Bruno Monteiro Belloni

Análise da Aplicação de Grid Mask no Aprendizado por Reforço com MuJoCo

Passo Fundo

2021

Bruno Monteiro Belloni

Análise da Aplicação de Grid Mask no Aprendizado por Reforço com MuJoCo

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO

Orientador: Prof. Dr. João Mário Lopes Brezolin
Coorientador: Prof. Me. Carlos Alexandre Silva dos Santos

Passo Fundo
2021

Resumo

O Aprendizado por Reforço é uma área que ganhou grande notoriedade com a publicação de um trabalho do *Google* publicado em 2013 em que foi aplicado AR no ambiente do *Atari* com êxito pela primeira vez. Desde então novos algoritmos foram apresentados e publicados por diversos pesquisadores de Inteligência Artificial.

No entanto, grande parte dos ganhos que os pesquisadores vinham obtendo em novos algoritmos foi estagnado por terem seu foco apenas no modelo. Com isso, começou-se a explorar outras áreas que envolvem o treinamento de agentes com Redes Neurais Artificiais. Portanto, ao aplicar aumento de dados artificiais no Aprendizado por Reforço, diversos autores obtiveram resultados expressivamente superiores em comparação aos trabalhos que realizavam ajustes apenas no modelo matemático.

Por mais que trabalhos que combinam técnicas de aumento de dados com Aprendizado por Reforço já tenham sido publicados, ainda há um vasto cenário a ser explorado. Com isso, este trabalho apresentou a análise da aplicação de *Grid Mask* no algoritmo *Deep Deterministic Policy Gradient* (DDPG) no cenário *Quadruped Walk* do ambiente de aprendizado *MuJoCo*.

Ao realizar os experimentos e treinamento do agente foi possível concluir que aplicar *Grid Mask* para resolver o problema do *Quadruped Walk* se mostrou um método extremamente eficaz e simples de ser implementado. Além do mais foi possível obter uma eficácia no processamento dos frames superior a de trabalhos análogos, onde obteve-se uma melhora no treinamento nas fases médias e finais do treinamento do agente.

Palavras-chave: Aprendizado por Reforço, Aprendizado Profundo, MuJoCo, Grid Mask

Abstract

Reinforcement Learning is an area that gained great notoriety with the publication of a Google work published in 2013 in which AR was successfully applied in the Atari environment for the first time. Since then, new algorithms have been presented and published by several Artificial Intelligence researchers.

However, much of the gains that researchers had been obtaining in new algorithms was stagnant because they focused only on the model. With that, it began to explore other areas that involve the training of agents with Artificial Neural Networks. Therefore, when applying artificial data increase in Reinforcement Learning, several authors obtained significantly superior results compared to works that performed adjustments only in the mathematical model.

Although works that combine data augmentation techniques with Learning by Reinforcement have already been published, there is still a vast scenario to be explored. Thus, this work presented the analysis of the application of Grid Mask in the Deep Deterministic Policy Gradient (DDPG) algorithm in the Quadruped Walk scenario of the MuJoCo learning environment.

When performing the experiments and training the agent, it was possible to conclude that applying the Grid Mask to solve the Quadruped Walk problem proved to be an extremely effective and simple method to be implemented. Furthermore, it was possible to obtain an efficiency in the processing of frames superior to that of analogous works, where it was obtained an improvement in the training in the middle and final phases of the agent's training.

Keywords: Reinforcement Learning, Deep Learning, MuJoCo, Grid Mask

Lista de ilustrações

Lista de abreviaturas e siglas

ADALINE	*ADAptive LINear Element*
ALE	*The Arcade Learning Environment*
API	*Application Programming Interface*
AR	Aprendizado por Reforço
CNN	*Convolutional Neural Network*
DQN	*Deep Q-Networks*
DDPG	*Deep Deterministic Policy Gradient*
DMC	*DeepMind Control Suite*
DRQ	*Deep-regularized Q*
GAN	*Generative adversarial network*
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
RAD	*Reinforcement Learning with Augmented Data*
RGB	*Red, Green, Blue*
SAC	*Soft Actor-Critic*
TD	*Temporal difference*
XML	*Extensible Markup Language*
WANDB	*Weights & Biases*

Lista de símbolos

ϵ	Letra grega Epsilon
λ	Lambda

Sumário

1 Introdução

Este trabalho irá abordar as principais técnicas da área de Aprendizado por Reforço com intuito de aplicar os algoritmos no ambiente de aprendizado do *MuJoCo*, mais precisamente no cenário *Quadruped Walk*, onde há a necessidade de tomada de decisões e controle do agente no emulador.

Serão analisados algoritmos de Aprendizado por Reforço que obtiveram sucesso ao treinar agentes para atuar em jogos digitais. Serão abordados trabalhos que aplicaram com exito técnicas de aumento de dados no treinamento de agentes.

Foi escolhido o aumento de dados *Grid Mask* para realização deste trabalho. Por ser uma técnica de aumento de dados simples de ser aplicada e replicável para qualquer problema de visão computacional, optou-se por realizar sua eficácia no ambiente *MuJoCo*.

A principal contribuição deste trabalho será analisar a aplicação da técnica de aumento de dados *Grid Mask* no algoritmo *Deep Deterministic Policy Gradient* (DDPG) no cenário *Quadruped Walk* do *MuJoCo*.

1.1 Objetivos

Desenvolver um agente inteligente que seja capaz de aprender a atuar no ambiente *MuJoCo* com a utilização de *Grid Mask* utilizando apenas os estados e recompensas disponibilizados pelo ambiente. O agente deverá aprender a atuar no ambiente apenas explorando-o, sem conhecimento prévio do cenário e dos obstáculos que serão descobertos. Os experimentos foram realizados no cenário *Quadruped Walk* do ambiente de aprendizado *MuJoCo*.

1.1.1 Objetivos Específicos

- Estudar e analisar técnicas e conceitos de Aprendizado por Reforço que serão utilizadas pelo agente inteligente;
- Revisar as principais técnicas do estado da arte utilizadas por agentes em Jogos Digitais;
- Treinar um agente capaz de aprender com base nos *pixels* brutos fornecidos pelo ambiente e nas recompensas numéricas atribuídas para cada estado;
- Analisar a aplicação de técnicas de aumento de dados artificiais em algoritmos da literatura;

- Comparar os resultados obtidos pelo agente implementado com resultados de trabalhos análogos encontrados na literatura.

1.2 Organização do Trabalho

O presente trabalho é constituído por capítulos, o que possibilita uma melhor organização e compreensão dos temas abordados. O Capítulo 2 apresenta o Referencial Teórico onde é abordado os principais conceitos tratados neste trabalho, entre eles: Aprendizado por Reforço, Redes Neurais Artificiais, Redes Neurais Convolucionais e o ambiente de desenvolvimento *MuJoCo*.

O Capítulo 3 aborda os principais Trabalhos Relacionados desta monografia. É abordado as *Deep Q-Networks* o trabalho que introduziu a combinação de Redes Neurais com *Q-Learning* para resolver o problema do Atari.

Além do mais, são abordados os principais trabalhos que relacionam Aumento de Dados Artificial e Aprendizado por Reforço. Entre eles: *Reinforcement Learning with Augmented Data* (RAD) e *Data-regularized Q* (DRQ). E por fim, os principais trabalhos que fundamentaram a seção de desenvolvimento e resultados: *Data-regularized Q V2* (DRQ-V2) e a técnica de aumento de dados *Grid Mask*.

