Caio de Souza Barbosa Costa

Otimização de Carteiras de Ações com Aprendizado por Reforço

São Paulo, SP

2023

Caio de Souza Barbosa Costa

Otimização de Carteiras de Ações com Aprendizado por Reforço

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Profa. Dra. Anna Helena Reali Costa

São Paulo, SP

2023

Catalogação-na-publicação

Costa, Caio de Souza Barbosa
Otimização de Carteiras de Ações com Aprendizado por Reforço / C. S. B. Costa -- São Paulo, 2023.
85 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

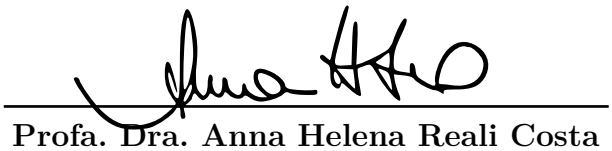
1.Otimização de Portfólio 2.Aprendizado de Máquina 3.Aprendizado por Reforço 4.Mercado de Ações 5.Finanças Quantitativas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Caio de Souza Barbosa Costa

Otimização de Carteiras de Ações com Aprendizado por Reforço

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

De acordo,



Profa. Dra. Anna Helena Reali Costa
Orientador

São Paulo, SP
2023

Agradecimentos

Este trabalho foi apoiado pelo Programa de Bolsas Itaú (PBI) do Centro de Ciências de Dados (C^2D) da Escola Politécnica da Universidade de São Paulo, programa que é mantido pelo Itaú Unibanco S.A.

Resumo

Este projeto de conclusão de curso tem como objetivo replicar e avaliar a eficácia de arquiteturas estado-da-arte na otimização de carteiras de ações usando aprendizado por reforço no mercado brasileiro. Muitas técnicas têm sido empregadas para automatizar esse processo, visando a redução de perdas e a maximização de retornos a longo prazo, com o objetivo de prever o valor futuro das ações e compor as carteiras da maneira mais eficiente possível. Devido à natureza do problema, que busca maximizar retornos a longo prazo, as técnicas de aprendizado por reforço mostram-se altamente adequadas. Curiosamente, essas técnicas ainda não foram amplamente exploradas nesse tipo de problema, especialmente no mercado brasileiro. Este trabalho aborda essa lacuna e, além de replicar e testar arquiteturas notáveis, introduz uma série de ferramentas para impulsionar ainda mais a pesquisa nessa área.

Palavras-chave: Otimização de Portfólio, Aprendizado de Máquina, Aprendizado por Reforço, Mercado de Ações, Finanças Quantitativas.

Abstract

This project aims to replicate and evaluate the effectiveness of state-of-the-art architectures in optimizing stock portfolios using reinforcement learning in the Brazilian market. Many techniques have been employed to automate this process, aiming to reduce losses and maximize long-term returns by predicting the future value of stocks and composing portfolios in the most efficient way possible. Due to the nature of the problem, which seeks to maximize long-term returns, reinforcement learning techniques are highly suitable. Interestingly, these techniques have not been extensively explored in this type of problem, particularly in the Brazilian market. This work addresses this gap and, in addition to replicating and testing state-of-the-art architectures, introduces a series of tools to further advance research in this area.

Keywords: Portfolio Optimization, Machine Learning, Reinforcement Learning, Stock Market, Quantitative Finance.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

A3C	Asynchronous Advantage Actor-Critic
CPU	Central Processing Unit
DDPG	Deep Deterministic Policy Gradient
EI^3	Ensemble of Identical Indenpendent Inception
EIIE	Ensemble of Identical Independent Evaluators
EMA	Exponential Moving Average
EMN	Equity Market Neutral
ETF	Exchange-Traded Fund
fAPV	Final Accumulative Portfolio Value
GB	GigaByte
GDDR5X	Graphics Double Data Rate 5 eXtreme
GNN	Graph Neural Network
GPU	Graphics Processing Unit
GTX	Giga Texel Shader eXtreme
IBOVESPA	Índice da Bolsa de Valores de São Paulo
MACD	Moving Average Convergence Divergence
MDD	Maximum DrawDown
MDP	Markov Decision Process
NASDAQ	National Association of Securities Dealers Automated Quotations
NYSE	New York Stock Exchange
POE	Portfolio Optimization Environment
POMDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization

ReLU	Rectified Linear Unit
RGB	Red Green Blue
SPDQ	Stochastic Policy with Distributional Q-network
SR	Sharpe Ratio
UBAH	Uniform Buy And Hold

Sumário

1 Introdução

1.1 Motivação

Por ser consideravelmente volátil, investidores que utilizam o mercado de ações precisam constantemente aplicar estratégias para diminuir perdas, aumentar ganhos e maximizar seus retornos a longo prazo. Para isso, é oportuno que tais estratégias consigam prever o valor futuro das ações em que se está investindo para que as compras e vendas sejam feitas apropriadamente. Por isso, técnicas de inteligência artificial têm sido aplicadas cada vez mais, em especial o aprendizado de máquinas (NASSIRTOUSSI et al., 2014; HU et al., 2015; HENRIQUE; SOBREIRO; KIMURA, 2019), alcançando altas capacidades preditivas.

Apesar de seus bons resultados, o aprendizado supervisionado pode não ser a escolha ideal para o mercado acionário, visto que um agente *trader* deseja maximizar o lucro a longo prazo e, portanto, ele deve ser capaz de estimar o retorno financeiro futuro de suas posições tomadas. Devido a esse motivo, sistemas supervisionados acabam sendo sensíveis a fatores do mundo real como, por exemplo, a existência de taxas de corretagem (DENG et al., 2017).

Por outro lado, o aprendizado por reforço é apropriado para esse tipo de problema visto que sua modelagem principal visa maximizar, exatamente, o retorno futuro de uma dada grandeza expressa na recompensa (no caso do mercado financeiro, o lucro). Além disso, esse tipo de técnica possui a capacidade de aprender em produção (*online learning*) e é bastante adaptável a situações incomuns, podendo criar comportamentos efetivos mesmo em condições iniciais do aprendizado (SUTTON; BARTO, 2018).

Dito isto, a aplicação do aprendizado por reforço se mostra promissora para resolver o problema de otimização de portfólio, em que o agente deve, a cada instante de tempo, definir o balanço ideal de investimento de sua carteira de modo a adquirir lucro a longo prazo. Mas a utilização desse algoritmo nesse tipo de problema é algo relativamente recente e que precisa ser mais estudado, mas vem ganhando certa atenção principalmente pelo advento do aprendizado por reforço profundo e seus bons resultados, de modo que diversos artigos têm sido publicados nos últimos anos (FELIZARDO et al., 2022; GUNJAN; BHATTACHARYYA, 2022; HAMBLY; XU; YANG, 2023) principalmente após a publicação do trabalho de Jiang *et al.* (JIANG; XU; LIANG, 2017). Apesar disso, ainda há uma certa escassez de ferramentas *open source* para desenvolver e reproduzir trabalhos da área, o que torna a pesquisa mais difícil e a aplicação pela indústria mais demorada por haver poucos métodos de reprodutibilidade e comparação de desempenho.

1.2 Objetivo

Este projeto de formatura visa desenvolver as ferramentas necessárias para implementar um sistema otimizador de carteira de ações utilizando aprendizado por reforço: um ambiente de simulação para agentes, o algoritmo de treinamento e algumas arquiteturas notáveis. Além disso, um breve estudo avaliando o desempenho das arquiteturas notáveis é feito, visando compreender seu comportamento em alguns mercados, incluindo o brasileiro.

1.3 Justificativa

Cada vez mais instituições financeiras estão fazendo uso de modelos estatísticos e de aprendizado de máquina para auxiliarem equipes de *asset allocation* e é esse contexto que faz o projeto apresentado ao longo deste documento ter considerável relevância. Com a proposta, este projeto possui grande potencial para permitir a pesquisa, o desenvolvimento e a aplicação de métodos bastante eficazes na otimização de carteiras de ações. Outro ponto de destaque deste projeto é o grande foco em realizar testes utilizando dados da bolsa de valores do Brasil. Essa ideia contrasta com a grande maioria da literatura científica correlata, que costuma focar em mercados de ações internacionais. Pesquisar no mercado financeiro brasileiro permite a descoberta de possíveis nuances da bolsa do País, algo que contribui para alavancar a aplicação de modelos estatísticos no Brasil.

1.4 Organização do trabalho

A seguir, no capítulo 2, os principais conceitos utilizados neste projeto de formatura são introduzidos. Posteriormente, no capítulo 3, a metodologia empregada durante o desenvolvimento deste trabalho é apresentada e, no capítulo 4, os requisitos funcionais e não-funcionais do projeto serão listados. Em seguida, uma revisão bibliográfica de trabalhos relacionados é feita no capítulo 5, bem como a classificação de artigos e a identificação de tendências e lacunas na área de pesquisa. Logo após, o capítulo 6 introduz as ferramentas desenvolvidas para permitirem a otimização de carteiras de ações e faz um breve estudo sobre a aplicação dessas ferramentas no mercado brasileiro e americano. Por fim, o capítulo 7 conclui este trabalho e discorre sobre possíveis trabalhos futuros.

2 Aspectos Conceituais

2.1 Sistemas de negociação autônoma

Existem dois problemas principais relacionados a sistemas autônomos de negociação de ações: o problema de *single asset* e de *portfolio optimization*, descritos a seguir.

Single Asset: o sistema tenta operar uma única ação por vez analisando, entre diversas características possíveis, sua série histórica. Em geral, as posições tomadas por um sistema *single asset* são de compra, venda e espera (o sistema segura a ação). Nota-se que, apesar do nome, é possível que um sistema gerencie diversas ações, mas de forma totalmente independente.

Portfolio Optimization: o sistema deve ser capaz de administrar uma carteira de investimentos com base em indicadores do mercado acionário (entre eles, a série histórica de cada ação). Esse sistema deve, periodicamente, modificar a composição das ações presentes na carteira, fazendo um reajuste das alocações de cada ação no *portfolio*.

Neste projeto, tomaremos como problema principal o ***portfolio optimization***, pois seu desafio é bastante interessante: diferentemente do problema de *single asset*, é necessário pensar em uma forma de combinar a série histórica de diversas ações e ainda considerar o problema de que a ação tomada pelo agente deve ser contínua, visando o maior retorno a longo prazo, considerando todo o conjunto de ações na carteira.

2.2 Otimização de portfólio

A otimização de portfólio é uma tarefa em que o sistema deve periodicamente fazer a alocação de recursos em um portfólio de investimentos. Nessa tarefa, o sistema inicia com uma quantidade específica de dinheiro em caixa e, procurando obter lucro, ele deve rebalancear constantemente todos os investimentos após ter analisado o mercado por um determinado intervalo de tempo: os pequenos períodos entre essas análises de mercado em que o sistema é capaz de rebalancear o portfólio é chamado de *período de realocação* (como pode-se ver na figura 1). Um *framework* de otimização de carteiras ideal deve ser capaz de agir nesses períodos de realocação para aumentar o lucro e mitigar perdas.

2.2.1 Definição matemática

Dado um portfólio com n ações, pode-se definir a cada instante de tempo um vetor de pesos ($\mathbf{W}_t$) que contém as porcentagens (ou pesos) de todas as ações presentes no portfólio e do dinheiro remanescente (e, portanto, não investido). Dessa forma, o tamanho de $\mathbf{W}_t$ é $|\mathbf{W}_t| = n + 1$ e a soma de seus elementos deve ser igual a 1. Formalmente, sendo $\mathbf{W}_t(i)$ o i-ésimo elemento de $\mathbf{W}_t$, o *weights vector* deve ser limitado pela expressão

$$\sum_{i=0}^{n} \mathbf{W}_t(i) = 1. \tag{2.1}$$

Para cada instante de tempo, pode-se também definir um vetor de preços $\mathbf{P}_t$ com a seguinte forma: $[1, \mathbf{P}_t(1), \mathbf{P}_t(2), ..., \mathbf{P}_t(n)]$. O primeiro elemento desse vetor é sempre 1 pois está relacionado ao capital não investido, cujo preço é invariável - caso o sistema possua, por exemplo, R$ não-investido, assume-se que, desde que esse valor não seja realocado em nenhum outro investimento, ele sempre terá o mesmo valor. Esse primeiro elemento também pode ser visto como um preço de referência visto que todos os outros preços são calculados em relação a ele.

Se forem conhecidos o último valor do portfólio (V_{t-1}), o atual vetor de preços ($\mathbf{P}_t$), o último vetor de preços ($\mathbf{P}_{t-1}$) e o último vetor de pesos ($\mathbf{W}_{t-1}$), é possível calcular o valor atual do portfólio V_t utilizando a fórmula recursiva:

$$V_t = V_{t-1}\Big(\mathbf{W}_{t-1} \cdot (\mathbf{P}_t \oslash \mathbf{P}_{t-1})\Big), \tag{2.2}$$

em que $\cdot$ é o produto escalar de dois vetores e $\oslash$ é a divisão elemento a elemento. Note que esse método assume que cada uma das ações possui preços positivos, visto que uma divisão por zero causaria erros na operação de divisão.

Na equação 2.2, o valor do portfólio mudou de V_{t-1} para V_t pois os preços das ações se modificaram, mas isso também afeta o vetor de pesos, que, ao final do último instante de tempo, assume um valor diferente dado por

$$\mathbf{W}^f_{t-1} = \frac{(\mathbf{P_t} \oslash \mathbf{P_{t-1}}) \odot \mathbf{W}_{t-1}}{(\mathbf{P_t} \oslash \mathbf{P_{t-1}}) \cdot \mathbf{W}_{t-1}}, \tag{2.3}$$

em que $\odot$ é a multiplicação elemento a elemento. A intuição de $\mathbf{W}^f_{t-1}$ é que ele representa o vetor de pesos no momento final do instante $t-1$, uma informação que é necessária quando taxas de corretagem são impostas ao sistema (algo que será explorado na seção 2.2.2).

A figura 1 representa o processo de otimização de portfólio ao longo do tempo, note que $\mathbf{W}_t$ é determinado durante o período de realocação.

É comum modelar o lucro no instante t como uma taxa de retorno logarítmica (r_t):

$$r_t = ln\Big(\frac{V_t}{V_{t-1}}\Big). \tag{2.4}$$

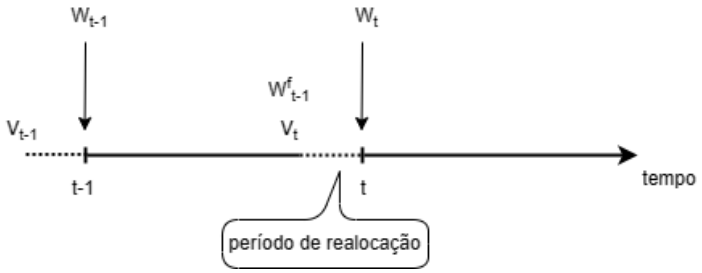


Figura 1 – Diagrama representando o processo de otimização de portfólio ao longo do tempo.

Dessa forma, o valor final do portfólio pode ser expressado como:

$$V_{T+1} = V_0 \; exp\Big(\sum_{t=1}^{T+1} r_t\Big), \tag{2.5}$$

em que T é o último instante em que uma alocação de ações do portfólio ocorre e r_{T+1} é o lucro dessa última alocação.

Assim, pode-se definir o problema de otimização de portfólio como o processo de achar a melhor sequência de vetores de pesos ($[\mathbf{W_0}, \mathbf{W_1}, ..., \mathbf{W_T}]$) que maximiza o valor final do portfólio V_{T+1}.

2.2.2 Taxas de corretagem

A formulação apresentada na seção 2.2.1 não considera taxas de corretagem, que são comumente cobradas quando ações são compradas ou vendidas. Mas considerar essas taxas é bastante importante visto que, no mundo real, esse tipo de tarifa é uma prática comum e pode influenciar muito o desempenho do sistema já que o custo de se assumir posições no mercado aumenta conforme o número de ordens emitidas aumenta. Normalmente, em muitas corretoras, as taxas são definidas como uma porcentagem do valor absoluto da transação e, durante o período de realocação do portfólio, o valor do portfólio diminui devido aos custos relacionados às movimentações. Por isso, é necessário diferenciar o valor do portfólio no final de um instante de tempo (V_{t-1}^f) do valor do portfólio no início do próximo instante de tempo (V_t) (após os encargos terem sido aplicados).

Existem duas formas de se modelar taxas de corretagem em problemas de otimização de portfólio:

Modificador do vetor de pesos: Nessa primeira abordagem, no início do instante t, o agente define a ação na forma do vetor de pesos W_t. Após isso, as tarifas de operação são calculadas somando-se o valor absoluto da variação de dinheiro investido em cada ação e aplicando a taxa percentual. A taxa de corretagem é, então, retirada do dinheiro não investido, o que faz o vetor de pesos W_t ser modificado para um novo

vetor W'_t e o valor do portfólio mudar de V^f_{t-1} para V_t. Esse método tem uma falha: se a taxa de corretagem for maior do que o dinheiro disponível em caixa, a realocação do portfólio não pode ser feita e o último vetor de pesos W^f_{t-1} é considerado como W'_t. A figura 2 representa como o modificador de vetor de pesos é aplicado ao processo de otimização.

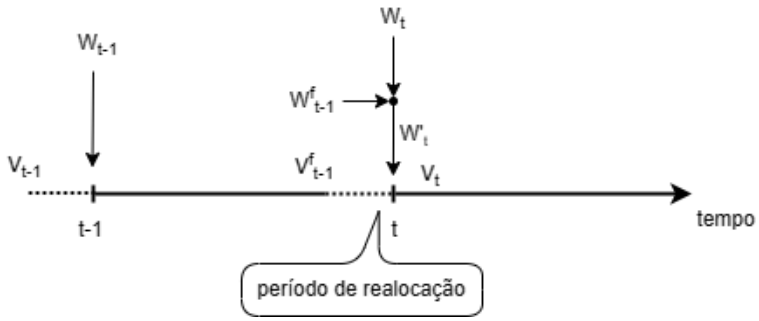


Figura 2 – Processo de otimização de portfólio com modificador do vetor de pesos.

Fator residual de transações: Outra forma de simular os custos de transação é definir um fator residual μ_t que modifica o valor do portfolio durante a realocação utilizando a seguinte fórmula:

$$V_t = \mu_t V^f_{t-1}. \tag{2.6}$$

μ_t assume um valor diferente dependendo do instante de tempo visto que cada realocação possui um número diferente de ordens de compra e venda a serem executadas. A fórmula para calcular μ_t e sua demostração está disponível em (JIANG; XU; LIANG, 2017). A figura 3 mostra a influência do fator residual de transações no valor do portfólio.

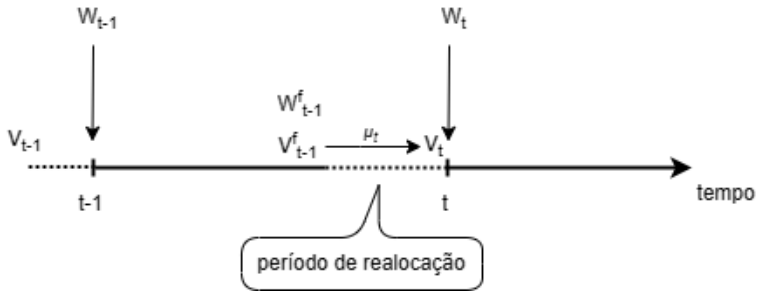


Figura 3 – Portfolio optimization process with transaction remainder factor trading fee.

Como o modelo do fator residual de transações não sofre as limitações descritas no modelo do modificador do vetor de pesos, é preferível utilizá-lo na grande maioria das vezes.

2.2.3 Métricas de desempenho

De modo a comparar diferentes sistemas de otimização de portfólio, é importante utilizar algumas métricas de desempenho. Assim como em (JIANG; XU; LIANG, 2017), as métricas adotadas neste trabalho são o valor final acumulado do portfólio (fAPV, do inglês *final accumulative portfolio value*), o máximo *drawdown* (MDD, do inglês *maximum drawdown*) e a razão de Sharpe (SR, do inglês *Sharpe ratio*). Tais métricas são descritas abaixo.

Valor final acumulado do portfólio (fAPV): Essa métrica de desempenho é responsável por calcular o lucro gerado pelo portfólio durante um intervalo de tempo específico. Quanto maior for o fAPV, maior é o lucro gerado pela abordagem implementada. Sendo V_0 o valor inicial do portfólio e V_{T+1} o valor final do portfólio, é possível calcular o valor final acumulado do portfólio da seguinte forma:

$$fAPV = \frac{V_{T+1}}{V_0}. \tag{2.7}$$

Máximo DrawDown (MDD): O fAPV é uma boa métrica de lucro, mas ele ignora o risco assumido para alcançar esse lucro. Uma das formas de modelar o risco é analisar o *drawdown* máximo da série temporal do valor de um portfólio. Pode-se definir o máximo *drawdown* como a maior queda da série temporal (Magdon-Ismail et al., 2004):

$$MDD = \max\left(\max_{t<\tau} \frac{V_t - V_\tau}{V_t}\right). \tag{2.8}$$

Sharpe Ratio (SR): Outra métrica relacionada ao risco do portfólio é a razão de Sharpe (SHARPE, 1994). Essa é uma métrica do retorno médio ajustado pelo risco, que é definido como o retorno médio livre de risco dividido pelo seu desvio padrão:

$$SR = \frac{\mathbb{E}_t[\rho_t - \rho_F]}{\sqrt{var_t[\rho_t - \rho_F]}}, \tag{2.9}$$

em que $\rho_t = V_t/V_{t-1}$ é a razão de retorno, ρ_F é a razão de retorno do investimento livre de risco (que costuma ser assumido como zero) e o denominador é o desvio padrão do numerador.

