Evaluating Deep Reinforcement Learning for Asset Management in the Crypto Market

Singature Work submitted in partial fulfilment of the requirements for the degree of

Bachelor of Science

in

Data Science

By

Carlos Gustavo Salas Flores
netID: cs582

Mentor: Ph.D. Luyao Zhang
Approval: Marcia B. France, Dean of Undergraduate Studies

Department of Natural Sciences
Duke Kunshan University, Kunshan, Jiangsu, 215316
China

February 2023

Abstract

Data Science is a very interdisciplinary field. As such, several disciplines need to perform some data analysis or statistical modeling to predict the future. In this paper, I propose a Reinforcement Learning approach to train two modified Vision Transformer (ViT) models and two CNN models using Q-Learning to trade in the crypto and Defi tokens market. The action space consists of 9 options which are different sets of digital assets and a neutral option. The results show that the actions taken by ViT models produce a higher Sharpe Ratio than CNN models in the evaluation dataset.

Contents

List of Figures

List of Tables

1 Introduction

Previous models using Transformer Encoders have been proposed such as AlphaPortfolio [1] for asset management in the stock market. On the other hand, other models such as DeepTrader [2] use Convolutional Layers, Attention, and LSTMs to achieve a similar task. Yet, these models are very complex, even at a high level.

The procedure used in this paper is focused on simplifying the process and models as much as possible. Instead of making a rule to handle a huge Action space $\mathcal{A}$, I rather use Unsupervised Learning to discretize the options that the model can take at a time and reduce it from $|\mathcal{A}| = 2^{121}$ to $|\mathcal{A}| = 9$. This approach to cluster assets is popular for Pairs Trading [3] but has not been applied to asset management before.

Furthermore, to the best of my knowledge, this is the first work to implement a Computer Vision Attention-based architecture such as the Vision Transformer (ViT) for asset management using Reinforcement Learning in the High-Frequency crypto market.

1.1 Motivation

Economists, Bankers, and Data Scientists all share something in common–they want to predict the future.

Everybody who works as an investor wants to beat the market and make good margins of returns. Furthermore, advances in technology and computer resources have made the Banking and Finance industry turn its eyes to Artificial Intelligence (AI) and Machine Learning (ML).

Today, several finance companies are reliant on automated high-frequency algorithms, AI-driven, and or blockchain technology to power their businesses. From old players such as BlackRock and J.P. Morgan to newly established quantum trading firms such as Citadel and Jane Street [4] [5].

While there is a well-developed financial theory around the stock market. The same cannot yet be said about the crypto market, especially in asset management. Classical models such as the Fama French models are not well-suited for studying cryptocurrencies or Defi tokens.

Therefore, the goal of this paper is to contribute to the study of asset management in the crypto market. This study makes use of Reinforcement Learning to train different Deep Q-Learning models with the main goal of letting it find the optimal trading strategy to maximize the reward (return of investment or shape ratio). Inspired by AlphaFold, this study transforms this time series problem into a Computer Vision problem, instead of feeding the model with a handful of indicators, the model is fed with a covariance matrix, which serves as the input image to the computer vision algorithm [6] [7]. This model will then be studied to find patterns in its decision-making process and unveil what factor(s) is(are) behind a good expected return.

1.2 Problem Overview

Broadly speaking, the solution to asset management using Reinforcement Learning is tackled by choosing which options to hold at each point t in time.

First, the original problem allows for a very broad spectrum of options to hold at every point in time. I constrain this by creating a pre-defined portfolio of options, which is nothing but sets of assets that one can hold or sell at a time. Therefore, reducing the action space size from more than trillions of options to less than 20.

Second, the Reinforcement Learning algorithm will learn the best trading strategy at each point in time t by using a Q-approximation function as in [8]. In this paper, I test two different models to approximate Q with two different input formats, Plan A is the state-of-the-art for image classification tasks Visualization Transformer (ViT) [9] which takes a snapshot of historical prices as input. Plan

B is the old-reliable Convolutional Neural Networks (CNNs) which will take a covariance matrix as input. Both approaches are summarized and shown in Figure 5.

While the CNN approach is more widely used, it falls short in attention since it focuses on a local area to find relationships. However, because of the nature of Transformers, it is not easy to scale the solutions to even low-resolution images of size 256×256 pixels, yet, [9] tackle this issue by dividing the image into 16×16 patches. However, because of the nature of the asset management problem, it makes more sense to rather use each row (or column since both share essentially the same information).

2 Literature Review

2.1 The Economics: Asset Management and the Crypto Market

Despite the general interest in being able to forecast the future of the stock market [10]. Previous works have shown that forecasting the stock market is essentially unreliable as it essentially behaves like a random walk [11] [12]. Yet, if considered on a broader scale, it is possible to make accurate estimations of future returns if we consider the whole market. Already in 1992, Fama and French developed a three-factor model to calculate the expected returns of a portfolio by using historical data about the stocks and the market [13]. Which they later expanded by adding two more parameters to their model [14].

While this shows the huge advance and interest in developing techniques and metrics to forecast future returns from the stock market. New forms of assets have emerged within the last 10 years–cryptocurrencies and Defi tokens–that don't necessarily follow the same trends and patterns as the stock market.