No Capítulo 4 é explanado o funcionamento do algoritmo de Aprendizado por Reforço, assim como a combinação com *Grid Mask* que foi realizada. O Capítulo 5 aborda os resultados da aplicação de aumento de dados abordado por este trabalho no algoritmo DRQ-V2. E por fim, o Capítulo trata as principais considerações do trabalho, bem como sugestões de trabalhos futuros.

2 Referencial teórico

Nesta seção serão abordados os principais conceitos de Inteligência Artificial utilizados neste trabalho: Aprendizado por Reforço, Redes Neurais Artificiais e Redes Neurais Convolucionais. Também será relatado a respeito do ambiente de desenvolvimento utilizado para realização da pesquisa: *MuJoCo*.

2.1 Aprendizado por Reforço

Agentes inteligentes podem qualificar suas ações se implementarem a capacidade de aprendizado. Há diversas formas de se implementar o aprendizado de um agente: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço. Um agente inteligente adota a melhor ação possível após analisar o estado atual do ambiente.

Com o Aprendizado por Reforço (AR) o agente aprende a como mapear situações em ações, com objetivo de maximizar o sinal numérico de recompensa. No processo de aprendizado do agente não é informado ao agente quais ações devem ser executadas, mas em vez disso, ele deve descobrir quais ações geram maior recompensas ao experimentá-las por tentativa e erro (SUTTON; BARTO, 2018).

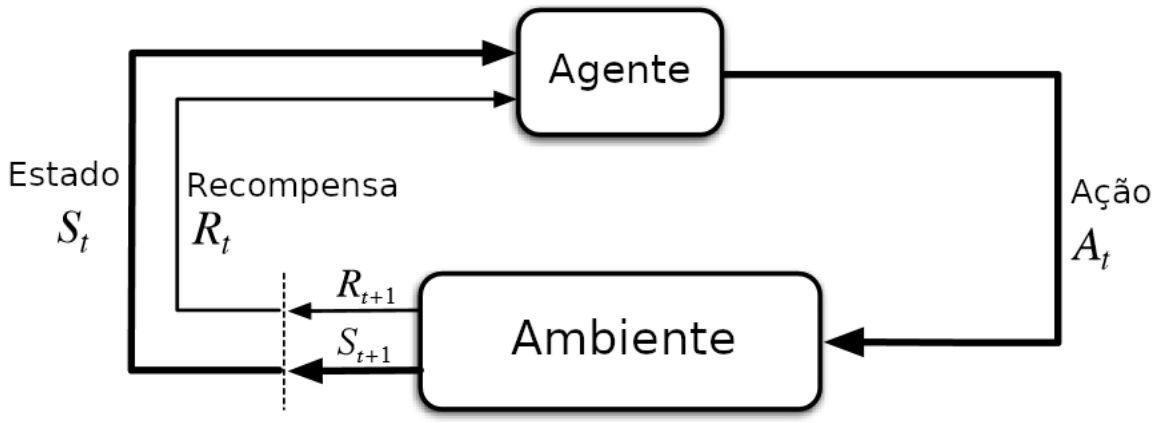
Há algoritmos de AR baseados em Modelos e livres de Modelos. A principal diferença entre os dois é que os algoritmos que utilizam modelos permitem que o agente planeje pensando no futuro, vendo o que aconteceria com uma gama de escolhas possíveis. Um exemplo famoso dessa categoria é o *AlphaZero* (SILVER et al., 2017a).

Embora os métodos sem modelo renunciem aos ganhos potenciais na eficiência da amostra com o uso de um modelo, eles tendem a ser mais fáceis de implementar e ajustar. Neste trabalho serão analisados algoritmos de AR Livres de Modelo. Na categoria de AR Livre de Modelos há duas categorias: Técnicas de Otimização de Política (RUMMERY; NIRANJAN, 1994) e técnicas de *Q-Learning* (WATKINS; DAYAN, 1992).

AR é o terceiro campo de Aprendizado de Máquina e fica entre entre o Aprendizado Supervisionado e a completa falta de rótulos predefinidos. Por um lado, ele usa muitos métodos bem estabelecidos de aprendizado supervisionado, como redes neurais profundas (*Deep Learning*) para aproximação de função, descida gradiente estocástica e retropropagação, para aprender a representação de dados. Por outro lado, geralmente os aplica de maneira diferente para cada problema (LAPAN, 2020).

Como é possível observar na Figura 1, após observar o estado (S_t) e a recompensa (R_t), a cada novo passo o agente toma uma decisão (A) no tempo (t) dentro do ambiente e

Figura 1 – Fluxograma Aprendizado por Reforço



Fonte: Google, 2021.

observa seu novo estado (S_{t+1}) e sua nova recompensa (R_{t+1}). Com a utilização de técnicas de IA mencionadas no trabalho, será efetuado esse ciclo diversas vezes, de forma que o agente aprenda quais ações o levam a maiores recompensas.

No Aprendizado por Reforço a Recompensa (R) é um valor escalar é que obtido como resposta a uma ação tomada pelo agente. Pode ser um valor positivo se a ação aplicada ao estado foi bem sucedida, ou um valor negativo caso a ação tomada não tenha efeito positivo no ambiente. O sinal de recompensa, portanto, define quais são os eventos bons e ruins para o agente (SUTTON; BARTO, 2018). Dessa forma, o propósito da recompensa é dizer ao agente o quão bem ele está atuando no ambiente.

Basicamente, o termo reforço vem do fato de que a recompensa obtida por um agente deve reforçar seu comportamento de forma positiva ou negativa (LAPAN, 2020). O objetivo do agente é obter a maior recompensa acumulada possível a partir de uma sequência de ações executadas no ambiente.

Outro formalismo bastante importante no aprendizado por reforço é o agente. O agente é basicamente algo ou alguém que interage com em um ambiente, na maioria dos cenários de AR, o agente pode ser interpretado como um robô que opera ações, ou um pedaço de software. Ao interagir com o ambiente, o agente recebe a observação resultante da ação aplicada e a recompensa que representa a qualidade da ação que foi tomada.

O agente também possui ações que permitem a alteração do ambiente. Cada ação executada pelo agente retorna uma recompensa e transforma o estado do ambiente. As ações de um agente podem ser Discretas ou Contínuas.

As ações são discretas quando possui um conjunto limitado pré definido, um exemplo de ambiente que possui ações discretas é o ALE (Bellemare et al., 2013), um *framework* que permite que pesquisadores desenvolvam agentes de IA para jogos do *Atari*

2600.

As ações são contínuas quando o agente precisa prever um valor do mundo real, como a força N que deve ser aplicada no motor de um veículo, um exemplo dessa categoria seria o ambiente *Mountain Car*, presente no ambiente de aprendizado *OpenAI Gym* (BROCKMAN et al., 2016), onde o agente precisa prever a quantidade de força que deve ser aplicada no carro para que ele chegue ao topo da montanha.

O ambiente é tudo com que o agente interage. A comunicação do agente com o ambiente é limitada apenas às observações (S), recompensas (R) e ações (A).