Um detalhe importante é que a razão de Sharpe costuma ser apresentada de forma anualizada, em que se assume que os retornos são calculados de forma anual. Uma boa aproximação para transformar o SR em anualizado é multiplicar seu valor por $\sqrt{252}$.

Dadas as métricas apresentadas, um bom portfólio deve possuir os maiores $fAPV$ e SR possíveis, visto que essas são métricas de retorno. Além disso, ele deve possuir um MDD baixo para garantir que a abordagem é o mais livre de risco quanto possível.

2.2.4 Hipóteses assumidas

Em muitas soluções para o problema de otimização de portfólio, em especial aquelas baseadas em aprendizado por reforço, é necessário que o sistema finja estar em um momento passado do mercado e sinta a simulação da passagem de tempo (JIANG; XU; LIANG, 2017). Para permitir isso, duas hipóteses são assumidas:

Não há *slippage*: Assume-se que o mercado tenha liquidez consideravelmente alta, de modo que não ocorre *slippage* ao se executar ordens, ou seja, as ordens são imediatamente completadas pelo último preço de mercado.

Não há impacto no mercado: As ordens executadas pelo sistema não afetam o mercado pois é assumido que a quantidade de ações administradas é pequena.

Essas hipóteses conseguem simplificar consideravelmente a simulação do problema de otimização de portfólio visto que, no mundo real, um alto volume executado pode afetar os preços de mercado e também não há garantia de que o sistema conseguirá completar as ordens sem *delay* e a preço de mercado.

2.3 Redes neurais artificiais

Grande parte da larga adoção da inteligência artificial no dia-a-dia da sociedade moderna se deve ao advento das redes neurais artificiais, que podem ser aplicadas em diversos problemas de aprendizado de máquina, como nos problemas de classificação de padrões e de regressão. O bom desempenho das redes neurais se deve principalmente pelo teorema da aproximação universal (HORNIK; STINCHCOMBE; WHITE, 1989), que declara que uma rede desse tipo consegue aproximar funções contínuas no domínio $\mathbb{R}^n$ desde que possua ao menos uma camada oculta com um número finito de neurônios. Dessa forma, ao assumir que existe uma função que mapeia os dados de entrada com a solução ideal do problema, é possível ensinar uma rede neural, por meio de algoritmos de aprendizado de máquina, a aproximar essa função, sendo o desempenho dessa aproximação principalmente influenciada pela arquitetura da rede, pelo algoritmo utilizado e pelos hiperparâmetros de treinamento.

2.3.1 Rede perceptron multicamada

Define-se uma rede neural do tipo perceptron multicamada como aquela que é composta por uma série de camadas que fazem a operação:

$$\mathbf{Y_i} = f_i(\mathbf{W_i X_{i-1}} + \mathbf{B_i}), \tag{2.10}$$

em que i é a camada da rede, $\mathbf{X_{i-1}}$ são os dados de entrada da camada (um vetor de dimensão igual à dimensão dos dados de entrada), $\mathbf{W_i}$ denota a matriz de pesos da camada, $\mathbf{B_i}$ denota a matriz de viés da camada e f_i é uma função não linear chamada de *função de ativação*. Os dados de saída da rede são gerados pela concatenação dessa operação ao longo de todas as camadas da rede (exceto a camada de entrada, que somente recebe os dados de entrada), de modo que, na primeira camada oculta, tem-se $\mathbf{X_{i-1}} = \mathbf{X_0}$, que são os dados de entrada e, na camada de saída, $\mathbf{Y_i}$ representa a saída da rede. A figura 4 mostra a representação visual dessa concatenação em uma rede de duas camadas ocultas.

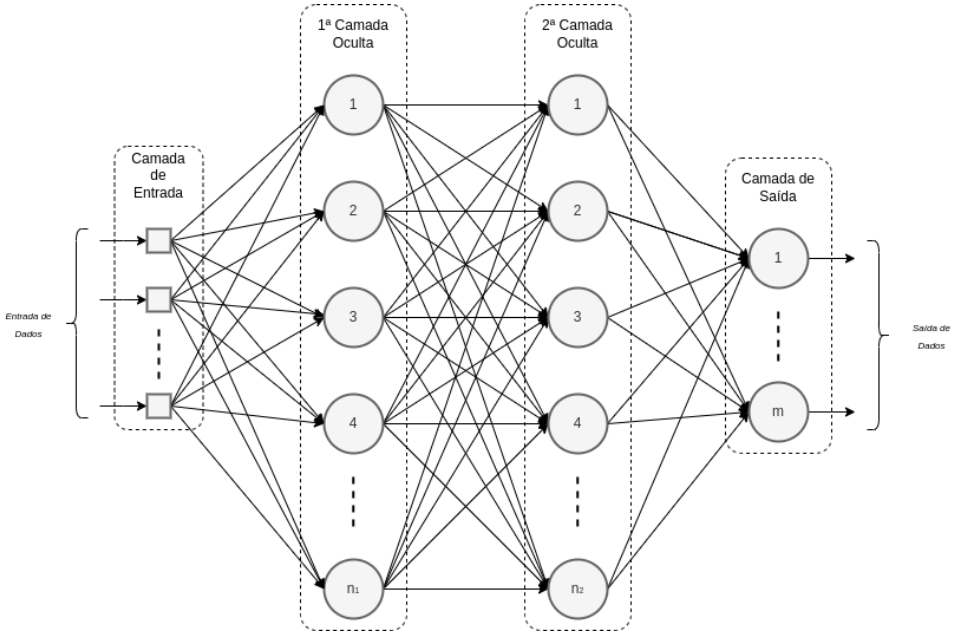


Figura 4 – Exemplo de rede perceptron multicamada com duas camadas ocultas.

O objetivo, ao se treinar uma rede neural, é descobrir as matrizes de pesos $\mathbf{W_i}$ e de viés $\mathbf{B_i}$ ideal em cada uma das camadas da rede de modo que a saída aproxime alguma função que resolva o problema, sendo o domínio dessa função $\mathbb{R}^n \rightarrow \mathbb{R}^m$, em que n é a dimensão dos dados de entrada e m é a dimensão dos dados de saída.

2.3.2 Redes neurais convolucionais

Uma limitação da rede perceptron multicamada está no fato de que os dados de entrada e de saída precisam ser vetores n-dimensionais de modo que, para aplicá-la, por exemplo, em imagens (que costumam ser representadas por uma matriz bidimensional - ou até tridimensional - de *pixels*), é necessário linearizar os dados, criando um vetor de entrada com dimensões muito grandes e perdendo dados relacionados ao relacionamento de *pixels* próximos. Para lidar com essa questão, costumam-se utilizar redes neurais convolucionais, que aplicam filtros convolucionais de modo a manter na rede o conhecimento relacionado ao posicionamento de cada um dos valores dos dados de entrada.

A figura 5 mostra a ideia por trás da operação de convolução em uma imagem. Nessa operação, um *kernel* representado pelo quadrado vermelho pontilhado percorre toda a imagem agrupando os *pixels* próximos e aplicando sobre eles uma transformação linear - ou, como costuma ser chamado, um filtro - que tem como resultado um valor que os represente. Dependendo do filtro aplicado, o resultado da operação convolucional pode representar uma característica da imagem, como um determinado tipo de forma ou geometria.

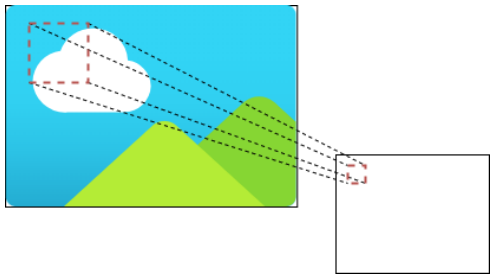


Figura 5 – Filtro convolucional sendo aplicado a uma imagem.

A concatenação desses diversos filtros convolucionais forma uma rede neural convolucional, que tem como objetivo aprender as transformações lineares capazes de extrair características espaciais dos dados de entrada que melhor descrevam os dados. A figura 6 possui a representação de uma rede neural convolucional aplicada a uma imagem formada por três canais de dados representado os valores de cada *pixel* em vermelho, verde e azul (RGB). Pode-se ver que diversos filtros diferentes são aplicados sobre esses canais de forma a gerar um *output* com ainda mais canais, sendo que ao invés de representarem os valores de cores, eles representam características espaciais locais da imagem.

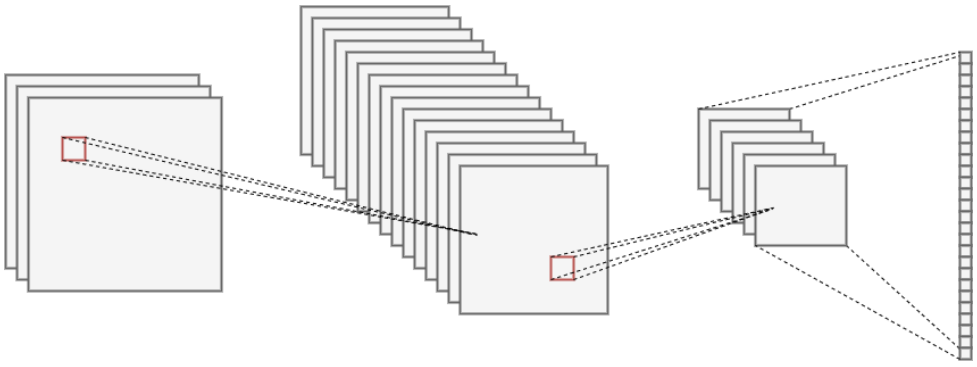


Figura 6 – Exemplo de linearização de uma imagem RGB em uma rede neural convolucional.

Assim como nas redes de perceptron multicamadas, as redes convolucionais podem ter um número variado de camadas e quanto mais profundas forem, maior será a concatenação desse processo de aplicação de filtros. Outra similaridade está no fato de que esse processo de concatenação também conta com uma função de ativação não-linear que

é aplicada após as transformações lineares dos filtros, algo importante para que a rede consiga aprender de fato.

Por fim, ressalta-se que, na grande maioria das vezes, esses dois tipos de rede trabalham em conjunto: em geral, a rede convolucional é responsável por extrair características dos dados de entrada e, por fim, linearizá-los em um vetor unidimensional (como na figura 6), que pode ser utilizado como entrada na rede perceptron multicamadas.

2.3.3 Funções de ativação

Ao longo das seções 2.3.1 e 2.3.2, falou-se sobre a utilização de funções não lineares entre as camadas de redes neurais. Essas funções são importantes pois, sem elas, as diversas camadas da rede seriam análogas a uma única transformação linear fazendo com que a rede neural não fosse diferente de um simples modelo de regressão linear. Nas arquiteturas apresentadas neste documento, duas funções de ativação são utilizadas:

Rectified Linear Unit (ReLU): Essa é uma função simples mas muito utilizada como função de ativação na área de aprendizado profundo. A função $ReLU$ é não-linear pois atribui valor zero a qualquer entrada negativa. Matematicamente, a função é dada por

$$ReLU(x) = max(0, x), \tag{2.11}$$

em que $max(a, b)$ denota o maior valor entre a e b, sendo $a, b \in \mathbb{R}$.

Softmax: Essa função costuma ser utilizada na última camada de uma rede neural quando há a necessidade de que a saída da rede represente uma distribuição de probabilidades. Isso é possível pois, para cada valor $X(i)$ de um vetor unidimensional de entrada $\mathbf{X}$ de tamanho N, a função aplica a transformação

$$Softmax(\mathbf{X}(i)) = \frac{e^{\mathbf{X}(i)}}{\sum_{j=1}^{N} e^{\mathbf{X}(j)}}. \tag{2.12}$$

Isso gera um novo vetor de mesmo tamanho cujos elementos possuem valor entre 0 e 1 e soma exatamente igual a 1, assim como em uma distribuição de probabilidades discreta.

2.4 Aprendizado por reforço no problema de otimização de portfólio

Uma forma de resolver o problema de otimização de portfólio é treinar um agente usando aprendizado por reforço. Nessa abordagem, o agente aprende uma política (π) ao interagir diretamente com o ambiente. A cada instante de tempo, o ambiente provê ao agente observações (O_t) que definem um estado (S_t) do sistema, que pode ser utilizado pelo agente para escolher uma ação (A_t) a ser executada. Após a execução da ação, o

ambiente também fornece ao agente uma recompensa (R_t), que é utilizado pelo algoritmo de aprendizado por reforço para saber se a ação escolhida teve efeitos positivos. O objetivo de um agente de aprendizado por reforço é encontrar uma política de ações que maximize a soma esperada das recompensas recebidas. A figura 7 representa um ciclo de treino de um agente de aprendizado por reforço interagindo com o ambiente.

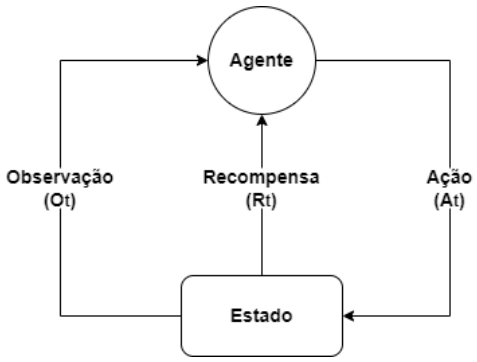


Figura 7 – Diagrama representando um ciclo do aprendizado por reforço.

Um dos grandes desafios de se utilizar aprendizado por reforço no problema de otimização de portfólio é ser capaz de lidar com estados complexos, visto que uma carteira de investimentos é composta por diversas ações, cada uma contendo sua própria série temporal. (SUTTON; BARTO, 2018) introduz, em seu livro, algumas formas de lidar com espaços de estados complexos e contínuos ao utilizar funções de aproximação como as redes neurais, que conseguem resultados consideravelmente bons em diversas tarefas. Assim, os algoritmos mais convenientes são aqueles que empregam redes neurais na aproximação de estados e, por isso, são conhecidos como algoritmos de *deep reinforcement learning* (aprendizado por reforço profundo).

2.4.1 Algumas considerações

Antes de modelar o espaço de estados, de ações e as recompensas, é importante ter em mente algumas considerações assumidas na formulação do aprendizado por reforço aplicado ao problema de otimização de portfólio apresentado nesta seção.

Para respeitar as hipóteses introduzidas na seção 2.2.4, o aprendizado por reforço deve ser modelado como um problema de *contextual bandits* ao invés de um problema de aprendizado por reforço completo. A grande diferença é que, no aprendizado completo, uma ação A_t executada no estado S_t não só afeta a recompensa R_t que o agente receberá como também afeta o próximo estado S_{t+1} em que o agente estará no próximo instante de simulação, enquanto que, no *contextual bandits*, uma ação A_t executada no estado S_t somente afeta a recompensa R_t que o agente receberá. Pode-se dizer que o *contextual bandits* é uma formulação simplificada do aprendizado por reforço, como é possível observar na figura 8.

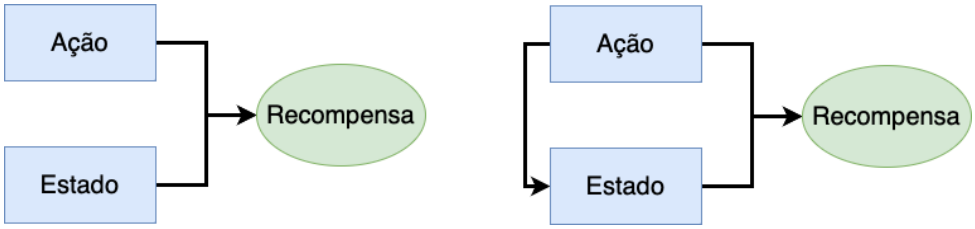


Figura 8 – Comparação entre *contextual bandits* (à esquerda) e aprendizado por reforço (à direita).

Outra consideração importante feita é que o problema é modelado como um processo de decisão de Markov (MDP, do inglês *Markov decision process*) (SUTTON; BARTO, 2018) e, portanto, respeita a propriedade de Markov, que diz que toda a informação necessária para o agente encontrar a melhor política de ações está presente no estado observado. No mundo real, considerando a grande complexidade do mercado de ações, não se pode assumir que a propriedade de Markov é verdadeira, o que forçaria o problema a ser modelado como um processo de decisão de Markov parcialmente observável (POMDP, do inglês *partially observable Markov decision process*), algo que está fora do escopo deste trabalho.

2.4.2 Representação de estados

Para capturar tendências do mercado e escolher uma ação a ser executada, o agente precisa de informação sobre as séries temporais de preços de todas as ações presentes no *portfólio*. Por isso, se o portfólio possui n ações com um histórico de preços de t intervalos de tempo no passado (ou seja, uma janela de tempo de tamanho t), é necessário uma matriz de estados de dimensões (n, t) para representar todas essas informações.

Mas uma matriz bidimensional pode não ser o suficiente visto que somente uma *feature* pode ser representada quando, na verdade, podem ser necessárias várias *features* para que o agente de fato entenda as tendências do mercado. (WENG et al., 2020) mostra que as *features* mais importantes para o problema de otimização de carteira são as séries temporais dos preços de fechamento, o preço em alta e em baixa das ações, mas *features* como volume, preço de abertura e até mesmo indicadores econômicos podem ser utilizados na representação de estados. Assim, o estado pode ser melhor representado como uma matriz multi-dimensional de tamanhos (f, n, t), em que f é o número de *features*, n é o número de ações do portfólio e t é o tamanho da janela de tempo considerada para as séries temporais. A figura 9 permite visualizar a representação de estados para o problema da otimização de portfólio.

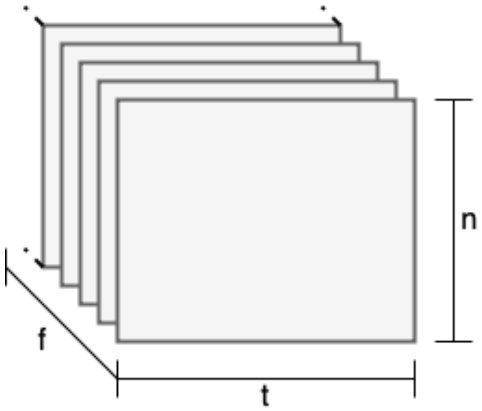


Figura 9 – Representação de estados para otimização de portfólio.

2.4.3 Espaço de ações

Considerando a formulação matemática introduzida na seção 2.2.1, o espaço de ações para agentes de aprendizado por reforço treinado para o problema de otimização de portfólio é idêntico ao vetor de pesos $\mathbf{W}_t$ e segue as mesmas restrições. Note que, no contexto de agentes de *deep reinforcement learning*, essas restrições são facilmente implementadas utilizando uma função de ativação *softmax*.

2.4.4 Função de recompensa

Muitas funções podem ser utilizadas para modelar a recompensa do ambiente, mas a mais comum para o problema em questão é a *taxa de retorno logarítmico* r_t introduzida na equação 2.4, visto que ela está disponível naturalmente a cada espaço de simulação. Alguns artigos (ALMAHDI; YANG, 2017; BETANCOURT; CHEN, 2021) têm procurado incorporar alguns indicadores (em especial, o index SR) na função de recompensa de modo que o agente aprenda, também, a lidar com o risco do mercado.

2.4.5 Política de ações

Considerando as formulações apresentadas ao longo desta seção, define-se uma política de ações como uma função $f : \mathbb{R}^{f \times n \times t} \rightarrow \mathbb{R}^{n+1}$ que mapeia o espaço de estados do agente com um respectivo vetor portfólio $\mathbf{W}_t$, sendo f, n e t os valores apresentados na seção 2.4.2. Essa política de ações pode ser de dois tipos:

Política determinística: A política funciona de fato como uma função matemática simples, mapeando cada estado a uma ação específica. Com isso, um agente seguindo esse tipo de política escolherá uma mesma ação sempre que um mesmo estado for observado.

Política estocástica: Nesse caso, a política de ações é dada por uma distribuição de probabilidades, de modo que, para um mesmo estado, é possível que o agente escolha fazer uma ação diferente.

Ao longo deste trabalho, as políticas implementadas serão determinísticas pois políticas estocásticas são mais difíceis de serem implementadas considerando a complexidade do espaço de ações do problema. Além disso, políticas determinísticas podem ser facilmente implementadas utilizando redes neurais convolucionais para processar o espaço de estados e a função de ativação *softmax* para lidar com as restrições do espaço de ações.

2.4.6 Políticas convolucionais

Um dos grandes desafios de se trabalhar no problema de otimização de carteiras é saber tratar independentemente os dados de diversas ações mas uni-los para serem utilizados no aprendizado da atuação do sistema. Em (JIANG; XU; LIANG, 2017), Jiang *et al.* propôs uma arquitetura chamada *ensemble of identical independent evaluators* (EIIE), que permite que fluxos de dados diferentes compartilhem parâmetros em um modelo de aprendizado por reforço. Tal arquitetura utiliza-se de redes neurais convolucionais cujos filtros são aplicados ao longo de cada série temporal presente no espaço de estados apresentado na seção 2.4.2 de forma independente (ou seja, os filtros percorrem cada série temporal ao longo do tempo independentemente). Tais filtros conseguem, com isso, capturar as tendências temporais das ações por meio de *features* e, ao concatenar essas *features* em um único tensor final, é possível que todas as ações da carteira compartilhem os mesmos parâmetros de aprendizado de máquina. A figura 10 contém um diagrama da arquitetura EIIE presente no trabalho de Jiang *et al.*

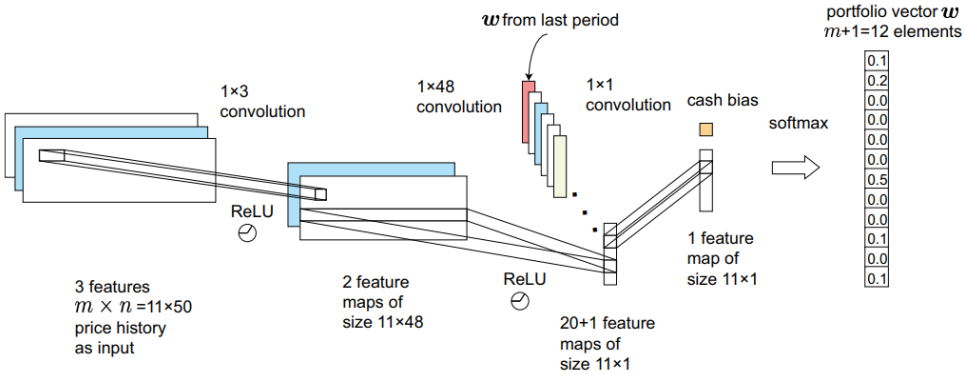


Figura 10 – Representação da arquitetura EIIE retirada de (JIANG; XU; LIANG, 2017).

