



دانشگاه تهران  
پردیس دانشکده‌های فنی  
دانشکده مهندسی برق و کامپیوتر



پیاده سازی عامل هوشمند با یادگیری تقویتی عمیق برای معاملات الگوریتمی در بازار های مالی

پایان نامه برای دریافت درجه کارشناسی  
در رشته مهندسی کامپیوتر

پارسا صدری سینکی

شماره دانشجویی

۸۱۰۱۹۵۵۲۶

استاد راهنما:

دکتر سعید صفری

شهریورماه ۱۴۰۰

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

## تعهدنامه اصالت اثر

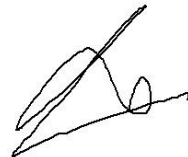
### باسمه تعالی

اینجانب پارسا صدری سینکی تأیید می‌کنم که مطالب مندرج در این پایان‌نامه حاصل تلاش اینجانب است و به دستاوردهای پژوهشی دیگران که در این نوشته از آنها استفاده شده است مطابق مقررات ارجاع گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم سطح یا بالاتر ارائه نشده است. کلیه حقوق مادی و معنوی این اثر متعلق به دانشکده فنی دانشگاه تهران می‌باشد.

نام و نام خانوادگی دانشجو:

پارسا صدری سینکی

امضای دانشجو:



## چکیده<sup>۱</sup>

امروزه سرمایه‌گذاری و معامله در بازار رمزارزها به یکی از بازارهای جذاب برای افراد و کارگزاری‌ها تبدیل شده‌است. در سال‌های اخیر معاملات در این بازارها از معاملات دستی به سمت معاملات الگوریتمی و معاملات براساس هوش مصنوعی سوق پیدا کرده‌است. هدف این پژوهش پیاده‌سازی عامل هوشمندی بر مبنای یادگیری تقویتی عمیق برای تشخیص زمان و مقدار خرید و فروش یک جفت رمزارز است. روش انتخابی برای این پیاده‌سازی به این صورت است که بخش‌های مختلف این عامل هوشمند از یکدیگر جدا شوند. فرایند معاملات در بازارهای رمزارز توسط عامل‌های هوشمند شامل دو بخش اصلی، پیش‌بینی روند قیمتی رمزارز و مدیریت خرید و فروش بر اساس سیگنال‌های ورودی است. در این پایان‌نامه هدف پیاده‌سازی عامل هوشمندی است که در دسته دوم قرار می‌گیرد و با استفاده از سیگنال‌های ورودی، تصمیم به زمان و مقدار مناسب برای خرید و فروش رمزارزها می‌گیرد. این نوع جداسازی اجزای یک عامل هوشمند به تنظیم، اشکال‌زدایی و درک بهتر از سیستم کلی منتهی خواهد شد. در گذشته مقدار زیادی تحقیق بر روی روش‌های مختلف پیش‌بینی روند بازار از جمله پیش‌بینی با استفاده از شبکه‌های عصبی، تحلیل تکنیکال و جمع‌آوری و تحلیل متن اخبار انجام شده‌است. به محققین آتی توصیه می‌شود برای پژوهش‌های آینده ترکیب نتایج انواع پژوهش‌های مربوط به پیش‌بینی روند و قیمت بازار و تحقیق‌های مشابه این پایان‌نامه را بررسی کنند.

کلمات کلیدی: رمزارز، یادگیری تقویتی عمیق، معاملات الگوریتمی، مدیریت سبد دارایی

---

<sup>1</sup> Abstract

## فهرست

۱	فصل اول: مقدمه و بیان مساله
۲	۱.۱. مقدمه
۳	۲.۱. تاریخچه‌ای از موضوع تحقیق
۴	۳.۱. شرح مساله تحقیق
۵	۴.۱. تعریف موضوع تحقیق
۵	۵.۱. اهداف و آرمان‌های کلی تحقیق
۶	۶.۱. روش انجام تحقیق
۶	۷.۱. ساختار پایان‌نامه
۷	۲. فصل دوم: مفاهیم اولیه و پیش زمینه
۸	۱.۲. مقدمه
۸	۲.۲. مفاهیم اولیه
۸	۲.۲.۱. بازارهای مالی
۸	۲.۲.۱.۱. بازار تبادل ارز خارجی
۹	۲.۲.۱.۲. بازار رمزارزها
۹	۲.۲.۲. یادگیری عمیق
۱۱	۲.۲.۲.۱. نحوه اتصال لایه‌ها در شبکه‌های عصبی برای ساخت مدل‌های یادگیری عمیق
۱۲	۲.۲.۲.۲. لایه پنهان
۱۲	۲.۲.۲.۳. لایه خروجی
۱۲	۳.۲.۲. یادگیری تقویتی
۱۳	۴.۲.۲. یادگیری تقویتی عمیق
۱۳	۲.۲.۴.۱. الگوریتم یادگیری تقویتی Q-learning
۱۴	۲.۲.۴.۲. الگوریتم یادگیری تقویتی Advantage Actor-Critic
۱۵	۲.۲.۴.۳. الگوریتم یادگیری تقویتی Proximal Policy Optimization
۱۶	۳.۲. خلاصه و جمع‌بندی
۱۷	۳. فصل سوم: روش تحقیق
۱۸	۱.۳. مقدمه
۱۸	۲.۳. روش پیشنهادی
۱۸	۳.۲.۱. جمع‌آوری داده‌ها
۱۹	۳.۲.۲. تولید سیگنال ورودی
۲۰	۳.۲.۳. محیط یادگیری تقویتی
۲۱	۴.۲.۳. مدل یادگیری تقویتی

۲۲	۳.۲.۴.۱. لایه ورودی
۲۲	۳.۲.۴.۲. لایه پنهان
۲۲	۳.۲.۴.۳. لایه خروجی
۲۳	۳.۳. ابزارهای مورد استفاده
۲۳	۳.۳.۱. TensorTrade
۲۳	۳.۳.۲. Stable-Baselines3
۲۴	۳.۳.۳. Finnhub
۲۴	۳.۴. معیار ارزیابی
۲۵	۳.۵. خلاصه و جمع‌بندی
۲۶	۴. فصل چهارم: پیاده‌سازی
۲۷	۴.۱. مقدمه
۲۷	۴.۲. پیاده‌سازی
۲۷	۴.۲.۱. جمع‌آوری داده‌ها
۲۸	۴.۲.۲. تولید سیگنال‌های ورودی بر روی داده‌ها
۲۹	۴.۲.۳. تغییرات انجام شده بر محیط یادگیری تقویتی در کتابخانه TensorTrade
۳۱	۴.۲.۳.۱. پیاده‌سازی شروع از زمان تصادفی به هنگام ریست شدن محیط
۳۳	۴.۲.۳.۲. پیاده‌سازی زمان پایان محیط هنگام ضرر بیش از ۵۰ درصدی
۳۴	۴.۲.۳.۳. پیاده‌سازی اقدام سهگانه خرید، فروش و نگهداری ارز برای محیط
۳۵	۴.۲.۳.۴. پیاده‌سازی اقدام خرید و فروش ارز با مقادیر پیوسته برای محیط
۳۶	۴.۲.۳.۵. پیاده‌سازی تابع پاداش محاسبه سود منهای بنچمارک برای محیط
۳۶	۴.۲.۳.۶. پیاده‌سازی تابع پاداش محاسبه سود منهای بنچمارک و جریمه عدم معامله برای محیط
۳۷	۴.۲.۳.۷. پیاده‌سازی تابع پاداش محاسبه سود بر حسب هر معامله برای محیط
۳۸	۴.۲.۳.۸. پیاده‌سازی تابع پاداش محاسبه سود با جریمه عدم معامله برای محیط
۳۸	۴.۲.۴. مدل یادگیری تقویتی عمیق
۴۰	۴.۲.۵. پیاده‌سازی معیارهای ارزیابی
۴۰	۴.۲.۵.۱. نسبت شارپ
۴۰	۴.۲.۵.۲. معیارهای مربوط به زمان و مقدار ضرردهی متوالی
۴۰	۴.۲.۵.۳. مقدار سوددهی
۴۱	۴.۳. نتایج به‌دست آمده از ارزیابی مدل
۴۴	۴.۴. تحلیل نتایج
۴۶	۴.۵. خلاصه و جمع‌بندی
۴۷	۵. فصل پنجم: جمع‌بندی، نتیجه‌گیری و پیشنهادها

۴۸	۵. ۱. جمع بندی .....
۴۸	۵. ۲. نتیجه گیری .....
۴۸	۵. ۲. ۱. دستاوردها .....
۴۹	۵. ۲. ۲. محدودیت ها .....
۴۹	۵. ۲. ۳. پیشنهادها .....
۵۰	۶. مراجع .....

## فهرست شکل‌ها

- شکل (۱-۲) نمونه یک شبکه عصبی ..... ۱۰
- شکل (۲-۲) نمونه عملیات در یک نورون شبکه عصبی ..... ۱۱
- شکل (۳-۲) معماری Advantage Actor-Critic ..... ۱۵
- شکل (۱-۴) نمودار سود حاصل نگهداری رمزارز بیت کوین ..... ۴۳
- شکل (۲-۴) نمودار سود حاصل از استفاده از مدل پیشنهادی ..... ۴۴



## فهرست جدول‌ها

جدول (۱-۳) انواع نویز سیگنال‌های ورودی .....	۱۹
جدول (۲-۳) انواع اقدام‌های ممکن .....	۲۰
جدول (۳-۳) انواع تابع پاداش .....	۲۱
جدول (۱-۴) نتایج پیاده‌سازی .....	۴۲

## فهرست علائم اختصاری

CNN	Convolutional Neural Network
LSTM	Long short-term memory
DDPG	Deep Deterministic Policy Gradient
DDQN	Dueling Deep Q Networks
ReLU	Rectified Linear Activation Unit
Tanh	Hyperbolic Tangent Function
TD3	Twin Delayed DDPG
DQN	Deep Q Networks
A2C	Advantage Actor-Critic
PPO	Proximal Policy Optimization

## ۱. فصل اول: مقدمه و بیان مساله

## ۱.۱. مقدمه

امروزه بازار تبادل رمزارزها<sup>۱</sup> بسیار رونق یافته و روزانه حجم عظیمی بالغ بر ۱۵۰ میلیارد دلار در این بازار معامله می شود. موفقیت در معاملات این بازار نیازمند توانایی پیش‌بینی روند قیمت رمزارزها و تصمیم‌گیری درست و به موقع بر اساس آنهاست.

به غیر از بازار رمزارزها، بازارهای مالی دیگری نیز وجود دارند؛ مثل بازار تبادل ارز خارجی<sup>۲</sup> (فارکس) که پیش‌تر بر روی آنها تحقیقات بیشتری انجام شده است. پیش‌بینی در بازارهای رمزارزها دارای شباهت‌ها و تفاوت‌هایی با دیگر بازارهای مالی همچون بازار تبادل ارز خارجی است.

شباهت آنها در آن است که می‌توان در هر دو آنها از روش‌های تحلیل تکنیکال برای پیش‌بینی استفاده نمود. تفاوت آنها در آن است که بازار رمزارزها با توجه به ماهیتش دارای میزان تغییرات و نوسانات شدیدتری نسبت به دیگر بازارها هستند. کنترل و کاهش ریسک ناشی از این میزان نوسانات یکی از چالش‌های اصلی حضور موثر در بازار رمزارزها است.

در سال اخیر بسیاری از صندوق‌های سرمایه‌گذاری و مدیران دارایی شروع به درج دارایی‌های مرتبط با رمزارزها در سبد دارایی خود کرده‌اند. همچنین در [۱] نوشته شده است که بیش از ۸۵ درصد مقالات این حوزه از سال ۲۰۱۸ ظاهر شده‌اند. که نشان‌دهنده ظهور بازار رمزارزها به عنوان یک حوزه سرمایه‌گذاری و تحقیقاتی جدید است.

---

<sup>1</sup> Cryptocurrency

<sup>2</sup> Foreign exchange market

## ۲.۱. تاریخچه‌ای از موضوع تحقیق

یادگیری عمیق<sup>۱</sup> چندین سال می باشد که حوزه پر مخاطبی در بازار های مالی می باشد. از یادگیری عمیق برای پیش بینی قیمت یا روند قیمت ارز در بازار های مالی استفاده می شود [۲، ۳]. که گروهی از پژوهشگران برای پیشبینی روند قیمت بیت کوین<sup>۲</sup> به دقت پیشبینی ۷۱ درصد نیز رسیده بودند [۳]. کاربرد یادگیری تقویتی عمیق<sup>۳</sup> در بازار های مالی در سال های اخیر طرفداران زیادی میان پژوهشگران پیدا کرده است. دو مورد از مهمترین این کاربردها که در مقالات به چشم می خورد. کاربرد اول استفاده عامل یادگیری تقویتی به طور مستقیم در برخورد با بازار و با مسئولیت خرید و فروش می باشد [۴، ۵، ۶، ۷]. کاربرد دوم مربوط به استفاده از یادگیری تقویتی به عنوان مدیر سبد دارایی است [۷]. هر دو این کاربردها عملاً یک کاربرد مشترک مدیریت سرمایه و خرید و فروش می باشد و تفاوت آن ها بیشتر در نحوه طرح مساله می باشد.

پژوهشگران اکثراً به عنوان داده کمکی برای مدل از داده های اخبار و شبکه اجتماعی استفاده می کنند [۳، ۷، ۸]. همچنین قبل از ورود مستقیم داده ها به مدل یادگیری تقویتی پژوهشگران برای پیدا کردن ویژگی های جدید از داده های ورودی قیمت و اخبار، داده ها را از شبکه عصبی پیچشی<sup>۴</sup> (CNN) و شبکه عصبی بازگشتی با حافظه طولانی کوتاه-مدت<sup>۵</sup> (LSTM) عبور می دهند [۶، ۸].