As observações (S) em AR representam o contexto atual em que o ambiente de aprendizado se encontra (SEWAK, 2019). Neste trabalho foi utilizado imagens RGB como fonte de observação do ambiente, mas poderia ser utilizado a posição no plano cartesiano juntamente com o ângulo do agente em ambientes de *grid*, como é comumente realizado em conjunto com o *Q-Learning* tabular.

As observações são informações que o ambiente fornece ao agente para dizer o que está acontecendo ao redor (LAPAN, 2020), no entanto a observação sem a presença do valor de recompensa não diz ao agente o quão bom foi a ação tomada.

A Política é o que define o comportamento do agente em um determinado momento do aprendizado (SUTTON; BARTO, 2018). A política também pode ser vista como o mapeamento dos estados observados no ambiente em ações a serem tomadas durante o estado. No início do aprendizado o agente executa uma política aleatória com intuito de explorar o ambiente e à medida que os pesos da rede neural vão sendo ajustados a política passa a se tornar mais assertiva.

A aprendizagem por reforço é conhecida por ser instável ou mesmo divergir quando um aproximador de função não linear, como uma rede neural, é usado para representar a função de valor de ação (também conhecido como Q) (TSITSIKLIS; ROY, 1997). Essa instabilidade tem várias causas: as correlações presentes na sequência de observações, o fato de que pequenas atualizações em Q podem alterar significativamente o aprendizado (MNIH et al., 2015).

2.2 Redes Neurais Artificiais

Uma rede neural é composta por vários neurônios artificiais (SEWAK, 2019) e foi originalmente modelada baseando-se no funcionamento de um neurônio biológico. A introdução da unidade lógica de limiar como um modelo de neurônio abstrato foi proposto por McCulloch e Pitts (MCCULLOCH; PITTS, 1943), e é considerado o início das redes neurais artificiais (RNAs).

A história das RNAs como métodos de aprendizagem para classificação e regressão

passou por vários estágios: Perceptron (ROSENBLATT, 1958) e ADALINE (WIDROW; HOFF, 1960). O estágio de retropropagação de erros proposto por LeCun (LECUN, 1986) e Rumelhart, Hinton e Williams (RUMELHART; HINTON; WILLIAMS, 1986). E o estágio de aprendizado profundo atual com ênfase no aprendizado de representação (BENGIO; COURVILLE; VINCENT, 2012) e (GOODFELLOW; BENGIO; COURVILLE, 2016).

Redes Neurais como aproximador de função para Aprendizado por Reforço foram combinadas primeiramente por Farley e Clark (FARLEY; CLARK, 1954), onde os autores utilizaram a aprendizagem por reforço para modificar os pesos das funções de limiar linear que representavam políticas. Posteriormente Widrow, Gupta e Maitra (WIDROW; GUPTA; MAITRA, 1973) apresentaram uma unidade de limiar linear semelhante a um neurônio, implementando um processo de aprendizagem que eles chamaram de aprendizagem com uma adaptação crítica ou *bootstrap* seletiva, é uma variante de aprendizagem por reforço do algoritmo ADALINE (WIDROW; HOFF, 1960).

Hinton e Williams (RUMELHART; HINTON; WILLIAMS, 1988) descreveram várias maneiras pelas quais a retropropagação e o aprendizado por reforço podem ser combinados para treinar RNAs. Gullapalli (GULLAPALLI, 1990) e Williams (WILLIAMS, 1992) desenvolveram algoritmos de aprendizagem por reforço para unidades semelhantes a neurônios com saídas contínuas, em vez de binárias. Barto e Sutton (SUTTON et al., 1998) argumentaram que as RNAs podem desempenhar papéis significativos para aproximar funções necessárias para resolver problemas de decisão sequencial.

O TD-Gammon da Tesauro (TESAURO, 1995) influentemente demonstrou as habilidades de aprendizagem do algoritmo TD (λ) com aproximação de função por RNAs de múltiplas camadas para aprender a jogar gamão.

Os programas *AlphaGo* (SILVER et al., 2016), *AlphaGo Zero* (HOLCOMB et al., 2018) e *AlphaZero* (SILVER et al., 2017b) usaram aprendizado por reforço com RNAs convolucionais profundas para alcançar resultados impressionantes com o jogo *Go*. Schmidhuber (SCHMIDHUBER, 2015) analisa as aplicações de RNAs na aprendizagem por reforço, incluindo aplicações de RNAs recorrentes.

2.3 Redes Neurais Convolucionais

As redes neurais convolucionais (*Convolutional Neural Networks*, CNN), são redes especializadas no processamento e análise de imagens digitais. Elas também são conhecidas como redes neurais artificiais invariantes a deslocamento ou invariantes no espaço. Seu funcionamento baseia-se em uma arquitetura em que pesos são compartilhados entre os núcleos de convolução ou filtros que deslizam ao longo dos recursos de entrada e fornecem respostas equivariantes de tradução conhecidas como mapas de recursos (LAPAN, 2020).

LeCun (LECUN et al., 1989) usaram retropropagação para aprender os coeficientes do *kernel* de convolução diretamente de imagens de números escritos à mão. O aprendizado era, portanto, totalmente automático, tinha um desempenho melhor do que o design de coeficiente manual e era adequado a uma ampla gama de problemas de reconhecimento de imagem e tipos de imagem. Essa abordagem se tornou a base da visão computacional moderna.

A camada de convolução é a parte principal de uma CNN. Ela possui um conjunto de filtros que são aprendidos durante o treinamento. Cada filtro é envolvido ao longo da largura e altura do volume de entrada, computando o produto escalar entre as entradas do filtro e a entrada, produzindo um mapa de ativação bidimensional desse filtro. Como resultado, a rede aprende filtros que são ativados quando detecta algum tipo específico de recurso em alguma posição espacial na entrada (SEWAK, 2019).

2.4 *MuJoCo*

MuJoCo (*Multi-Joint Dynamics with Contact*) (TODOROV; EREZ; TASSA, 2012) é uma *engine* física que tem como objetivo facilitar a pesquisa e o desenvolvimento de robótica, biomecânica, gráficos e animação, e outras áreas onde uma simulação rápida e precisa é necessária. Recentemente o ambiente de aprendizado por reforço *MuJoCo* ganhou notoriedade devido a diversos trabalhos científicos que exploram a interação de agentes nesse ambiente.

O *MuJoCo* é uma plataforma que requisita licença, apesar de fornecer o primeiro mês para testes gratuitos, a partir do segundo mês já é necessário uma licença completa. Apesar disso, é fornecida uma licença para estudantes.

Possui uma biblioteca dinâmica com API C/C++ que inclui um analisador XML, compilador de modelo, simulador e visualizador *OpenGL* (WOO et al., 1999) interativo. Apesar de ser escrita em C, a biblioteca *mujoco-py* permite usar *MuJoCo* do *Python* 3. Neste trabalho foi utilizada a implementação *DeepMind Control Suite* (TASSA et al., 2018) que fornece um ambiente de aprendizado por reforço amigável utilizando a *engine* física *MuJoCo*.

O DMC (TASSA et al., 2018) é um ambiente de controle contínuo e portanto não possui um ponto de parada como é visto nos ambientes do *Atari* 2600 (Bellemare et al., 2013). Por ser um ambiente de aprendizado infinito, os pesquisadores definem um número estático para avaliar os algoritmos de AR em ambientes contínuos.