Posteriormente, (SHI et al., 2019) combinou essa ideia com redes do tipo *inception* (SZEGEDY et al., 2015), comumente utilizadas na classificação de imagens para capturar pedrões em diferentes escalas na imagem. Tal arquitetura proposta por Shi *et al.* (chamada EI^3, do inglês *ensemble of identical independent inception*) faz um processo

parecido com a arquitetura EIIE, mas utiliza a ideia de múltiplas escalas para capturar tendências de mercado no curto, médio e no longo prazo, algo que *traders* costumam fazer naturalmente por meio de indicadores clássicos de mercado. Usando essa técnica, o EI^3 alcançou bons resultados no problema de otimização de portfólio, conseguindo superar o EIIE na maximização de lucros. A figura 11 mostra a implementação EI^3 introduzida por Shi *et al.*

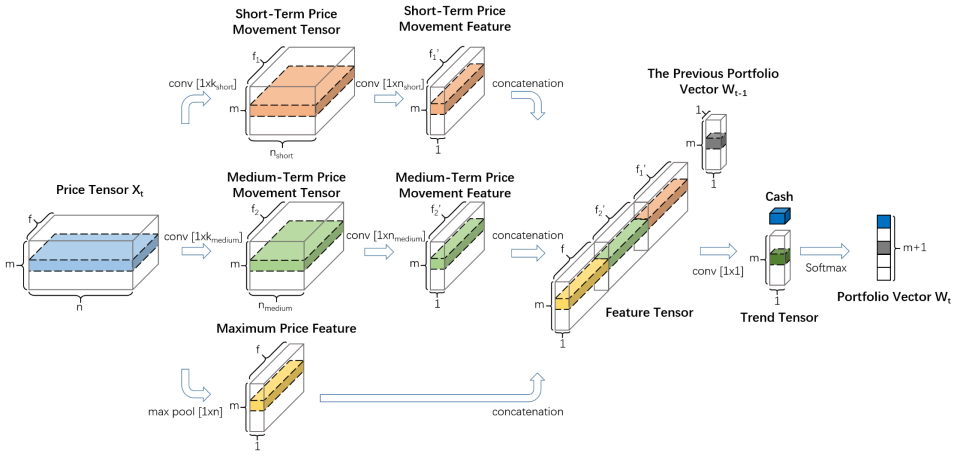


Figura 11 – Representação da arquitetura EI^3 retirada de (SHI et al., 2019).

É importante ressaltar que ambas as arquiteturas têm como entrada o último vetor portfólio definido pelo agente, ou seja, a última ação executada afeta a escolha de ação do agente. Isso não costuma ser feito em algoritmos de aprendizado por reforço, mas nesse caso, essa técnica melhora o desempenho pois permite que o agente conheça a última alocação feita e possa lidar com taxas de corretagem.

2.4.7 Algoritmos de treinamento

Os algoritmos de aprendizado por reforço são responsáveis por fazer com que a política de ações de determinado agente alcance um bom desempenho considerando a recompensa a longo prazo. No contexto do aprendizado por reforço profundo, isso corresponde a utilizar as interações do agente com o ambiente para melhorar os parâmetros da rede neural responsável por gerar a política de ações utilizando estratégias como gradiente descendente (ou ascendente, dependendo do objetivo da otimização) de forma muito parecida com o aprendizado supervisionado. Na maioria dos algoritmos, o processo funciona da seguinte forma:

1. O agente interage com o ambiente ao executar uma ação. Essa ação é salva junto com dados úteis para o algoritmo específico (como a recompensa recebida e o estado antes e depois de se fazer a ação) em uma estrutura de memória, geralmente chamada de *replay buffer*;

2. Quando o *replay buffer* tiver uma quantidade razoável de experiências, um *batch* de dados é recuperado dele da forma como o algoritmo determinar: os dados podem ser, por exemplo, aleatórios ou ordenados;

3. Esse *batch* de dados é utilizado para treinar a(s) rede(s) neural(is) responsável(is) pelo comportamento do agente, seguindo a formulação específica do algoritmo escolhido;

4. O processo retorna para o item 1 até que alguma condição de parada (como número de *steps* de treinamento ou algum valor de desempenho) seja atingido.

Ressalta-se que o tamanho do *replay buffer* costuma ser um dos hiperparâmetros de treinamento e que, conforme ele vai se enchendo, experiências mais antigas costumam ser substituídas por experiências novas já consequentes da rede neural treinada, fazendo com que o agente vá se especializando cada vez mais na solução do problema. Isso, porém, gera outro problema: se um agente segue sempre uma determinada estratégia já aprendida ao longo do treinamento, ele nunca explorará outras possibilidades, ocasionando em uma solução que pode ser boa, mas não necessariamente a melhor. Essa questão é um dos pontos de grande estudo na área de aprendizado por reforço e é conhecida como "o problema do *exploration* e *exploitation*"(SUTTON; BARTO, 2018). Os diversos algoritmos existentes tentam, cada vez mais, trazer um balanço entre exploração e aproveitamento.

Há, também, uma classificação bem definida que divide os algoritmos de aprendizado por reforço profundo em três possíveis abordagens: os algoritmos atores, críticos e atores-críticos.

Algoritmos atores: Esses algoritmos tem o objetivo de treinar diretamente a política de ações, que é dada por uma rede neural artificial, por meio da maximização de uma métrica conhecida. Esse tipo de algoritmo é interessante quando o desenvolvedor sabe uma forma de medir o desempenho da política, apesar de haver implementações que procuram explorar o espaço de ações. São exemplos os algoritmos de *Policy Gradient*, como o *REINFORCE* (WILLIAMS, 1992).

Algoritmos críticos: Já nesse tipo de algoritmo, redes neurais costumam ser responsáveis por aproximar o valor de se tomar uma ação em todos os possíveis estados do agente. Essa aproximação costuma ser chamada de *Q-value* e a política do agente é comumente definida como a ação que possui maior *Q-value* em um determinado estado. Esses algoritmos tendem a ter problemas para lidar com ações contínuas, apesar de algumas adaptações permitirem isso. São exemplos os algoritmos *Q-learning* (SUTTON; BARTO, 2018), *Double Q-Learning* (van Hasselt; GUEZ; SILVER, 2015) e *Rainbow* (HESSEL et al., 2017).

Algoritmos atores-críticos: Esses algoritmos combinam as estratégias dos algoritmos atores e críticos utilizando, portanto, pelo menos duas redes neurais na arquitetura

do agente: a rede neural atora e a rede crítica. A rede neural atora é basicamente a política de ações, enquanto que a crítica funciona de forma análoga à rede dos algoritmos críticos. A ideia é que a rede crítica aprende os *Q-value* das diversas ações em diversos estados e esse valor é utilizado para atualizar os parâmetros da rede neural atora. Esses algoritmos são os mais famosos atualmente: são exemplos *Deep Deterministic Policy Gradient* (LILLICRAP et al., 2015) e o *Proximal Policy Optimization* (SCHULMAN et al., 2017).

Neste trabalho, o foco será na utilização de um algoritmo ator específico para o problema de otimização de portfólio, apresentado a seguir.

2.4.8 *Policy gradient* para otimização de portfólio

Além da arquitetura EIIE, Jiang *et al.* também introduziu em seu trabalho (JIANG; XU; LIANG, 2017) um algoritmo de *policy gradient* específico para a resolução do problema da otimização de portfólio. O método introduzido por ele tem como principal vantagem o fato de não haver a necessidade de se utilizar mais de uma rede neural, algo que, como argumenta o autor, faz o treinamento ser mais estável que outros algoritmos. Posteriormente, Liang *et al.* (LIANG et al., 2018) demonstrou que, de fato, esse algoritmo de *policy gradient* alcança um desempenho maior que outros algoritmos como PPO (SCHULMAN et al., 2017) e DDPG (LILLICRAP et al., 2015).

Assim como outros algoritmos de aprendizado por reforço, o algoritmo proposto faz uso de um *replay buffer* que é utilizado para salvar as interações do agente com o ambiente de modo a permitir um treinamento em *batch*. A principal diferença está no fato de esse *buffer* guardar os dados de forma sequencial e, além disso, quando dados são recolhidos, eles são imediatamente apagados do *replay buffer*, de modo que só ficam presentes nele dados temporais ainda não utilizados no treinamento durante um episódio.

Essa ordenação das experiências no *replay buffer* é importante pois permite que o algoritmo consiga maximizar a função objetivo, que pode ser calculada entre dois instantes de tempo T_1 e T_2 por meio da fórmula

$$\sum_{t=T_1}^{T_2} \frac{ln(\mu_t(\mathbf{W_t} \cdot \mathbf{P_t}))}{T_2 - T_1 + 1}, \tag{2.13}$$

em que μ_t representa o fator residual de transações no instante t, $\mathbf{P_t}$ representa um vetor contendo a variação de preços de cada ação do instante t e $\mathbf{W_t}$ representa a ação feita no período t. Note que, como utilizamos uma política determinística ϕ, $\mathbf{W_t} = \phi(s_t, \mathbf{W_{t-1}}|\theta^\phi)$ (considerando uma política recorrente), ou seja, o valor de $\mathbf{W_t}$ pode ser calculado diretamente utilizando a rede neural, o que permite o cálculo dos gradientes para a otimização. O pseudocódigo 1 contém o algoritmo de *policy gradient* para otimização de portfólio.

Algorithm 1 *Policy gradient* para otimização de portfólio

Inicializa a rede neural de política $\phi(s, W|\theta^{\phi})$
Inicializa o *replay buffer* R
for $ep = 1$ até `num_episodes` **do**
    Reseta o ambiente e recebe o estado inicial s_1
    for $step = 1$ até `episode_size` **do**
        Seleciona ação $W_t = \phi(s_t, W_{t-1}|\theta^{\phi})$
        Executa ação no ambiente e observa r_t e s_{t+1}
        Salva (s_t, W_{t-1}, P_t, μ) no *replay buffer*
        if |R| = `batch_size` **then** ▷ *buffer* com tamanho suficiente para um *batch*
            Atualiza os parâmetros da rede neural usando gradiente ascendente:

$$\nabla_{\theta^{\phi}} J = \nabla_{\theta^{\phi}} \sum_{t=step}^{step+batch\_size-1} \frac{ln(\mu_t(W_t \cdot P_t))}{batch\_size}$$

        end if
    end for
    if |R| > 0 **then** ▷ Ainda há experiências no *buffer* que devem ser consideradas
        Atualiza os parâmetros da rede neural usando gradiente ascendente:

$$\nabla_{\theta^{\phi}} J = \nabla_{\theta^{\phi}} \sum_{t=step}^{step+|R|-1} \frac{ln(\mu_t(W_t \cdot P_t))}{|R|}$$

    end if
end for

3 Metodologia do trabalho

Para alcançar seu objetivo, o trabalho segue uma metodologia dada pelas etapas descritas a seguir.

3.1 Revisão bibliográfica

Antes de tudo, é necessário estudar e compreender as tendências dos sistemas de otimização de carteiras de ações com aprendizado por reforço para ser capaz de definir o contexto em que as arquiteturas convolucionais se encontram na área de pesquisa. Dessa forma, uma revisão bibliográfica de alguns trabalhos publicados posteriormente ao (JIANG; XU; LIANG, 2017) deve ser feita, permitindo uma análise teórica da área de pesquisa e das suas lacunas em aberto.

3.2 Desenvolvimento de um ambiente de simulação

Após a revisão bibliográfica, desenvolve-se um ambiente de simulação que permita a aplicação da formulação apresentada nas seções 2.2 e 2.4. Assim, esse ambiente deve ser capaz de, dado um conjunto de dados financeiros, simular a passagem de tempo de um agente de aprendizado por reforço atuando no mercado, determinar as consequências das ações tomadas pelo agente e fornecê-lo uma recompensa associada a essas ações. Ressalta-se que tal ambiente deve seguir as hipóteses e considerações assumidas nas seções 2.2.4 e 2.4.1. Por fim, espera-se que o ambiente desenvolvido seja *open source*, permitindo que outros pesquisadores o utilizem no futuro, e que siga os padrões do *Gymnasium* (TOWERS et al., 2023), um dos *frameworks* de simulação de ambientes de aprendizado por reforço mais utilizado no mundo.

3.3 Testes com arquiteturas notáveis

Tendo o ambiente de simulação em mãos, a implementação de duas arquiteturas notáveis (a EIIE, por Jiang *et. al* (JIANG; XU; LIANG, 2017), e a EI^3, por Shi *et. al* (SHI et al., 2019)) é feita e testada exaustivamente com dados do mercado americano, visando reproduzir os resultados encontrados em outros artigos. Posteriormente, a melhor dessas arquiteturas é utilizada em testes no mercado brasileiro. Essa etapa é importante pois as duas arquiteturas tem como objetivo otimizar uma carteira de criptomoedas e, por isso, o desempenho no mercado brasileiro ainda é uma incógnita. Além disso, como o mercado

brasileiro é volátil em comparação com o americano, é interessante fazer testes neste último, para ver como o algoritmo se comporta em diferentes situações de variabilidade da bolsa.

4 Especificação de requisitos

Neste capítulo, são apresentados os requisitos funcionais e não-funcionais do sistema final a ser desenvolvido.

4.1 Requisitos funcionais

Aquisição de dados: O trabalho deve ser capaz de adquirir os dados das séries temporais do histórico de preços das ações da carteira. Essas séries temporais precisam possuir granularidade diária e dados sobre o preço máximo, mínimo e de fechamento das respectivas ações. É importante ressaltar que os dados adquiridos precisam estar tratados, de modo que as séries temporais de todas as ações possuam a mesma quantidade de dados e não possuam dados faltantes;

Ambiente de simulação: Uma vez que sistemas de aprendizado por reforço aprendem por meio da experiência adquirida ao interagir com um determinado ambiente, é necessário o desenvolvimento de um ambiente que consiga simular a dinâmica do mercado de ações ao longo do tempo considerando as séries históricas das ações da carteira seguindo a formulação apresentada na seção 2.2;

Métricas ao final da simulação: De modo a poder comparar o desempenho de arquiteturas com o de modelos do estado-da-arte da otimização de carteiras, é necessário que as principais métricas de desempenho empregadas na área possam ser calculadas e exibidas ao final da simulação. Em especial, utilizaremos o ganho percentual da carteira, o *drawdown* máximo e o índice de Sharpe.

Reprodução de arquiteturas notáveis: O estudo apresentado deve reproduzir arquiteturas consideradas notáveis pela área de pesquisa e avaliar seu desempenho em mercados relevantes.

***Open source*:** Todo o trabalho apresentado ao longo deste documento deve ser disponibilizado de forma *open source* por meio de contribuições à biblioteca *FinRL* (LIU et al., 2022), biblioteca com recursos de aprendizado por reforço para finanças.

4.2 Requisitos não-funcionais

Testes no mercado brasileiro: Entre os testes que serão feitos com arquiteturas notáveis, deve-se, em ao menos um deles, utilizar portfólios do mercado brasileiro;

Boa documentação: Os códigos-fonte do ambiente de simulação, do algoritmo de treinamento e das arquiteturas utilizadas nos testes devem estar bem documentados de modo que qualquer desenvolvedor que queira utilizá-los tenha as informações necessárias para fazê-lo;

Utilização de *frameworks* bem estabelecidos: De modo a dar mais confiabilidade ao sistema e evitar que problemas de desempenho estejam associados a áreas que não são o foco do estudo proposto, este projeto deve utilizar, sempre que possível, *frameworks open source* bem estabelecidos na área.

5 Revisão bibliográfica

Neste capítulo, é feita uma breve análise dos trabalhos mais relacionados ao descrito neste documento. Essa análise tem o objetivo de compreender o que outros pesquisadores já investigaram, quais metodologias mais utilizam, qual a tendência da área de pesquisa e, principalmente, quais as suas lacunas.

5.1 Trabalhos relacionados

Existem duas formas principais de se desenvolver um sistema de otimização de carteiras. A primeira, que não é o foco deste trabalho e que geralmente alcança desempenho menor, é considerar um portfólio como uma série de ações não relacionadas, como um *single asset* com múltiplos agentes, um para cada ação da carteira. Essa estratégia foi utilizada, por exemplo, por (TRAN; Pham-Hi; BUI, 2023) para criar um agente que, para cada ação, consegue gerar um sinal de compra e venda de ações.

Por outro lado, a segunda forma consegue alcançar desempenho maior ao considerar *features* de todas as ações do portfólio simultaneamente. Essa abordagem foi popularizada por Jiang *et al.*, que, em seu trabalho, desenvolveu a arquitetura EIIE, sigla de *ensemble of identical independent evaluators* (JIANG; XU; LIANG, 2017), que utiliza redes neurais convolucionais para combinar as séries temporais dos preços de diversas ações permitindo que o agente utilize um único modelo de aprendizado por reforço. Esse trabalho também introduziu uma sólida formulação matemática para o problema e um algoritmo de *policy gradient* que permite o treinamento eficaz de agentes, alcançando bons resultados em comparação com técnicas clássicas.

A técnica de Jiang *et al.* teve grande influência nessa área de pesquisa pois abriu a possibilidade de se utilizar técnicas comumente aplicadas à visão computacional no problema de otimização de carteiras. (LIANG et al., 2018), por exemplo, aplicou ideias de *data augmentation* (MUMUNI; MUMUNI, 2022) ao EIIE, ao adicionar um ruído randômico às series temporais das ações e teve bom desempenho. Posteriormente, (SHI et al., 2019) explorou as redes de *inception* (SZEGEDY et al., 2015) para que *features* de diversas janelas de tempo (longo, médio e curto prazo) das séries temporais das ações de carteiras fossem geradas, criando a arquitetura EI^3, que desempenha consideravelmente melhor que a arquitetura EIIE no mercado de criptomoedas. (WENG et al., 2020), por outro lado, seguiu a ideia das redes *Xception* em conjunto com técnicas de *attention gating* e (SOLEYMANI; PAQUET, 2020) utilizou arquiteturas de *autoencoders* em conjunto com o EIIE para combinar as séries temporais dos preços das ações com diversos outros indicadores. Além desses, (JANG; SEONG, 2023) segue a mesma proposta de tentar

utilizar indicadores econômicos técnicos mas adiciona também uma matriz de correlação das ações do portfólio no estado do agente para que ele compreenda quais ações têm comportamentos parecidos. Para evitar que a representação de estados seja muito grande, Jang *et al.* faz uso da decomposição de Tucker (TUCKER, 1966) para compactar a matriz de estados.

Alguns artigos tentaram modificar alguns pontos na formulação e no algoritmo introduzidos por Jiang *et al.* (KANG; ZHOU; KANG, 2018), por exemplo, tentou treinar uma versão simplificada da arquitetura EIIE com o algoritmo A3C (MNIH et al., 2016), mas apesar de a convergência ocorrer mais rapidamente, o resultado dos testes foram insatisfatórios. (YU et al., 2019), por outro lado, faz uso do algoritmo DDPG (LILLICRAP et al., 2015) para treinar uma arquitetura convolucional em conjunto com três módulos: um de *data augmentation*, outro de predição de preços e, por fim, mais um de *imitation learning* (ZHENG et al., 2022). Com esses módulos, Yu *et al.* conseguiu alcançar desempenho melhor que arquiteturas clássicas. (PARK; SIM; CHOI, 2020) faz o treinamento utilizando *Deep Q-Learning* (SUTTON; BARTO, 2018) fazendo uso de um espaço de ações discreto e também consegue desempenhar melhor que todos os algoritmos clássicos. (BETANCOURT; CHEN, 2021) faz algumas mudanças na formulação do problema de modo a permitir um tamanho variável de portfólio além de utilizar o algoritmo PPO (SCHULMAN et al., 2017) para o treinamento, obtendo bons desempenhos tanto em métricas de lucro quanto de risco. (WANG; KU, 2022) faz algumas modificações no algoritmo DDPG de forma a torná-lo hierárquico e distribuído, permitindo a criação de um agente com parâmetro de sensibilidade ao risco ajustável. Por fim, (SONG et al., 2023) introduz um algoritmo chamado SPDQ (*Stochastic Policy with Distributional Q-network*) para conseguir treinar uma política estocástica e alcança resultados melhores do que as políticas determinísticas previamente utilizadas.

Já outros artigos implementam uma solução própria muito diferente da apresentada por Jiang *et al.* (ALMAHDI; YANG, 2017), por exemplo, abandona a ideia de redes neurais convolucionais e faz uso de redes neurais recorrentes. Ao fim do trabalho, Almahdi *et al.* cria um sistema *trader* completo que consegue desempenho melhor do que fundos *hedge*. (LIU et al., 2018) faz uso de um espaço de ações e de um espaço de estados completamente diferente de outros trabalhos: o agente desenvolvido utiliza somente o último preço das ações presentes no portfólio e o rebalanceamento não modifica o vetor portfólio, mas sim define quais ações comprar, vender e segurar. (LIM; CAO; QUEK, 2022) faz uso de indicadores como MACD e EMA (VAIDYA, 2020) ao invés das séries temporais das ações e define, a cada instante de tempo, a melhor ação da carteira, investindo nela.