تعدادی پژوهش برای پیاده سازی مدل یادگیری تقویتی عمیق برای خروجی های پیوسته از مدل Deep Deterministic Policy Gradient (DDPG)<sup>۱</sup> استفاده کردند [۶، ۷]. همچنین برای خروجی های پیوسته از الگوریتم Dueling Deep Q Networks (DDQN)<sup>۲</sup> استفاده کردند [۱، ۴، ۹]. برای تابع فعال سازی در شبکه عصبی از توابع مرسوم ReLU در [۴] و Tanh در [۵] استفاده شده است. قسمت مهمی از طراحی و پیاده سازی یادگیری تقویتی بخش طراحی تابع پاداش دهنده استفاده شده در محیط آموزش مدل می باشد. پژوهشگری نوشته است که در ابتدا از تابع پاداش دهنده ای بر اساس

<sup>1</sup> Deep Learning

<sup>2</sup> Bitcoin

<sup>3</sup> Deep Reinforcement Learning

<sup>4</sup> Convolutional Neural Network

<sup>5</sup> Long short-term memory

سود یا ضرر کسب شده بعد از فروش به عنوان پاداش استفاده کرده بود که موجب شده بود عامل هوشمند معاملات در ضرر را بفروش نرساند تا شاید بعداً به سود برسد، که در نسخه های بعدی تابع پاداش دهنده مدل خود را به سود تحقق یافته نشده با احتساب ریسک و با عادی سازی<sup>۱</sup> کردن مقدار پاداش تغییر داد [۴]. استفاده از پاداش نهایی هر معامله بعد از فروش در پژوهشی دیگر نیز کاربر داشته است که در آن پژوهش نیز منجر به موفقیت نشد [۶]. تابع پاداشی که اکثراً در مقالات استفاده شده است و موجب کسب سود در بازارهای مالی شده است، تابع پاداش سود تحقق یافته نشده با احتساب ریسک با استفاده از نسبت شارپ می باشد [۴، ۵، ۷]. دو تابع پاداش دیگر نیز موفق به کسب موفقیت هایی در پژوهش های دیگر شدند. یک تابع پاداش سود محقق نشده منهای کارمزد تراکنش است [۷]. تابع پاداش موفق دیگر استفاده از یک پاداش باینری برای مشخص کردن سودده یا ضررده بودن معامله بود [۶].

### ۳.۱. شرح مساله تحقیق

برای کسب سود مناسب و جلوگیری از ضررهای هنگفت و ناگهانی در بازارهای مالی نیاز به پیش بینی روند کلی بازار و مدیریت ریسک در معاملات است. زیرا هیچ بازار مالی وجود ندارد که دائماً روند ثابتی داشته باشد و در سیکل های مختلفی از سوددهی و ضرردهی می افتد. برای پیش بینی روند قیمت و استفاده بهینه از این پیش بینی ها نیاز به تولید عاملی هوشمند است که با استفاده از یادگیری الگوهای گذشته بازار و تطبیق آن با منطق مورد نظر برای سوددهی، منجر به افزایش سرمایه با حداقل ریسک شود. هدف این پایان نامه پیاده سازی سیستمی برای استفاده از سیگنال های پیش بینی روند قیمت بازار رمزارزها و مدیریت خرید و فروش در این بازار است.

<sup>1</sup> Normalize

## ۴.۱. تعریف موضوع تحقیق

تحقیقات گسترده‌ای در حوزه معاملات در بازارهای مالی انجام شده‌است. با بهبود روزبه‌روز هوش مصنوعی و الگوریتم‌های یادگیری ماشین<sup>۱</sup> نیز بستر مناسبی برای استفاده از یادگیری ماشین در این حوزه بوجود آمده‌است. حوزه معاملات در بازارهای رمزارز توسط عامل‌های هوشمند شامل دو زیرحوزه اصلی، اول، پیش‌بینی روند قیمتی رمزارز و دوم، مدیریت خرید و فروش بر اساس سیگنال‌های ورودی است. در این پایان‌نامه هدف پیاده‌سازی عامل هوشمندی است که در دسته دوم از این دو دسته قرار می‌گیرد و با استفاده از سیگنال‌های ورودی تصمیم به زمان و مقدار مناسب برای خرید و فروش رمزارزها می‌گیرد. به طور کلی مسئول مدیریت معاملات بر بستر بازار مالی مورد نظر خواهد بود.

## ۵.۱. اهداف و آرمان‌های کلی تحقیق

به علت تعدد جفت‌ارزها در این بازار، افراد امکان تحلیل تمامی آنها را ندارند. همچنین مواردی را که برای تحلیل و بررسی انتخاب می‌کنند، در بازه‌های کوتاه مدت ممکن است به سرعت کافی موفق به بررسی کامل نشوند یا در طول شبانه‌روز به طور مداوم به تحلیل بازار نپردازند و فرصت‌های خرید و فروش بسیاری را از دست بدهند. به همین دلایل ضرورت وجود عامل هوشمندی که به‌طور خودکار در طول شبانه‌روز به معامله بپردازد، حس می‌شود. اکثر تحقیقات این حوزه به بررسی پیش‌بینی روند قیمت در آینده بسنده کرده‌اند در حالی که نیاز به یک سیستم ثانویه برای استفاده از سیگنال‌های تولید شده وجود دارد. هدف این تحقیق بررسی و تولید این سیستم ثانویه است تا با استفاده از آن بتوان به سوددهی حقیقی رسید.

در این پژوهش، هدف، علاوه بر پیاده‌سازی این عامل هوشمند و بررسی موفقیت آن، دستیابی به یک بستر مناسب برای آزمایش عامل‌های هوشمند دیگر که در آینده تولید می‌شوند نیز هست.

<sup>1</sup> Machine Learning

## ۶.۱. روش انجام تحقیق

در این پروژه ابتدا یک رابط برنامه‌نویسی کاربردی<sup>۱</sup> (API) برای دریافت داده‌های قیمتی بازار مالی پیاده‌سازی می‌کنیم. سپس با استفاده از این داده‌ها سیگنال‌های پیش‌بینی روند قیمت در دوره‌های کوتاه، میان و بلند مدت را همراه با درصد قابل قبولی از خطا تولید می‌کنیم. در مرحله بعد مدل یادگیری تقویتی عمیقی را برای کسب سود از سیگنال‌های ورودی طراحی می‌کنیم. در مرحله آخر مدل تولید شده را در بستر انتخابی برای آزمایش مورد آزمون قرار می‌دهیم تا میزان موفقیت آن را با نگهداری ساده آن رمزارزها مقایسه کنیم.

## ۷.۱. ساختار پایان‌نامه

فصل دوم، شامل بررسی مفاهیم اولیه مربوط به حوزه بازارهای مالی و شبکه‌های عصبی، تعاریف اساسی یادگیری عمیق و یادگیری تقویتی، مروری بر پیشینه‌ی تحقیق و پیش‌زمینه‌های مورد نیاز برای درک هرچه بهتر موضوع کاربرد یادگیری تقویتی عمیق در بازارهای مالی خواهیم بود.

فصل سوم در برگیرنده‌ی بررسی ابزارها و کتابخانه‌های استفاده شده، معیار ارزیابی نتایج به دست آمده، توضیح مربوط به مدل پیشنهادی و پیاده‌سازی مدل و محیط آموزش آن شده است.

در فصل چهارم در مورد پیاده‌سازی مدل یادگیری تقویتی عمیق و محیط آن صحبت خواهیم کرد. در انتها فصل نیز به نتایج پیاده‌سازی پروژه پرداخته خواهد شد.

در نهایت، در فصل پنجم، نتیجه‌گیری‌های کلی حاصل شده در این تحقیق، دستاوردها و محدودیت‌ها مورد بحث قرار می‌گیرد و پیشنهادهایی برای ادامه‌ی مسیر به علاقمندان این حوزه‌ی ارائه خواهد شد.

<sup>1</sup> Application Programming Interface



## ۲. فصل دوم: مفاهیم اولیه و پیش زمینه

## ۱.۲. مقدمه

در این فصل مفاهیم اولیه و پیش زمینه‌هایی را که جهت درک هرچه بهتر موضوع‌های مطرح شده در این پایان نامه مورد نیاز است بررسی می‌کنیم. این مفاهیم شامل مفاهیم مربوط به بازارهای مالی، یادگیری عمیق و یادگیری تقویتی است.

## ۲.۲. مفاهیم اولیه

### ۱.۲.۲. بازارهای مالی

دو عدد از بازارهای مالی جهانی که امکان دسترسی به آن‌ها به سادگی وجود دارد عبارتند از بازار تبادل ارزهای خارجی و بازار رمزارزها. در ادامه به توصیف بیشتر این دو بازار می‌پردازیم و دلیل انتخاب بازار رمزارزها را برای این پایان‌نامه توضیح می‌دهیم.

### ۱.۱.۲.۲. بازار تبادل ارز خارجی

بازار تبادل ارز خارجی یا همان بازار فارکس یک بازار نامتمرکز دوطرفه، برای تبادل و معامله‌گری ارز است. در حال حاضر فارکس از لحاظ حجم معاملات، بزرگترین بازار مالی در جهان است و در این بازار تمامی بانک‌ها، مؤسسات مالی بزرگ، شرکت‌های چند ملیتی، شرکت‌های بیمه، شرکت‌های صادرات و واردات، صندوق‌های بازنشستگی و اشخاص حقیقی در آن ارزهای مختلفی همچون دلار آمریکا، یورو، پوند، ین و دیگر ارزهای رسمی کشورها را داد و ستد می‌کنند.

خصوصیت اصلی این بازار مالی این است که قیمت‌های جفت‌ارز در طول زمان تغییرات خیلی شدیدی ندارند و فقط دارای مقداری نوسان در قیمت هستند. به همین دلیل به طور کلی کسب سود از این بازار معمولاً صرفاً از طریق معامله بین ارزهای مختلف به دست می‌آید و صرف خرید یک ارز و نگهداری آن روش مناسبی برای کسب سود نیست. به همین دلیل نیز این بازار معمولاً یکی از جذاب‌ترین بازارها برای پیاده‌سازی عاملی هوشمند بر روی آن است.

## ۲.۱.۲.۲. بازار رمزارزها

بازار رمزارزها یک بازار بسیار جدید است به طوری که اولین رمزارز به نام بیت کوین در سال ۲۰۰۹ معرفی شد. در حال حاضر تعداد بسیار زیادی جفت‌ارز در این بازار موجود است و روزانه به این تعداد افزوده می‌شود. از خصوصیات اصلی این بازار می‌توان به نوسانات شدید آن اشاره کرد. یکی از موارد تفاوت این بازار با فارکس این است که به غیر از کسب سود از معاملات، می‌توان از این بازار با صرفاً نگهداری بعضی از ارزها در طول زمان به سود رسید.

صرافی‌های مربوط به این بازار API های مناسبی ایجاد کرده‌اند که امکان معاملات خودکار و الگوریتمی را بسیار ساده می‌کنند.

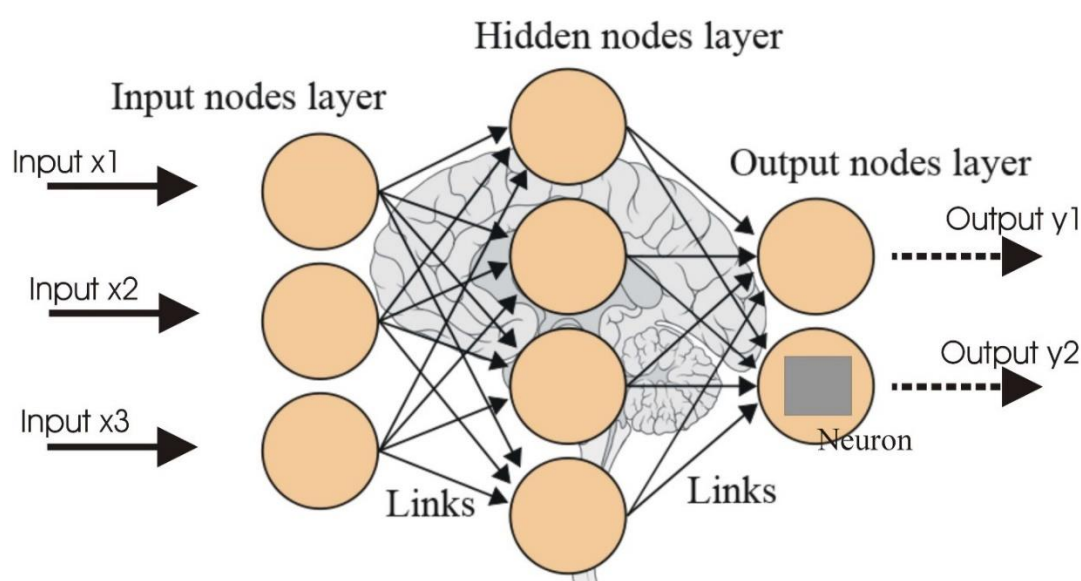
به علت وجود نوسانات زیاد فرصت‌های زیادی برای معامله در این بازار ایجاد می‌شود و همچنین این نوسانات موجب ایجاد ریسک زیادی برای معامله‌گر می‌شود. به همین دلایل این بازار، بازار مناسبی برای پیاده‌سازی عامل هوشمند برای معاملات الگوریتمی است و ما نیز این بازار را به عنوان مرجع آزمایش قرار دادن عامل هوشمند پیاده‌سازی شده خود قرار دادیم.