O DMC possui um vasto número de ambientes. 24 destes ambientes foram classificados entre fácil, médio e difícil pelo trabalho DRQ-V2 (YARATS et al., 2021). Onde 9 destes ambientes são fáceis e são resolvidos em média em 1 milhão de *frames*. 12 são

classificados como de dificuldade média e são resolvidos em 3 milhões de *frames* em média. E por fim, 3 ambientes foram classificados como difícil e são resolvidos em 30 milhões de *frames* em média.

3 Trabalhos Relacionados

Nesta seção serão abordados os principais trabalhos que atingiram resultados satisfatórios ao executar algoritmos de Aprendizado por Reforço combinados com Redes Neurais Convolucionais. O primeiro artigo abordado foi o trabalho que introduziu as *Deep Q-Networks* e em seguida foram abordados alguns trabalhos que utilizaram Aumento de Dados para realizar o treinamento de um agente de Aprendizado por Reforço Profundo.

3.1 DQN

O trabalho que inspirou esta monografia foi publicado em 2013 por um grupo de pesquisa denominado *DeepMind*, que em 2014 foi vendido para a *Google*. Foi apresentado o primeiro modelo de aprendizado profundo que foi capaz de aprender com sucesso as políticas de controle diretamente da entrada sensorial de alta dimensão usando aprendizado por reforço (MNIH et al., 2013).

O método apresentado foi capaz de aprender por meio de imagens RGB que foram processadas por uma rede neural convolucional com intuito de extrair os atributos de interesse para uma outra rede neural totalmente conectada que realizou a regressão linear para aprimorar os valores de Q.

O sucesso de técnicas de aprendizado profundo capazes de extrair características de alto nível de dados sensoriais brutos como a *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) vencedora da competição *ImageNet* (RUSSAKOVSKY et al., 2015) em 2013 possibilitaram a realização do trabalho por parte dos pesquisadores do *DeepMind* (MNIH et al., 2013). Além do mais, os autores introduziram o conceito de Aprendizado por Reforço combinado com o algoritmo *Q-Learning*.

Os autores utilizaram a mesma arquitetura de rede neural e aplicaram o mesmo método de aprendizado, com alguns ajustes de hiperparâmetros, a sete jogos do *Atari* 2600. Em seis jogos os autores obtiveram resultados superiores aos anteriores e superou o resultado de um especialista humano em três jogos.

3.2 RAD

Os autores (LASKIN et al., 2020) começam o artigo afirmando que um agente aprender por meio de observações visuais é um problema fundamental em Aprendizado por Reforço, porém, desafiador. Embora os avanços dos algoritmos combinados com redes neurais terem provado ser uma receita para o sucesso, os métodos que utilizam dessa

receita necessitam de uma eficiência maior na aprendizagem e uma maior capacidade de generalização. Com isso, os autores realizaram o primeiro estudo extensivo de aumento de dados em Aprendizado por Reforço.

Foram testados alguns métodos de aumento de dados convencionais, entre eles: Corte da Imagem, Translação, Escala de Cinza, *Cutout* (DEVRIES; TAYLOR, 2017), Espelhamento, Rotação, *Cutout-color*, *Random Convolution* e *Color-jitter*. Segundo os autores, o método que obteve maior eficiência foi o *Crop*.

3.3 DRQ

No trabalho publicado em 2020 (YARATS; KOSTRIKOV; FERGUS, 2021) os autores propuseram uma técnica de aumento de dados simples que pode ser aplicada a qualquer algoritmo de Aprendizado por Reforço. A abordagem aproveita as perturbações de entrada comumente usadas em tarefas de visão computacional para transformar exemplos de entrada da rede neural, com intuito de regularizar a função de valor e a política.

Além do mais, os autores afirmam que o algoritmo de Aprendizado por Reforço utilizado como base da pesquisa, o *Soft Actor Critic* (SAC) (HAARNOJA et al., 2018) não é capaz de treinar redes profundas com base nos *pixels* da imagem como recurso. No entanto, a adição de aumento de dados realizada pelos autores tornou possível com que o algoritmo SAC atingisse resultados em estado da arte nos ambientes de controle do *DeepMind* superando técnicas anteriores.

Além do mais, os autores também validaram a técnica de aumento de dados no algoritmo DQN (MNIH et al., 2013) melhorando significativamente os resultados obtidos pelo agente no ambiente de aprendizado do *Atari* 100k.

3.4 DRQ-V2

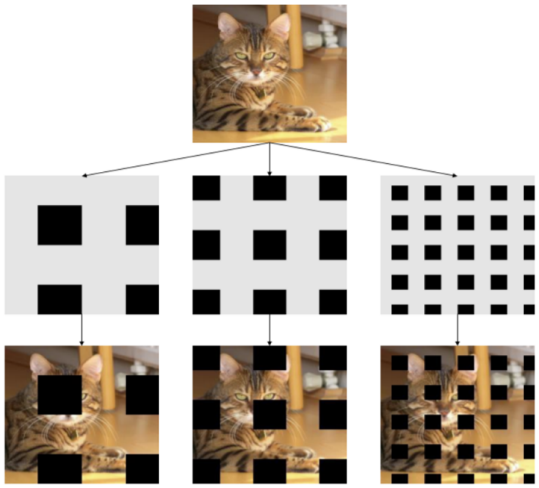
Os autores apresentaram uma melhoria do DRQ (YARATS; KOSTRIKOV; FERGUS, 2021), onde o algoritmo de Aprendizado por Reforço base foi alterado do algoritmo *Soft Actor Critic* (SAC) para o *Deep Deterministic Policy Gradient* (DDPG) (YARATS et al., 2021). Com isso, os autores atingiram o estado da arte pela primeira vez em ambientes do *MuJoCo* antes nunca solucionados.

O DRQ-V2 é capaz de resolver tarefas complexas de locomoção humanoide com base na observação de *pixels*. Para isso, os autores utilizaram um método de aumento de dados chamado *Random Shift*, na qual move a imagem 4 *pixels* em uma direção aleatória repetindo os *pixels* equivalentes.

3.5 *Grid Mask*

Grid Mask (CHEN et al., 2020) é um método de aumento de dados que remove parte das informações da imagem de entrada para obter resultados de estado da arte em diversos problemas de visão computacional. Segundo os autores, a regularização dos dados é fator importante para obter sucesso no treinamento de um modelo matemático. Um exemplo da transformação que aplicada na imagem pode ser visualizada na Figura 2.

Figura 2 – Exemplo da transformação aplicada pelo *Grid Mask*



Fonte: Grid Mask (CHEN et al., 2020).

Além do mais, os autores exibiram alguns métodos de regularização que obtiveram sucesso nos últimos anos. *Dropout* (SRIVASTAVA et al., 2014) que é utilizado nas camadas totalmente conectadas para desligar alguns valores de saída da rede neural. *Dropconnect* (WAN et al., 2013) muito semelhante ao *Dropout*, porém ele não descarta os valores de saída e sim os entrada. E muitos outros como *Dropout* Adaptativo (BA; FREY, 2013), *Pooling* Estocástico (ZEILER; FERGUS, 2013), *Droppath* (LARSSON; MAIRE; SHAKHNAROVICH, 2016), Regulação de *Shake-Shake* (GASTALDI, 2017) e *DropBlock* (GHIASI; LIN; LE, 2018) também obtiveram sucesso em suas implementações.