Além das pesquisas anteriores, uma parte dos pesquisadores têm focado em desenvolver soluções com múltiplos agentes para tornar o sistema total mais versátil e generalista. (WU et al., 2021a), por exemplo, faz uso de dois agentes de aprendizado por reforço -

um agente *long* e outro *short* - de modo que um agente é especializado em definir quais ações vender e outro em quais comprar. A cada instante de tempo, o agente cuja ação será considerada é escolhido por meio da estratégia *equity market neutral* (PATTON, 2009). (LIN et al., 2022), por outro lado, faz uso de diversos agentes, sendo alguns deles locais, que são treinados para trabalhar com risco baixo, e um agente global, que concatena os agentes locais de modo a maximizar o lucro. Essa ideia foi capaz de alcançar desempenho maior que o trabalho de Jiang *et al.* em diversos indicadores de desempenho. Outra arquitetura multiagente cujo desempenho ultrapassa a arquitetura EIIE foi apresentada em (MA et al., 2022), em que dois agentes são treinados: um para lidar com o mercado com tendência geral de subida e outro para lidar com o mercado com tendência de descida ou sem tendência aparente. A cada instante de tempo, um desses agentes será utilizado para definir a ação final do sistema. Por fim, (YANG, 2023) também faz uso de um sistema multiagente, mas, nesse caso, cada agente é treinado por um algoritmo diferente e, dependendo do momento atual do mercado, um agente toma controle do sistema. Com essa estratégia, Yang *et al.* consegue ter desempenho melhor que algoritmos clássicos.

Por fim, outra parte de pesquisadores se debruça sobre como lidar com o fato de que as ações possuem algum relacionamento entre si e não determinam séries temporais completamente independentes como considerado em (JIANG; XU; LIANG, 2017). Para resolver esse problema, (SOLEYMANI; PAQUET, 2021) combinou a utilização de *graph neural networks* (GNNs) ao EIIE e, posteriormente, (SHI et al., 2022) seguiu o mesmo caminho ao inserir GNNs em sua arquitetura EI^3. As redes grafos neurais mostraram alcançar um bom desempenho no rebalanceamento de ações ao se utilizar de uma estrutura em grafos em que as diversas ações das carteiras são consideradas nós e os relacionamentos são modelados como arestas.

5.2 Identificando tendências

Os trabalhos apresentados na seção anterior mostram uma certa tendência na área de pesquisa. Focando especificamente nos trabalhos que fazem uso da formulação apresentada em (JIANG; XU; LIANG, 2017), vê-se que diversas possibilidades relacionadas mudanças pontuais na formulação, no algoritmo ou na arquitetura convolucional já foram muito exploradas. Por outro lado, pode-se ver que há duas áreas em ascensão e que são consideravelmente recentes: a utilização de aprendizado de máquina em grafos e de sistemas multiagentes. Na figura 12, pode-se ver um panorama geral dos artigos avaliados e essa tendência é visível quando se percebe que os trabalhos que aplicam redes neurais em grafos e sistemas multiagentes começaram a surgir somente a partir de 2021.

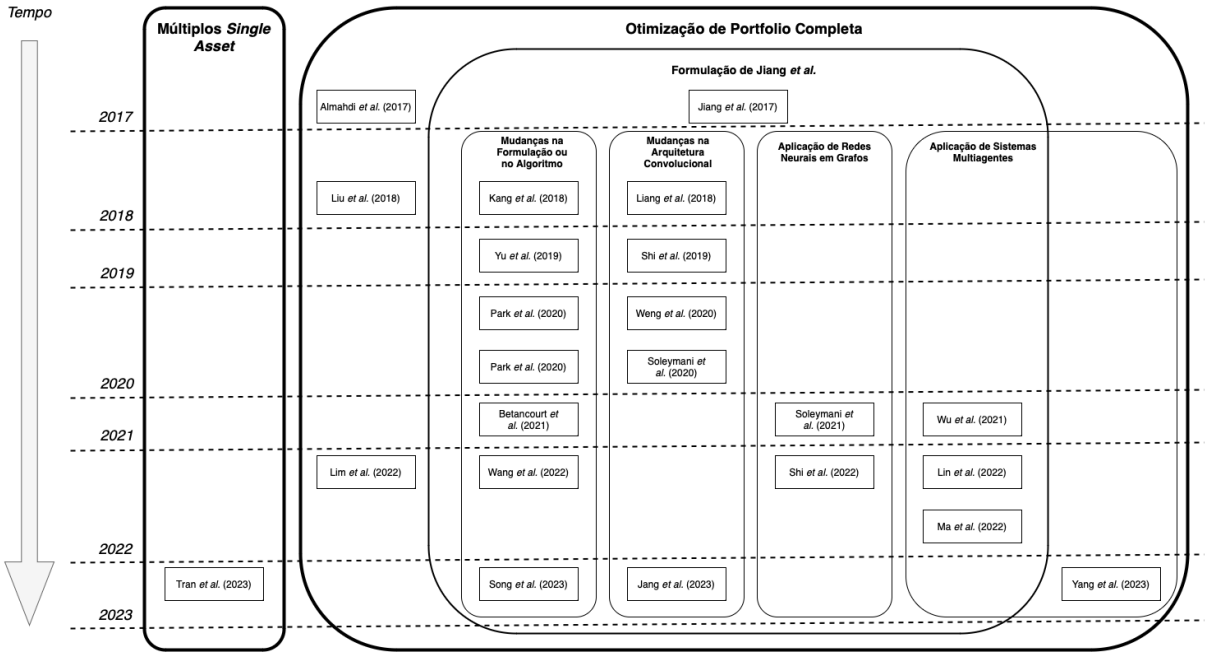


Figura 12 – Panorama dos artigos analisados.

5.3 Classificação dos trabalhos

Nesta seção, os artigos previamente apresentados são classificados quanto a alguns parâmetros mais específicos, de modo que seja possível ter um panorama das principais metodologias aplicadas.

5.3.1 Quanto ao algoritmo utilizado

Analisando a classificação desta seção, é possível perceber que os algoritmos atores-críticos dominam a área de pesquisa. De modo a investigar mais a fundo a eficácia desses algoritmos na solução do problema, (LIANG et al., 2018) faz uma breve comparação entre os três mais utilizados: PPO, DDPG e o método de *policy gradient* apresentado na seção 2.4.8. Esse trabalho chegou a conclusão de que, de fato, o *policy gradient* é o que alcança maior desempenho.

Apesar do resultado de Liang *et al.*, diversos trabalhos fazem uso do algoritmo que mais os convém, muitas vezes, não explicitando o motivo que levou a essa escolha. A seguir, tem-se a classificação dos trabalhos. Note que alguns trabalhos não estão presentes pois não especificaram o algoritmo utilizado. Outros, por outro lado, utilizaram mais de um algoritmo em seu estudo.

***Policy Gradient*:** (JIANG; XU; LIANG, 2017; LIANG et al., 2018; SHI et al., 2019; WU et al., 2021a; MA et al., 2022; SHI et al., 2022).

DDPG e derivados: (LIANG et al., 2018; LIU et al., 2018; YU et al., 2019; LIN et al., 2022; WANG; KU, 2022; JANG; SEONG, 2023; YANG, 2023).

PPO: (LIANG et al., 2018; BETANCOURT; CHEN, 2021; YANG, 2023).

Algorítmos críticos: (PARK; SIM; CHOI, 2020; SOLEYMANI; PAQUET, 2020; LIM; CAO; QUEK, 2022; TRAN; Pham-Hi; BUI, 2023; SONG et al., 2023).

Outros algoritmos atores-críticos: (KANG; ZHOU; KANG, 2018; SOLEYMANI; PAQUET, 2021; YANG, 2023).

5.3.2 Quanto à representação de estados

Grande parte dos trabalhos analisados fazem uso das séries temporais dos preços das ações e alguns também adicionam a série temporal de volume transacionado à representação de estados. (WENG et al., 2020) fez um estudo interessante que analisa quais *features* têm mais relação com o desempenho final e o resultado pode ser visualizado na figura 13: a série temporal de preços de fechamento das ações é, de longe, a mais importante para o modelo, mas o preço mais alto e mais baixo no dia também possuem influência considerável.

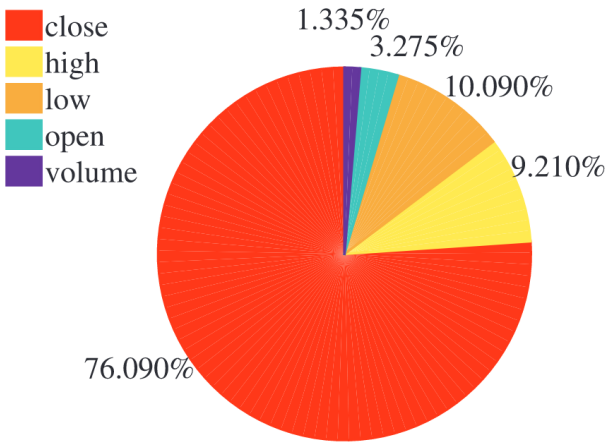


Figura 13 – *Features* temporais mais importantes para a otimização de carteiras, retirado de (WENG et al., 2020).

A seguir, tem-se uma classificação do tipo de dado financeiro utilizado na representação de estados. Considera-se como espaço de estados padrão a utilização das séries temporais de preços (abertura, fechamento, máximo e/ou mínimo) com ou sem a série temporal de volume de transações

Estado padrão: (ALMAHDI; YANG, 2017; JIANG; XU; LIANG, 2017; KANG; ZHOU; KANG, 2018; LIANG et al., 2018; SHI et al., 2019; YU et al., 2019; PARK; SIM; CHOI, 2020; WU et al., 2021a; LIN et al., 2022; MA et al., 2022).

Estado padrão com dados extras (como indicadores): (SOLEYMANI; PAQUET, 2020; BETANCOURT; CHEN, 2021; SOLEYMANI; PAQUET, 2021; SHI et al., 2022; JANG; SEONG, 2023; SONG et al., 2023).

Estado sem utilizar séries temporais : (LIU et al., 2018; LIM; CAO; QUEK, 2022; TRAN; Pham-Hi; BUI, 2023; YANG, 2023).

Uma ideia interessante introduzida na arquitetura EIIE (JIANG; XU; LIANG, 2017) está no fato de se utilizar a última ação executada pelo agente para que seja possível que ele leve em consideração a mudança no balanço do portfólio e, com isso, os possíveis custos com taxas operacionais e de corretagem. Diversas pesquisas fazem uso desse artifício, como pode-se ver na classificação abaixo.

Utiliza última ação: (JIANG; XU; LIANG, 2017; KANG; ZHOU; KANG, 2018; SHI et al., 2019; YU et al., 2019; PARK; SIM; CHOI, 2020; SOLEYMANI; PAQUET, 2020; WENG et al., 2020; SOLEYMANI; PAQUET, 2020; LIN et al., 2022; MA et al., 2022; SHI et al., 2019; SONG et al., 2023).

Não utiliza última ação: (LIU et al., 2018; WU et al., 2021a; LIM; CAO; QUEK, 2022; WANG; KU, 2022; TRAN; Pham-Hi; BUI, 2023; YANG, 2023).

5.3.3 Quanto ao tipo de ação do agente

A maioria dos modelos pesquisados utilizam o vetor portfólio como ação do agente, como explicado na formulação apresentada na seção 2.4. O problema dessa formulação é que ela é pouco realista e, muitas vezes, alguns trabalhos optam por utilizar ações que definem claramente a quantidade de ações que devem ser vendidas ou compradas, por exemplo. Já outros artigos optam por seguir um caminho próprio seja por conveniência ou para testar se é possível obter resultados melhores. A seguir, tem-se os principais tipos de ação utilizadas pelo agente.

Vetor Portfólio: (ALMAHDI; YANG, 2017; JIANG; XU; LIANG, 2017; KANG; ZHOU; KANG, 2018; LIANG et al., 2018; SHI et al., 2019; YU et al., 2019; SOLEYMANI; PAQUET, 2020; WENG et al., 2020; BETANCOURT; CHEN, 2021; WU et al., 2021a; LIN et al., 2022; MA et al., 2022; SHI et al., 2022; WANG; KU, 2022; JANG; SEONG, 2023; SONG et al., 2023).

Vetor indicando quais ações comprar, vender e segurar: (LIU et al., 2018; PARK; SIM; CHOI, 2020; YANG, 2023).

Uma ação específica: (LIM; CAO; QUEK, 2022), em especial, define uma ação como a melhor a ser investida e investe majoritariamente nela naquele momento.

Além disso, é importante ressaltar que alguns trabalhos fazem uso de uma etapa de pós-processamento da ação, utilizando alguma regra definida pelo desenvolvedor do sistema, que por vezes é baseada em alguma estratégia do mercado, para definir a ação final. (BETANCOURT; CHEN, 2021), por exemplo, usa um módulo chamado otimizador de transações para transformar o vetor portfólio em uma ação aplicável no mundo real, com a indicações de quais ações comprar e vender. (WU et al., 2021a), por outro lado, usa a estratégia EMN (PATTON, 2009) para modificar a ação final.

5.3.4 Quanto ao mercado alvo

Ao observar os principais mercados utilizados pelos pesquisadores para testar o desempenho de sistemas otimizadores de carteiras de ações, percebe-se que não há uma diversidade considerável. A maioria dos artigos fazem seus experimentos no mercado americano, seguido pelo mercado chinês. Além disso, de modo a haver uma forma de comparação com (JIANG; XU; LIANG, 2017), que é considerado um *baseline* da área, alguns artigos optam por utilizar o mercado de criptomoedas, mercado utilizado por Jiang *et al.*. Infelizmente, essa classificação deixa claro que o mercado brasileiro é negligenciado pela área de pesquisa, visto que somente um dos artigos, que opera a nível global por meio de ETFs, possui em sua carteira o índice IBOVESPA.

A seguir, tem-se a classificação dos trabalhos quanto ao mercado alvo. Ressalta-se que um artigo pode estar em mais de uma classificação caso faça experimentos em mais de um mercado.

Estados Unidos: (KANG; ZHOU; KANG, 2018; LIU et al., 2018; YU et al., 2019; PARK; SIM; CHOI, 2020; SOLEYMANI; PAQUET, 2020; SOLEYMANI; PAQUET, 2021; LIM; CAO; QUEK, 2022; LIN et al., 2022; SHI et al., 2022; WANG; KU, 2022; JANG; SEONG, 2023; YANG, 2023; SONG et al., 2023).

Criptomoedas: (JIANG; XU; LIANG, 2017; SHI et al., 2019; WENG et al., 2020; BETANCOURT; CHEN, 2021; TRAN; Pham-Hi; BUI, 2023).

China: (LIANG et al., 2018; MA et al., 2022). (WU et al., 2021a) faz uso de um portfólio no mercado taiwanês, que estamos classificando como China.

Mercado Global: (ALMAHDI; YANG, 2017; LIM; CAO; QUEK, 2022; WANG; KU, 2022) utilizam carteiras que investem em índices de diversos países, como IBOVESPA ou NASDAQ.

Coreia do Sul: (PARK; SIM; CHOI, 2020).

5.3.5 Quanto ao tamanho do portfólio

Define-se como tamanho do portfólio o número de ações que estão sendo utilizadas nele. Ao analisar os principais tamanhos de portfólio utilizados nos trabalhos citados, vê-se que há considerável variedade e alguns artigos até mesmo testam diversos tamanhos diferentes pois seu melhor valor é algo que depende principalmente da arquitetura do modelo e do mercado alvo. É importante ressaltar que a maioria dos artigos faz uso de um tamanho de carteira fixo (ou seja, define-se que a carteira é constituída por um conjunto de N ações e, em nenhum momento, uma ação pode ser trocada por outra, por exemplo), mas (BETANCOURT; CHEN, 2021) implementa uma forma de se utilizar um tamanho de portfólio variável.

A seguir, classificamos os trabalhos relacionados em três tamanhos de portfólio: portfólio pequeno, possuindo de um a 10 ações, portfólio médio, possuindo de 11 a 25 ações e portfólio grande, que possui mais de 26 ações. Além disso, há uma classe para portfólios variáveis.

Pequeno: (LIANG et al., 2018; YU et al., 2019; PARK; SIM; CHOI, 2020; LIM; CAO; QUEK, 2022; LIN et al., 2022; MA et al., 2022; WANG; KU, 2022).

Médio: (JIANG; XU; LIANG, 2017; SHI et al., 2019; SOLEYMANI; PAQUET, 2020; WENG et al., 2020; SHI et al., 2022; SONG et al., 2023).

Grande: (KANG; ZHOU; KANG, 2018; LIU et al., 2018; SOLEYMANI; PAQUET, 2021; WU et al., 2021a; SHI et al., 2022; JANG; SEONG, 2023; YANG, 2023).

Variável: (BETANCOURT; CHEN, 2021).

5.3.6 Quanto à granularidade das séries temporais

Como visto na seção 5.3.2, diversos artigos fazem uso das séries temporais de preços e indicadores de ações. Mas em todos os trabalhos apresentados, essas séries temporais possuem uma granularidade fixa, ou seja, os intervalos entre as medições presentes nas séries são fixos. É possível perceber que artigos diferentes usam granularidades diferentes considerando o mercado em que operam: para mercados acionários, por exemplo, é mais comum que *datasets* publicamente disponíveis tenham dados intervalados de hora em hora, enquanto que as cotações de criptomoedas são mais facilmente encontradas em intervalos menores.

Além disso, a grande maioria dos artigos utilizam o intervalo entre as séries temporais idêntico ao intervalo de balanceamento, ou seja, se a série temporal possui dados de hora em hora, o rebalanceamento do portfólio ocorrerá, também, de hora em hora. Isso faz com que a escolha da granularidade também seja influenciada se o modelo foca em

ganhos em pequenas ou grandes oscilações do mercado. Essa característica, porém, não está presente em (JANG; SEONG, 2023), que, apesar de utilizar dados diários das ações, somente pratica o rebalanceamento a cada 10 dias.

A seguir, tem-se as granularidades encontradas nos artigos avaliados. Trabalhos que investigaram seu desempenho em diferentes granularidades temporais podem aparecer em mais de uma classificação.

30 minutos: (JIANG; XU; LIANG, 2017; SHI et al., 2019; BETANCOURT; CHEN, 2021; LIN et al., 2022).

1 hora: (YU et al., 2019).

6 horas: (BETANCOURT; CHEN, 2021).

1 dia: (KANG; ZHOU; KANG, 2018; LIU et al., 2018; PARK; SIM; CHOI, 2020; SOLEYMANI; PAQUET, 2020; BETANCOURT; CHEN, 2021; SOLEYMANI; PAQUET, 2021; WU et al., 2021a; LIM; CAO; QUEK, 2022; MA et al., 2022; SHI et al., 2022; WANG; KU, 2022; JANG; SEONG, 2023; SONG et al., 2023).

1 semana: (ALMAHDI; YANG, 2017).

5.4 Lacunas da área de pesquisa

É possível observar algumas lacunas na área de pesquisa que podem servir como possíveis pontos de pesquisa em trabalhos futuros.

Falta de reprodutibilidade: Uma das grandes lacunas da área de pesquisa é a falta de reprodutibilidade dos artigos e de haver uma forma fácil de comparação entre diferentes arquiteturas. Esse problema ocorre pois a grande maioria dos trabalhos não disponibiliza seu código-fonte e torna-se difícil garantir que as comparações feitas entre as diversas arquiteturas representa a realidade. Dessa forma, é necessário haver um esforço para se desenvolver ferramentas *open source* que sejam adotadas pela comunidade científica e que permitam a reprodução de algumas arquiteturas notáveis para serem utilizadas como *baseline* em artigos científicos.

Formulação pouco realística em alguns cenários: Diversos trabalhos se baseiam na formulação apresentada por (JIANG; XU; LIANG, 2017), mas os próprios autores deste artigo discorrem sobre o fato de que a formulação não leva em consideração a influência do agente no mercado e a possível impossibilidade de se comprar ou vender uma ação a determinado preço. Isso faz com que trabalhos que se baseiem nesta formulação somente sejam realísticos ao utilizarem mercados com alto volume

de negociação, restringindo consideravelmente as possibilidades de aplicação prática dos modelos desenvolvidos.

Desconsideração da correlação de ações: Grande parte dos trabalhos utilizam as séries temporais de preços e indicadores de ações como estado do agente mas desconsideram a possível correlação entre ações. Isso pode ser um problema pois faz com que o agente rebalanceie o portfólio para lidar com a queda de determinada ação investindo em outras ações que também estarão em queda em um futuro próximo. Uma forma de lidar com isso é utilizar redes neurais em grafos (WU et al., 2021b), mas esse tipo de arquitetura foi explorado em somente dois dos artigos analisados (SOLEYMANI; PAQUET, 2021; SHI et al., 2022) e, apesar de terem bom resultado, foram somente adicionadas às arquiteturas convolucionais pré-existentes, sem haver um estudo mais aprofundado da melhor forma de se utilizar *graph neural networks*.