## ۲.۲.۲. یادگیری عمیق

یادگیری عمیق زیرشاخه‌ای از یادگیری ماشین است. الگوریتم‌ها و ساختارهای استفاده شده در یادگیری عمیق به شکلی مشابه شبکه عصبی موجود در مغز است که در نتیجه آن فرایند یادگیری مشابه آنچه در مغز انسان هست، انجام می‌گیرد. در این نوع یادگیری سعی می‌شود با در نظر گرفتن داده‌ها به روابط پنهان میان آنها پی برده شود. همچنین این نوع ساختار یادگیری توانایی انطباق با تغییرات داده یا ورودی را دارد و برای این کار نیازی به تغییر اساسی ساختار شبکه یادگیری نیست. یکی از مهم‌ترین دلایل استفاده بیشتر از یادگیری عمیق در سال‌های اخیر، حجم بالای داده‌ای است که امروزه تولید می‌شود.

امروزه به دلیل پیشرفت‌هایی که در یادگیری عمیق وجود داشته، این نوع یادگیری در زمینه‌های مختلفی استفاده شده است. به دلیل عملکرد مناسب یادگیری عمیق در زمینه‌هایی مانند تحلیل سری‌های زمانی، یادگیری عمیق در این زمینه‌ها بسیار مورد توجه قرار گرفته است و کارهای پژوهشی زیادی بر اساس آن انجام می‌شود. مدل‌های یادگیری عمیق به صورت لایه‌ای عمل می‌کنند. یک مدل ابتدایی در این زمینه

حداقل دارای سه لایه است؛ اولین لایه، لایه ورودی نامیده می‌شود که داده ورودی مدل، ورودی این لایه است. آخرین لایه، لایه خروجی نامیده می‌شود که خروجی مدل، خروجی این لایه است. هر لایه اطلاعات را از لایه قبل از خود دریافت می‌کند و بعد از انجام عملیات بر روی ورودی، داده را به لایه بعد از خود تحویل می‌دهد. در هر لایه نیز تعداد سلول عصبی<sup>۱</sup> وجود دارد. به همین دلیل مدل‌های یادگیری عمیق با پیاده‌سازی یک شبکه از این گره‌ها که به آن شبکه عصبی می‌گویند به وجود می‌آیند. این ساختار لایه‌ای در شکل ۱-۲ اقتباس شده از [۱۰] نمایش داده شده است.



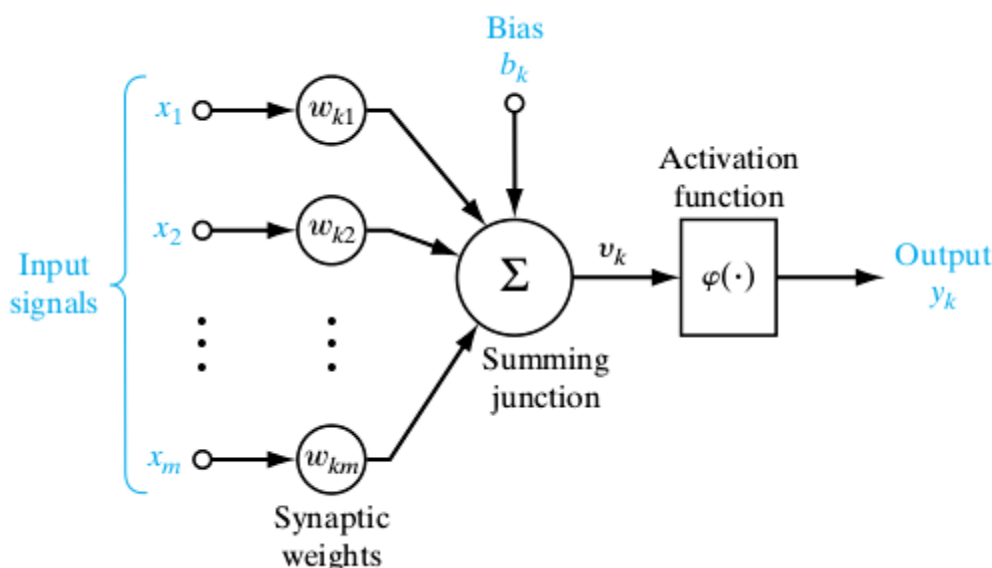
شکل ۱-۲ نمونه یک شبکه عصبی [۱۰]

هر سلول عصبی در این ساختار، ورودی خود را از سلول‌های لایه قبل که به آنها متصل است، می‌گیرد و تابع فعال‌سازی<sup>۲</sup> را بر روی جمع وزن‌دار ورودی‌ها اعمال می‌کند. نتیجه این محاسبات به عنوان خروجی به سلول‌های متصل لایه بعد فرستاده می‌شود. یال متصل‌کننده دو سلول دارای وزن است که در محاسبات سلول‌های عصبی مورد استفاده قرار می‌گیرد. وزن یال‌ها بر اساس عملکرد یا دقت شبکه با توجه به داده‌های دیده شده به‌روزرسانی می‌شود. اگر دقت شبکه بالا باشد، وزن‌ها تغییر نمی‌کنند و اگر دقت پایین باشد، وزن‌ها

<sup>1</sup> Neuron

<sup>2</sup> Activation Function

برای بهبود عملکرد به روز می شوند. وزن ها از طریق گرادیان نزولی<sup>۱</sup> و پسانتشار<sup>۲</sup> به دست می آیند. محاسبات یک سلول به صورت کلی در شکل ۲-۲ اقتباس شده از [۱۱] نمایش داده شده است. در ادامه به نحوه اتصال شبکه های عصبی برای ساخت مدل های یادگیری عمیق خواهیم پرداخت.



شکل ۲-۲ نمونه عملیات در یک نورون شبکه عصبی [۱۱]

۱.۲.۲.۲. نحوه اتصال لایه ها در شبکه های عصبی برای ساخت مدل های یادگیری

### عمیق

برای ساخت مدل یادگیری عمیق، چندین لایه از شبکه عصبی به شکل های جریان داده به طرف جلو<sup>۳</sup> یا جریان داده با بازخورد<sup>۴</sup> به یکدیگر متصل می شوند. در حالت جریان داده به طرف جلو، اتصالات دور تشکیل نمی دهند و داده ورودی در یک جهت مستقیم به طرف خروجی جریان می یابند. این نوع ساختار به طور گسترده برای تشخیص الگو<sup>۵</sup> به کار می رود.

<sup>1</sup> Gradient Descent

<sup>2</sup> Back Propagation

<sup>3</sup> Feed Forward

<sup>4</sup> Feedback

<sup>5</sup> Pattern Recognition

## ۲.۲.۲.۲. لایه پنهان

این لایه به دنبال ترکیب ویژگی‌های ورودی و استخراج ویژگی‌های انتزاعی جدید از آن است.

## ۳.۲.۲.۲. لایه خروجی

وظیفه این لایه تعیین خروجی بر اساس مجموع ویژگی‌های یافته‌شده در لایه پنهان شبکه است. تعداد نورون‌های این لایه با توجه به نوع خروجی مورد نیاز از شبکه عصبی عمیق تعیین می‌شود. برای مثال در صورت نیاز به خروجی گسسته‌ای مثل انتخاب بین خرید، فروش و نگهداری، نیاز به سه نورون خروجی برای هر یک از این انتخاب‌ها است.

## ۳.۲.۲. یادگیری تقویتی

یادگیری تقویتی یکی از گرایش‌های یادگیری ماشینی است که از روانشناسی رفتارگرایی الهام می‌گیرد. این روش بر رفتارهایی تمرکز دارد که ماشین باید برای پیشینه کردن پاداشش انجام دهد. تفاوت اصلی بین روش‌های سنتی و الگوریتم‌های یادگیری تقویتی این است که در یادگیری تقویتی نیازی به داشتن اطلاعات راجع به فرایند تصمیم‌گیری ندارد. یادگیری تقویتی با یادگیری با نظارت معمول دو تفاوت عمده دارد، نخست اینکه در آن زوج‌های صحیح ورودی و خروجی در کار نیست و رفتارهای ناکارآمد نیز از بیرون اصلاح نمی‌شوند، و دیگر آنکه تمرکز زیادی روی اجرای زنده وجود دارد که نیازمند پیدا کردن یک تعادل مناسب بین اکتشاف چیزهای جدید و بهره‌برداری از دانش کسب شده دارد.

نکته مهمی که در اینجا وجود دارد این است که یادگیری تقویتی برای مسائلی که در آن‌ها بیشترین سود در کوتاه مدت تضمین‌کننده بیشترین سود در دراز مدت نیست بسیار مناسب است. یادگیری تقویتی شامل دو دسته الگوریتم‌های مبتنی بر سیاست و ارزش است.

## ۴.۲.۲. یادگیری تقویتی عمیق

یادگیری تقویتی عمیق زیرمجموعه‌ای از یادگیری ماشین است که یادگیری تقویتی و یادگیری عمیق را ترکیب می‌کند.

### ۱.۴.۲.۲. الگوریتم یادگیری تقویتی Q-learning

Q-learning معروف ترین الگوریتم یادگیری تقویتی است که یک نوع الگوریتم یادگیری تقویتی مبتنی بر ارزش است. هدف این الگوریتم یادگیری تابعی است که ارزش یک اقدام در یک وضعیت خاص را مشخص می‌کند. یکی از نقاط قوت این روش، توانایی یادگیری تابع مذکور بدون داشتن مدل معینی از محیط است.

در اینجا مدل مسئله تشکیل شده از یک عامل، وضعیت‌ها  $S$  و مجموعه از اقدامات  $A$  برای هر وضعیت. با انجام یک اقدام  $a \in A$ ، عامل از یک وضعیت به وضعیت بعدی حرکت کرده و هر وضعیت پاداشی به عامل می‌دهد. هدف عامل حداکثر کردن پاداش دریافتی کل خود است. این کار با یادگیری اقدام بهینه برای هر وضعیت انجام می‌گردد. الگوریتم دارای تابعی است که ترکیب وضعیت/اقدام را محاسبه می‌نماید:

$$Q : S \times A \rightarrow \mathbb{R}$$

قبل از شروع یادگیری،  $Q$  مقدار ثابتی را که توسط طراح انتخاب شده بر می‌گرداند. سپس هر بار که به عامل پاداش داده می‌شود، مقادیر جدیدی برای هر ترکیب وضعیت/اقدام محاسبه می‌گردد. هسته الگوریتم از یک به‌روزرسانی تکراری ساده تشکیل شده است. به این ترتیب که بر اساس اطلاعات جدید مقادیر قبلی اصلاح می‌شود.

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \left[ \underbrace{R(s_t)}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})}_{\text{max future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right]$$

که  $R(s_t)$  پاداش  $s_t$  و  $\alpha_t(s, a)$  است. نرخ یادگیری  $(0 < \alpha < 1)$  ممکن است برای همه زوج‌ها یکسان باشد. مقدار عامل تخفیف  $\gamma$  نیز به این شکل است:

$$(0 < \gamma < 1)$$

یک اپیزود الگوریتم وقتی  $S_{t+1}$  به وضعیت نهایی می‌رسد پایان می‌یابد. توجه کنید که برای همه وضعیت‌های نهایی  $S_f$  و  $Q(S_f, a)$  مربوطه هیچگاه به‌روز نمی‌شود و مقدار اولیه خود را حفظ می‌کند. Q-learning می‌تواند با تابع تقریب‌زننده ترکیب شود. این باعث می‌شود که الگوریتم برای مشکلات بزرگ‌تر اعمال شود، حتی زمانی که فضای حالت پیوسته است. Q-learning عمیق عبارت است از استفاده از یک شبکه عصبی مصنوعی عمیق به عنوان تابع تقریب‌زننده.

در این تکنیک از تکرار تجربه استفاده می‌شود، مکانیسم الهام گرفته شده از بیولوژی که از نمونه تصادفی اقدامات قبلی به جای جدیدترین اقدام برای آموزش استفاده می‌کند. این مکانیسم نیاز به پیوستگی در دنباله مشاهدات را حذف می‌کند و تغییرات را در توزیع داده‌ها هموار می‌کند.

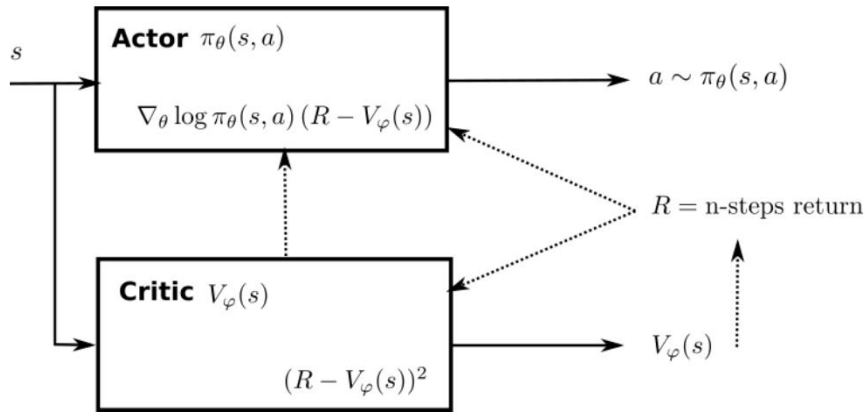
## ۲.۴.۲.۲ الگوریتم یادگیری تقویتی Advantage Actor-Critic

در زمینه یادگیری تقویتی، الگوریتم Advantage Actor-Critic (A2C) دو نوع الگوریتم یادگیری تقویتی مبتنی بر سیاست و ارزش را با هم ترکیب می‌کند. عوامل مبتنی بر سیاست (توزیع احتمالی اقدامات) مستقیماً یک سیاست را یاد می‌گیرند که هر یک از انواع حالات ورودی را به یک اقدام خروجی مرتبط می‌کند. الگوریتم‌های مبتنی بر ارزش یاد می‌گیرند که اقدامات را بر اساس مقدار پیش‌بینی شده ارزش محیط نهایی بر اساس هر اقدام عامل انتخاب کنند.