Cryptocurrencies and Defi tokens have grown in popularity these last few years. Figure 1 shows the market cap of BTC and ETH (the most popular coins by market cap) throughout the years since their conception. As of Jan 19 of 2023, their market cap adds up to \$594.74B USD which is more than 50% the value of the whole crypto market cap (\$1.0T USD) [15]. Not only that but according to Panigirtzoglou (2021) Bitcoin has surpassed Gold in risk capital terms; theoretically speaking, they also expect BTC's long-term price to be set at \$149,000 USD which would also imply by then that the volatility of BTC would be lower than that of Gold, even though the opposite is being observed today [4].

J.P. Morgan is not the only motivated agent, investors and other financial institutions have a growing interest in the crypto market. Which makes Cryptocurrencies and Defi tokens exciting to invest in and include in portfolios. Yet, it also comes with bigger risks. Due to the unregulated nature of these currencies, they are prone to higher and more unpredictable volatility [16]. A paper suggested that BTC had some similarities with the dollar and gold [17]. But due to its nature, it does not seem to fit into classical financial theories such as the Efficient Market Hypothesis (EMH); instead, it fits better into more contemporary theories such as the Adaptive Market Hypothesis (AMH) [18].

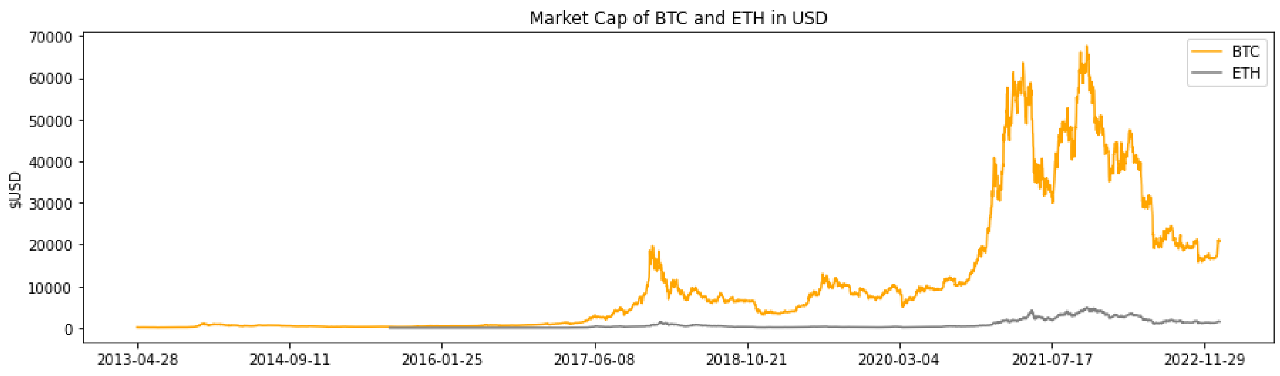


Figure 1: Cumulative explained variance.
Daily Market Cap (in USD) of Bitcoin (BTC) and Ethereum (ETH) from their release up until Jan 19th, 2023.

2.2 Reinforcement Learning

Little is discussed nowadays on how, prior to 2012, Machine Learning was not taken seriously by most people in Computer Science. It wasn't until AlexNet was shown to the public, that Scientists and Engineers saw its potential [19]. AlexNet was arguably the major breakthrough in Machine Learning. While the original model is rarely used today, it laid the ground for future research, not only in Computer Science but also in Finance, Archeology, and even Linguistics.

However, another major breakthrough that is not often discussed is that of Reinforcement Learning. Not long after the publication of AlexNet, a new company called DeepMind published its first paper titled ”Playing Atari with Deep Reinforcement Learning” [8].

Reinforcement Learning was able to do tasks that seemed impossible or (very far in time) to teach a computer to do. In less than ten years they taught an algorithm how to play Atari 2600 games, Chess, Go, and Dota 2, but they also revolutionized other fields such as medicine when they introduced AlphaFold [7] [8] [20] [21]. Not long ago, they also created a general AI that was able to perform different tasks but also in the fields of mathematics and computational sciences, they trained an algorithm to find the best way to perform matrix multiplications [22] [23].

Looking in retrospect, it is very clear that AI, Reinforcement Learning specifically in particular, is a very powerful tool that can help us to solve real-life problems. Therefore, some scholars have started to implement Reinforcement Learning strategies for portfolio management or automated trading.

2.3 Attention & Transformers

The biggest breakthrough in Machine Learning so far has been the use of Attention mechanisms through Transformers.

While Attention operations themselves were not first introduced in [24], this work was the first to show their true potential by creating the Transformer architecture. While the original paper simply proposed it as a better sequential model to translate between German and English, its potential demonstrated to be key in developing some of the most important AI tools and technologies of today such as GPT-3, Stable Diffusion, and even Full Self Driving models [25] [26] [27].

According to Andrej Karpathy, former director of AI at Tesla and founding member of OpenAI, the Transformer architecture is one of the most surprising and beautiful ideas since it is a general-purpose computer (it can be trained on an arbitrary task) that is optimizable with a simple algorithm such as is back-propagation and gradient descent. Furthermore, it is able to run efficiently in GPUs (today’s hardware) by allowing parallelism i.e. run several operations simultaneously but not sequentially [28].