Pouca quantidade de dados financeiros: Um dos grandes problemas do aprendizado de máquina aplicado ao mercado financeiro está na pequena quantidade de dados disponíveis se comparado com outras áreas. Esse problema também afeta os agentes apresentados nesta revisão bibliográfica visto que a maioria deles não faz uso de nenhuma técnica de *data augmentation*. Algumas exceções são (LIANG et al., 2018), que aplica um ruído aleatório aos dados de treinamento, e (YU et al., 2019), que faz uso de redes neurais generativas (GOODFELLOW et al., 2014) para gerar dados sintéticos, e ambos os artigos alcançam bons resultados. Apesar disso, esse tipo de aplicação se mostra pouco explorada em trabalhos posteriores.

Poucas análises da efetividade de algoritmos: Foi possível observar na seção 5.3.1 que diversos algoritmos diferentes são utilizados no treinamento de agentes, mas a grande maioria dos artigos não justifica o motivo pelo qual certo algoritmo foi escolhido. Um possível motivo para isso está no fato de que não há, até onde se sabe, uma análise mais aprofundada do desempenho de algoritmos no problema, como foi feito de forma mais simples por (LIANG et al., 2018).

Falta de trabalhos no mercado brasileiro: A seção 5.3.4 tornou evidente o fato de que há uma enorme disparidade entre os mercados alvos escolhidos pela pesquisa científica. Uma vez que cada mercado possui suas características (o mercado brasileiro, por exemplo, é bem mais volátil que o americano), não é possível garantir que as soluções implementadas em um serão eficazes em outro. Dessa forma, é necessário que mais estudos foquem não só no mercado brasileiro, mas também em outros tipos de economias, como as subdesenvolvidas e emergentes, que estão sendo deixadas de lado.

6 Desenvolvimento

6.1 Tecnologias Utilizadas

De modo a garantir boa reprodutibilidade e confiabilidade, o projeto faz uso de diversas tecnologias modernas e *open source*.

Linguagem de programação *Python*: Por ser uma linguagem de programação *open source*, gratuita e largamente mantida e utilizada por desenvolvedores, o *Python* consegue ser aplicado em diversos contextos. Um desses contextos é o de desenvolver e treinar arquiteturas de aprendizado de máquina: possuindo uma série de bibliotecas confiáveis com foco em aprendizado profundo e aprendizado por reforço, que são os temas principais deste trabalho, essa linguagem possui grande aderência ao projeto desenvolvido.

Biblioteca *Gymnasium* (TOWERS et al., 2023): Essa biblioteca do *Python* permite a criação de ambientes de simulação para agentes de aprendizado por reforço de forma padronizada. Tal padronização faz o ambiente desenvolvido ser facilmente integrado a algoritmos existentes mantidos pela comunidade de desenvolvedores, permitindo melhor reprodutibilidade de trabalhos.

Biblioteca *PyTorch* (PASZKE et al., 2019): Também desenvolvida e mantida em *Python*, essa biblioteca possui todo o ferramental necessário para se desenvolver e treinar uma arquitetura de redes neurais, desde perceptrons simples até redes convolucionais profundas. O *PyTorch* é largamente utilizado ao redor do mundo pois permite a implementação de aprendizado profundo com confiabilidade e simplicidade ao abstrair grande parte da matemática por trás do processo de treinamento.

Biblioteca *FinRL* (LIU et al., 2022): Sendo o primeiro *framework* em *Python* para aprendizado por reforço em finanças, o *FinRL* possui uma série de ferramentas que permitem desde a aquisição de dados histórico de diversas fontes do mercado de ações até a fácil implementação de agentes previamente desenvolvidos em problemas de finanças.

Biblioteca *QuantStats* (AROUSSI, 2023): Essa biblioteca possui todo o ferramental necessário para calcular métricas relacionadas a dados financeiros e exibi-las como gráficos intuitivos.

***Framework Optuna* (AKIBA et al., 2019):** Por fim, este *framework* proporciona um ambiente para automatização de treinamentos de agentes, permitindo que a escolha de hiperparâmetros seja feita de forma mais abrangente e rápida.

6.2 Desenvolvimento de Simulador

Com o advento do aprendizado por reforço profundo em mercados financeiros, alguns *frameworks* e ambientes de simulação foram desenvolvidos para permitir a pesquisa na área. Muitos desses ambientes, como o *gym-anytrading* (HAGHPANAH, 2023), *gym-mtsim* (HAGHPANAH, 2021) e os disponíveis no *FinRL-Meta* (LIU, 2022) seguem a arquitetura proposta pelo *Gymnasium* (TOWERS et al., 2023) e são integradas a várias bibliotecas bibliotecas de aprendizado por reforço mas não seguem a representação de Markov apresentada na seção 2.4. Outros, como (AMROUNI, 2022), são desenvolvidos utilizando as formulações matemáticas previamente apresentadas mas não são bem integráveis às bibliotecas de aprendizado por reforço.

Para melhorar esse cenário, foi desenvolvido um ambiente *open source* para agentes de aprendizado por reforço chamado POE (COSTA; COSTA, 2023), sigla de *Portfolio Optimization Environment*, que é capaz de simular os principais aspectos da otimização de portfólio considerando as definições apresentadas nas seções 2.2 e 2.4 e possui integrações com bibliotecas modernas. Esse ambiente[1] é uma contribuição para o *FinRL-Meta* (LIU, 2022), um repositório contendo múltiplos ambientes *open source* do *FinRL* (LIU et al., 2022), um dos principais *frameworks* de aprendizado por reforço para finanças.

6.2.1 Diretrizes

O ambiente desenvolvido possui as seguintes diretrizes:

Integração completa com *FinRL*: O *FinRL* fornece uma *pipeline* completa para desenvolver, avaliar e implantar agentes de aprendizado por reforço profundo que operam no mercado financeiro, razão pela qual o ambiente desenvolvido foi pensado para ser parte dele. Ter integração completa com o *FinRL* permite que o simulador seja consideravelmente *plug-and-play*, aumentando a conveniência para outros pesquisadores.

Generalidade: Conforme visto em seções anteriores, o problema de otimização de portfólio possui muitos parâmetros para serem considerados: tamanho do portfólio, tamanho da janela de tempo, porcentagem de taxa de corretagem, etc. Por essa

[1] O código está disponível em <https://github.com/AI4Finance-Foundation/FinRL-Meta/tree/master/meta/env_portfolio_optimization> e é parte do repositório do *FinRL-Meta* (<https://github.com/AI4Finance-Foundation/FinRL-Meta>).

razão, o ambiente desenvolvido segue a diretriz de ser tão genérico quanto possível de modo que o usuário não sinta a necessidade de modificar seu código na maioria dos casos.

Estrutura análoga ao *Gymnasium*: Como o *Gymnasium* é, no momento, o *framework* mais famoso para padronizar ambientes de aprendizado por reforço, o simulador desenvolvido segue sua estrutura, permitindo sua utilização fora do contexto do *FinRL* de forma fácil.

Integração com bibliotecas de aprendizado por reforço confiáveis: Uma das principais razões de se utilizar a estrutura do *Gymnasium* está no fato de o ambiente desenvolvido poder ser utilizado com outras bibliotecas de aprendizado por reforço confiáveis que utilizam-se dessa estrutura. A habilidade de utilizar esse ambiente com ferramentas poderosas como *stable-baselines-3* (RAFFIN et al., 2021) ou *ElegantRL* (LIU et al., 2022) pode ajudar consideravelmente a área de pesquisa.

Contribuição aberta: O simulador tem o objetivo de ser *open source* e contribuições da comunidade ao longo do tempo permitirão que novas *features* sejam adicionadas a ele. Por esse motivo, o código foi adicionado ao repositório do *FinRL-Meta*, que possui uma vasta comunidade que será capaz de utilizá-lo e contribuir para ele.

6.2.2 Implementação

O ambiente foi implementado utilizando a linguagem de programação *Python* e, como dito na seção 6.2.1, é um ambiente com a estrutura do *Gymnasium*, que possui uma interface `reset`, `step` e `render`. Como qualquer objeto *Python*, para utilizar essa interface, é necessário instanciar o `PortfolioOptimizationEnv`, definir o valor inicial da carteira de ações e determinar um *dataframe Pandas* (TEAM, 2023) que contenha dados históricos do mercado financeiro. Seguindo a diretriz de generalidade, muitos parâmetros de simulação podem ser determinados, como:

- Um método de normalização de dados, que pode ser um dos pré-incluídos na biblioteca ou um criado pelo usuário;
- Uma janela de tempo que represente o estado (veja a seção 2.4);
- Um modelo de simulação de corretagem (modificador do vetor de pesos ou fator residual de transações) e uma taxa de corretagem;
- A lista de colunas do *dataframe* que deve ser considerada como *features* pelo ambiente (colunas não listadas são ignoradas). É também possível modificar a *feature* que é utilizada para calcular o valor do portfólio (por padrão, é considerado o valor de fechamento das ações);

- Opções para lidar com os nomes e os formatos de tempo do *dataframe* caso o usuário não esteja utilizando o padrão do *FinRL*;
- Um repositório onde gráficos serão salvos.

Depois de determinados todos os parâmetros, é finalmente possível interagir com o ambiente. Como ele segue a estrutura *Gymnasium*, qualquer agente que receba um estado e determine uma ação (no caso, o vetor de pesos) em cada instante de tempo pode ser utilizado, seja um agente de aprendizado por reforço ou não. Esse agente deve interagir com o ambiente por meio da interface de funções `reset`, `step` e `render`.

`reset`: Esse método é responsável por resetar o ambiente e retornar a data de simulação para a primeira data do *dataframe*. Ele também modifica os atributos do ambiente para seus valores padrão, permitindo que a simulação execute novamente. Esse método retorna o estado atual após o *reset*.

`step`: Como o nome sugere, esse método é responsável por executar um passo da simulação. Como visto na seção 2.2, um passo modifica o valor do portfólio considerando flutuações de preço durante o período. A quantidade de tempo que um passo de simulação representa depende do *dataframe* utilizado pelo ambiente: se o *dataframe* contiver uma série temporal diária, então é assumido que o *step* representa um período de um dia. Esse método precisa receber uma ação como parâmetro (que é o novo vetor de pesos do portfólio que o agente deseja utilizar) e retorna informações comumente necessárias para um algoritmo de aprendizado por reforço: um novo estado, a recompensa considerando a ação tomada no estado anterior e uma variável booleana definindo se o próximo estado é terminal ou não.

`render`: Renderiza o ambiente de modo que um humano possa visualizar o que está ocorrendo. No momento, este método somente retorna o valor numérico dos estados.

É importante entender o que é definido como um episódio nesse ambiente: considerando o *dataframe* ordenado por tempo e sendo t o tamanho da janela de tempo e T o último passo temporal do dataframe, o episódio vai começar no t-ésimo instante de tempo e termina quando o ambiente tentar acessar o instante inexistente $T + 1$. No momento de término, o método `step` vai parar de simular a passagem de tempo e irá exibir as métricas de desempenho do portfólio naquele episódio bem como salvar seus respectivos gráficos. Para começar um novo episódio, o agente deve utilizar o método `reset`.

6.2.3 Visualizando o desempenho

Após um episódio, é importante avaliar o desempenho do agente utilizando as métricas apresentadas na seção 2.2.3. Por isso, o ambiente desenvolvido faz uso da biblioteca

QuantStats (AROUSSI, 2023) para prover não só cálculos confiáveis dessas métricas mas também gráficos para que pesquisadores possam entender o comportamento do agente desenvolvido.

As figuras 14 e 15 mostram dois gráficos gerados pelo ambiente: um resumo do desempenho do portfólio e a recompensa ao longo de um episódio. Como pode ser visto, apesar de o portfólio ter bons resultados considerando o retorno acumulado da figura 14, o gráfico da figura 15 mostra que o agente recebe recompensas erráticas durante o episódio e, portanto, não está aprendendo com a experiência. De fato, o agente utilizado nesse exemplo aplica uma simples estratégia de *buy and hold* em um portfólio de 19 ações do mercado brasileiro durante o período de 2010 a 2020, de modo que não se ajusta de acordo com a dinâmica de mercado e somente obtém lucro devido ao crescimento geral do mercado brasileiro no período.

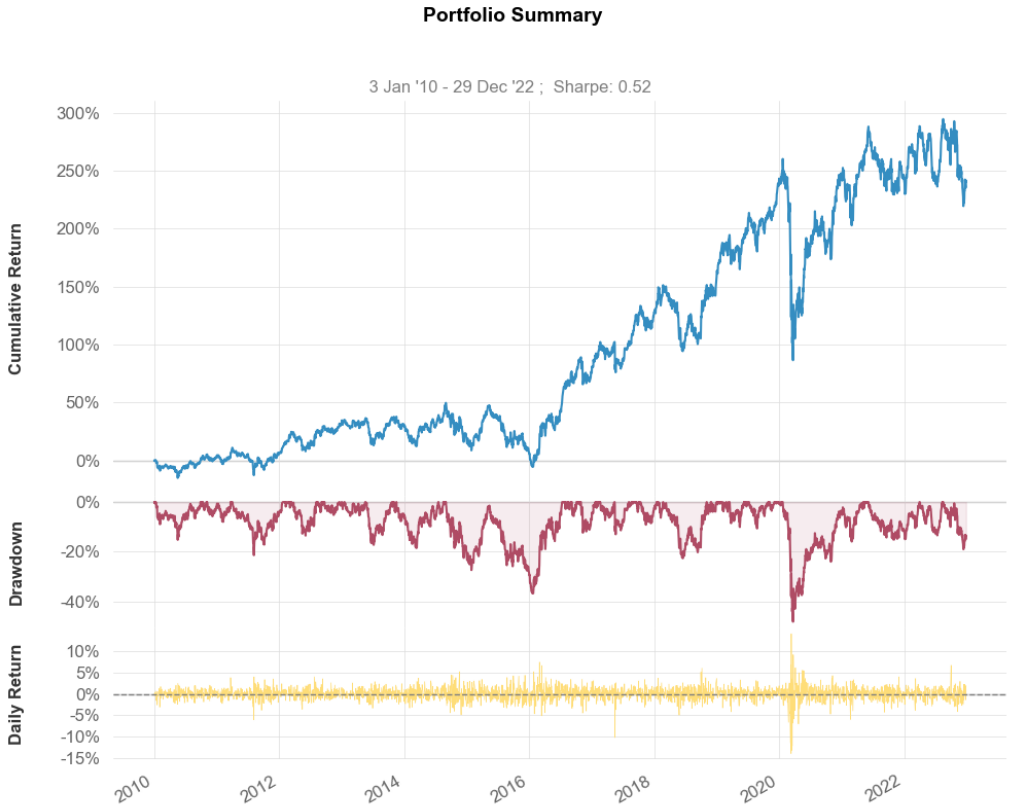


Figura 14 – Visualização do resumo do portfólio.

6.3 Testes com arquiteturas notáveis

Utilizando o ambiente de simulação apresentado na seção 6.2, é possível reproduzir algumas arquiteturas relevantes e verificar seu desempenho não só em mercados bem estabelecidos como também no mercado brasileiro, de modo a checar se essas arquiteturas são utilizáveis no País. Neste trabalho, foi escolhido reproduzir as arquiteturas convolucionais

Figura 15 – Visualização da recompensa do portfólio.

introduzidas na seção 2.4.6: a EIIE e EI^3. A primeira foi escolhida por ter grande influência na área e ser vista como *baseline* por diversos trabalhos posteriores. Já a segunda foi selecionada por ser uma melhoria direta em relação à primeira, não havendo a necessidade de modificar nada além da própria arquitetura em si. Assim, é interessante observar se essa melhoria se reproduz.

6.3.1 Reprodução das arquiteturas

As arquiteturas foram reproduzidas utilizando a biblioteca *PyTorch* de forma idêntica à apresentada em seus artigos originais (JIANG; XU; LIANG, 2017; SHI et al., 2019). Sabe-se que é possível melhorar o desempenho de uma arquitetura convolucional modificando, por exemplo, a quantidade de filtros convolucionais, a profundidade da rede ou o tamanho dos *kernels*, mas optamos por manter esses parâmetros idênticos ao original para que fosse possível comparar o desempenho.

Um ponto importante a ser destacado está no fato da definição do *cash bias*, elemento unidimensional que é adicionado à última camada da rede antes da aplicação da função *softmax* e que pode ser visto nas figuras 10 e 11. Nos artigos de referência, o valor desse *bias* não é explicitado e com isso, duas possibilidades foram testadas:

1. Tentou-se, inicialmente, utilizar um *cash bias* igual ao último valor de dinheiro não investido em ações no portfólio, mas isso não atingiu bom desempenho: a política se tornava mais difícil de ser treinada e parecia não generalizar bem nos ambientes de testes;

2. Posteriormente, resolveu-se utilizar um valor constante (especificamente, zero) e isso surtiu efeitos positivos no treinamento.

A intuição por trás do *cash bias* está no fato de que ele somente existe para que haja um ajuste de dimensionalidade do *output* da rede neural convolucional de modo que a utilização de um valor constante faz com que a rede consiga definir os valores intermediários ideais para que o balanceamento final após a aplicação da função *softmax* seja satisfatório.

6.3.2 Aquisição e processamento de dados

Os dados do mercado foram adquiridos utilizando o módulo *YahooDownloader* do *FinRL* (LIU et al., 2022). Os dados utilizados foram as séries temporais dos preços de fechamento, dos preços máximos e mínimos de ações com granularidade diária. Os mercados analisados foram ações da NASDAQ, da NYSE e da IBOVESPA. As ações americanas escolhidas tentaram se assemelhar ao máximo aos portfólios utilizados em (SHI et al., 2022), pois haveria uma boa base de comparação de desempenho. Já as ações brasileiras escolhidas foram aquelas que apresentam maior volume de negociação e que poderiam ser utilizadas nos períodos escolhidos para testes (algumas ações são muito recentes, por exemplo). As ações escolhidas foram:

NASDAQ: AAPL, MU, CSCO, MSFT, META, QQQ, CMCSA, INTC, HBAN, NVDA.

NYSE: BAC, F, RF, WFC, GE, PFE, C, T, MRO, X, JPM.

IBOVESPA: VALE3, PETR4, ITUB4, BBDC4, BBAS3, RENT3, LREN3, PRIO3, WEGE3, ABEV3, SBSP3, ENEV3, VIVT3, B3SA3, MGLU3, EQTL3, CMIG4, ELET3, GOAU4, AMER3, BRFS3, CPLE6, TIMS3, JBSS3, RADL3.

Após adquiridos, os dados passam por uma etapa de processamento em que eles são normalizados dividindo-se os valores de cada instante de tempo pelo valor anterior. Dessa forma, as séries temporais ficam com valores próximos de 1 e ainda assim conseguem representar bem quando houve uma alta ou queda.

6.3.3 Treinamento e validação dos agentes

O treinamento dos agentes foi feito utilizando o algoritmo de gradiente de política introduzido na seção 2.4.8 com o objetivo de maximizar somente o lucro alcançado. Dessa forma, espera-se que os agentes consigam adquirir lucro mas sofram com as oscilações do mercado. Essa abordagem de maximizar somente o lucro têm dado certo em trabalhos anteriores, mas como eles tem um foco muito maior no mercado americano, é interessante observar seus resultados no mercado volátil brasileiro.

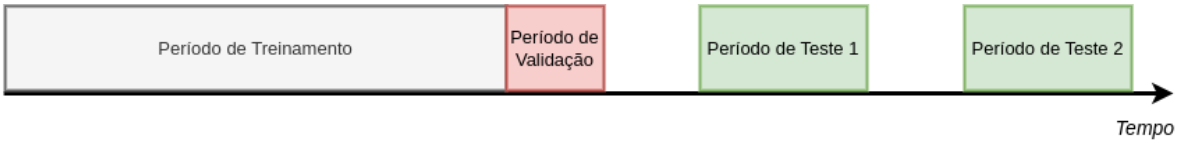


Figura 16 – Metodologia de treinamento de agentes.

A metodologia de treinamento utilizada neste trabalho segue o diagrama temporal mostrado na figura 16. Primeiramente, define-se um período cujos dados temporais são utilizados para treinar o modelo utilizando o algoritmo escolhido. Finalizado o treinamento, um período curto de validação é utilizado para que seja possível avaliar o desempenho do agente com dados desconhecidos por ele e utilizar essa avaliação para escolher os melhores hiperparâmetros do modelo. Os hiperparâmetros utilizados neste trabalho são:

Taxa de aprendizado: A taxa que é utilizada para atualizar os parâmetros da rede neural a cada gradiente ascendente. Procuramos valores entre 10^{-5} e 0.1;

Tamanho do *batch*: O tamanho do *batch* de dados utilizados em cada processo de atualização de parâmetros da rede neural. Definimos valores inteiros entre 10 e 200;

Número de episódios: Define basicamente o número de vezes que o período de treinamento será varrido completamente. Utilizamos valores entre 20 e 300;

Período de treinamento *online*: Durante os testes e avaliações, o agente é capaz de fazer um treinamento *online* com os novos dados que vai recebendo. Daí, define-se o período de treinamento *online* como o intervalo (em *steps*) entre duas atualizações *online* da rede. Nesse caso, procuram-se valores entre 10 e 50.

Para fazer a busca de hiperparâmetros, foi utilizado o *framework* Optuna (AKIBA et al., 2019), que executava diversos treinamentos em um servidor com as seguintes especificações:

Processador: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz;

Memória RAM: 32 GB;

Modelo de GPU: GeForce GTX 1080;

Memória VRAM: 8GB GDDR5X.

É importante destacar que, apesar do tamanho da janela temporal observada pelo agente poder ser considerada um hiperparâmetro e, por isso, poder ser otimizada na etapa de validação, optou-se por utilizar seu valor fixo como 50 *steps*. O motivo disso está no

fato de que este é o valor utilizado nos artigos que estão sendo reproduzidos (JIANG; XU; LIANG, 2017; SHI et al., 2019).