الگوریتم Advantage Actor-Critic شامل دو شبکه بازیگر و منتقد است که برای حل یک مشکل خاص با هم کار می‌کنند. در سطح بالا، تابع Advantage خطای پیش‌بینی را محاسبه می‌کند. شبکه بازیگر در هر مرحله یک عمل یا اقدام را انتخاب می‌کند و شبکه منتقد کیفیت یا مقدار  $Q$  حالت ورودی را محاسبه می‌کند. همانطور که شبکه منتقد می‌فهمد کدام حالت‌ها بهتر یا بدتر هستند، بازیگر از این اطلاعات برای یادگیری عامل به دنبال وضعیت‌های خوب و اجتناب از حالات بد استفاده می‌کند.

معماری الگوریتم Advantage Actor-Critic را در شکل ۲-۳ اقتباس شده از [۱۲] می‌بینید.





شکل ۲-۳ معماری Advantage Actor-Critic [۱۲]

که در آن شبکه بازیگر سیاست  $\pi_\theta$  را برای حالت  $s$  خروجی می‌دهد، یعنی بردار احتمالات برای هر اقدام و شبکه منتقد ارزش  $V_\phi(s)$  را برای یک حالت  $s$  خروجی می‌دهد.

## ۳.۴.۲.۲ الگوریتم یادگیری تقویتی Proximal Policy Optimization

در یادگیری تحت نظارت، به راحتی با تنظیم هایپرپارامترهای کمی به راحتی می‌توان تابع هزینه‌ای پیاده‌سازی کرد و آن را با استفاده از گرادینان نزولی بهینه کرد. اما همین فرایند در یادگیری تقویتی بسیار دشوارتر می‌باشد چون قسمت‌های متغیر بیشتری وجود دارد. الگوریتم Proximal Policy Optimization (PPO) بین سهولت پیاده‌سازی، تعداد نمونه ورودی مورد نیاز و سهولت تنظیم هایپرپارامترها تعادل ایجاد می‌کند و سعی می‌کند در هر گام یک به روزرسانی را که تابع هزینه را به حداقل برساند، محاسبه کند. ایده اصلی این است که پس از به روزرسانی، سیاست جدید نباید از سیاست قبلی خیلی دور باشد. برای این منظور، PPO از clipping برای جلوگیری از به روزرسانی بیش از حد استفاده می‌کند.

الگوریتم PPO از زیر مجموعه الگوریتم‌های یادگیری تقویتی Actor-Critic است که برای بروزرسانی سیاست الگوریتم زیر را با استفاده از تابع clipping زیر پیشنهاد کرده است.

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

$\theta$  پارامتر سیاست است.

$\hat{E}_t$  نشان دهنده ارزش تجربی به دست آمده طی گام‌ها است.

$r_t$  نسبت احتمال بر حسب سیاست جدید به قدیم.  $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$  و در نتیجه

$$r(\theta_{old}) = 1$$

Advantage  $\hat{A}_t$  تخمین زده شده در زمان  $t$  است.

$\epsilon$  نیز هایپرپارامتری برای هدف تعیین میزان قابل قبول تغییر سیاست نسبت به حالت قبلی است که

معمولا عدد ۰٫۱ یا ۰٫۲ است. [۱۳]

الگوریتم کلی PPO عبارت است از :

---

**Algorithm 1** PPO, Actor-Critic Style

---

```

for iteration=1, 2, ... do
  for actor=1, 2, ...,  $N$  do
    Run policy  $\pi_{\theta_{old}}$  in environment for  $T$  timesteps
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
  end for
  Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
   $\theta_{old} \leftarrow \theta$ 
end for

```

---

الگوریتم PPO [۱۳]

## ۳.۲. خلاصه و جمع بندی

در این فصل با مفاهیم اولیه و پیش زمینه‌هایی که جهت درک هرچه بهتر بازارهای مالی و یادگیری تقویتی عمیق آشنا شدیم. این مفاهیم به درک بهتر مسئله کمک می‌کنند و ما را برای ارائه‌ی مدل پیشنهادی در فصل بعد کمک خواهند کرد.

### ۳. فصل سوم: روش تحقیق

### ۱.۳. مقدمه

در این فصل نخست روش پیشنهادی خود برای مدیریت معاملات در بازارهای مالی با کمک سیگنال‌های ورودی خرید و فروش توسط عاملی هوشمند را معرفی می‌کنیم. در ادامه به بررسی ابزارهای متفاوتی که در مراحل پیاده‌سازی، آزمون و اندازه‌گیری معیارها استفاده شده است، خواهیم پرداخت. همچنین برای مقایسه بین روش‌های متفاوت و تشخیص موفقیت در حل مسئله، نیاز است معیارهایی ارائه شوند که این معیارها نیز در این فصل بررسی می‌شوند. این فصل مختص بیان روش پیشنهادی و نحوه جمع‌آوری مجموعه داده‌ها است و نحوه دقیق‌تر پیاده‌سازی و نتایج به‌دست آمده را در فصل بعد توضیح خواهیم داد.

### ۲.۳. روش پیشنهادی

همان‌طور که اشاره شد، روش پیشنهادی، یادگیری تقویتی عمیق است. به همین منظور، روش پیشنهادی چهار فاز پیاده‌سازی خواهد داشت. در فاز اول یک برنامه برای جمع‌آوری داده‌های بازار رمزارزها پیاده‌سازی خواهد شد. در گام بعدی، سیگنال‌های ورودی، که شامل سه سیگنال کوتاه، میان و بلند مدت برای خرید و فروش هستند، همراه با مقداری نويز پیاده‌سازی خواهد شد. در گام سوم، محیطی برای آموزش و تست عامل یادگیری تقویتی پیاده‌سازی خواهد شد. در گام آخر، مدل یادگیری تقویتی پیاده‌سازی شده و در محیط پیاده‌سازی شده در گام قبل، آموزش خواهد دید.

### ۱.۲.۳. جمع‌آوری داده‌ها

برای آموزش و آزمون مدل نهایی دو رمزارز بیت‌کوین و اتریوم<sup>۱</sup> انتخاب شدند که داده‌ها برای هر کدام داده‌های مربوط به قیمت تبادل رمزارز مورد نظر با رمزارز تتر<sup>۲</sup> است که قیمت آن معادل یک دلار است. به

<sup>۱</sup> Ethereum

<sup>۲</sup> Tether

جای روش مرسوم که داده را به دو قسمت برای آموزش و تست تقسیم کنیم، فرایند آموزش و تست را بر روی داده‌های رمز ارزهای متفاوتی انجام دادیم. آموزش مدل بر روی رمز ارز اتریوم انجام می‌شود و تست نهایی بر روی داده‌های موجود از رمز ارز بیت کوین است.

برای داده‌ها از داده‌های ابتدای سال ۲۰۱۸ تا انتهای ماه پنجم سال ۲۰۲۱ استفاده شده است. این داده‌های با دقت زمانی ۵ دقیقه‌ای ذخیره شده اند که در مجموع شامل ۴۱ ماه داده قیمت تبادل برای دو ارز می‌شود.

### ۲.۲.۳. تولید سیگنال ورودی

برای تولید سیگنال‌های ورودی از دو روش استفاده کردیم. یک روش این که ابتدا به قیمت‌ها مقداری نویز اضافه کردیم و بعد با استفاده از داده‌های قیمت آینده نویزی در صورت افزایش و کاهش قیمت در بازه زمانی‌های کوتاه، میان و بلند مدت، سیگنال‌های خرید و فروش تولید کردیم.

روش دوم مورد استفاده، تولید سیگنال مشابه روش قبل بود با این تفاوت که برای ایجاد خطا در سیگنال ورودی، در صدی از سیگنال‌های خرید را به عنوان سیگنال فروش و در صدی از سیگنال‌های فروش را به عنوان سیگنال خرید قرار دادیم.

در انتها، در هر دو روش مورد استفاده، در صد خطای ایجاد شده در سیگنال‌ها حدود ۳۵ در صد بود تا دقت سیگنال‌ها تا حدی مشابه پیش‌بینی‌های شبکه‌های عصبی عمیق بر روی بازارهای مشابه باشد. در جدول (۱-۳) اسامی انواع نویز استفاده شده مطابق با نام استفاده شده در نتایج آورده شده است.

جدول (۱-۳) انواع نویز سیگنال‌های ورودی

نام معادل در نتایج	نوع نویز
Noise Type1	نویز تولید شده توسط روش اول
Noise Type2	نویز تولید شده توسط روش دوم

### ۳.۲.۳. محیط یادگیری تقویتی

محیط<sup>۱</sup> یادگیری تقویتی فضایی است که عامل هوشمند در آن زندگی می‌کند و امکان تعامل با آن را دارد. برای پیاده سازی این محیط از رابط استاندارد محیط‌های OpenAI Gym استفاده کردیم تا این محیط به راحتی بتواند با مدل‌های معروف یادگیری تقویتی تعامل داشته باشد.

مهم‌ترین خصوصیات محیط پیاده سازی شده عبارتند از کیف پول، صرافی و عوامل اصلی محیط مثل اقدام‌های ممکن و سیستم‌های پاداش موجود.

کیف پول کاربر که در آن مقدار موجودی رمزارزهای او مشخص است و کنترل می‌شود که مقادیر غیرقابل قبول پیدا نکند. مازول صرافی مسئول اعمال اقدام‌های عامل و محاسبه قیمت معامله و کارمزد تراکنش است که برای کارمزد هر معامله مقدار ۰,۲ درصد ارزش هر معامله را در نظر گرفتیم که مطابق کارمزد صرافی‌های مطرح در دنیا است.

برای اقدام‌های ممکن در محیط ۴ نوع اقدام پیاده سازی شد؛ نوع اول خرید و فروش و نگهداری به طور کامل. نوع دوم خرید و فروش و نگهداری به اندازه مقادیر گسسته پیش فرض ۰,۱، ۰,۲، ۰,۳، ... و ۱,۰ از کل سرمایه. نوع سوم خرید و فروش و نگهداری به هر مقدار پیوسته دلخواه. نوع آخر نیز مشابه نوع دوم خرید و فروش و نگهداری به اندازه مقادیر گسسته پیش فرض ۰,۱، ۰,۲، ۰,۳، ... و ۱,۰ از کل سرمایه همراه با تراکنش‌های کمکی Stop-Loss و Take-Profit مکمل آن‌ها. تعداد اقدام ممکن در محیط تعداد نوروں‌های خروجی مدل را مشخص می‌کند. در جدول (۳-۲) سامی استفاده شده در بخش نتایج برای هر یک از انواع اقدام‌های ممکن در محیط آورده شده است.

جدول (۳-۲) انواع اقدام‌های ممکن

نوع اقدام ممکن	نام معادل در نتایج
اقدام نوع اول	Total buy, sell and hold
اقدام نوع دوم	Buy, sell and hold in discrete amounts
اقدام نوع سوم	Buy and sell in continuous amounts
اقدام نوع چهارم	Risk Managed Buy and sell

<sup>1</sup> Environment

پنجره مورد استفاده برای بازه محاسبات پاداش‌ها و مشاهدات در محیط را مقدار ۱۵ قرار دادیم که معادل ۷۵ دقیقه است.

در نهایت برای تابع پاداش و جریمه عامل هوشمند نیز ۵ نوع تابع پاداش پیاده‌سازی کردیم؛ نوع اول پاداش و جریمه بر اساس میزان سود و ضرر در تعداد گام مشخص شده در گذشته است. نوع دوم بر اساس میزان نسبت شارپ و سورتینو سود و ضرر در تعداد گام مشخص شده در گذشته است. نوع سوم نیز بر اساس میزان سود و ضرر در تعداد گام مشخص شده در گذشته منهای مقدار تغییر قیمت رمزارز نگهداری شده در کل طول این بازه است. نوع چهارم مشابه تابع پاداش نوع اول است ولی به همراه جریمه در صورت عدم معامله در طول بازه مشخص شده. نوع آخر نیز مشابه تابع پاداش نوع اول است ولی به همراه پاداشی اضافه برای هر جفت معامله خرید و فروش پس از فروش. در جدول (۳-۳) اسامی استفاده شده در بخش نتایج برای هر یک از توابع پاداش آورده شده است.

جدول (۳-۳) انواع توابع پاداش

نام معادل در نتایج	تابع پاداش
Profit of observation window	تابع پاداش نوع اول
Risk Adjusted Profit of observation window	تابع پاداش نوع دوم
Profit of observation window minus the profit of Buy and Hold in the same window	تابع پاداش نوع سوم
Profit of observation window with cash penalty	تابع پاداش نوع چهارم
Profit of observation window with Trade profit	تابع پاداش نوع پنجم

### ۴.۲.۳. مدل یادگیری تقویتی

مدل پیشنهادی ما برای یادگیری تقویتی عمیق مدل PPO است که برای این مدل شبکه عصبی تمام متصل با ۶۴ نورون در دو لایه پنهان برای هر دو شبکه بازیگر و منتقد و برای تابع فعال سازی از Tanh استفاده می‌کنیم.

برای مقایسه نتایج مدل پیشنهادی برای خروجی های گسسته نتایج را با نتایج مدل DQN و برای خروجی های پیوسته نتایج را با مدل TD3 نیز مقایسه می‌کنیم.