Os métodos supracitados alteram a estrutura da rede neural e adicionam ruído a alguns parâmetros no processo de treinamento da rede com diferentes regras que possuem objetivo de evitar o sobre-ajuste dos dados e tornar a generalização do modelo matemático mais eficiente (CHEN et al., 2020).

Com isso, os autores afirmam que além dos métodos que alteram a estrutura do modelo matemático, o Aumento de Dados é uma regularização muito eficaz e muito mais vantajoso comparado com os métodos mencionados, pois não necessita a alteração da estrutura da rede neural, visto que a única transformação é aplicada no conjunto de dados de treino quando é enviado à rede neural durante o processo de treinamento.

Os autores também indicaram alguns métodos de aumento de dados que ocultam parte das informações da imagem e obtiveram sucesso recentemente. Entre eles, *Random Erasing* (ZHONG et al., 2017), *Hide-and-Seek* (SINGH; LEE, 2017), *Cutout* (DEVRIES; TAYLOR, 2017) e entre outros. Assim como os métodos citados, o *Grid Mask* também pertence a esse grupo de técnicas de aumento de dados que excluem parte da informação. Contudo, os autores afirmam que o *Grid Mask* superou todos os outros métodos citados anteriormente em vários conjuntos de dados.

4 Desenvolvimento

4.1 Treinamento Do Agente

Para realizar o treinamento do agente foi utilizado o algoritmo de Aprendizado por Reforço *off-policy* e livre-de-modelo, Deep Deterministic Policy Gradient (DDPG). Foi utilizado como base o algoritmo publicado no trabalho DRQ-V2 (YARATS et al., 2021).

O DDPG utiliza quatro redes neurais (Figura 3) para realizar o seu aprendizado, entre elas: Uma Rede Neural para prever os valores de Q (*Q Network*), Uma Rede Neural para sincronizar os valores de Q (*Target Q Network*), Uma Rede Neural Determinística de Política (*Deterministic Policy Network*) e por fim, Uma Rede Neural para sincronizar a Política (*Target Policy Network*).

Figura 3 – Parâmetros de um *Deep Deterministic Policy Gradient*

Parameters:

θ^{Q} : Q network

θ^{μ} : Deterministic policy function

$\theta^{Q'}$: target Q network

$\theta^{\mu'}$: target policy network

Fonte: Google, 2021.

Esse sistema de utilizar redes neurais cópias que são sincronizadas ao fim de cada episodio é um importante elemento para o sucesso do treinamento do agente. Conceitualmente, é como dizer: "Tenho uma ideia de como jogar bem, vou tentar um pouco até encontrar algo melhor", em vez de dizer "Vou treinar-me novamente para jogar isso jogo inteiro após cada movimento".

Dessa forma, ao utilizar as redes *target* o modelo de aprendizado do agente não é atualizado a todo momento, mas somente quando o agente encontra um modelo matemático mais robusto. As redes *Target* foram introduzidas pela primeira vez em *Double Q-Learning* (HASSELT, 2010).

A rede neural Q e a rede neural de política são muito parecidas com o *Advantage Actor-Critic* (MNIH et al., 2016), mas no DDPG, o Ator mapeia diretamente os estados para as ações em vez de produzir a distribuição de probabilidade em um espaço de ação discreto, como é realizado nos algoritmos que atuam no ambiente do Atari.

O fluxograma do algoritmo DDPG pode ser visualizado conforme o pseudocódigo da Figura 4. A Figura foi retirada do trabalho que introduziu o DDPG (LILLICRAP et al., 2016).

Figura 4 – Fluxograma do Algoritmo *Deep Deterministic Policy Gradient*

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Fonte: *Continuous control with deep reinforcement learning.* (LILLICRAP et al., 2016)

O primeiro passo é inicializar os pesos das Redes Neurais aleatoriamente. Portanto a rede neural *Critic* (*Q Network*) e a rede neural *Actor* (*Deterministic Policy Function*). Logo após, as redes neurais *target* são inicializadas por meio da sincronização dos pesos. A *Target Q Network* é sincronizada com os pesos aleatórios gerados para a Q Network, assim como a *Target Policy Function* é sincronizada com a rede neural *Deterministic Policy Function*.

O próximo passo é inicializar o *Buffer* de Experiências. Ele será responsável por armazenar as experiências processadas pelo agente e são utilizadas de forma aleatória no treinamento da rede neural. Para o cenário do *Quadruped Walk* foi definido o tamanho máximo de 1 milhão de registros para o *Buffer* de Experiências, o mesmo número de frames que serão processados até o fim do treinamento.

Com isso, o algoritmo inicia o processo de treinamento realizando a primeira ação aleatoriamente. O ambiente então retorna o estado e a recompensa resultante após a alteração do ambiente ocasionada pela execução da ação aleatória.

Então o algoritmo requisita a melhor ação a ser executada para Rede Neural. Nesse momento a Rede Neural não está com os pesos ajustados, portanto, provavelmente as primeiras ações realizadas no ambiente resultarão em recompensas negativas.

Essa requisição de ação é realizada para as redes neurais *target* para evitar que haja perda no aprendizado do agente, pois as redes neurais originais têm seus pesos atualizados a cada ação do agente no ambiente. Enquanto são sincronizadas apenas no final do Episódio.

Após selecionar a ação, o agente executa-a no ambiente. Nesse momento o ambiente é alterado e retorna uma recompensa positiva ou negativa de acordo com a qualidade da ação que foi aplicada. Além do mais, o ambiente retorna a observação resultante da aplicação da ação na observação anterior.

Com isso é gerada uma transição que é composta por uma tupla: Observação Atual, Ação Atual, Recompensa Atual e Próxima Observação. Esta transição é armazenada no buffer de experiências e será utilizada para treinamento e ajustes dos pesos das redes neurais posteriormente.

O próximo passo é realizar o treinamento da rede neural para ajuste dos pesos e para que o agente selecione ações melhores. Então é selecionado 256 transições do buffer de experiências aleatoriamente para remover qualquer relação entre as transições experienciadas. Dessa forma o agente não irá decorar as transições.

Com isso, é realizado o ajuste dos pesos por meio da minimização da perda com o otimizador *Adam* (KINGMA; BA, 2014). Ambas, *Q-Network* e *Deterministic Policy Function* são treinadas separadamente. A Partir de então o agente começará a qualificar suas ações com base nas experiências vivenciadas por meio da tentativa e erro.

Na próxima iteração será realizada a sincronização dos pesos das redes neurais originais com as redes neurais target apenas se possuírem uma política mais eficiente. Dessa forma, é garantido que o agente apenas teste ações e simule um aprendizado. Onde o conhecimento é compartilhado apenas se houver ganho.

4.2 Aplicação do Aumento de Dados

O aumento de dados *Grid Mask* (CHEN et al., 2020) foi aplicado logo após a transformação Random Shift que foi aplicada pelos autores do DRQ-V2 (YARATS et al., 2021). A técnica de aumento de dados é uma técnica simples que pode ser aplicada em qualquer outro problema de visão computacional.