Além disso, todos os testes também utilizam o mesmo valor inicial de investimento (R$ 100000,00), a mesma taxa de corretagem (0.25%) e as mesmas *features*, que foram as séries temporais dos preços de fechamento e dos preços máximos e mínimos de cada ação, conforme também foi utilizado por Jiang *et al.*

Por fim, ressalta-se que o código[2] das arquiteturas e do algoritmo de treinamento desenvolvido ao longo deste trabalho ficará disponível no *FinRL* quando for aceito pelos mantenedores da biblioteca.

6.3.4 Testes dos agentes

Tendo completado a etapa de treinamento e validação, o melhor agente é testado em um ou mais períodos de testes, que são, como mostra a figura 16, períodos posteriores à etapa de treino. Vale salientar que, mesmo durante o período de teste, o agente continua sendo atualizado de tempos em tempos utilizando novas informações do mercado para que ele consiga se manter em operação por mais tempo, sem a necessidade de haver um novo treinamento.

Primeiramente, propõe-se verificar o desempenho das arquiteturas EIIE (JIANG; XU; LIANG, 2017) e EI^3 (SHI et al., 2019) no mercado americano pois um trabalho desenvolvido por Shi *et al.* (SHI et al., 2022) apresenta todos os dados necessários para reproduzir o experimento feito (os períodos e as ações utilizadas), apesar de ele ter sido feito somente com a arquitetura EI^3. Para fazer essa comparação, foram executados os seguintes testes.

1. Testar as arquiteturas EIIE e EI^3 na NASDAQ com o portfólio composto pelas seguintes ações: AAPL, MU, CSCO, MSFT, META, QQQ, CMCSA, INTC, HBAN, NVDA. O teste deve utilizar os mesmos períodos utilizados no trabalho de Shi *et al.* (SHI et al., 2022) para permitir comparações.

    - Período de treinamento: 02/01/2013 a 30/06/2016;
    - Período de validação: 01/07/2016 a 31/12/2016;
    - Período de Teste: 01/01/2017 a 08/12/2017;

2. Testar as arquiteturas EIIE e EI^3 na NYSE com o portfólio composto pelas seguintes ações: BAC, F, RF, WFC, GE, PFE, C, T, MRO, X, JPM. O teste deve utilizar os

[2] No momento, o código está disponível em <https://github.com/C4i0kun/FinRL>, mas eventualmente sera adicionado ao repositório oficial do *FinRL* (<https://github.com/AI4Finance-Foundation/FinRL>)

mesmos períodos utilizados no trabalho de Shi *et al.* (SHI et al., 2022) para permitir comparações.

- Período de treinamento: 02/01/2013 a 30/06/2016;
- Período de validação: 01/07/2016 a 31/12/2016;
- Período de Teste: 01/01/2017 a 08/12/2017;

Tendo feito esses testes iniciais, propõe-se trazer a arquitetura que alcançar melhor desempenho para o mercado brasileiro. Por isso, definimos três portfólios diferentes: um portfólio pequeno, um médio e outro grande. Nos três casos, as ações escolhidas são ações de alto volume de negociação da IBOVESPA. Os portfólios são:

Portfólio pequeno: Composto por 5 ações de alto volume da IBOVESPA - VALE3, PETR4, ITUB4, BBDC4, BBAS3;

Portfólio médio: Composto por 10 ações de alto volume da IBOVESPA - VALE3, PETR4, ITUB4, BBDC4, BBAS3, RENT3, LREN3, PRIO3, WEGE3, ABEV3;

Portfólio grande: Composto por 25 ações de alto volume da IBOVESPA - VALE3, PETR4, ITUB4, BBDC4, BBAS3, RENT3, LREN3, PRIO3, WEGE3, ABEV3, SBSP3, ENEV3, VIVT3, B3SA3, MGLU3, EQTL3, CMIG4, ELET3, GOAU4, AMER3, BRFS3, CPLE6, TIMS3, JBSS3, RADL3.

Com essas carteiras definidas, propõe-se o seguinte teste:

3. Testar a arquitetura EIIE ou EI^3 (dependendo de qual desempenhar melhor nos testes 1 e 2) no mercado brasileiro com os três portfólios previamente listados. O teste deve utilizar o mesmo período de treino descrito no trabalho de (SHI et al., 2022), mas períodos de testes diferentes também foram escolhidos.

   - Período de treinamento: 02/01/2013 a 30/06/2016;
   - Período de validação: 01/07/2016 a 31/12/2016;
   - Período de Teste 1: 01/01/2017 a 08/12/2017;
   - Período de Teste 2: 01/01/2017 a 31/12/2018;
   - Período de Teste 3: 01/01/2019 a 31/12/2020;
   - Período de Teste 4: 01/01/2021 a 31/12/2022;

Esse último teste tem o objetivo de verificar como a arquitetura com melhor desempenho funciona no mercado brasileiro, além de entender qual é a influência do

tamanho do portfólio e como o agente desempenha em períodos de testes mais distantes do treinamento.

Por fim, destaca-se que os testes usam como base de comparação a estratégia *uniform buy and hold* (UBAH), em que um agente investe igualmente em todas as ações e não mais faz rebalanceamentos. Essa estratégia é bastante utilizada pois, a longo prazo, costuma adquirir lucro por meio da diversificação do portfólio (LEKOVIĆ, 2018), mas pode ser bastante sensível quando há algum período de crise econômica, por exemplo.

6.4 Resultados dos testes

6.4.1 Portfólio NASDAQ

Para o portfólio utilizando 10 ações da NASDAQ introduzido na seção 6.3.2, o resultado das duas arquiteturas foi muito parecido, visto que elas alcançam um valor final muito próximo. É possível observar, porém, que a arquitetura EI^3 é mais estável que a EIIE, que conta com mais oscilações. Isso fica evidente quando se observa os resultados presentes na tabela 1, em que a arquitetura EIIE possui uma máxima queda (MDD) maior e uma razão Sharpe (SR) menor, o que indica que ela é mais suscetível ao risco do mercado. Apesar disso, ambas as arquiteturas alcançam mais lucro do que a estratégia clássica *uniform buy and hold* (UBAH), o que mostra que o rebalanceamento do agente foi eficaz.

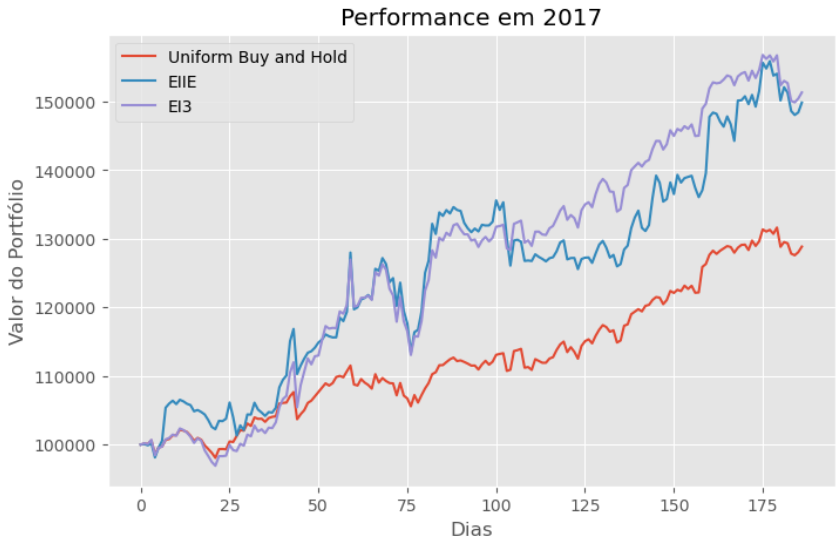


Figura 17 – Performance do portfólio NASDAQ no período de testes.

É interessante também pontuar como os resultados encontrados neste experimento para a arquitetura EI^3 são próximos aos apresentados no trabalho de (SHI et al., 2022), que estão compilados na tabela 2. Pode-se dizer que os resultados não são idênticos por dois motivos: a inicialização das redes neurais não se dá da mesma forma e o portfólio utilizado neste trabalho contém uma ação a menos quando comparado com a carteira

Tabela 1 – Métricas de desempenho do experimento em NASDAQ. Os melhores valores estão em negrito.

	fAPV	**MDD**	**SR**	**SR (anualizado)**
UBAH	1,288	**0,053**	0,162	2,574
EIIE	1,498	0,114	0,134	2,131
EI^3	**1,513**	0,109	**0,168**	**2,647**

utilizada por Shi *et al.* pois tal ação (cuja sigla é "XIV") não está mais listada no site do *Yahoo Finanças* (nossa fonte de dados).

Tabela 2 – Métricas de desempenho retiradas de (SHI et al., 2022) para o mercado NASDAQ.

	fAPV	**MDD**	**SR**
EI^3	1,497	0,13	0,1257

Os hiperparâmetros utilizados neste experimento podem ser observados na tabela 3. Como dito anteriormente, esses hiperparâmetros foram definidos utilizando o Optuna. É importante deixar claro que foi utilizada uma técnica de *early stopping* caso o treinamento começasse a ter problemas com *overfitting*: quando o valor do portfólio, durante o treinamento, alcançava um valor muito alto, ele era interrompido. Por isso, é possível que o treinamento tenha utilizado menos episódios que os definidos na tabela de hiperparâmetros.

Tabela 3 – Hiperparâmetros utilizados experimento em NASDAQ.

	EIIE	**EI^3**
Taxa de aprendizado	0,0125	0,085
Tamanho do *batch*	180	50
Número de episódios	45	88
Período do treinamento *online*	35	20

6.4.2 Portfólio NYSE

No caso do portfólio NYSE, tem-se um período de teste em que o mercado está passando por uma oscilação. Observando as séries temporais de valor do portfólio para as arquiteturas presentes na figura 18, é possível observar que a estratégia UBAH é a que melhor consegue lidar com a oscilação do mercado. Em contraste com o resultado apresentado no caso NASDAQ, neste caso é possível perceber que a arquitetura EI^3 teve desempenho consideravelmente melhor que a arquitetura EIIE, conseguindo praticamente anular as perdas no ano.

Na tabela 4, é possível observar as métricas de desempenho do portfólio NYSE e, na tabela 5, têm-se os resultados apresentados no trabalho de referência (SHI et al., 2022). Assim como no caso NASDAQ, os resultados são muito parecidos mas não idênticos (a

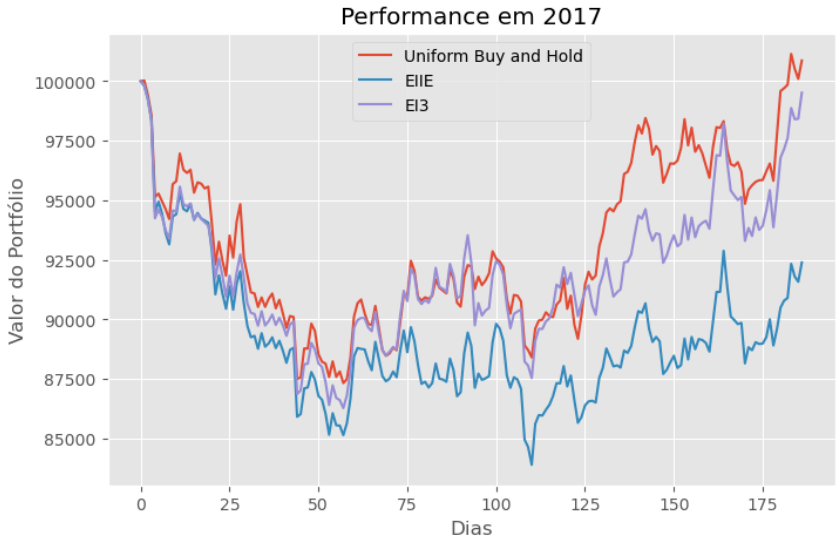


Figura 18 – Performance do portfólio NYSE no período de testes.

Tabela 4 – Métricas de desempenho do experimento em NYSE. Os melhores valores estão em negrito.

	fAPV	**MDD**	**SR**	**SR (anualizado)**
UBAH	**1,009**	**0,127**	**0,0097**	**0,153**
EIIE	0,924	0,161	-0,0425	-0,675
EI^3	0,995	0,137	0,0019	0,030

arquitetura EI^3 tem desempenho minimamente maior no trabalho de Shi *et al.*). Nesse caso, os portfólios são iguais e essa diferença se deve pelo fato de tanto a inicialização das arquiteturas quanto os hiperparâmetros de treinamento utilizados serem diferentes.

Tabela 5 – Métricas de desempenho retiradas de (SHI et al., 2022) para o mercado NYSE.

	fAPV	**MDD**	**SR**
EI^3	1,004	0,146	0,0062

Na tabela 6, apresentam-se os hiperparâmetros utilizados neste experimento, adquiridos utilizando o mesmo método do experimento da seção 6.4.2.

Tabela 6 – Hiperparâmetros utilizados experimento em NYSE.

	EIIE	**EI^3**
Taxa de aprendizado	0,085	0,0125
Tamanho do *batch*	130	110
Número de episódios	45	35
Período do treinamento *online*	42	18

6.4.3 Comentários sobre o treinamento no mercado americano

Como pode ser observado nas duas seções anteriores, o treinamento no mercado americano foi satisfatório, visto que foi possível reproduzir o desempenho apresentado em (SHI et al., 2022). O êxito na reprodução de resultados de um artigo de referência mostra que o trabalho desenvolvido até então é consistente: tanto o ambiente de simulação implementado quanto as arquiteturas e o algoritmo reproduzidos funcionam bem.

Apesar desse bom resultado, o processo de treinamento encontrou alguns obstáculos. Por ser um método *full exploitation* (ou seja, um método em que o agente não explora o ambiente por meio de ações subótimas), o algoritmo utilizado no treinamento sofre com problemas de *overfitting*. É comum, principalmente ao se utilizar taxas de aprendizagem altas, conseguir aumentar o valor do portfólio em algo como 50 ou 100 vezes durante o treinamento, mas esse bom desempenho não será refletido no período de testes e isso pode, na verdade, fazer o modelo perder grande parte de seu dinheiro inicial. Isso ocorre porque o agente se tornará extremamente eficiente em maximizar o valor do portfólio considerando a variação dos preços das ações no período de treinamento e, ao encontrar novos padrões no período de testes, escolherá ações muito ineficazes. Durante os experimentos, algumas formas de lidar com o *overfitting* foram testadas mas, considerando o tempo de treinamento, a técnica que mais surtiu efeito foi o *early stopping*, em que o treinamento é parado quando alguma métrica de desempenho é alcançada.

Outro problema que é consequência do agente não desbravar o espaço de ações é o fato de que, por vezes, é possível que o modelo defina como política ideal não investir em nenhuma ação. Isso cria um problema no treinamento pois, dessa forma, não há como utilizar o algoritmo de gradiente ascendente pois o valor da métrica que está sendo maximizada (o valor final do portfólio) nunca varia. Com isso, este inconveniente pode acabar atrasando o processo.

6.4.4 Experimentos no mercado brasileiro

Uma vez que a arquitetura EI^3 alcançou desempenho um pouco melhor que a arquitetura EIIE nos experimentos no mercado americano, sua eficiência deverá também ser analisada no mercado brasileiro utilizando os três portfólios com ações da IBOVESPA apresentados na seção 6.3.4. Nas tabelas 7, 8, 9 e 10, é possível verificar o desempenho da arquitetura em diversos períodos de teste por meio das métricas de desempenho introduzidas na seção 2.2.3.

Pode-se observar que a arquitetura EI^3, no mercado brasileiro, não parece ter resultados tão diferentes da estratégia *uniform buy and hold* (UBAH) e, na maioria dos casos, a estratégia clássica se mostra melhor do que a arquitetura testada. Assim, é possível concluir que a arquitetura pode ser utilizada no Brasil (já que ela não apresenta perdas

Tabela 7 – Métricas de desempenho em 2017 para o mercado brasileiro. Os melhores valores entre EI³ e UBAH estão em negrito.

	EI³			**UBAH**		
Portfólio	**fAPV**	**MDD**	**SR (Anualizado)**	**fAPV**	**MDD**	**SR (Anualizado)**
Pequeno	1,003	0,226	0,153	**1,065**	**0,167**	**0,457**
Médio	1,195	0,135	1,329	**1,256**	**0,130**	**1,641**
Grande	1,178	0,128	1,108	**1,182**	**0,120**	**1,193**

Tabela 8 – Métricas de desempenho no período de 2017 e 2018 para o mercado brasileiro. Os melhores valores entre EI³ e UBAH estão em negrito.

	EI³			**UBAH**		
Portfólio	**fAPV**	**MDD**	**SR (Anualizado)**	**fAPV**	**MDD**	**SR (Anualizado)**
Pequeno	1,497	0,413	0,824	**1,607**	**0,268**	**1,095**
Médio	1,633	**0,210**	1,373	**1,703**	0,212	**1,452**
Grande	**1,622**	**0,160**	**1,317**	1,596	0,179	1,316

Tabela 9 – Métricas de desempenho no período de 2019 e 2020 para o mercado brasileiro. Os melhores valores entre EI³ e UBAH estão em negrito.

	EI³			**UBAH**		
Portfólio	**fAPV**	**MDD**	**SR (Anualizado)**	**fAPV**	**MDD**	**SR (Anualizado)**
Pequeno	**1,233**	**0,461**	**0,492**	1,065	0,514	0,293
Médio	1,560	0,488	0,835	**1,576**	**0,482**	**0,843**
Grande	**1,839**	0,453	**1,112**	1,708	**0,441**	1,022

Tabela 10 – Métricas de desempenho no período de 2021 e 2022 para o mercado brasileiro. Os melhores valores entre EI³ e UBAH estão em negrito.

	EI³			**UBAH**		
Portfólio	**fAPV**	**MDD**	**SR (Anualizado)**	**fAPV**	**MDD**	**SR (Anualizado)**
Pequeno	**1,348**	**0,193**	**0,786**	1,217	0,234	0,583
Médio	1,156	0,219	0,483	**1,162**	**0,199**	**0,496**
Grande	0,961	0,225	0,004	**0,967**	**0,208**	**0,019**

muito altas), mas outras estratégias podem ser muito mais apropriadas.

É importante, porém, ressaltar que, apesar desse resultado abaixo do esperado, o fato de a arquitetura gravitar muito proximamente da estratégia UBAH mostra que ela possui grande potencial de melhorias. Como dito na seção 6.3.4, optou-se por utilizar neste trabalho a arquitetura exatamente igual à utilizada nos trabalhos de referência, algo que pode ser modificado de modo a adaptá-la para o mercado brasileiro. Assim, esse resultado pode ser visto como uma oportunidade para que estudos futuros tentem, por exemplo, modificar o número de camadas da rede ou a quantidade de filtros aplicados, adicionar uma etapa não-convolucional à rede, entre outras possibilidades que podem fazê-la desempenhar melhor no caso específico brasileiro.

A tabela 11 lista os hiperparâmetros utilizados no treinamento de cada um dos

portfólios utilizados. Assim como no caso do mercado americano, é importante salientar a questão da utilização do *early stopping*, o que faz com que o número de episódios possa ser menor do que o indicado.

Tabela 11 – Hiperparâmetros utilizados experimento em NYSE.

	Pequeno	**Médio**	**Grande**
Taxa de aprendizado	0,05	0,032	0,075
Tamanho do batch	130	170	170
Número de episódios	112	46	95
Período de treinamento online	35	29	46

6.4.4.1 Análise do desempenho do portfólio pequeno

As figuras 19, 20, 21 e 22 mostram, respectivamente, o desempenho da arquitetura EI^3 treinada para otimizar o portfólio pequeno no mercado brasileiro nos períodos de 2017, 2017 a 2018, 2019 a 2020 e 2021 a 2022. De imediato, é possível perceber que, em todos os períodos, o agente segue uma política que lembra a estratégia *uniform buy and hold* (UBAH), mas nota-se que a estratégia convolucional peca um pouco ao lidar com quedas do mercado: tanto em 2017 quanto em 2018, grande parte das perdas do agente se dá por, em comparação com o UBAH, ter perdido mais dinheiro.

Figura 19 – Desempenho do portfólio pequeno em 2017.

Curiosamente, no caso do portfólio pequeno, o agente consegue ultrapassar o desempenho da estratégia clássica nos períodos de 2019 a 2020 e 2021 a 2022 (figuras 21 e 22, respectivamente). É importante lembrar que esses períodos foram marcados pela pandemia de COVID-19 e por uma eventual recuperação econômica do mercado brasileiro. Dessa forma, faz sentido que a estratégia UBAH tenha desempenho menor. Além disso, como o portfólio é treinado para maximizar o lucro, ele é favorecido pelo fato de o mercado

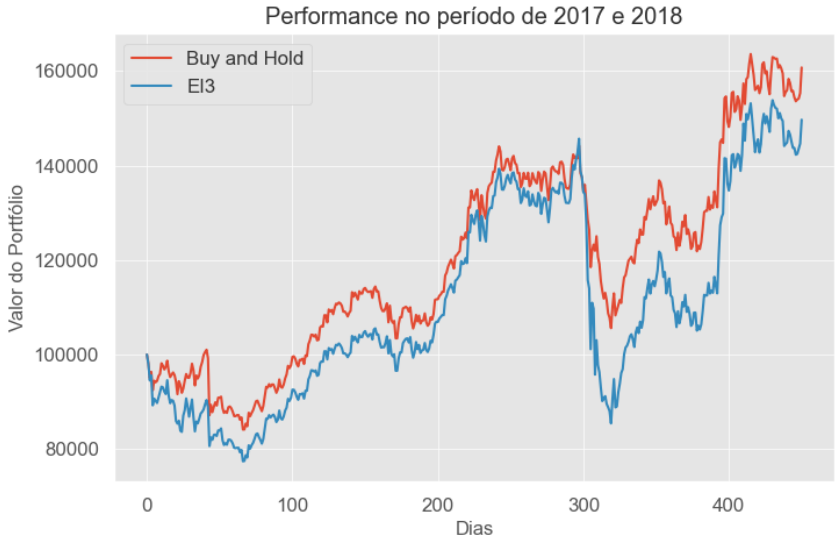


Figura 20 – Desempenho do portfólio pequeno no período de 2017 a 2018.

estar se recuperando de uma crise fiscal e, como ele é composto por 5 ações de grande volume de empresas bem consolidadas, ele consegue explorar bem isso e alcançar lucro acima do esperado.

