017/04/02/notes-on-the-generalized-advantage-estimation-paper/>. Acessado em 16 nov. 2019.
- [38] SILVER, D.; ET AL; HASSABIS, D. **Mastering the game of go with deep neural networks and tree search**. *Nature*, 529,:484–489, Jan. 2016.
- [39] SILVER, D. **Deep reinforcement learning**. Disponível para visualização em: <https://deepmind.com/blog/article/deep-reinforcement-learning>. Acessado em 13 nov. 2019.
- [40] SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. The MIT Press, Cambridge, MA, 1998.
- [41] TORRADO, R. R.; BONTRAGER, P.; TOGELIUS, J.; 0001, J. L.; PÉREZ-LIÉBANA, D. **Deep reinforcement learning for general video game ai**. *CoRR*, abs/1806.02448, 2018.
- [42] WATKINS, C. J. C. H.; DAYAN, P. **Technical note Q-learning**. *Machine Learning*, 8:279, 1992.
- [43] WU, Y.; MANSIMOV, E.; LIAO, S.; RADFORD, A.; SCHULMAN, J. **Openai baselines: Acktr & a2c**. Disponível para visualização em: <https://openai.com/blog/baselines-acktr-a2c/>. Acessado em 02 nov. 2019.

Hiperparâmetros

Tabela A.1: *Hiperparâmetros para o algoritmo PPO.*

Hiperparâmetro	Valor
Iterações (total)	3×10^5
Taxa de aprendizado	2.5^{-4}
Coeficiente de entropia	0.01
Parâmetro ϵ de corte (PPO)	0.2
Tamanho do episódio	128
Tamanho do lote	32
γ	0.99
λ	0.95