2.4 Previous work

Previous studies have researched using Q-learning for algorithmic trading and portfolio management.

On trading strategies involving focusing on a single asset data at a time, Jeong and Kim (2019) used transfer learning to avoid overfitting in trading stock indexes but also used a two-branch model to train one on finding the number of shares to trade and another one to predict the best actions [29]. Using a slightly different approach, Carta et. al. (2020) use a Double Q-Learning approach to trade indexes and compare using different agents, namely Long, Short, Long+Short, and a Full Agent [30]. A slightly simpler approach was proposed by Teate and Ernst (2021) whom proposed a modification of the DQN algorithm called the TDQN algorithm that they tested on US stocks [31].

Now, in portfolio management, Park, Sim, and Choi (2020) implemented a DQN algorithm to manage Exchange Trading Funds (ETFs) portfolios with a Deep Neural Network (DNN) and a Long-Short Term Memory (LSTM) and tested their results on the US market and the Korean market [32]. Another approach was proposed by Chakole et al. (2020) they used a Markov Decision Process (MDP) along with Q-learning, they decided to implement a dynamic trading strategy on US and Indian stocks, which is why they constrained their algorithm to trade only a single asset at a time [33]. Another different approach has been recently taken by Cong et. al. (2022) who use financial data and transformers to create an asset management algorithm on US stocks [1].

Several of these methods have shown positive results. Yet, not all of them can be easily extrapolated to cryptocurrency data or DeFi tokens. Furthermore, some works [1] rely on huge amounts of data that are not easily available on the crypto or DeFi token market. Therefore, I decided to take a different approach inspired by AlphaFold [6] [7].

2.5 Inspiration

Senior et. al. (2020) use Convolutional Neural Networks to solve the problem of protein folding, more specifically they use Residual Neural Networks as their model and feed it with a covariance matrix [6]. Later, Jumper et. al. (2021) improve the method by introducing a novel block called Evoformer which is based on attention mechanisms, thereby improving the results obtained just a year ago [7]. While Jumper et. al. (2021) use attention mechanisms, the current state-of-the-art in attention mechanism for computer vision algorithms is Vision Transformers developed by Dosovitskiy et. al. (2021) [9]. In this study, I compare using both approaches, on the one hand, Convolutional Neural Networks [19] and, on the other hand, Vision Transformers [9].

3 Methodology

3.1 Data Preprocessing

3.1.1 Datasource

The original data was retrieved from AlphaVantage API on Oct 1st of 2022 and stored in a public github repository it contains hourly data (Close Price in USD) on 121 different cryptocurrencies and governance tokens from Aug 21st of 2022 to Oct 1st of 2022. The resulting 121 options came after filtering out those with any missing information and are detailed in Tables 9, 10, and 11 in the Appendix.

3.1.2 Action Space

The Action Space is an important element of the Environment $\mathcal{E}$ (which is discussed in detail later in this work) since it contains the possible moves we can make at each point in time t. However, selecting the right assets at every point in time is not an easy task since it escalates very quickly.

The number of possible tokens to hold is in principle $n \in \{0, 1, 2, ..., \mathrm{A}\}$, where A is the total number of assets. Then we can take a position $\mathrm{P} \in \{0, 1\}$ for each asset. If this is the case, then the action space would grow exponentially 2^{A}. Therefore, for all 121 tokens, the action space will be of size 2^{121} which is just unfeasible to compute.

Therefore, I opted for reducing my action space by clustering similar assets with each other and then focusing only on a few options at a time. For this, I propose an Unsupervised Learning approach.

Before running the UL algorithm, I performed dimensionality reduction using Principal Component Analysis, I set the threshold to 0.90, resulting in 41 components.