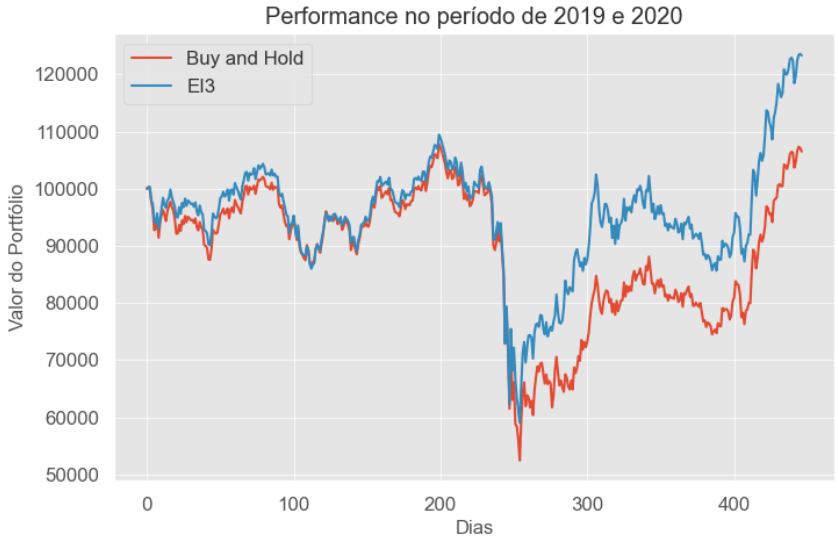


Figura 21 – Desempenho do portfólio pequeno no período de 2019 a 2020.

6.4.4.2 Análise do desempenho do portfólio médio

Os desempenhos do portfólio médio podem ser visualizadas nas figuras 23, 24, 25 e 26, cobrindo os mesmos intervalos de tempo utilizados na análise da seção anterior. De imediato, é possível perceber que o agente sofre com o mesmo problema de perdas maiores durante oscilações do mercado em comparação com a arquitetura clássica. Isso fica evidente, por exemplo, nas figuras 23 e 25, em que o agente está praticamente seguindo o

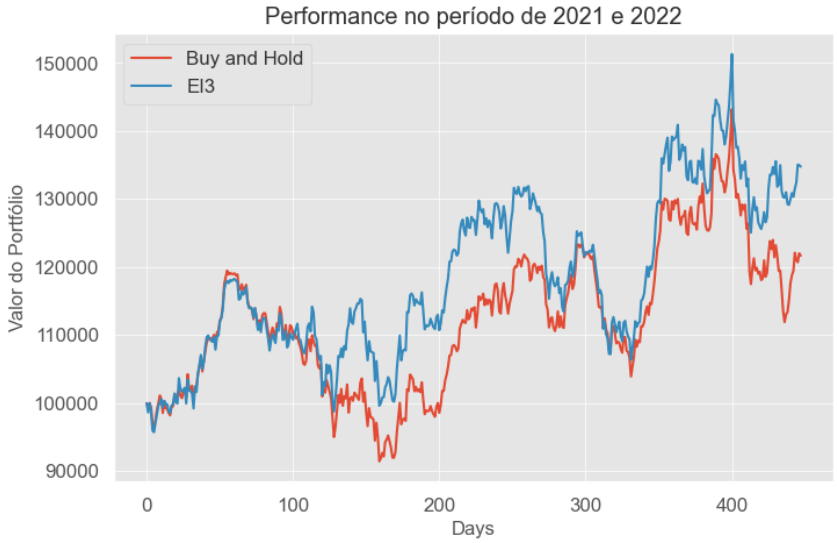


Figura 22 – Desempenho do portfólio pequeno no período de 2021 a 2022.

benchmark estabelecido, mas, ao encontrar uma oscilação, perde mais dinheiro do que o normal. Por outro lado, é interessante observar, ainda na figura 25, o agente explorando uma subida repentina do mercado e alcançando a estratégia clássica.

Figura 23 – Desempenho do portfólio médio em 2017.

Ao analisar as figuras do experimento dessa seção, pode-se questionar o motivo pelo qual o agente está desempenhando muito mais próximo da estratégia UBAH do que na seção anterior. O principal motivo para isso está no fato de que, conforme se aumenta o número de ações em um portfólio, mais a diversificação da carteira beneficia o investimento (LEKOVIĆ, 2018), fazendo o desempenho da estratégia UBAH ser mais alto e, com isso, a arquitetura em teste tende a gravitar mais nessa estratégia que já tem bom retorno.

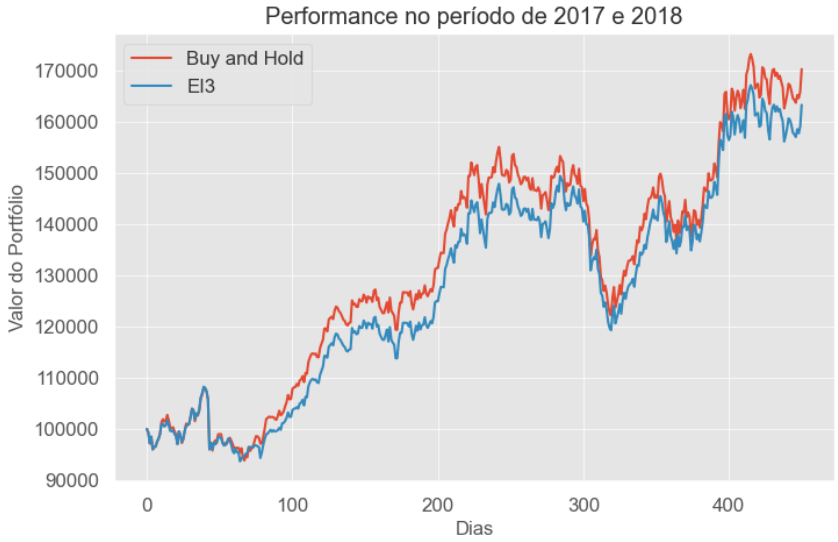


Figura 24 – Desempenho do portfólio médio no período de 2017 a 2018.

Figura 25 – Desempenho do portfólio médio no período de 2019 a 2020.

6.4.4.3 Análise do desempenho do portfólio grande

O último experimento feito no mercado brasileiro tem seu desempenho comparado com o *benchmark* clássico nas figuras 27, 28, 29 e 30. Assim como todos os portfólios testados no mercado brasileiro, o portfólio grande também sofre com problemas de perdas durante oscilações do mercado, algo que pode ser observado claramente na segunda metade da figura 27. Apesar disso, é perceptível que essas perdas são minimizadas pela diversificação do portfólio, visto que, em todos os casos, a arquitetura alcançou desempenho muito parecido com o desempenho da estratégia UBAH. Destacam-se os períodos apresentados nas figuras 28 e 29, em que o agente consegue manter as perdas em um nível próximo do *benchmark* mas consegue maximizar os lucros, sendo mais eficiente.

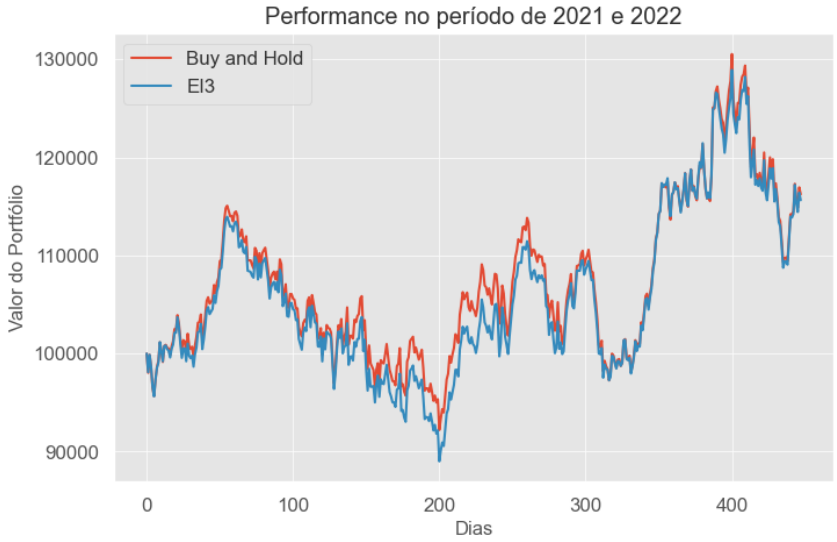


Figura 26 – Desempenho do portfólio médio no período de 2021 a 2022.

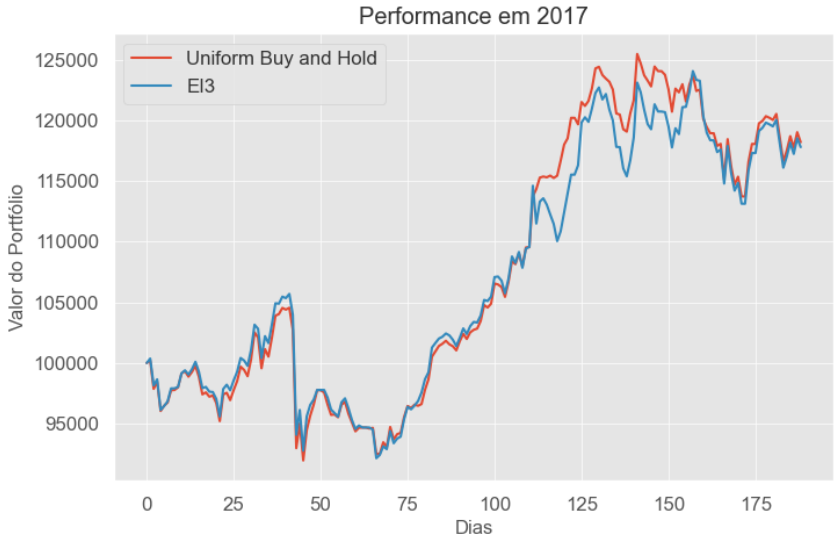


Figura 27 – Desempenho do portfólio grande em 2017.

6.4.4.4 Efeitos do tamanho do portfólio

As seções 6.4.4.1, 6.4.4.2 e 6.4.4.3 mostram, numa análise visual, que quanto maior o tamanho de um portfólio no mercado brasileiro, mais ele desempenha similar à estratégia *uniform buy and hold*. Isso pode trazer a falsa sensação de que não há grandes benefícios em aumentar o tamanho da carteira. Mas podemos perceber, por meio das figuras 31, 32 e 33, que essa sensação é equivocada e que o reflexo positivo da diversificação da carteira (LEKOVIĆ, 2018) é um fator de grande impacto. Nessas figuras, é possível observar que o portfólio grande alcança não só valor final maior que o portfólio pequeno, mas também é menos influenciado pelas quedas bruscas durante períodos de oscilação, algo que pode ser visto com grande ênfase na segunda metade da figura 32, por exemplo, em que o portfólio

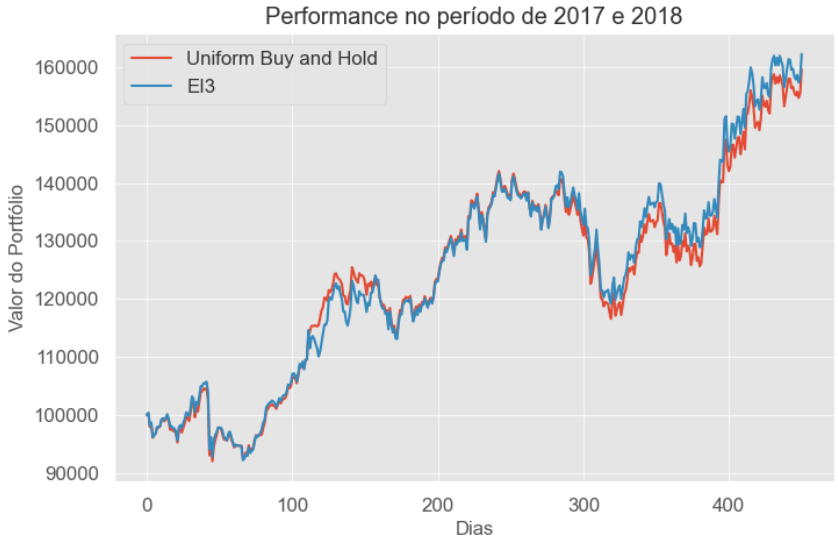


Figura 28 – Desempenho do portfólio grande no período de 2017 a 2018.

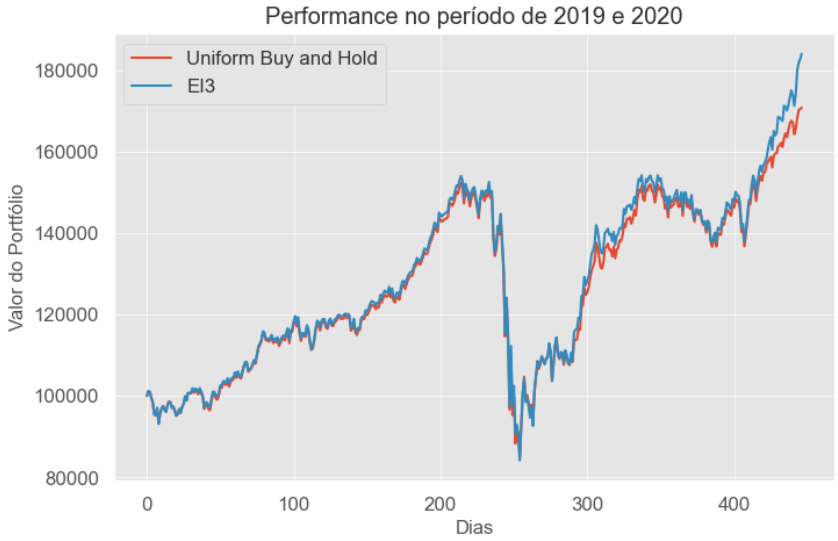


Figura 29 – Desempenho do portfólio grande no período de 2019 a 2020.

pequeno sofre uma grande queda em comparação aos outros dois.

Curiosamente, porém, nos períodos de 2021 a 2022, que pode ser visto na figura 34, a tendência se inverte e o menor portfólio apresenta maior desempenho. Um estudo mais aprofundado sobre essa questão precisaria ser feito de modo a haver uma resposta assertiva, mas uma possível explicação para esse comportamento está no fato de que, quando construímos os portfólios, utilizamos as ações de maior volume no portfólio pequeno e fomos adicionando ações de menor volume nos portfólios médio e grande, de modo que carteiras menores estão contidas em carteiras maiores. Isso foi feito pois é necessário garantir que ações de alto volume sejam utilizadas devido às hipóteses apresentadas na seção 2.2.4.

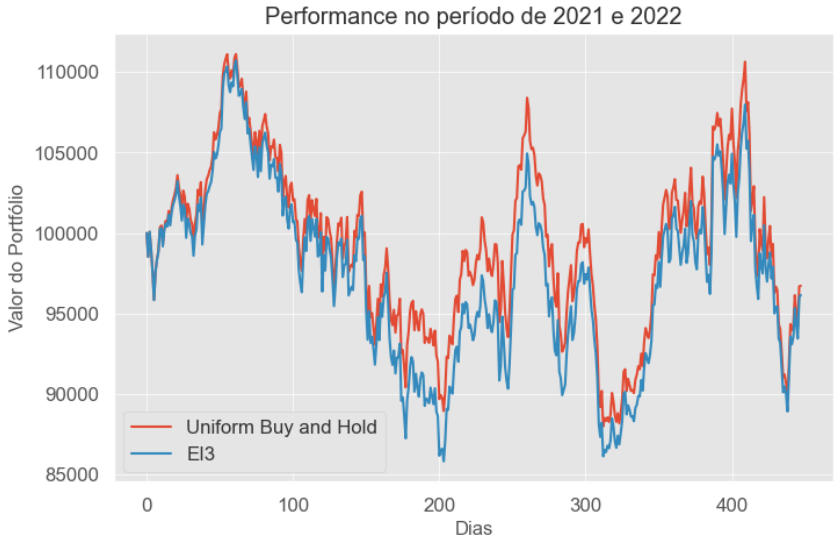


Figura 30 – Desempenho do portfólio grande no período de 2021 a 2022.

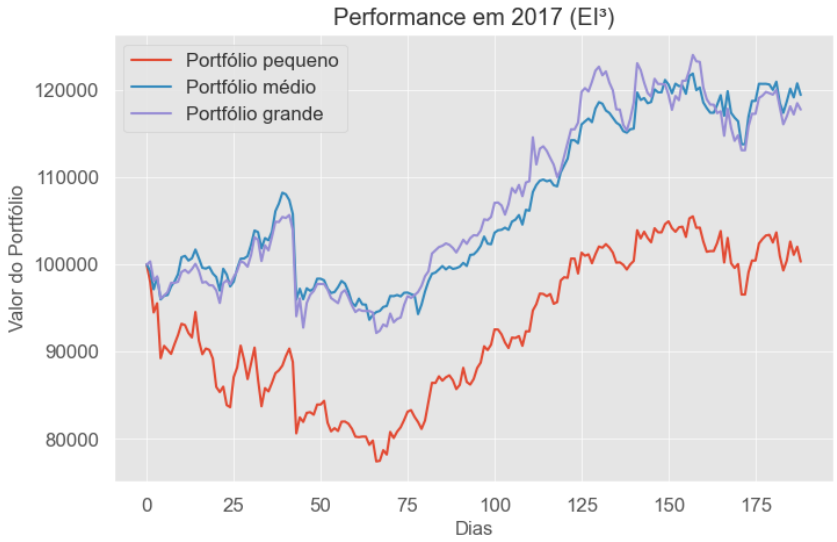


Figura 31 – Comparação de diferentes tamanhos de portfólio em 2017.

Mas como, geralmente, ações de alto volume são aquelas associadas a empresas grandes ou, no caso do mercado brasileiro, empresas que possuem participação do Governo, pode-se dizer que o portfólio pequeno utilizado nestes experimentos é composto por ações mais seguras e que, portanto, possuem maior capacidade de se recuperarem no período pós-pandemia. Dito isso, uma hipótese é que o resultado presente na figura 34 mostra que empresas muito bem consolidadas conseguiram se recuperar bem da crise da COVID-19, algo que não se reflete em todo o mercado.

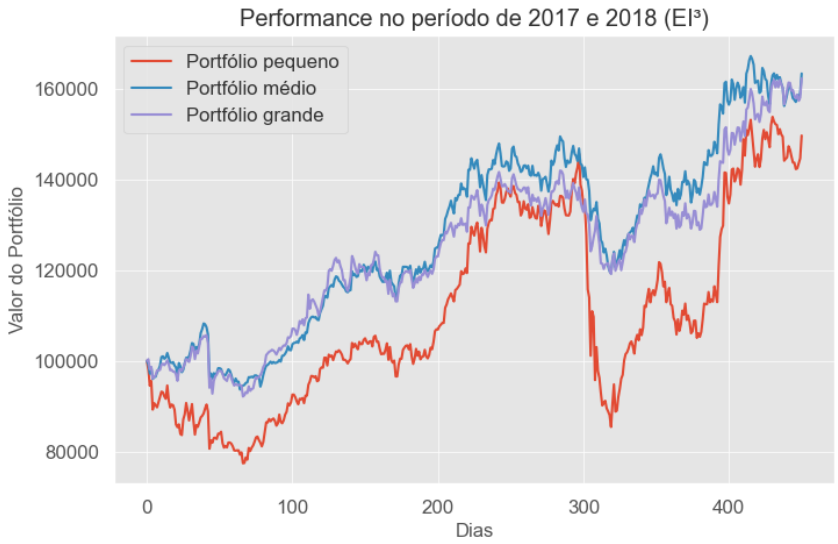


Figura 32 – Comparação de diferentes tamanhos de portfólio no período de 2017 a 2018.

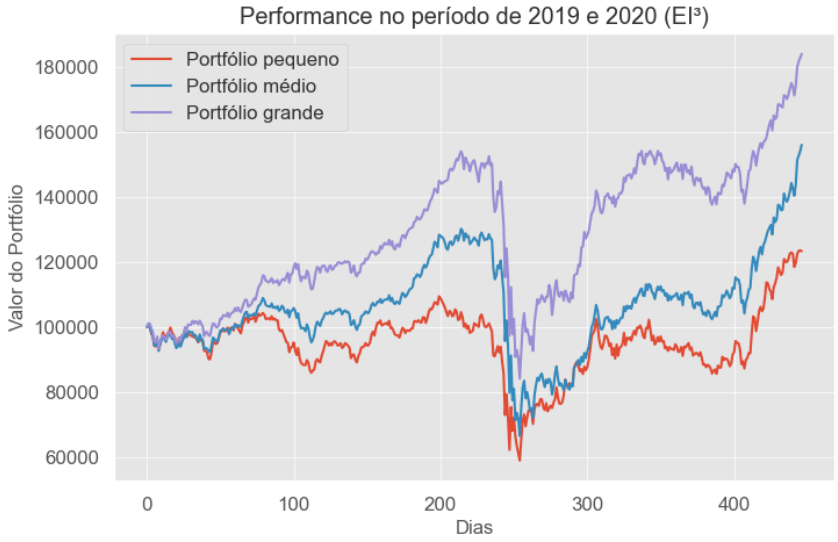


Figura 33 – Comparação de diferentes tamanhos de portfólio no período de 2019 a 2020.