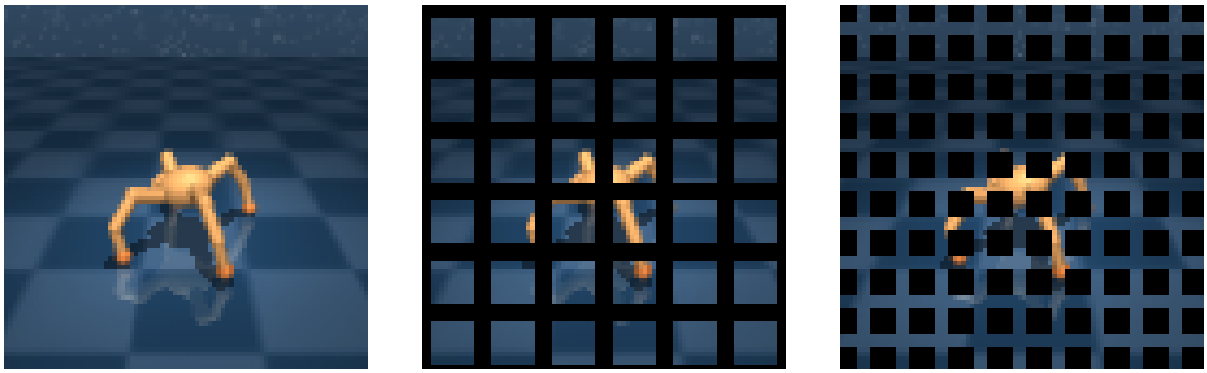
Os autores aplicaram o aumento de dados Random Shift em todas imagens que são enviadas à rede neural, devido ao fato de que esse aumento de dados muda relativamente poucas informações da imagem. Já o aumento de dados *Grid Mask* foi aplicado em apenas 25% das imagens, no melhor experimento realizado.

Foram realizados 3 experimentos diferentes com a aplicação de *Grid Mask* no cenário *Quadruped Walk* do ambiente de desenvolvimento *MuJoCo*. Um experimento em que foi aplicado *Grid Mask* do tipo 0 em 10% das imagens. Um experimento em que foi aplicado *Grid Mask* 0 em 25% das imagens. E por fim, o experimento mais bem sucedido, em que foi aplicado *Grid Mask* do tipo 1 em 25% das imagens.

O *frame* original do cenário pode ser visualizado na Figura 5 (a). Enquanto a aplicação de *Grid Mask* do tipo 0 pode ser visualizada na Figura 5 (b), enquanto a aplicação de *Grid Mask* do tipo 1 pode ser visualizada na Figura 5 (c). Além do mais foi definido para que o tamanho de cada bloco varie entre 8 e 16 *pixels* enquanto que a razão entre cada bloco foi definida para 0.5.

Ambas transformações ocluem parte das informações da imagem, no entanto o tipo 1 é ideal para o cenário do *Quadruped Walk* pois ainda torna possível a visualização do agente. Diferentemente do tipo 0 que oclui grande parte das informações da imagem.

Figura 5 – Cenário *Quadruped Walk* antes e depois da aplicação do Aumento de Dados



(a) Imagem Original (b) *Grid Mask* Tipo 0 (c) *Grid Mask* Tipo 1

Fonte: Própria, 2021.

Após a aplicação da transformação de dados *Grid Mask*, a rede neural é forçada a generalizar os espaços que foram obstruídos no processo de aumento de dados. Como é possível observar na Figura 5, parte do ambiente do *MuJoCo* no cenário *Quadruped Walk* foi obstruído. No entanto, ainda é possível obter algumas informações do agente por meio do reflexo no chão do ambiente. Cabe a politica da rede neural identificar este padrão.

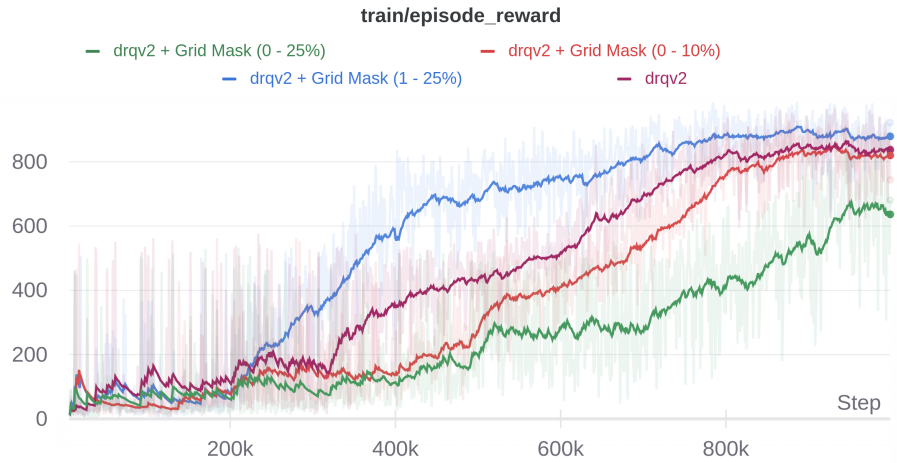
5 Avaliação dos Resultados

Ao final do treinamento do agente foi possível perceber que o agente aprendeu a executar ações no cenário *Quadruped Walk* do ambiente de aprendizado *MuJoCo* e ainda superou resultado dos autores relacionados. O aprendizado do agente levou cerca de 5h40min para ser concluído, na qual foi utilizada uma placa gráfica para aceleração dos cálculos de tensores. A placa gráfica utilizada foi uma *GPU NVIDIA GeForce GTX Titan X* de 12GB.

Como o *MuJoCo* possui um conjunto de ambientes que possuem a característica de ter Controle Contínuo, ele não possui um ponto de chegada ou fim de jogo como é visto nos ambientes do Atari. Portanto o agente foi treinado durante 1M de iterações, conforme definido pelo autor para cenários de dificuldade média.

Todos os gráficos dessa seção foram gerados por meio da ferramenta *Weights & Biases* (WANDB) (BIEWALD, 2020) que possui um painel central em nuvem para armazenar os hiperparâmetros e métricas do treinamento do agente.

Figura 6 – Gráfico da Recompensa sobre cada *Frame*



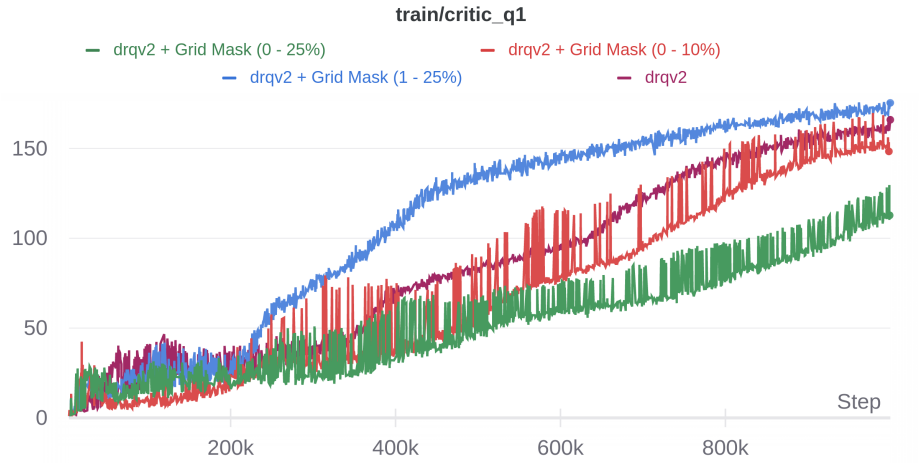
Fonte: Própria (WANDB), 2021.