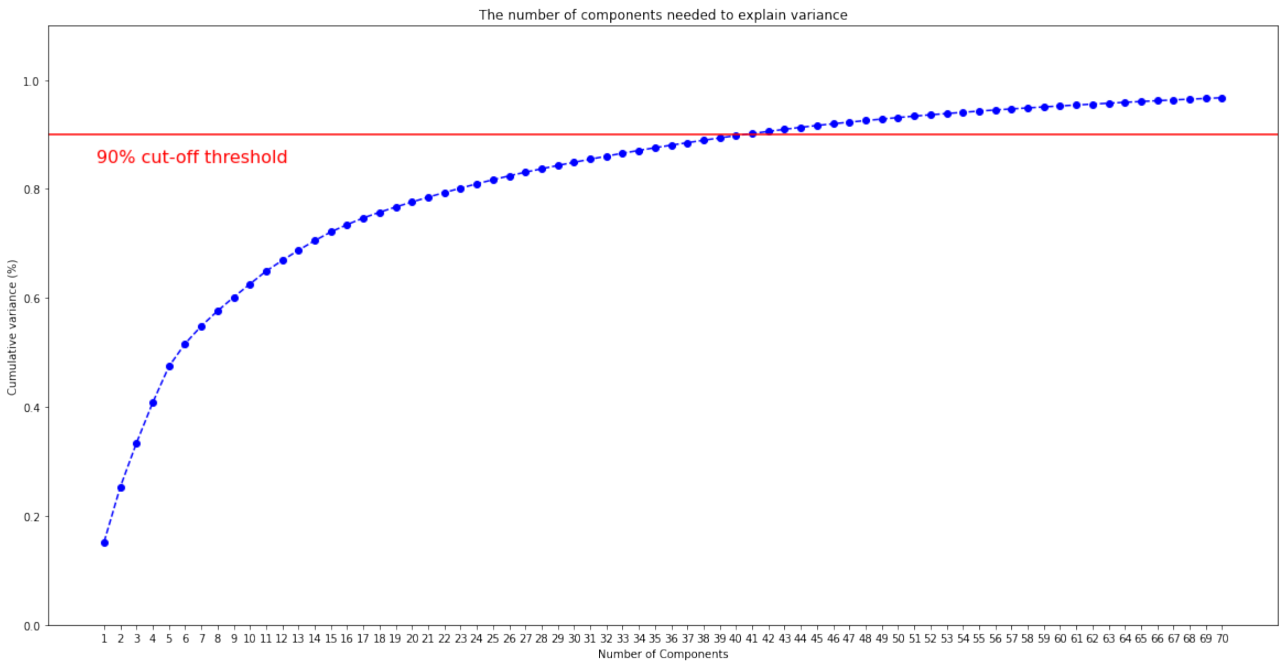


Figure 2: Cumulative explained variance.
This figure shows the cumulative explainability variance of the first 70 components.

I then input the resulting transformation into OPTICS to create groups of options of at least 2 assets each (without overlap). After running different experiments with OPTICS and DBSCAN and $min_s \in \{2, 3, 4, 5, 7, 10\}$. I choose the OPTICS model with $min_s = 2$ since this would be the only one to produce more than 2 groups. See Figure 3.

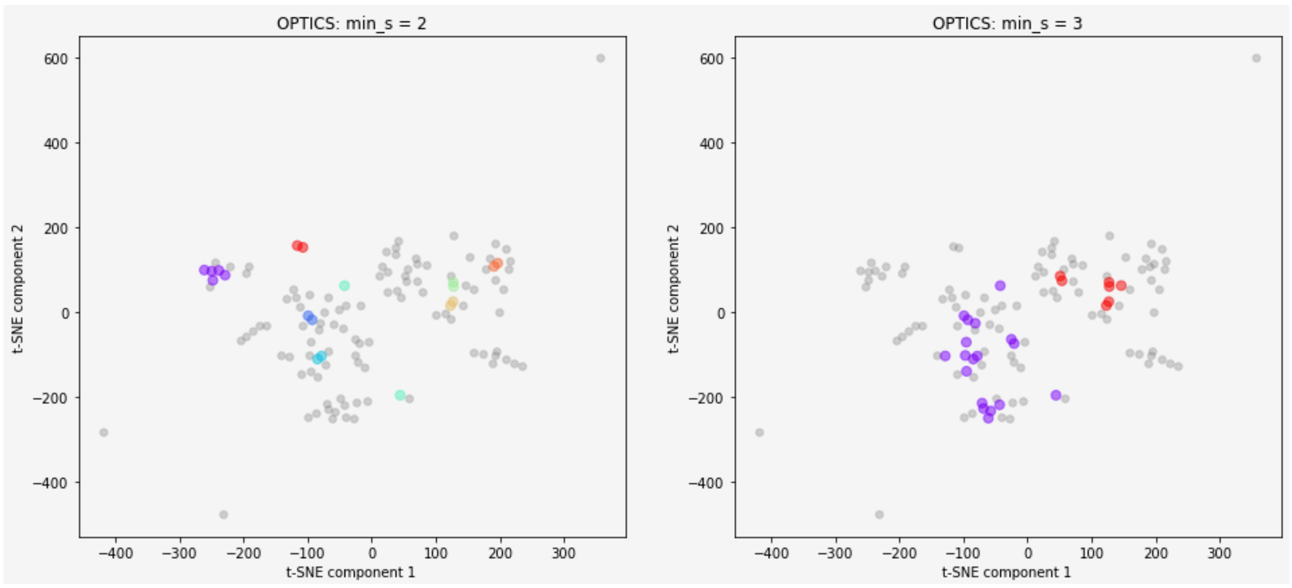


Figure 3: t-SNE projection of the resulting 121 tokens.
On the left is the t-SNE projection of using OPTICS with $min_{size} = 2$. on the right is the same projection but with OPTICS $min_{size} = 3$.

This clustering approach would result in 8 different asset options, I then added a neutral option 0 to simply opt out and hold only cash, therefore resulting in 9 different options with only 1 possible action to take at each time t. Therefore, resulting in $\mathcal{A} = \{0, 1, 2, 3, ..., 8\}$ as the action space.

3.2 Observations

It is worth discussing the observations/images x_t that the environment $\mathcal{E}$ will produce at each time t. First of all, the information contained in the observations is constrained to those found in $\mathcal{A}$ and are listed in Table 12 in the Appendix.