### ۱.۴.۲.۳. لایه ورودی

دو مقدار تعیین کننده تعداد لایه ورودی است. یکی تعداد سطر داده از گذشته که مدل به آن‌ها در هر لحظه دسترسی دارد که این مقدار معادل متغیر پنجره در محیط است. دومین مقدار تعداد سیگنال‌های ورودی مدل که ما سه سیگنال کوتاه، میان و بلند مدت برای خرید و فروش ایجاد کردیم که در مجموع می‌شود ۶ سیگنال ورودی.

ضرب مقدار اول، متغیر پنجره، در مقدار دوم، تعداد سیگنال‌ها، تعداد نوروں‌های ورودی مدل را مشخص می‌کند که معادل ۹۰ نوروں در لایه ورودی است.

### ۲.۴.۲.۳. لایه پنهان

برای لایه‌های پنهان میانی نیز ما مقدار ۶۴ نوروں در دو لایه را در نظر گرفتیم. این دو لایه به صورت تمام متصل هستند.

این لایه پنهان برای دو شبکه بازیگر و منتقد مشترک نیست و هر کدام از آن‌ها از همین معماری دو لایه با ۶۴ نوروں استفاده می‌کنند.

### ۳.۴.۲.۳. لایه خروجی

تعداد نوروں لایه خروجی معادل تعداد اقدام ممکن در محیط است.

در حالتی که سه اقدام خرید، فروش و نگهداری وجود دارد، سه نوروں خروجی لازم هست. در حالتی که خرید و فروش و نگهداری به اندازه مقادیر گسسته پیش فرض ۱، ۲، ۳، ... و ۱۰ از کل سرمایه باشد، ۲۱ نوروں در لایه خروجی موجود است و در حالتی که مقدار خرید به صورت گسسته باشد، یک نوروں با مقدار ۱- تا ۱+ در لایه خروجی موجود است.



### ۳.۳. ابزارهای مورد استفاده

در طی پیاده‌سازی این پروژه نیاز به استفاده از ابزارها و کتابخانه‌های مختلفی بود. برای دریافت داده‌ها از رابط برنامه‌نویسی سایت FinnHub، برای تولید سیگنال‌ها از کتابخانه Pandas پایتون، برای محیط یادگیری تقویتی از کتابخانه متن-باز TensorTrade<sup>۱</sup> استفاده شد و در نهایت برای مدل یادگیری تقویتی از کتابخانه متن باز Stable-Baselines3. بقیه کتابخانه‌های مورد استفاده عبارتند از کتابخانه Numpy و فریم‌ورک PyTorch شرکت فیسبوک برای پیاده‌سازی‌های Stable-Baselines3.

#### ۳.۳.۱. TensorTrade

این کتابخانه متن-باز برای محیط آموزش مدل یادگیری تقویتی عمیق مورد استفاده قرار گرفت. محیط پیاده‌سازی شده در این کتابخانه مطابق با اینترفیس محیط‌های OpenAI Gym است. محیط این کتابخانه مخصوص آموزش مدل‌های یادگیری تقویتی مختص بازارهای مالی است. به همین دلیل محیط آن شامل پیاده‌سازی مدل کیف پول، صرافی و سفارش نیز می‌شود. در این کتابخانه متن-باز تغییراتی اعمال شد که در فصل بعد به این تغییرات می‌پردازیم.

#### ۳.۳.۲. Stable-Baselines3

این کتابخانه متن-باز انواع پیاده‌سازی‌های الگوریتم‌های یادگیری تقویتی عمیق را بر اساس مقالات روز بر روی فریم‌ورک PyTorch دارد. تمامی این الگوریتم‌ها بر روی محیط‌های مطابق با اینترفیس محیط‌های OpenAI Gym قابل اجرا هستند. برای پیاده‌سازی مدل یادگیری تقویتی عمیق استفاده شده در این پژوهش، از این کتابخانه استفاده شده است که امکان شخصی‌سازی قسمت‌های مختلف مثل تعداد لایه شبکه عصبی یا تعداد نوروها یا تابع‌های فعال‌سازی را به کاربر می‌دهد.

### ۳.۳.۳. Finnhub

این سایت رابط برنامه‌نویسی کاربردی برای دریافت داده‌های مربوط به انواع بازارهای مالی از جمله فارکس و رمزارز را در اختیار برنامه‌نویسان قرار می‌دهد. برای دریافت داده‌ها از این رابط برنامه‌نویسی کاربردی استفاده شد.

### ۳.۴. معیار ارزیابی<sup>۱</sup>

برای ارزیابی این پروژه به علت این که یک پروژه یادگیری با نظارت<sup>۲</sup> نیست و ما صرفاً نمی‌توانیم دقت مدل نهایی را بررسی کنیم، مجبور هستیم که مدل را بر روی داده‌ها اعمال کنیم و بر روی مقدار سود کسب‌شده توسط مدل در طول زمان تحلیل انجام دهیم. در انتها برای مقایسه نتیجه مدل نهایی، مقادیر معیارهای ارزیابی مدل را با بازده رمزارز در بازه زمانی مشابه و استراتژی ساده‌ای که به صورت دستی طراحی شده است مقایسه می‌کنیم.

برای این کار از چند معیار استفاده می‌کنیم:

معیار اول سود تعدیل‌شده با ریسک توسط نرخ Sharpe است.

نرخ Sharpe سود به‌دست‌آمده را گرفته و نرخ سود بدون ریسک را از آن کم می‌کند و سپس آن مقدار را بر انحراف معیار بازده دارایی تقسیم می‌کند.

$$\text{Sharpe Ratio} = \frac{R_p - r_f}{\sigma_p}$$

$R_p$  سود به‌دست‌آمده

$r_f$  سود بدون ریسک

$\sigma_p$  انحراف معیار بازده دارایی

معیار دوم تعداد معاملات و میانگین زمان بین معاملات است.

<sup>1</sup> Evaluation metric

<sup>2</sup> Supervised Learning

معیار سوم حداکثر میزان ضرر در طول زمان<sup>۱</sup> است.  
معیار چهارم حداکثر زمانی که مدل در حال ضرر<sup>۲</sup> کردن بوده است.  
معیار پنجم میزان سود کلی و میزان سود در واحد سال است.

### ۳.۵. خلاصه و جمع‌بندی

فصل سوم به طور عمده در برگیرنده‌ی ساختار مدل پیشنهادی برای عامل هوشمند و جزئیات مربوط به تابع پاداش و انواع اقدام پیشنهادی می‌باشد. در این فصل ساختار مدل استفاده شده و معیارهای ارزیابی معرفی شده مورد بررسی قرار گرفته‌اند. همچنین در این فصل ابزارها و کتابخانه‌های استفاده شده معرفی شدند.

---

<sup>1</sup> Maximum Drawdown

<sup>2</sup> Maximum Drawdown Time

## ۴. فصل چهارم: پیاده سازی

## ۱.۴ . مقدمه

در این فصل ابتدا به نحوه جمع‌آوری داده‌ها و تولید سیگنال‌های ورودی می‌پردازیم و سپس نحوه پیاده‌سازی مدل یادگیری تقویتی و محیط یادگیری تقویتی را توضیح می‌دهیم. در انتها نیز نتایج پیاده‌سازی را مشاهده می‌کنیم.

## ۲.۴ . پیاده‌سازی

### ۱.۲.۴ . جمع‌آوری داده‌ها

برای جمع‌آوری داده‌ها از سایت [finnhub.io](https://finnhub.io) استفاده کردیم. این سایت رابط برنامه‌نویسی کاربردی برای دریافت داده‌های گذشته و در لحظه انواع بازارهای مالی را در اختیار برنامه‌نویس قرار می‌دهد. دو مورد از بازارهای مالی مهمی که توسط این سایت پشتیبانی می‌شوند عبارتند از بازار مالی فارکس و بازار مالی رمزارزها.

```
def crypto_candles(self, symbol, resolution, _from, to, _format='json'):
    return self._get("crypto/candle", params={
        "symbol": symbol,
        "resolution": resolution,
        "from": _from,
        "to": to,
        "format": _format
    })
```

به علت این که رابط برنامه‌نویس این سایت فقط در هر بار درخواست تعداد محدودی داده را به ما باز می‌گرداند، ما نمی‌توانیم با یک درخواست، داده‌ها را از هر تاریخی تا هر تاریخی دریافت کنیم. همچنین به علت وجود محدودکننده نرخ درخواست‌ها در سایت ما نمی‌توانیم به طور مکرر درخواست بفرستیم. برای رسیدگی به موارد ذکر شده از قطعه کد زیر استفاده کردیم.

```

class API():
    def __init__(self):
        self.finnhub_client = Client(api_key=API_KEY)
        self.utility = utility()

    def get_candles(self, ticker_name, timeframe, from_time, to_time):
        constant_time = 500
        time.sleep(5)
        diff_time = to_time - from_time
        time_list = []
        if (diff_time > (self.utility.timeframe_to_timestamp() * constant_time)):
            new_to_time = from_time + self.utility.timeframe_to_timestamp() * constant_time
            candles = self.finnhub_client.crypto_candles(ticker_name, timeframe, from_time, new_to_time)
            diff_time = to_time - new_to_time
            while (diff_time > (self.utility.timeframe_to_timestamp() * constant_time)):
                time.sleep(1)
                new_from_time = new_to_time
                new_to_time = new_from_time + self.utility.timeframe_to_timestamp() * constant_time
                candles_temp = self.finnhub_client.crypto_candles(ticker_name, timeframe, new_from_time, new_to_time)
                for key, value in candles_temp.items():
                    if key == "s":
                        continue
                    candles[key].extend(value)
                diff_time = to_time - new_to_time
            candles_temp = self.finnhub_client.crypto_candles(ticker_name, timeframe, new_to_time, to_time)
            for key, value in candles_temp.items():
                if key == "s":
                    continue
                candles[key].extend(value)
        else:
            candles = self.finnhub_client.crypto_candles(ticker_name, timeframe, from_time, to_time)
        return candles

```

در انتها نیز با کد زیر داده‌ها را با فرمت CSV به شکل زیر ذخیره کردیم.

```

def _ohlcv_save_to_csv(self, ticker, ticker_name):
    timeframe = TIMEFRAME
    ticker.sort_index(inplace=True)
    ticker['date'] = ticker['date'].apply(lambda x: datetime.datetime.utcfromtimestamp(x).strftime("%Y/%m/%d %H:%M:%S"))
    ticker.insert(0, "time", ticker['date'])
    ticker = ticker.drop('date', axis=1)
    ticker.to_csv(DATA+ticker_name+"_"+timeframe+'.csv', index = False)

```

## ۲.۲.۴. تولید سیگنال‌های ورودی بر روی داده‌ها

برای تولید سیگنال‌ها از دو روش استفاده کردیم.

روش اول، اضافه کردن مقداری نویز به قیمت‌ها و بعد محاسبه سیگنال‌ها از روی داده‌های نویزی.

```

df = df.assign(close_noise=0)
noise = np.random.normal(0, df["close"].std(), len(df)) * 0.015
df['close_noise'] = df['close'] + noise

```

سپس مقادیر سیگنال کوتاه مدت را به روش زیر محاسبه می کنیم.

```
df = df.assign(diff_consecutive=df['close_noise'].shift(-1) - df['close_noise'])
max_consecutive_diff = df['diff_consecutive'].max()
min_consecutive_diff = df['diff_consecutive'].min()
df = df.drop(['diff_consecutive'], axis=1)
df = df.assign(probability_short=0)
df.loc[df['prediction_short']==2, 'probability_short'] = (df['close_noise'].shift(-1) - df['close_noise'])/max_consecutive_diff
df.loc[df['prediction_short']==1, 'probability_short'] = (df['close_noise'].shift(-1) - df['close_noise'])/min_consecutive_diff
df = df.assign(buy_probability_short=0)
df['buy_probability_short'] += df['probability_short']*(df['prediction_short']-1)
df = df.assign(sell_probability_short=0)
df['sell_probability_short'] += (-1)*df['probability_short']*(df['prediction_short']-2)
```

برای محاسبه مقادیر سیگنال های میان و بلندمدت نیز به روش مشابه عمل کردیم. با این تفاوت که به جای شیفت داده ها فقط به اندازه یک ردیف برای میان مدت ۱۰، ۱۱، ۱۲، ۱۳، ۱۴ ردیف شیفت می دهیم و برای بلند مدت ۴۳، ۴۴، ۴۵، ۴۶، ۴۷، ۴۸، ۴۹، ۵۰، ۵۱، ۵۲، ۵۳ ردیف شیفت می دهیم و با محاسبه حداکثر و حداقل مقدار قیمت در این بازه های شیفت داده شده تصمیم به تولید سیگنال خرید یا فروش میان و بلندمدت می گیریم.

روش دوم، بعد از مراحل قبل درصدی از سیگنال های خرید و فروش را با یکدیگر جابجا می کنیم تا درصدی مشخص خطا به سیگنال ها اضافه می کنیم.

#### ۳.۲.۴. تغییرات انجام شده بر محیط یادگیری تقویتی در کتابخانه

### TensorTrade

برای پیاده سازی محیط یادگیری تقویتی از کتابخانه TensorTrade استفاده کردیم. ولی برای به دست آوردن نتایج بهتر هنگام آموزش مدل یادگیری تقویتی در این محیط، نیاز به پیاده سازی تعدادی امکانات جدید داریم. در ادامه این ویژگی های اضافه شده را بررسی می کنیم.