Como é possível observar na Figura 6, o experimento em que foi aplicado o aumento de dados *Grid Mask* com Tipo 1 em 25% das imagens foi o que obteve melhor desempenho. Isso deve-se ao fato de que o *Grid Mask* do Tipo 1 é o que menos oclui informações da imagem, dessa forma a generalização a ser realizada pela rede neural é sintetizada. Para simplificar a visualização foi aplicado uma suavidade de linha de 90% no gráfico de recompensa sobre iterações. É possível observar as linhas originais com opacidade baixa no fundo da Figura 6.

O experimento com *Grid Mask* ultrapassa o experimento original do DRQ-V2 após

as 200 k iterações e permanece superior até o fim do treinamento. A partir da iteração 200 k até 800 k é possível observar uma grande diferença no desempenho da implementação proposta por este trabalho e o trabalho original que não utiliza *Grid Mask*. No entanto, ao final do treinamento ambos gráficos se equiparam, com uma pequena vantagem no experimento com *Grid Mask*.

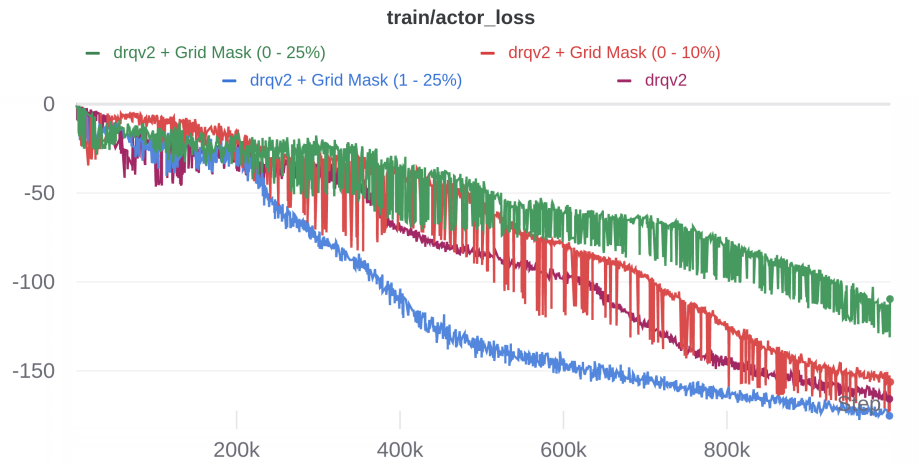
Figura 7 – Gráfico do Valor de Q sobre cada *Frame*



Fonte: Própria (WANDB), 2021.

Assim como no gráfico de recompensa sobre as iterações, o gráfico do *Critic* Q na Figura 7 no experimento com *Grid Mask* se mostrou superior na maioria do experimento. Em comparação com o trabalho original do DRQ-V2, obteve-se um grande aumento no valor de Q durante as fases medias de treinamento. No entanto ao final do treinamento esses valores se igualam com uma pequena vantagem no experimento em que foi aplicado *Grid Mask*. Desta vez não foi aplicado suavidade nas linhas do gráfico.

Figura 8 – Gráfico da *Loss* do Agente



Fonte: Própria (WANDB), 2021.

Além do mais, também foi coletada a função de perda do agente que pode ser visualizada na Figura 8. Assim como nos outros gráficos é possível observar que o experimento em que foi aplicado *Grid Mask* com tipo 1 obteve resultados superiores em comparação com o trabalho original DRQ-V2 e com os experimentos com *Grid Mask* 0.

A função de perda é ajustada juntamente com o ajuste dos pesos que é realizado durante o treinamento do agente. Para o treinamento do DDPG os autores escolheram o otimizador *Adam* (KINGMA; BA, 2014) para normalizar a perda do treinamento.

6 CONSIDERAÇÕES FINAIS

Após realização deste trabalho, conclusão dos experimentos e analise dos resultados é possível perceber que a utilização de técnica de aumento de dados no treinamento de agentes de Aprendizado por Reforço é uma técnica extremamente eficaz, porém ainda inexplorada por pesquisadores de AR.

Além da técnica de aumento de dados *Grid Mask* ser simples de ser implementada ela é amplamente escalável para outros problemas de Aprendizado por Reforço que envolvem processamento dos estados por meio de Visão Computacional.

Ao realizar os experimentos de treinamento do agente no cenário *Quadruped Walk* com algoritmo DRQ-V2 combinado com a técnica de aumento de dados *Grid Mask* percebeu-se um grande ganho na eficiência do treinamento do agente, principalmente nas fase media e final.

Por ser uma área de reconhecimento muito nova, no Aprendizado por Reforço ainda há muitos cenários que ainda não foram explorados por pesquisadores de AR. Com isso, sugerimos que como trabalhos futuros sejam exploradas a geração de dados artificiais por meio de Generative adversarial networks (GANs), que podem ser aplicadas em qualquer problema de visão de computacional.

Referências

BA, J.; FREY, B. Adaptive dropout for training deep neural networks. In: BURGES, C. J. C. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013. v. 26. Disponível em: <https://proceedings.neurips.cc/paper/2013/file/7b5b23f4aadf9513306bcd59afb6e4c9-Paper.pdf>. Citado na página 18.

Bellemare, M. G. et al. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, v. 47, p. 253–279, jun 2013. Citado 2 vezes nas páginas 11 e 14.

BENGIO, Y.; COURVILLE, A. C.; VINCENT, P. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012. Disponível em: <http://arxiv.org/abs/1206.5538>. Citado na página 13.

BIEWALD, L. *Experiment Tracking with Weights and Biases*. 2020. Software available from wandb.com. Disponível em: <https://www.wandb.com/>. Citado na página 24.

BROCKMAN, G. et al. *OpenAI Gym*. 2016. Cite arxiv:1606.01540. Disponível em: <http://arxiv.org/abs/1606.01540>. Citado na página 12.

CHEN, P. et al. *GridMask Data Augmentation*. 2020. Citado 2 vezes nas páginas 18 e 22.

DEVRIES, T.; TAYLOR, G. W. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. Disponível em: <http://arxiv.org/abs/1708.04552>. Citado 2 vezes nas páginas 17 e 19.

FARLEY, B.; CLARK, W. A. Simulation of self-organizing systems by digital computer. *Trans. IRE Prof. Group Inf. Theory*, v. 4, p. 76–84, 1954. Citado na página 13.

GASTALDI, X. Shake-shake regularization. *CoRR*, abs/1705.07485, 2017. Disponível em: <http://arxiv.org/abs/1705.07485>. Citado na página 18.

GHIASI, G.; LIN, T.; LE, Q. V. Dropblock: A regularization method for convolutional networks. *CoRR*, abs/1810.12890, 2018. Disponível em: <http://arxiv.org/abs/1810.12890>. Citado na página 18.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 13.

GULLAPALLI, V. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, v. 3, n. 6, p. 671–692, 1990. Disponível em: <https://doi.org/10.1016/0893-6080(90)90056-Q>. Citado na página 13.