However, since there are two different approaches that use different input data, each deserves its own explanation for clarity.

1. ViT-based model input matrix. This model takes a very simple input matrix x_t^{ViT} which is a snapshot of the daily return of investment δ last T hours from the current time t.

$$x_t^{ViT} = \begin{bmatrix} \delta^1_{t-T} & \delta^2_{t-T} & ... & \delta^A_{t-T} \\ \vdots & \vdots & \vdots & \vdots \\ \delta^1_t & \delta^2_t & ... & \delta^A_t \end{bmatrix} \tag{1}$$

Where $\delta^c_{t'} = (p^c_{t'} - p^c_{t'-1})/p^c_{t'-1}$ is the return of investment at time t' and $p^c_{t'}$ is the close price at hour t'.

2. CNN-based model input matrix. This model takes rather a covariance matrix x_t^{CNN} as input. Using the same notation as for the previous input matrix. The covariance matrix shows the correlation among all different tokens by using data from the last T days from the current time t.

$$x_t^{\text{CNN}} = Cov \left[\begin{bmatrix} \delta^1_{t-T} & \delta^2_{t-T} & ... & \delta^A_{t-T} \\ \vdots & \vdots & \vdots & \vdots \\ \delta^1_t & \delta^2_t & ... & \delta^A_t \end{bmatrix}, \begin{bmatrix} \delta^1_{t-T} & \delta^2_{t-T} & ... & \delta^A_{t-T} \\ \vdots & \vdots & \vdots & \vdots \\ \delta^1_t & \delta^2_t & ... & \delta^A_t \end{bmatrix} \right] \tag{2}$$

3.3 Agent & Environment

The Agent and Environment are an adaptation of the one in [8] but essentially the same, see Figure 4.

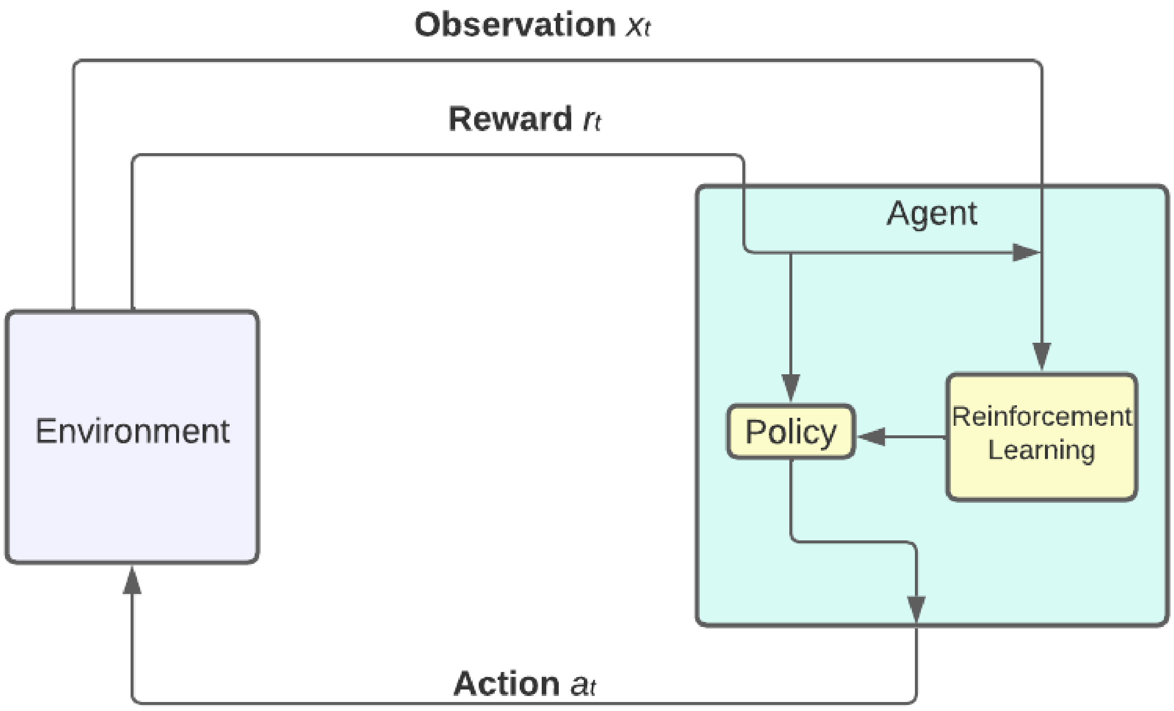


Figure 4: Reinforcement Learning Components
This figure shows the interaction between the Agent and the Environment and where Reinforcement Learning takes place.

In this case, the agent interacts with the Trading Environment $\mathcal{E}$ in a sequence of actions, observations, and rewards. The action a_t represents a series of transactions to perform on the portfolio, $\mathcal{E}$ takes the input and performs the transaction(s) modifying its internal state, shape ratio, and return of investment. It then observes image x_t and a reward r_t. This algorithm relies on the premise that relying solely on the current image is impossible. Instead, we consider a series of equations $s_t = x_1, a_1, x_2, ..., a_{t-1}, x_t$.