برای درک بهتر کارکرد کتابخانه، کد مربوط به هر گام محیط را در تصویر زیر مشاهده می کنید.

```
def step(self, action: Any) -> 'Tuple[np.array, float, bool, dict]':
    """Makes on step through the environment.

    Parameters
    -----
    action : Any
        An action to perform on the environment.

    Returns
    -----
    `np.array`
        The observation of the environment after the action being
        performed.
    float
        The computed reward for performing the action.
    bool
        Whether or not the episode is complete.
    dict
        The information gathered after completing the step.
    """
    self.action_scheme.perform(self, action)

    obs = self.observer.observe(self)
    reward = self.reward_scheme.reward(self)
    done = self.stopper.stop(self)
    info = self.informer.info(self)

    self.clock.increment()

    return obs, reward, done, info
```



و نحوه گرفتن یک نمونه از محیط در کد را در زیر مشاهده می کنید.

```
def environment(env_type, data_file_name, action_scheme, reward_scheme, window_size):
    df = pd.read_csv('data/'+data_file_name+'.csv')
    df.rename(columns = {'time':'date'}, inplace = True)
    binance = Exchange("binance", service=execute_order)
    (Stream.source(list(df['close']), dtype="float").rename("USDT-ETH"))
    price_history = df[['date', 'open', 'high', 'low', 'close', 'volume']]
    renderer_feed = DataFeed([
        Stream.source(price_history[c].tolist(), dtype="float").rename(c) for c in price_history]
    )
    df = df.drop(df.columns[0], axis=1)
    dataset = df.drop(columns=
        ['date', 'open', 'high', 'low', 'close', 'volume', 'close_noise']
        , inplace=False)
    with Namespace("binance"):
        binance_streams = [
            Stream.source(list(dataset[c]), dtype="float").rename(c) for c in dataset.columns
        ]
    feed = DataFeed(binance_streams)
    ETH = Instrument('ETH', 8, 'Ethereum')
    USDT = Instrument('USDT', 8, 'Thether')
    portfolio = Portfolio(USDT, [
        Wallet(binance, 10000 * USDT),
        Wallet(binance, 0 * ETH),
    ])
    if env_type == 'train':
        random_start = True
    elif env_type == 'eval':
        random_start = False
    chart_renderer = MatplotlibTradingChart(
        display=False,
        save_format="jpeg", # save the chart to a JPEG file
    )
    env = default.create(
        portfolio=portfolio,
        action_scheme=action_scheme,
        reward_scheme=reward_scheme,
        feed=feed,
        window_size=window_size,
        renderer_feed=renderer_feed,
        renderer=chart_renderer,
        enable_logger=False,
        random_start=random_start
    )
    return env
```

#### ۱۰.۳.۲.۴ پیاده سازی شروع از زمان تصادفی به هنگام ریست شدن محیط

برای جلوگیری از overfit شدن مدل بر روی داده های ابتدای بازه زمانی، نیاز بود تا این ویژگی را پیاده سازی کنیم تا در هر بار ریست شدن محیط، داده های محیط از ابتدا شروع نشوند.

برای این قسمت نیاز به تغییر کد مربوط به ریست شدن در سه کلاس مختلف بود که در ادامه هر سه آنها آورده شده‌اند.

### ریست شدن در کلاس `Environment`:

```
def reset(self) -> 'np.array':
    """Resets the environment.

    Returns
    -----
    obs : `np.array`
        The first observation of the environment.
    """
    if self._random_start:
        size = len(self.observer.feed.process[-1].inputs[0].iterable)
        random_start = randint(0, int(size*0.9))
    else:
        random_start = 0

    self.episode_id = str(uuid.uuid4())
    self.clock.reset()

    for c in self.components.values():
        if hasattr(c, "reset"):
            if isinstance(c, Observer):
                c.reset(random_start)
            else:
                c.reset()

    obs = self.observer.observe(self)

    self.clock.increment()

    return obs
```

### ریست شدن در کلاس `Feed`:

```
def reset(self, random_start = 0) -> None:
    for s in self.process:
        if isinstance(s, IterableStream):
            s.reset(random_start)
        else:
            s.reset()
```

## ریست شدن در کلاس Stream:

```
def reset(self, random_start = 0):
    if(random_start != 0):
        self._random_start = random_start

    if self.is_gen:
        self.generator = self.gen_fn()
    else:
        self.generator = iter(self.iterable[self._random_start:])
    self.stop = False

    try:
        self.current = next(self.generator)
    except StopIteration:
        self.stop = True
    super().reset()
```

## ۲.۳.۲.۴. پیاده سازی زمان پایان محیط هنگام ضرر بیش از ۵۰ درصدی

برای متوقف کردن سریع تر یک اپیزود هنگام ضرردهی و جریمه عامل به هنگام ضرر کردن بیش از ۵۰ درصدی در محیط، نیازمند به پیاده سازی زیر شدیم.

```
class MaxLossStopper(Stopper):
    """A stopper that stops an episode if the portfolio has lost a particular
    percentage of its wealth from maximum wealth.

    Parameters
    -----
    max_allowed_loss : float
        The maximum percentage of maximum funds that is willing to
        be lost before stopping the episode.

    Attributes
    -----
    max_allowed_loss : float
        The maximum percentage of maximum funds that is willing to
        be lost before stopping the episode.

    Notes
    -----
    This stopper also stops if it has reached the end of the observation feed.
    """

    def __init__(self, max_allowed_loss: float):
        super().__init__()
        self.max_allowed_loss = max_allowed_loss

    def stop(self, env: 'TradingEnv') -> bool:
        c1 = env.action_scheme.portfolio.profit_loss_max > self.max_allowed_loss
        c2 = not env.observer.has_next()
        return c1 or c2
```

## ۳.۳.۲.۴. پیاده سازی اقدام سه گانه خرید، فروش و نگهداری ارز برای محیط

پیاده سازی سیستم اقدام ساده خرید و فروش و نگهداری.

```
class SimpleOrdersThreeType(TensorTradeActionScheme):
    """A discrete action scheme that determines actions whith choises of
    Buying, Selling and Holding.

    @property
    def action_space(self) -> Space:
        if not self._action_space:
            self.actions = product(
                self.criteria,
                self.trade_sizes,
                self.durations,
                [TradeSide.BUY, TradeSide.SELL]
            )
            self.actions = list(self.actions)
            self.actions = list(product(self.portfolio.exchange_pairs, self.actions))
            self.actions = [None] + self.actions

            self._action_space = Discrete(len(self.actions))
        return self._action_space

    def get_orders(self,
                   action: int,
                   portfolio: 'Portfolio') -> 'List[Order]':
        if action == 0:
            return []
        (ep, (criteria, proportion, duration, side)) = self.actions[action]

        instrument = side.instrument(ep.pair)
        wallet = portfolio.get_wallet(ep.exchange.id, instrument=instrument)

        balance = wallet.balance.as_float()
        size = (balance * proportion)
        size = min(balance, size)

        if ((size == balance) and (size != 0.0)):
            size -= 10 ** -instrument.precision

        quantity = (size * instrument).quantize()

        if size < 10 ** -instrument.precision \
            or size < self.min_order_pct * portfolio.net_worth \
            or size < self.min_order_abs:
            return []

        order = Order(
            step=self.clock.step,
            side=side,
            trade_type=self._trade_type,
            exchange_pair=ep,
            price=ep.price,
            quantity=quantity,
            criteria=criteria,
            end=self.clock.step + duration if duration else None,
            portfolio=portfolio
        )

        if self._order_listener is not None:
            order.attach(self._order_listener)

        return [order]
```



## ۴.۳.۲.۴. پیاده سازی اقدام خرید و فروش ارز با مقادیر پیوسته برای محیط

پیاده سازی سیستم اقدام با یک متغیر پیوسته برای خرید، فروش و نگهداری.

```
class SimpleOrdersContinuous(TensorTradeActionScheme):
    """A continuous action scheme that determines actions but it can only be used with one exchange pair.

    @property
    def action_space(self) -> Space:
        if not self._action_space:
            self.actions = product(
                self.criteria,
                self.trade_sizes,
                self.durations,
                [TradeSide.BUY, TradeSide.SELL]
            )
            self.actions = list(self.actions)
            self.actions = list(product(self.portfolio.exchange_pairs, self.actions))
            self.actions = [None] + self.actions

            self._action_space = Box(low=-1, high=1,
                                     shape=(1,), dtype=float,)
        return self._action_space

    def get_orders(self,
                  action: float,
                  portfolio: 'Portfolio') -> 'List[Order]':

        if (action[0] < 0.05) and (action[0] > -0.05):
            return []

        proportion = round(abs(action[0]), 2)
        if action[0] > 0:
            side = TradeSide.BUY
        else:
            side = TradeSide.SELL
        criteria = self.criteria[0]
        duration = self.durations[0]
        ep = self.portfolio.exchange_pairs[0]

        instrument = side.instrument(ep.pair)
        wallet = portfolio.get_wallet(ep.exchange.id, instrument=instrument)
        balance = wallet.balance.as_float()
        size = (balance * proportion)
        size = min(balance, size)
        quantity = (size * instrument).quantize()

        if size < 10 ** -instrument.precision \
            or size < self.min_order_pct * portfolio.net_worth \
            or size < self.min_order_abs:
            return []

        order = Order(
            step=self.clock.step,
            side=side,
            trade_type=self._trade_type,
            exchange_pair=ep,
            price=ep.price,
            quantity=quantity,
            criteria=criteria,
            end=self.clock.step + duration if duration else None,
            portfolio=portfolio
        )

        if self._order_listener is not None:
            order.attach(self._order_listener)

        return [order]
```

### ۵.۳.۲.۴. پیاده سازی تابع پاداش محاسبه سود منهای بنچمارک برای محیط

پیاده سازی تابع پاداش با استفاده از محاسبه سود کسب شده منهای مقدار تغییر قیمت رمزارز مورد معامله.

```
class SimpleProfitMinusBuyandHold(TensorTradeRewardScheme):

    def __init__(self, window_size: int = 1):
        self._window_size = self.default('window_size', window_size)

    def get_reward(self, portfolio: 'Portfolio') -> float:
        net_worths = [nw['net_worth'] for nw in portfolio.performance.values()]
        benchmark = [nw['binance:/USDT-ETH'] for nw in portfolio.performance.values()]
        returns = [(b - a) / a for a, b in zip(net_worths[:-1], net_worths[1:])]
        returns = np.array([x + 1 for x in returns[-self._window_size:]]).cumprod() - 1
        if len(benchmark) < self._window_size:
            benchmark_returns = benchmark[-1]/benchmark[0] - 1
        else:
            benchmark_returns = benchmark[-1]/benchmark[-self._window_size] - 1
        if len(returns) > 0:
            diff = returns[-1] - benchmark_returns
            reward = np.sign(diff) * (diff)**2
        return 0 if len(returns) < 1 else reward
```

### ۶.۳.۲.۴. پیاده سازی تابع پاداش محاسبه سود منهای بنچمارک و جریمه عدم معامله

برای محیط

پیاده سازی تابع پاداش با استفاده از محاسبه سود کسب شده منهای مقدار تغییر قیمت رمزارز مورد معامله. همچنین در صورت نگه داشتن کل دارایی و عدم معامله بعد از مدت زمان معادل پنجره محیط مقداری جریمه از پاداش عامل کم می شود.

```

class SimpleProfitMinusBuyandHoldWithCashPenalty(TensorTradeRewardScheme):

    def __init__(self, window_size: int = 1):
        self._window_size = self.default('window_size', window_size)

    def get_reward(self, portfolio: 'Portfolio') -> float:
        cash_penalty = 0
        net_worths = [nw['net_worth'] for nw in portfolio.performance.values()]
        benchmark = [nw['binance:/USDT-ETH'] for nw in portfolio.performance.values()]
        returns = [(b - a) / a for a, b in zip(net_worths[:-1], net_worths[1:])]
        returns = np.array([x + 1 for x in returns[-self._window_size:]]).cumprod() - 1
        if len(benchmark) < self._window_size:
            benchmark_returns = benchmark[-1]/benchmark[0] - 1
        else:
            benchmark_returns = benchmark[-1]/benchmark[-self._window_size] - 1
        if len(returns) > 0:
            diff = returns[-1] - benchmark_returns
            reward = np.sign(diff) * (diff)**2
        if len(cash) > self._window_size:
            if all(x == net_worths[0] for x in cash):
                cash_penalty = 0.5
            else:
                cash_penalty = 0
        return 0 if len(returns) < 1 else reward-cash_penalty

```

## ۷.۳.۲.۴. پیاده‌سازی تابع پاداش محاسبه سود بر حسب هر معامله برای محیط

پیاده‌سازی تابع پاداش با استفاده از محاسبه سود کسب‌شده به علاوه مقدار سود کسب‌شده از هر معامله

مشخص در هنگام فروش.

```

class TradeBased(TensorTradeRewardScheme):

    def __init__(self, window_size: int = 1):
        self._window_size = self.default('window_size', window_size)

    def get_reward(self, portfolio: 'Portfolio') -> float:
        current_step = [step for step in portfolio.performance.keys()][-1]
        cash_total = [nw['binance:/USDT:/total'] for nw in portfolio.performance.values()]
        trade_steps = [i for i, (x, y) in enumerate(zip(cash_total[:-1], cash_total[1:])) if x!=y]
        trade_sides = [x>y for i, (x, y) in enumerate(zip(cash_total[:-1], cash_total[1:])) if x!=y]
        prices = [nw['binance:/USDT-ETH'] for nw in portfolio.performance.values()]
        trade_prices = [prices[i] for i in trade_steps]
        trade_profit = 0
        if len(trade_steps)>0:
            if trade_sides[-1] == False and trade_steps[-1] == current_step-1:
                for i in range(len(trade_sides)-1, 0, -1):
                    if trade_sides[i] == True:
                        trade_profit = (trade_prices[-1]/trade_prices[i])-1-0.002
                        break
        return 0 if len(trade_steps) < 1 else trade_profit