HAARNOJA, T. et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: DY, J. G.; KRAUSE, A. (Ed.). *ICML*. PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 1856–1865. Disponível em: <http://dblp.uni-trier.de/db/conf/icml/icml2018.html#HaarnojaZAL18>. Citado na página 17.

HASSELT, H. Double q-learning. In: LAFFERTY, J. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2010. v. 23. Disponível em: <https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf>. Citado na página 20.

HOLCOMB, S. D. et al. Overview on deepmind and its alphago zero ai. In: *Proceedings of the 2018 International Conference on Big Data and Education*. New York, NY, USA: Association for Computing Machinery, 2018. (ICBDE '18), p. 67–71. ISBN 9781450363587. Disponível em: <https://doi.org/10.1145/3206157.3206174>. Citado na página 13.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2014. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado 2 vezes nas páginas 22 e 26.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Citado na página 16.

LAPAN, M. *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods to Practical Problems of Chatbots, Robotics, Discrete Optimization, Web Automation, and More, 2nd Edition*. Packt Publishing, Limited, 2020. (Expert insight). ISBN 9781838826994. Disponível em: <https://books.google.com.br/books?id=Gy1ZzAEACAAJ>. Citado 4 vezes nas páginas 10, 11, 12 e 13.

LARSSON, G.; MAIRE, M.; SHAKHNAROVICH, G. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016. Disponível em: <http://arxiv.org/abs/1605.07648>. Citado na página 18.

LASKIN, M. et al. *Reinforcement Learning with Augmented Data*. 2020. Citado na página 16.

LECUN, Y. Learning processes in an asymmetric threshold network. In: BIENENSTOCK, E.; FOGELMAN-SOULIE, F.; WEISBUCH, G. (Ed.). *Disordered systems and biological organization, Les Houches, France*. [S.l.]: Springer-Verlag, 1986. p. 233–240. Citado na página 13.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 1, n. 4, p. 541–551, dez. 1989. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1989.1.4.541>. Citado na página 14.

LILLICRAP, T. P. et al. Continuous control with deep reinforcement learning. In: BENGIO, Y.; LECUN, Y. (Ed.). *ICLR*. [s.n.], 2016. Disponível em: <http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15>. Citado na página 21.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 127–147, 1943. Citado na página 12.

MNIH, V. et al. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. Disponível em: <http://arxiv.org/abs/1602.01783>. Citado na página 20.

MNIH, V. et al. *Playing Atari with Deep Reinforcement Learning*. 2013. Citado 2 vezes nas páginas 16 e 17.

MNIH, V. et al. Human-level control through deep reinforcement learning. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 518, n. 7540, p. 529–533, fev. 2015. ISSN 00280836. Disponível em: <http://dx.doi.org/10.1038/nature14236>. Citado na página 12.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958. ISSN 0033-295X. Disponível em: <http://dx.doi.org/10.1037/h0042519>. Citado na página 13.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Back-propagating Errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. Disponível em: <http://www.nature.com/articles/323533a0>. Citado na página 13.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. In: _____. *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988. p. 696–699. ISBN 0262010976. Citado na página 13.

RUMMERY, G. A.; NIRANJAN, M. *On-Line Q-Learning Using Connectionist Systems*. Cambridge, England, 1994. Citado na página 10.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado na página 16.

SCHMIDHUBER, J. *On Learning to Think: Algorithmic Information Theory for Novel Combinations of Reinforcement Learning Controllers and Recurrent Neural World Models*. 2015. Citado na página 13.

SEWAK, M. *Deep Reinforcement Learning: Frontiers of Artificial Intelligence*. Springer Singapore, 2019. ISBN 9789811382857. Disponível em: <https://books.google.com.br/books?id=B5WfDwAAQBAJ>. Citado 2 vezes nas páginas 12 e 14.

SILVER, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, v. 529, n. 7587, 2016. Citado na página 13.

SILVER, D. et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 12 2017. Citado na página 10.

SILVER, D. et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. Citado na página 13.

SINGH, K. K.; LEE, Y. J. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. *CoRR*, abs/1704.04232, 2017. Disponível em: <http://arxiv.org/abs/1704.04232>. Citado na página 19.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, JMLR.org, v. 15, n. 1, p. 1929–1958, jan. 2014. ISSN 1532-4435. Citado na página 18.

SUTTON, R.; BARTO, A. *Reinforcement Learning: An Introduction*. MIT Press, 2018. (Adaptive Computation and Machine Learning series). ISBN 9780262039246. Disponível em: <https://books.google.com.br/books?id=6DKPtQEACAAJ>. Citado 3 vezes nas páginas 10, 11 e 12.

SUTTON, R. et al. *Reinforcement Learning: An Introduction*. MIT Press, 1998. (A Bradford book). ISBN 9780262193986. Disponível em: <https://books.google.com.br/books?id=CAFR6IBF4xYC>. Citado na página 13.

TASSA, Y. et al. *DeepMind Control Suite*. 2018. Citado na página 14.

TESAURO, G. Temporal difference learning and td-gammon. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 3, p. 58–68, mar. 1995. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/203330.203343>. Citado na página 13.

TODOROV, E.; EREZ, T.; TASSA, Y. Mujoco: A physics engine for model-based control. In: *IROS*. IEEE, 2012. p. 5026–5033. ISBN 978-1-4673-1737-5. Disponível em: <http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12>. Citado na página 14.

TSITSIKLIS, J.; ROY, B. V. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, v. 42, n. 5, p. 674–690, 1997. Citado na página 12.

WAN, L. et al. Regularization of neural networks using dropconnect. In: DASGUPTA, S.; MCALLESTER, D. (Ed.). *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, Georgia, USA: PMLR, 2013. (Proceedings of Machine Learning Research, 3), p. 1058–1066. Disponível em: <https://proceedings.mlr.press/v28/wan13.html>. Citado na página 18.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. *Machine Learning*, v. 8, p. 279–292, 1992. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00992698>. Citado na página 10.

WIDROW, B.; GUPTA, N. K.; MAITRA, S. Punish/reward: learning with a critic in adaptive threshold systems. *IEEE Trans. Syst. Man Cybern.*, v. 3, n. 5, p. 455–465, 1973. Citado na página 13.

WIDROW, B.; HOFF, M. E. Adaptive switching circuits. In: *1960 IRE WESCON Convention Record, Part 4*. New York: IRE, 1960. p. 96–104. Citado na página 13.

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, v. 8, p. 229–256, 1992. Citado na página 13.

WOO, M. et al. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1999. Citado na página 14.

YARATS, D. et al. *Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning*. 2021. Citado 4 vezes nas páginas 14, 17, 20 e 22.

YARATS, D.; KOSTRIKOV, I.; FERGUS, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In: *International Conference on Learning Representations*. [s.n.], 2021. Disponível em: <https://openreview.net/forum?id=GY6-6sTvGaf>. Citado na página 17.

ZEILER, M.; FERGUS, R. Stochastic pooling for regularization of deep convolutional neural networks. *In: ICLR*, 01 2013. Citado na página 18.

ZHONG, Z. et al. Random erasing data augmentation. *CoRR*, abs/1708.04896, 2017. Disponível em: <http://arxiv.org/abs/1708.04896>. Citado na página 19.