It also applies a discount factor γ to each of the rewards r_t. The reward r_t is the Sharpe Ratio defined as follows.

$$r_t = d^{0.5} \frac{\mathbb{E}\left[\zeta_t\right]}{Var\left(\zeta_t\right)} \tag{3}$$

Where d is the number of hours up until time t; $\zeta_t = (\text{cash}_t + \text{assets\_value}_t)/(\text{cash}_{t-1} + \text{assets\_value}_{t-1})$; cash_t is the total amount of cash at time t and asset_value_t is the value of all units held at time t.

3.4 Model Description

3.4.1 The Optimization Problem

The agent's goal is to interact with the trading environment $\mathcal{E}$ and select the best action vector a_t that maximizes future discounted rewards resulting in a discounted return $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t$. Where T is the terminal time step. Therefore, the optimal action-value function is the one that maximizes the expected return defined after seeing some sequence s and taking an action a i.e. $Q^*(s, a) =$

$\max_\pi \mathbb{E}[R_t|s_t = s, a_t = a, \pi]$, where π is a policy mapping sequences (or distributions) to an action vector. In practice, the function Q^* is approximated as $Q(s,a) \approx Q^*(s,a)$.

The original paper makes use of a Convolutional Neural Network (CNN) as an approximator $Q_{\text{CNN}}(s', a') \approx Q^*$, however, the current trend is shifting from using CNN-based models such as Residual Networks to Attention-based models such as Vision Transformers (ViT) which are considered the new state-of-the-art in computer vision machine learning algorithms. [9] Therefore, the main focus of this project is to test a ViT model as approximator $Q_{\text{ViT}}(s', a') \approx Q^*$.

Both models optimize for some parameters θ. The resulting optimization process is then to minimize–by Stochastic Gradient Descent–an approximator of the series of loss functions $\hat{L}_i(\theta)$ that changes after each iteration i.

$$\hat{L}_i(\theta) \approx \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}[(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i))^2] \tag{4}$$

The model is also trained with a greedy strategy. That takes a random action a_t with probability ϵ.

3.4.2 Replay Memory

This paper also follows the same approach in [8] for Reinforcement Learning from experience replay. At each time step, the agent stores the experience $e_t = (s_t, a_t, r_t, s_{t+1})$ in a dataset $\mathcal{D} = e_1, ..., e_N$, and we draw a random mini-batch of experiences e from $\mathcal{D}$ i.e. $e \sim \mathcal{D}$

3.4.3 Vision Transformer

The first approach $Q_{\text{ViT}}(s', a')$ uses attention mechanisms. Hence, it is more similar to the model proposed in [7]. However, the approach taken in [7] was built specifically to tackle the problem of protein folding. Therefore, the embedding and attention mechanism is more similar to the one used in [9] but not the same.

The key difference from [9] is that instead of using squared patches, I use rows as patches. Since the input matrix is a covariance matrix. It makes more sense to build the patches from rows rather than squares.

To handle the 2D covariance matrix. I, therefore, don't need to map x_t matrix into a new matrix, but I do consider it as a series of patches $x_t^{j^T} \in \mathbb{R}^A$ where x_t^j is a row vector at row p.

The rest of the approach is the same as that in [9]. Where the Transformer takes a latent vector size D through all its layers with a learnable linear projection $E \in \mathbb{R}^{A\times D}$ and I also prepend a sequence of learnable patches $z_0^0 = x_{class}$ Eq. 5.

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; ...; x_p^A E] + E_{pos}, E_{poss} \in \mathbb{R}^{(N+1\times D)} \tag{5}$$

In the same way, the Transformer is formed by alternating layers of Multi-headed Self-Attention (MSA) and Multilayer-Perceptron (MLP) blocks with Layer Normalization (LN) before every block and ReLu after every connection. Lastly, each MLP block is formed by two layers with GELU non-linearity Eq. 6 and 7.

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \ell = 1, ..., L \tag{6}$$

$$z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \ell = 1, ..., L \tag{7}$$

Where L is the number of Transformer Encoder layers in the model. Lastly, the embedded patch at the output of the transformer z_L^0 serves as the image representation $\text{LN}(z_L^0)$.

Finally, the output is fed to an MLP head.

3.4.4 Convolutional Neural Network

The second approach is the standard approach for image classification problems. $Q_{\text{CNN}}(s', a')$ is formed by L convolutional neural layers (CNL) blocks of two-layered CNNs with ReLU after each layer and Batch Normalization (BN) after every layer.

The output of the Lth CNL block is fed to a fully connected layer and, lastly, to an MLP head.