```

## ۸.۳.۲.۴. پیاده‌سازی تابع پاداش محاسبه سود با جریمه عدم معامله برای محیط

پیاده‌سازی تابع پاداش با استفاده از محاسبه سود کسب‌شده منهای مقداری جریمه به هنگام عدم معامله.

```
class SimpleProfitWithCashPenalty(TensorTradeRewardScheme):

    def __init__(self, window_size: int = 1):
        self._window_size = self.default('window_size', window_size)

    def get_reward(self, portfolio: 'Portfolio') -> float:
        net_worths = [nw['net_worth'] for nw in portfolio.performance.values()]
        cash = [nw['binance:/USDT:/total'] for nw in portfolio.performance.values()]
        returns = [(b - a) / a for a, b in zip(net_worths[:-1], net_worths[1:])]
        returns = np.array([x + 1 for x in returns[-self._window_size:]]).cumprod() - 1
        cash_penalty = cash[-1] - net_worths[-1] * 0.3
        if cash_penalty > 0:
            cash_penalty = self._window_size * 0.2
        else:
            cash_penalty = 0
        return 0 if len(returns) < 1 else returns[-1] - cash_penalty
```

## ۴.۲.۴. مدل یادگیری تقویتی عمیق

برای پیاده‌سازی مدل یادگیری تقویتی عمیق از کتابخانه Stable\_Baselines3 استفاده کردیم. در تصویر زیر نحوه پیاده سازی و استفاده Callback های پیاده‌سازی شده را می‌بینید.



```

class TensorboardCallback(BaseCallback):
    """
    Custom callback for plotting additional values in tensorboard.
    """
    def __init__(self, verbose=0):
        super(TensorboardCallback, self).__init__(verbose)
        self.net_worth = 0

    def _on_rollout_end(self) -> None:
        performance = pd.DataFrame.from_dict(
            {
                'performance': self.model.env.get_attr('action_scheme')[0].portfolio.performance,
                'net_worth': self.net_worth,
            }, orient='index'
        )
        value = performance.tail(1)['net_worth'].values[0]
        self.net_worth = value
        self.logger.record("rollout/net_worth", self.net_worth)

    def _on_step(self) -> bool:
        return True

checkpoint_callback = CheckpointCallback(save_freq=1000000,
                                         save_path='./logs/',
                                         name_prefix='rl_model')

tensorboard_callback = TensorboardCallback()
eval_callback = EvalCallback(eval_env,
                             best_model_save_path='./logs/eval/best_model',
                             log_path='./logs/eval',
                             n_eval_episodes=1,
                             eval_freq=5000,
                             deterministic=True,
                             render=False)

callback = CallbackList([
    checkpoint_callback,
    tensorboard_callback,
    eval_callback
])

```

در ادامه نحوه نمونه گرفتن مدل PPO و آموزش و ذخیره آن را مشاهده می کنید.

```

env = environment('train', "BINANCE_ETHUSD_5_signal", "simple", "compared-to-BuyandHold", 15)

#Custom actor (pi) and value function (vf) networks
#of two layers of size 64 and 64 each with tanh activation function
policy_kwargs = dict(activation_fn=th.nn.tanh,
                     net_arch=[dict(pi=[64, 64], vf=[64, 64])])
agent = PPO(ActorCriticPolicy, env, policy_kwargs=policy_kwargs,
            verbose=1, tensorboard_log='./tensorboard/', learning_rate=0.01)
agent.learn(total_timesteps=1400000, tb_log_name="first_run", callback=callback)
agent.save("trader-test")

```

## ۵.۲.۴. پیاده‌سازی‌های معیارهای ارزیابی

### ۱.۵.۲.۴. نسبت شارپ

این نسبت برای محاسبه میزان سود اصلاح شده با ریسک معاملات استفاده می‌شود.

```
def sharpe_ratio(returns):
    risk_free_rate = 0
    return (np.mean(returns) - risk_free_rate + 1e-9) / (np.std(returns) + 1e-9)
```

### ۲.۵.۲.۴. معیارهای مربوط به زمان و مقدار ضرردهی متوالی

در ادامه پیاده‌سازی‌های مربوط به حداکثر میزان ضرر در طول تست، طولانی‌ترین مدتی که عامل در

حال ضرردهی بوده را می‌بینیم.

```
performance = pd.DataFrame.from_dict(env.action_scheme.portfolio.performance, orient='index')
net_worths = performance.net_worth

highwatermarks = net_worths.cummax()
drawdowns = 1 - (1 + net_worths) / (1 + highwatermarks)
max_drawdown = max(drawdowns)

drawdown_times = (drawdowns > 0).astype(np.int64)
max_drawdown_days = (max(accumulate(drawdown_times, lambda x,y: (x+y)*y))*5)/1440

total_drawdown_time = drawdown_times.groupby((drawdown_times != drawdown_times.shift()).cumsum()).cumsum().max()
```

### ۳.۵.۲.۴. مقدار سوددهی

مقدار سوددهی در کل طول زمان تست و سود سالیانه.

```
current_net_worth = round(net_worths[len(net_worths)-1], 1)
initial_net_worth = round(net_worths[0], 1)
profit_percent = round((current_net_worth - initial_net_worth) / initial_net_worth * 100, 2)
profit_percent_yearly = profit_percent/((duration/(8640*12)))
```

### ۳.۴. نتایج به دست آمده از ارزیابی مدل

ما برای ارزیابی مدل پیشنهادی از داده های رمزارز بیت کوین و تبدیل آن به رمزارز تتر که معادل ۱ دلار قیمت دارد استفاده کردیم. داده ها شامل ۴۱ ماه داده با دقت ۵ دقیقه ای از ابتدا سال ۲۰۱۸ می باشد.

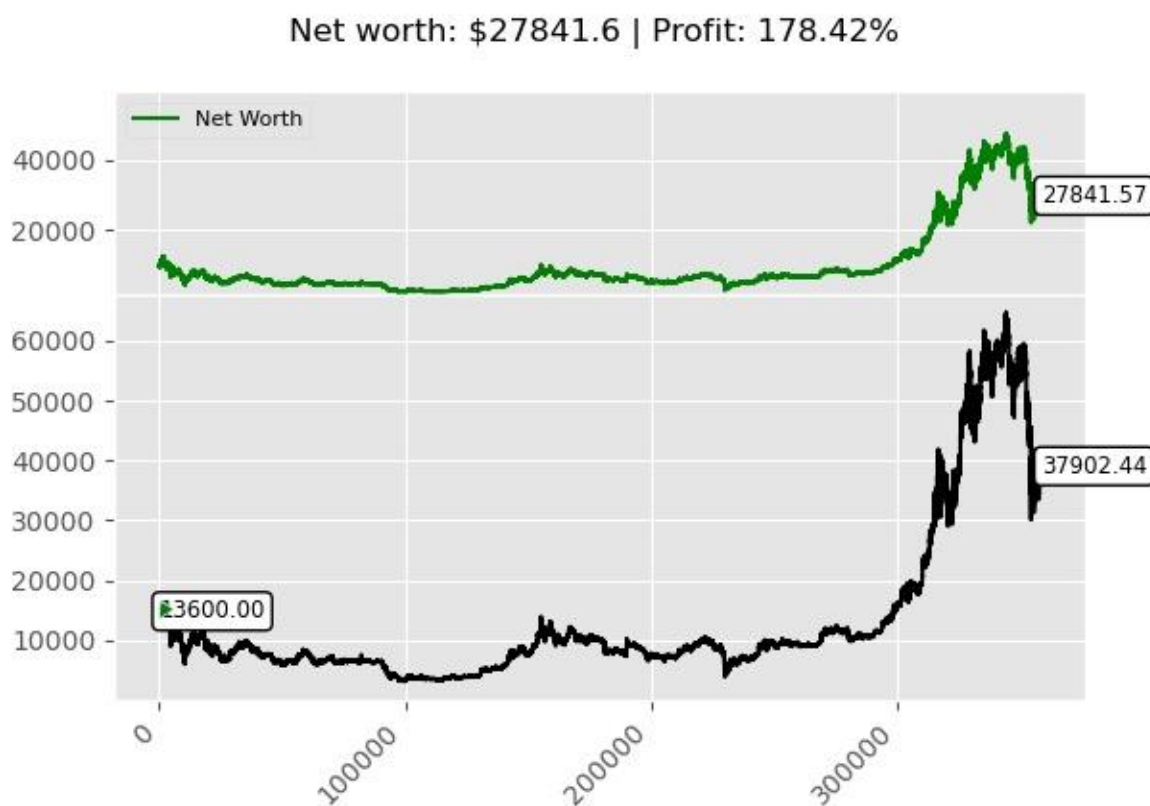
مدل ها همگی بر روی داده های رمزارز اتریوم آموزش دیده شده اند و برای جدا کردن داده های آموزش و تست و نشان دادن عدم وابستگی مدل به یک رمزارز خاص ارزیابی خود را بر روی داده های رمزارز بیت کوین انجام دادیم.

برای ارزیابی در تمامی حالات از مقدار سرمایه اولیه ۱۰۰۰۰ دلار شروع کردیم. سیگنال های ورودی دارای حدود ۳۵ درصد خطا می باشند. معیار های ارزیابی مدل های آموزش داده شده موفق در جدول (۴-۱) آمده است.

جدول ۴-۱ نتایج پیاده سازی

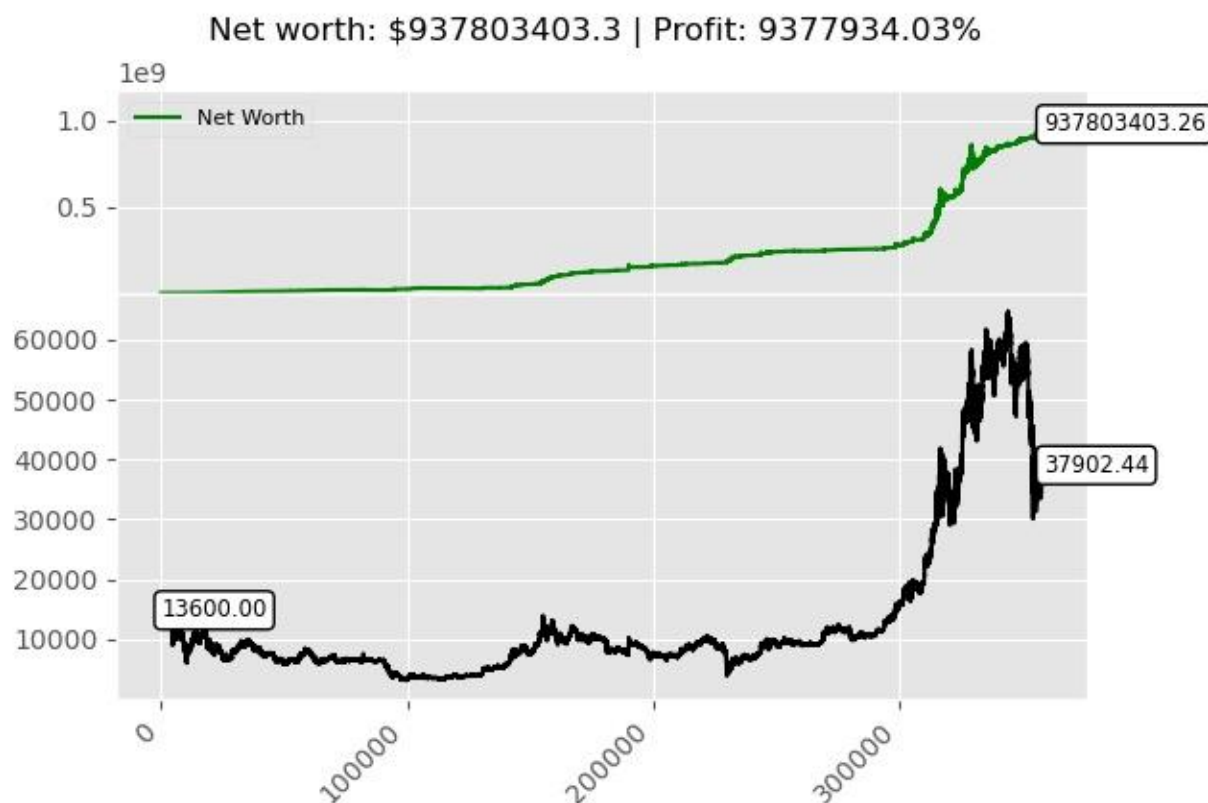
No	1	2	3	4	5	6	7
Model	-	Manual Strategy	PPO	PPO	PPO	PPO	TD3
Noise Type	-	Type2	Type1	Type1	Type2	Type2	Type2
Num Steps	-	-	4 million	2 million	2 million	2 million	4 million
Reward Function	-	-	Profit of observation window minus the profit of Buy and Hold in the same window	Profit of observation window minus the profit of Buy and Hold in the same window	Profit of observation window	Profit of observation window minus the profit of Buy and Hold in the same window	Profit of observation window
Action Type	Buy at beginning and Hold	Total buy, sell and hold	Total buy, sell and hold	Buy, sell and hold in discrete amounts	Buy, sell and hold in discrete amounts	Buy, sell and hold in discrete amounts	Buy and sell in continuous amounts
Sharpe Ratio	0.0024	0.0068	0.01618	0.0259	0.01563	0.01032	0.02380
Number of Trades	1	3494	59379	227482	323521	9842	33422
Average Time Between Trades	-	512 min	29 min	7.87 min	5.53 min	181.83 min	53.57 min
Max DrawDown	81.57%	16%	52.56%	18.52%	37.71%	7.38%	26.13%
Max time of a DrawDown	1042 days	502 days	271 days	85 days	61 days	116 days	144 days
Total Profit	178.42%	465.63%	3855922%	9377934%	385164%	116%	255212%
Profit Annually	51.64%	134.78%	1129136%	2714583%	111491%	33.68%	73875%

نمودار افزایش سرمایه در کنار قیمت رمزارز بیت کوین برای استراتژی خرید در ابتدا بازه و نگه داری آن، برای مقایسه در شکل ۱-۴ آمده است.