6.4.5 Comentários sobre os testes no mercado brasileiro

A aplicação da arquitetura EI^3 no mercado brasileiro foi uma experiência muito interessante pois, por ser um mercado mais volátil que os comumente utilizados na área de pesquisa, tornou evidente algumas questões relevantes à arquitetura utilizada. A mais relevante delas é o fato de que treinar o agente para maximizar somente o lucro é uma estratégia ruim de se fazer em uma carteira de ações brasileiras pois, como observado ao longo das últimas seções, o agente torna-se muito sensível a oscilações do mercado e os lucros obtidos em momentos de subida são perdidos logo depois, quando um momento volátil ocorre. Dessa forma, é imprescindível que se utilize alguma métrica de risco na função-objetivo a ser maximizada no algoritmo de treinamento.

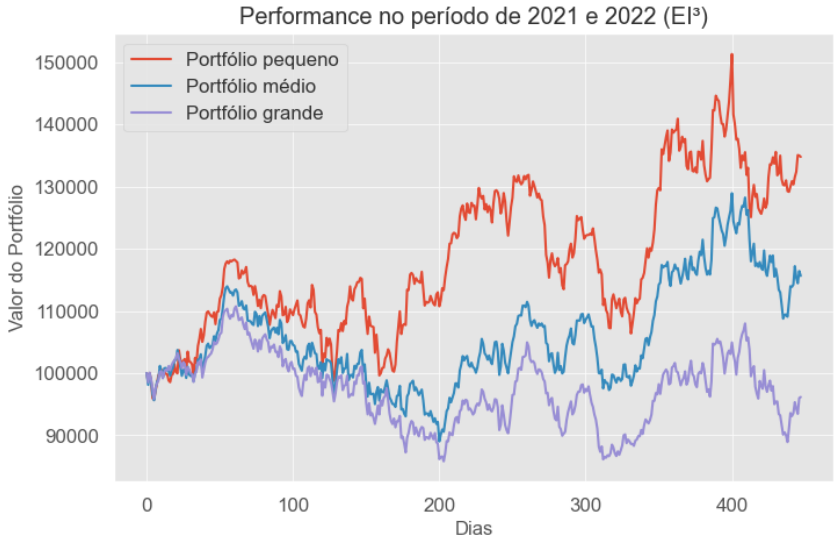


Figura 34 – Comparação de diferentes tamanhos de portfólio no período de 2021 a 2022.

Outro ponto interessante de ser levantado é que os agentes operando no mercado brasileiro possuem comportamentos muito mais parecidos com os da estratégia *uniform buy and hold* do que os agentes operando nos mercados NYSE e NASDAQ. Um possível motivo para isso está no fato de que o Brasil, sendo um mercado emergente, possui uma tendência de crescimento apesar de suas oscilações e, com isso, a estratégia de investir em diversas ações e aguardar torna-se uma boa opção. O problema é que espera-se que um agente otimizador de carteiras consiga explorar essas oscilações do mercado, algo que, como vimos, não está acontecendo tão bem.

Por fim, durante o treinamento, foi possível perceber que a arquitetura EI^3 teve mais dificuldades em observar padrões e definir uma política boa no mercado brasileiro, algo que acontecia com facilidade nos mercados americanos. Por um lado, isso é bom pois evita o problema de *overfitting* que aconteceu com frequência nos experimentos das seções 6.4.1 e 6.4.2, mas por outro, o agente ficava mais preso em máximos locais e parecia ter muito mais dificuldade para aprender novos comportamentos, além de demorar muito mais para terminar o treinamento.

7 Considerações finais

7.1 Conclusões

Em resumo, este documento faz uma breve revisão bibliográfica sobre a área de otimização de carteira de ações com aprendizado por reforço, introduz um ambiente de simulação *open source* e apresenta resultados de testes de duas arquiteturas relevantes para a área no mercado americano, sendo a melhor delas testada também no brasileiro.

Em relação à revisão bibliográfica, foi possível observar as tendências da área de pesquisa e verificar suas lacunas. Entre essas lacunas, observou-se que há poucas ferramentas que ajudem na reprodutibilidade de trabalhos e há poucos estudos realizados no mercado brasileiro. Essa conclusão foi importante pois ditou os passos seguintes do trabalho.

O primeiro desses passos foi o desenvolvimento de um ambiente de simulação utilizando a formulação introduzida em (JIANG; XU; LIANG, 2017), mas que fizesse uso de ferramentas mais modernas de aprendizado por reforço e fosse fácil de aplicar. Posteriormente, foi feita uma análise de duas arquiteturas relevantes para a área (a EIIE e a EI^3) no mercado americano e os resultados mostraram-se próximos dos de artigos de referência. Assim, foi possível concluir que o ambiente desenvolvido funciona como o esperado e que a reprodução das arquiteturas e dos algoritmos de treino foram bem-sucedidos.

Por fim, aplicaram-se essas mesmas arquiteturas no mercado brasileiro e observou-se que os agentes seguem um comportamento mais parecido com o da estratégia clássica *buy and hold*, de comprar ações e segurá-las indefinidamente. Apesar de o resultado ser relativamente fraco, ele evidencia que a aplicação dessas arquiteturas no mercado brasileiro é algo promissor, visto que ainda é possível melhorá-la consideravelmente para que ela consiga lidar com a volatilidade maior da bolsa de valores do País.

7.2 Contribuições

Este trabalho teve como principal contribuição o desenvolvimento de ferramentas que melhoram a reprodutibilidade e diminuem a dificuldade de desenvolvimento e pesquisa de sistemas otimizadores de carteiras de ações com aprendizado por reforço. Com o POE (COSTA; COSTA, 2023), desenvolvedores e pesquisadores podem facilmente ter acesso a um ambiente de simulação que implementa a formulação estado-da-arte do problema e podem utilizá-lo para fazer seus experimentos. Além disso, com a adição, tanto do

ambiente quanto do algoritmo e das duas arquiteturas reproduzidas (EIIE e EI^3) ao *FinRL*, espera-se que a área de pesquisa tenha um considerável aumento de contribuições conforme as ferramentas forem se popularizando.

Por fim, outra contribuição importante foi a análise da efetividade de uma arquitetura convolucional no mercado brasileiro, algo que não foi visto durante os artigos avaliados na revisão bibliográfica. Como os resultados trazem expectativas interessantes para o mercado brasileiro, imagina-se que estudos surjam na bolsa de valores brasileira, modificando a arquitetura e alcançando resultados melhores.

7.3 Perspectivas de continuidade

Ao introduzir o POE (COSTA; COSTA, 2023), este projeto de formatura abre uma gama de possibilidades para trabalhos futuros: pesquisadores podem utilizar facilmente a formulação estado-da-arte de otimização de portfólio para testar suas inovações arquiteturais, sem a necessidade de programar a simulação do mercado financeiro. A formulação utilizada, porém, assume as hipóteses introduzidas na seção 2.2.4, algo que pode não ser realístico dependendo do portfólio e mercado escolhido. Assim, uma continuidade interessante seria modificar a formulação para considerar, principalmente, o impacto dos agentes no mercado.

Além disso, houve, ao longo deste trabalho, uma grande preocupação em adicionar ao *FinRL* (LIU et al., 2022) as arquiteturas notáveis replicadas e testadas. Um bom trabalho futuro seria continuar esse esforço e adicionar outras arquiteturas, principalmente as mais recentes, para que pesquisadores possam comparar seus resultados de forma mais abrangente.

Outro ponto interessante que pode ser o foco de trabalhos futuros está em verificar formas de melhorar o desempenho das arquiteturas convolucionais no mercado brasileiro, seja modificando-as ou alterando o algoritmo de treinamento, testando, por exemplo, outras funções-objetivo (esse tipo de investigação já foi feita em outros mercados, mas não no mercado nacional) e formas de evitar o problema do *overfitting* que acontece ocasionalmente com o *policy gradient*. Além disso, um estudo acerca dos melhores algoritmos de aprendizado por reforço no contexto brasileiro também seria bem-vindo, visto que nem mesmo no mercado internacional há, pelo que se saiba, um estudo grande com esse propósito.

Por fim, ressalta-se que esse trabalho, sendo desenvolvido por um aluno do módulo de pesquisa em engenharia de computação, serve como base para a pesquisa de mestrado que está por vir. Nessa pesquisa, métodos de se modelar o relacionamento entre as ações utilizando redes neurais em grafos (WU et al., 2021b) serão avaliados de modo a desenvolver um agente que seja, de fato, melhor que o estado-da-arte atual.

Referências

AKIBA, T. et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. [S.l.]: arXiv, 2019. Citado 2 vezes nas páginas 56 e 62.

ALMAHDI, S.; YANG, S. Y. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, v. 87, p. 267–279, nov. 2017. ISSN 0957-4174. Citado 6 vezes nas páginas 34, 46, 49, 50, 51 e 53.

AMROUNI, S. *Selimamrouni/Deep-Portfolio-Management-Reinforcement-Learning: V2.0*. 2022. Zenodo. Citado na página 56.

AROUSSI, R. *Ranaroussi/Quantstats*. 2023. Citado 2 vezes nas páginas 55 e 59.

BETANCOURT, C.; CHEN, W.-H. Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications*, v. 164, p. 114002, fev. 2021. ISSN 0957-4174. Citado 7 vezes nas páginas 34, 46, 49, 50, 51, 52 e 53.

COSTA, C. d. S. B.; COSTA, A. H. R. POE: A General Portfolio Optimization Environment for FinRL. In: *Anais Do Brazilian Workshop on Artificial Intelligence in Finance (BWAIF)*. [S.l.]: SBC, 2023. p. 132–143. ISSN 0000-0000. Citado 3 vezes nas páginas 56, 79 e 80.

DENG, Y. et al. Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Transactions on Neural Networks and Learning Systems*, v. 28, n. 3, p. 653–664, mar. 2017. ISSN 2162-2388. Citado na página 21.

FELIZARDO, L. K. et al. *Reinforcement Learning Applied to Trading Systems: A Survey*. [S.l.]: arXiv, 2022. Citado na página 21.

GOODFELLOW, I. et al. Generative Adversarial Nets. In: *Advances in Neural Information Processing Systems*. [S.l.]: Curran Associates, Inc., 2014. v. 27. Citado na página 54.

GUNJAN, A.; BHATTACHARYYA, S. A brief review of portfolio optimization techniques. *Artificial Intelligence Review*, set. 2022. ISSN 1573-7462. Citado na página 21.

HAGHPANAH, M. A. *Gym-Mtsim*. 2021. Citado na página 56.

HAGHPANAH, M. A. *Gym-Anytrading*. 2023. Citado na página 56.

HAMBLY, B.; XU, R.; YANG, H. Recent advances in reinforcement learning in finance. *Mathematical Finance*, v. 33, n. 3, p. 437–503, 2023. ISSN 1467-9965. Citado na página 21.

HENRIQUE, B. M.; SOBREIRO, V. A.; KIMURA, H. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, v. 124, p. 226–251, jun. 2019. ISSN 0957-4174. Citado na página 21.

HESSEL, M. et al. *Rainbow: Combining Improvements in Deep Reinforcement Learning*. [S.l.]: arXiv, 2017. Citado na página 37.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2, n. 5, p. 359–366, jan. 1989. ISSN 0893-6080. Citado na página 28.

HU, Y. et al. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, v. 36, p. 534–551, nov. 2015. ISSN 1568-4946. Citado na página 21.

JANG, J.; SEONG, N. Deep reinforcement learning for stock portfolio optimization by connecting with modern portfolio theory. *Expert Systems with Applications*, v. 218, p. 119556, maio 2023. ISSN 0957-4174. Citado 6 vezes nas páginas 45, 49, 50, 51, 52 e 53.

JIANG, Z.; XU, D.; LIANG, J. *A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem*. [S.l.]: arXiv, 2017. Citado 19 vezes nas páginas 11, 21, 26, 27, 28, 35, 38, 41, 45, 47, 48, 49, 50, 51, 52, 53, 60, 63 e 79.

KANG, Q.; ZHOU, H.; KANG, Y. An Asynchronous Advantage Actor-Critic Reinforcement Learning Method for Stock Selection and Portfolio Management. In: *Proceedings of the 2nd International Conference on Big Data Research*. New York, NY, USA: Association for Computing Machinery, 2018. (ICBDR '18), p. 141–145. ISBN 978-1-4503-6476-8. Citado 6 vezes nas páginas 46, 49, 50, 51, 52 e 53.

LEKOVIĆ, M. Investment diversification as a strategy for reducing investment risk. *Ekonomski horizonti*, v. 20, n. 2, p. 173–187, 2018. ISSN 1450-863X. Citado 3 vezes nas páginas 65, 72 e 74.

LIANG, Z. et al. *Adversarial Deep Reinforcement Learning in Portfolio Management*. [S.l.]: arXiv, 2018. Citado 8 vezes nas páginas 38, 45, 48, 49, 50, 51, 52 e 54.

LILLICRAP, T. P. et al. Continuous control with deep reinforcement learning. set. 2015. Citado 2 vezes nas páginas 38 e 46.

LIM, Q. Y. E.; CAO, Q.; QUEK, C. Dynamic portfolio rebalancing through reinforcement learning. *Neural Computing and Applications*, v. 34, n. 9, p. 7125–7139, maio 2022. ISSN 1433-3058. Citado 6 vezes nas páginas 46, 49, 50, 51, 52 e 53.

LIN, Y.-C. et al. Multiagent-based deep reinforcement learning for risk-shifting portfolio management. *Applied Soft Computing*, v. 123, p. 108894, jul. 2022. ISSN 1568-4946. Citado 6 vezes nas páginas 47, 49, 50, 51, 52 e 53.

LIU, X.-Y. SSRN Scholarly Paper, *FinRL-Meta: Market Environments and Benchmarks for Data-Driven Financial Reinforcement Learning*. Rochester, NY: [s.n.], 2022. Citado na página 56.

LIU, X.-Y. et al. *ElegantRL-Podracer: Scalable and Elastic Library for Cloud-Native Deep Reinforcement Learning*. [S.l.]: arXiv, 2022. Citado na página 57.

LIU, X.-Y. et al. *Practical Deep Reinforcement Learning Approach for Stock Trading*. 2018. Https://arxiv.org/abs/1811.07522v3. Citado 6 vezes nas páginas 46, 49, 50, 51, 52 e 53.

LIU, X.-Y. et al. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. In: *Proceedings of the Second ACM International Conference on AI in Finance*. New York, NY, USA: Association for Computing Machinery, 2022. (ICAIF '21), p. 1–9. ISBN 978-1-4503-9148-1. Citado 5 vezes nas páginas 43, 55, 56, 61 e 80.

MA, C. et al. Multi-agent deep reinforcement learning algorithm with trend consistency regularization for portfolio management. *Neural Computing and Applications*, nov. 2022. ISSN 1433-3058. Citado 7 vezes nas páginas 47, 48, 49, 50, 51, 52 e 53.

Magdon-Ismail, M. et al. On the maximum drawdown of a Brownian motion. *Journal of Applied Probability*, Cambridge University Press, v. 41, n. 1, p. 147–161, mar. 2004. ISSN 0021-9002, 1475-6072. Citado na página 27.

MNIH, V. et al. Asynchronous Methods for Deep Reinforcement Learning. fev. 2016. Citado na página 46.

MUMUNI, A.; MUMUNI, F. Data augmentation: A comprehensive survey of modern approaches. *Array*, v. 16, p. 100258, dez. 2022. ISSN 2590-0056. Citado na página 45.

NASSIRTOUSSI, A. K. et al. Text mining for market prediction: A systematic review. *Expert Systems with Applications*, v. 41, n. 16, p. 7653–7670, nov. 2014. ISSN 0957-4174. Citado na página 21.

PARK, H.; SIM, M. K.; CHOI, D. G. An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems with Applications*, v. 158, p. 113573, nov. 2020. ISSN 0957-4174. Citado 6 vezes nas páginas 46, 49, 50, 51, 52 e 53.

PASZKE, A. et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. [S.l.]: arXiv, 2019. Citado na página 55.

PATTON, A. J. Are "Market Neutral" Hedge Funds Really Market Neutral? *The Review of Financial Studies*, v. 22, n. 7, p. 2495–2530, jul. 2009. ISSN 0893-9454. Citado 2 vezes nas páginas 47 e 51.

RAFFIN, A. et al. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, v. 22, n. 268, p. 1–8, 2021. ISSN 1533-7928. Citado na página 57.

SCHULMAN, J. et al. Proximal Policy Optimization Algorithms. jul. 2017. Citado 2 vezes nas páginas 38 e 46.

SHARPE, W. F. The Sharpe Ratio. *The Journal of Portfolio Management*, Institutional Investor Journals Umbrella, v. 21, n. 1, p. 49–58, out. 1994. ISSN 0095-4918, 2168-8656. Citado na página 27.

SHI, S. et al. A Multi-Scale Temporal Feature Aggregation Convolutional Neural Network for Portfolio Management. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Beijing China: ACM, 2019. p. 1613–1622. ISBN 978-1-4503-6976-3. Citado 13 vezes nas páginas 11, 35, 36, 41, 45, 48, 49, 50, 51, 52, 53, 60 e 63.

SHI, S. et al. GPM: A graph convolutional network based reinforcement learning framework for portfolio management. *Neurocomputing*, v. 498, p. 14–27, ago. 2022. ISSN 0925-2312. Citado 15 vezes nas páginas 13, 47, 48, 50, 51, 52, 53, 54, 61, 63, 64, 65, 66, 67 e 68.

SOLEYMANI, F.; PAQUET, E. Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath. *Expert Systems with Applications*, v. 156, p. 113456, out. 2020. ISSN 0957-4174. Citado 6 vezes nas páginas 45, 49, 50, 51, 52 e 53.

SOLEYMANI, F.; PAQUET, E. Deep graph convolutional reinforcement learning for financial portfolio management – DeepPocket. *Expert Systems with Applications*, v. 182, p. 115127, nov. 2021. ISSN 0957-4174. Citado 7 vezes nas páginas 47, 49, 50, 51, 52, 53 e 54.

SONG, Z. et al. From deterministic to stochastic: An interpretable stochastic model-free reinforcement learning framework for portfolio optimization. *Applied Intelligence*, v. 53, n. 12, p. 15188–15203, jun. 2023. ISSN 1573-7497. Citado 6 vezes nas páginas 46, 49, 50, 51, 52 e 53.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN 978-0-262-03924-6. Citado 5 vezes nas páginas 21, 32, 33, 37 e 46.

SZEGEDY, C. et al. Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. p. 1–9. ISSN 1063-6919. Citado 2 vezes nas páginas 35 e 45.

TEAM, T. P. D. *Pandas-Dev/Pandas: Pandas*. 2023. Zenodo. Citado na página 57.

TOWERS, M. et al. *Gymnasium*. 2023. Zenodo. Citado 3 vezes nas páginas 41, 55 e 56.

TRAN, M.; Pham-Hi, D.; BUI, M. Optimizing Automated Trading Systems with Deep Reinforcement Learning. *Algorithms*, Multidisciplinary Digital Publishing Institute, v. 16, n. 1, p. 23, jan. 2023. ISSN 1999-4893. Citado 4 vezes nas páginas 45, 49, 50 e 51.

TUCKER, L. R. Some mathematical notes on three-mode factor analysis. *Psychometrika*, v. 31, n. 3, p. 279–311, set. 1966. ISSN 1860-0980. Citado na página 46.

VAIDYA, R. Moving Average Convergence-Divergence (MACD) Trading Rule: An Application in Nepalese Stock Market "NEPSE". *Quantitative Economics and Management Studies*, v. 1, n. 6, p. 366–374, nov. 2020. ISSN 2722-6247. Citado na página 46.

van Hasselt, H.; GUEZ, A.; SILVER, D. *Deep Reinforcement Learning with Double Q-learning*. [S.l.]: arXiv, 2015. Citado na página 37.

WANG, M.; KU, H. Risk-sensitive policies for portfolio management. *Expert Systems with Applications*, v. 198, p. 116807, jul. 2022. ISSN 0957-4174. Citado 6 vezes nas páginas 46, 49, 50, 51, 52 e 53.

WENG, L. et al. Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism. *Neurocomputing*, v. 402, p. 171–182, ago. 2020. ISSN 0925-2312. Citado 7 vezes nas páginas 11, 33, 45, 49, 50, 51 e 52.

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, v. 8, n. 3, p. 229–256, maio 1992. ISSN 1573-0565. Citado na página 37.

WU, M.-E. et al. Portfolio management system in equity market neutral using reinforcement learning. *Applied Intelligence*, v. 51, n. 11, p. 8119–8131, nov. 2021. ISSN 1573-7497. Citado 7 vezes nas páginas 46, 48, 49, 50, 51, 52 e 53.

WU, Z. et al. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, v. 32, n. 1, p. 4–24, jan. 2021. ISSN 2162-237X, 2162-2388. Citado 2 vezes nas páginas 54 e 80.

YANG, B. *Deep Reinforcement Learning for Automated Stock Trading*. 2023. Https://towardsdatascience.com/deep-reinforcement-learning-for-automated-stock-trading-f1dad0126a02. Citado 5 vezes nas páginas 47, 49, 50, 51 e 52.

YU, P. et al. *Model-Based Deep Reinforcement Learning for Dynamic Portfolio Optimization*. [S.l.]: arXiv, 2019. Citado 7 vezes nas páginas 46, 49, 50, 51, 52, 53 e 54.

ZHENG, B. et al. *Imitation Learning: Progress, Taxonomies and Challenges*. [S.l.]: arXiv, 2022. Citado na página 46.