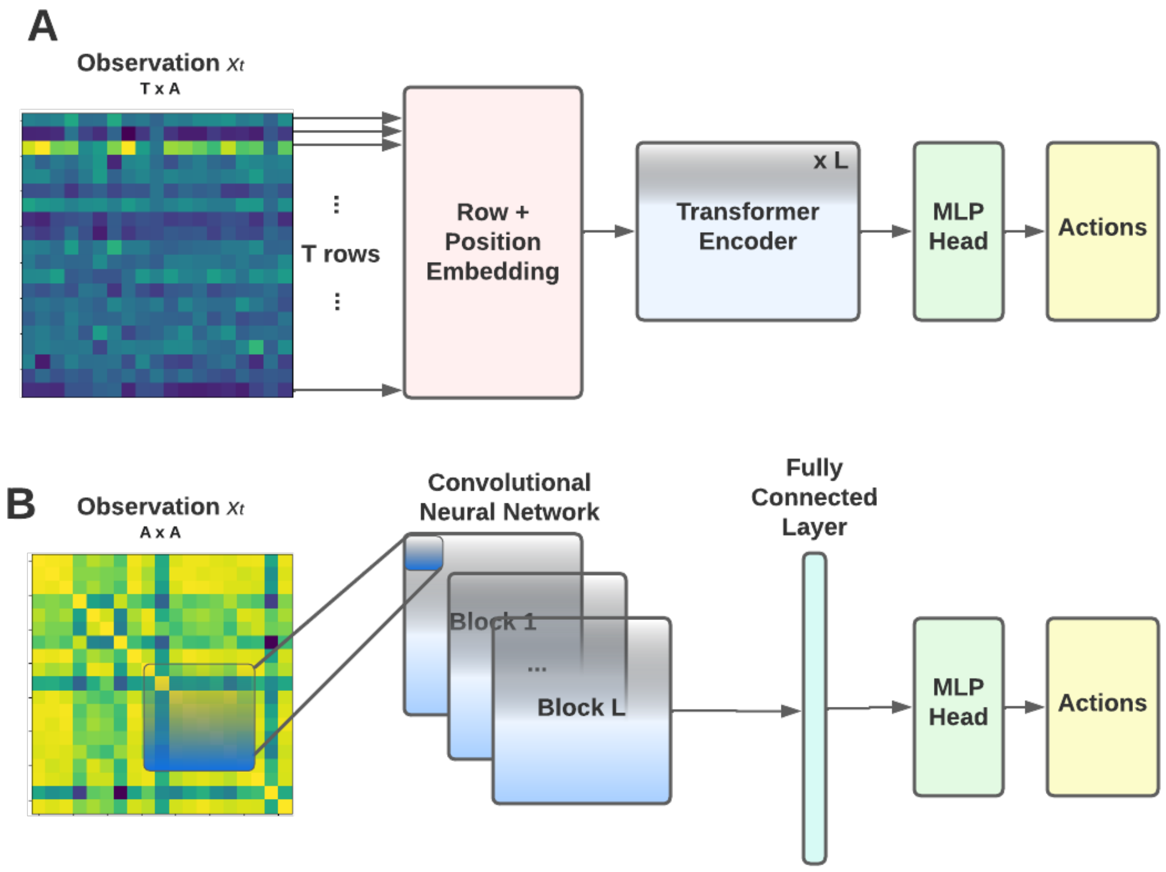


Figure 5: Overview of ViTs and CNNs' architectures side to side.
The figure above shows both approaches for the Q approximation function. **A.** Shows the vision transformer approach. At each timestep t, the input matrix x_t^{ViT} is fed into the positional embedding function. The output of this is then fed to the Transformer encoder. The output of that is then fed into an MLP which then produces the Q values that are then mapped to the trading positions. **B.** Shows the Convolutional Network approach. t each timestep t, the input matrix x_t^{CNN} is fed into the Convolutional Neural Network. That output is then fed into a Network of Linear Layers that produce the Q values that are then mapped to the trading positions.

3.5 Constrains

The environment was set with the following constraints: initial cash $= 100,000$ USD, a constant priority fee of 2 gwei, and a constant gas limit of 21000 gwei.

The agent's replay memory's max size was set to $10,000$ and the reward discount rate to $\gamma = 0.8$.

3.6 Optimization

The models are trained with stochastic gradient descent and backpropagation.

The hyperparameters were train following a batch size $= 128$, epsilon $e = 0.01$, decay rate $= 0.9$, minimum epsilon $= 0.0001$, lookback window $T = 10$.

3.7 Alternatives Studied

Even though there are two different approaches to the training of the ViT and CNN models given their input matrices. I still felt that it was better to do a direct comparison using x^{ViT} and x^{CNN} to train both models and then compare how using ViT compares to CNN when taking x^{ViT} and the same when taking x^{CNN}.

It is necessary to clarify that since x_t^{ViT} and x^{CNN_t} have different input sizes, the ViT model taking x_t^{ViT} is different from the one taking x^{CNN_t} as input. The same is true of the CNN model, the one taking x_t^{ViT} is different from the one taking x^{CNN_t}.

Therefore, to avoid confusion, I show in Table 1 which has all the information about all the different approaches including their models and other specifications used in this study.