شکل ۱-۴ نمودار سود حاصل نگهداری رمزارز بیت کوین

نمودار افزایش سرمایه در کنار قیمت رمزارز بیت کوین برای بهترین نتیجه به دست آمده و مدل پیشنهادی یعنی مدل PPO با آموزش همراه تابع پاداش سود منهای تغییرات قیمت رمزارز مورد معامله و اقدام های ۰، ۱، ۰، ...، ۱ درصد خرید، فروش و نگه داری، بعد از ۲ میلیون گام آموزش در شکل ۲-۴ آمده است.



شکل ۲-۴ نمودار سود حاصل از استفاده از مدل پیشنهادی

#### ۴.۴. تحلیل نتایج

در استفاده از تمامی توابع پاداش، انواع اقدام و مدل های یادگیری تقویتی که در جدول ۴-۱ آورده نشده اند به نتیجه قابل قبولی نرسیدیم.

با توجه به ستون های ۴ و ۵ جدول ۴-۱ مشخص است که مدل پیشنهادی با استفاده از هر دو نوع نويز تولید شده در سیگنال های ورودی به موفقیت رسیده است.

مدل پیشنهادی PPO در محیط با خروجی گسسته از مدل DQN بهتر عمل کرد به طوری که توسط مدل DQN با تعداد گام مشابه و شبکه عصبی مشابه به موفقیت دست پیدا نکردیم در حالی که مدل PPO بسیار موفق بود. در محیط با خروجی پیوسته مدل PPO با شبکه عصبی به اندازه پیشنهادی به موفقیت دست

نیافت ولی مدل TD3 با لایه پنهانی شامل ۴۰۰ و سپس ۳۰۰ نورون که بزرگتر از لایه پنهان مدل PPO است به موفقیت رسید.

برای آموزش تنها با استفاده از دو نوع از توابع پاداش موفق به آموزش مدلی موفق شدیم. تابع پاداش محاسبه سود و تابع پاداش محاسبه سود منهای سود رمزارز نگهداری شده. که از بین این دو با توجه به ستون‌های ۵ و ۶ جدول ۴-۱ می‌توانید مشاهده کنید که تابع پاداش محاسبه سود ساده خیلی بهتر عمل کرده است. مشکلی که در آموزش هنگام استفاده از تابع پاداش سود منهای سود رمزارز نگهداری شده و تابع پاداش محاسبه سود با احتساب ریسک (نسبت شارپ) مشاهده شد، این بود که مدل بعد از مدتی در اکسپلوریشن محلی گیر میکرد و دیگر مدل خرید یا فروش انجام نمی داد. برای حل این مشکل ممکن است ترکیب این توابع با تابع پاداش همراه با جریمه به هنگام معامله نکردن خروجی موفقیت آمیز تولید کند.

با توجه به نتایج با سه نوع اقدام موفق به آموزش مدل شدیم. ضعیف ترین نتیجه مربوط به اقدام خرید و فروش با کل پول هست که با مقایسه ستون ۳ و ۴ جدول ۴-۱ می‌توانید مشاهده کنید که این نوع اقدام در مقایسه با نوع اقدامی که شدت خرید و فروش با مقادیر گسسته تعیین می شود بسیار ضعیف تر عمل کرده است. برای مقایسه اقدام با مقادیر گسسته و پیوسته کفایت ستون های ۵ و ۷ را در جدول ۴-۱ با هم مقایسه کنید. ستون ۵ که مربوط به اقدام با مقادیر گسسته می باشد سود بیشتری را تولید کرده است و حداکثر مدت زمان ضرردهی کوتاه تری داشته است ولی در عوض نسبت شارپ کمتر و حداکثر ضرر بیشتری داشته است. با توجه به این که برای اقدام نوع پیوسته از مدلی با لایه پنهان بزرگتر و تعداد گام بیشتری برای آموزش استفاده کردیم، اقدام از نوع گسسته به نظر برتری کمی به نوع پیوسته دارد.

با توجه به موارد بالا در صورت نیاز به خروجی گسسته مشابه ستون ۵ جدول ۴-۱ مدل PPO با اقدام های خرید و فروش به مقادیر گسسته و تابع پاداش سود کسب شده موفق ترین حالت بود. و در صورت نیاز به خروجی پیوسته مدل مربوط به ستون ۷ یعنی مدل TD3 با تابع پاداش مشابه پیشنهاد می‌شود. در کل نیز به علت نتیجه مشابه ولی مدل کوچکتر حالت اول از دو حالت ذکر شده پیشنهاد می شود.

در انتها نیز می‌توانید مشاهده کنید که هر دو مدل مربوط به ستون ۵ و ۷ جدول ۴-۱ نتایج بسیار بهتری از خود رمزارز (ستون ۱) و استراتژی پیاده سازی شده دستی (ستون ۲) داشته است که نشان دهنده موفقیت این دو مدل است.

## ۵.۴. خلاصه و جمع بندی

در این فصل به بیان نحوه پیاده سازی مدل یادگیری تقویتی عمیق پیاده سازی به همراه محیط تعامل آن پرداخته شد. در ابتدا به بیان نحوه جمع آوری داده ها و سپس تولید سیگنال های خرید و فروش و اضافه کردن خطا و نویز به آن پرداخته شد و بعد به پیاده سازی محیط و مدل پرداخته شد. در انتها نیز نتایج به دست آمده از پیاده سازی بررسی شد.



۵. فصل پنجم: جمع‌بندی، نتیجه‌گیری و پیشنهادها

## ۱.۵. جمع‌بندی

در این پایان نامه در فصل نخست ما به تعریف مساله و بیان مقدمات پرداختیم. همچنین درباره هدف پایان‌نامه، کارهای انجام شده قبلی و اهمیت این موضوع صحبت شد.

در فصل دوم به بیان پیش‌زمینه‌های مورد نیاز برای تحلیل بازار رمزارزها پرداختیم. درباره بازارهای مالی و مفاهیم ابتدایی آن‌ها صحبت کردیم سپس به ساختار مدل‌های یادگیری عمیق و یادگیری تقویتی مورد استفاده در این پروژه پرداختیم.

در فصل سوم به معرفی نحوه جمع‌آوری داده‌ها، نحوه تولید سیگنال ورودی، محیط یادگیری تقویتی و مدل‌های مورد استفاده خود پرداختیم و معیارهایی را برای ارزیابی معرفی کردیم.

در فصل چهارم ابتدا به پیاده‌سازی نحوه جمع‌آوری داده‌ها و تولید سیگنال‌های ورودی مورد نیاز پرداختیم. سپس تغییرات پیاده‌سازی شده در کتابخانه استفاده شده برای محیط یادگیری تقویتی را بیان کردیم و سپس به نحوه پیاده‌سازی مدل یادگیری تقویتی پرداختیم. در انتها نیز نتایج پیاده‌سازی را مشاهده کردیم.

## ۲.۵. نتیجه‌گیری

### ۱.۲.۵. دستاوردها

در این پروژه دیدیم که امکان کسب سود از سیگنال‌های ورودی با خطای حدود ۳۵ درصد ممکن است و از این روش می‌توان نتایج پژوهش‌های مربوط به پیش‌بینی قیمت و روند بازارهای مالی را بهبود داد و از مجموعه این پژوهش و دیگر پژوهش‌ها استفاده مفید در روزمره داشت.

علاوه بر نتیجه کلی ذکر شده دستاوردهای دیگر این پروژه پیاده‌سازی رابط برنامه‌نویسی کاربردی کاملی برای دریافت اطلاعات گذشته و برخط قیمت، اندیکاتور و اطلاعات انواع بورس‌ها، فارکس و بازار رمزارزها می‌توان اشاره کرد.

دستاوردهای دیگر این پروژه محیط یادگیری عمیق استفاده شده است که امکان استفاده از آن برای انواع پروژه‌های یادگیری تقویتی در بازارهای مالی است.

## ۲.۲.۵. محدودیت‌ها

برای آموزش و تست انواع مدل‌های یادگیری تقویتی و سیستم‌های پاداش متنوع و انواع سیستم‌های اقدام، نیاز به قدرت پردازشی بسیار زیادی هست.

## ۳.۲.۵. پیشنهادها

در این پژوهش ما تلاش کردیم تا سیگنال‌های ورودی را با درصدی خطا به طور مصنوعی تولید کنیم. پیاده‌سازی چند پژوهش با هدف تولید سیگنال‌های پیش‌بینی روند قیمتی و ترکیب این سیگنال‌ها به عنوان ورودی به مدلی براساس روش پیشنهادی این پژوهش، می‌تواند نتایج حقیقی و بسیار مناسبی تولید کند که می‌توان زمینه‌ای برای تحقیقات آتی باشد.

همچنین با توجه به این که محیط پیاده‌سازی‌شده در این پروژه امکان معامله هم‌زمان بر روی چند رمزارز مختلف را دارد، پیشنهاد می‌شود با آموزش مدل یادگیری تقویتی بر روی چند رمزارز به طور هم‌زمان موفقیت یادگیری تقویتی در این حالت نیز بررسی شود تا مشخص شود که با چند مدل یادگیری تقویتی به طور هم‌زمان به عملکرد بهتری می‌توان دست پیدا کرد یا با آموزش یک مدل بر روی چند رمزارز مختلف به طور هم‌زمان.

٦. مراجع

- [1] F. Fang *et al.*, “Cryptocurrency trading: A comprehensive survey,” *arXiv [q-fin.TR]*, 2020.
- [2] T. Awoke, M. Rout, L. Mohanty, and S. C. Satapathy, “Bitcoin price prediction and analysis using deep learning models,” in *Communication Software and Networks*, Singapore: Springer Singapore, 2021, pp. 631–640.
- [3] F. Valencia, A. Gómez-Espinosa, and B. Valdés-Aguirre, “Price movement prediction of cryptocurrencies using sentiment analysis and machine learning,” *Entropy (Basel)*, vol. 21, no. 6, p. 589, 2019.
- [4] J. Carapuço, R. Neves, and N. Horta, “Reinforcement learning applied to Forex trading,” *Appl. Soft Comput.*, vol. 73, pp. 783–794, 2018.
- [5] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, “Deep direct reinforcement learning for financial signal representation and trading,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, 2017.
- [6] A. R. Azhikodan, A. G. K. Bhat, and M. V. Jadhav, “Stock trading bot using deep reinforcement learning,” in *Innovations in Computer Science and Engineering*, Singapore: Springer Singapore, 2019, pp. 41–49.
- [7] Y. Ye *et al.*, “Reinforcement-learning based portfolio management with Augmented asset movement prediction states,” *Proc. Conf. AAAI Artif. Intell.*, vol. 34, no. 01, pp. 1112–1119, 2020.
- [8] D. W. Lu, “Agent inspired trading using recurrent Reinforcement Learning and LSTM neural networks,” *arXiv [q-fin.CP]*, 2017.
- [9] Y. Li, P. Ni, and V. Chang, “Application of deep reinforcement learning in stock trading strategies and stock forecasting,” *Computing*, vol. 102, no. 6, pp. 1305–1322, 2020.

- [10] Kon Mamadou Tadiou. The future of human evolution - artificial neural networks. <http://futurehumanevolution.com/artificial-intelligence-future-human-evolution/artificial-neural-networks>, 2010. [Online; Accessed 24 May 2016]
- [11] S. B. S. AlMarri, “Real-Time Facial Emotion Recognition Using Fast R-CNN by Salem Bin,” 2019.
- [12] Julien Vitay. Deep Reinforcement Learning. <https://julien-vitay.net/deeprl/ActorCritic.html>. [Online; Accessed 1 Sep 2021]
- [13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” arXiv [cs.LG], 2017.

## **Abstract**

*Nowadays, investing and trading in the cryptocurrency market has become one of the most attractive markets for individuals and investment firms. In recent years, trading in these markets has shifted from manual trading to algorithmic trading and trading based on artificial intelligence. The purpose of this study is to implement an intelligence agent based on deep reinforcement learning to identify the optimal time and amount of buy and sell actions for a cryptocurrency pair. The method of choice for this implementation is to separate the different parts of this intelligent agent. The process of trading in Cryptocurrency market by smart agents consists of two main parts, predicting Cryptocurrency's price trend and managing the portfolio based on input signals. In this thesis, the aim is to implement an intelligent agent that is in the second category and uses input signals to decide the appropriate time and amount to buy and sell cryptocurrencies. This kind of separation of the components of an intelligent agent will lead to better tuning, debugging, and understanding of the overall system. In the past, a great deal of research has been done on various methods of predicting market trends, including forecasting using neural networks, technical analysis, and collecting and analyzing news texts. Future researchers are advised to review the combination of the results of various types of research related to trend and market price forecasting and similar research to this thesis for future research.*

## **Keywords:**

*Cryptocurrency, Deep Reinforcement Learning, Algorithmic Trading, Portfolio Management*



University of Tehran



College of Engineering  
School of Electrical and Computer Engineering

Implementation of Intelligent Agent with Deep Reinforcement Learning for  
Algorithmic Trading in Financial Markets

A thesis submitted to the Undergraduate Studies Office  
In partial fulfillment of the requirements for  
The degree of Bachelor of Science in  
Computer Engineering

By:  
Parsa Sadri Sinaki

Supervisor:  
Dr. Saeed Safari

September 2021