Table 1: Reinforcement Learning Approaches used in this Study

Approach	**Model**	**Input Matrix**	**Input Size**	T
A	ViT	x_t^{ViT}	24x19	24
B	CNN	x_t^{CNN}	19X19	24
C	ViT	x_t^{CNN}	19X19	24
D	CNN	x_t^{ViT}	24X19	24

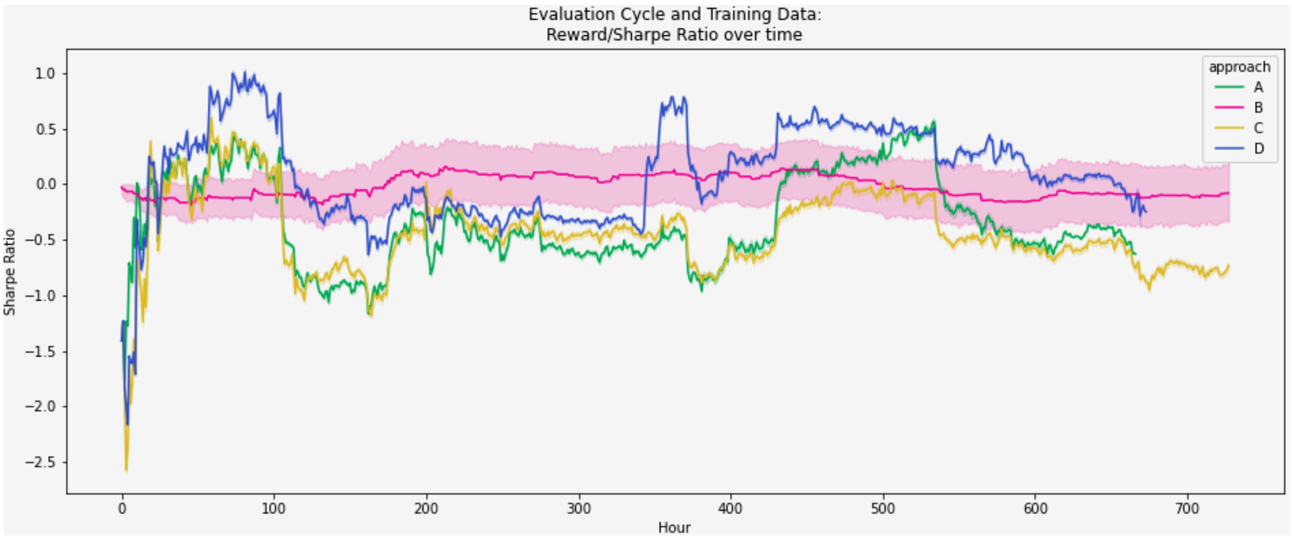
All four CNN and ViT architectures used in this study are described in tables 2 and 3.

Table 2: CNN architectures (Approaches B and D)

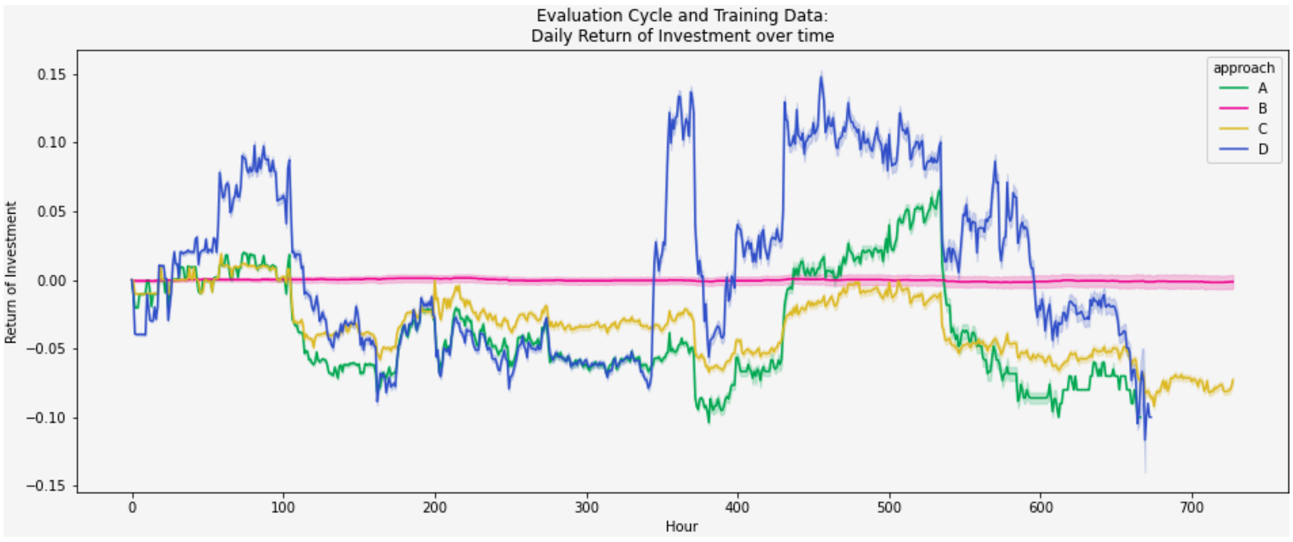
Block	**Layer**	**Output Size (App. B)**	**Output Size (App. D)**	**Kernel**	**Activation**
Dense Block 1	Conv	19x19x8	24x19x8	1x1	ReLU
	Conv	17x17x8	22x17x8	3x3	ReLU
	Max Pool	15x15x8	20x15x8	3x3	-
Dense Block 2	Conv	15x15x16	20x15x16	1x1	ReLU
	Conv	13x13x16	18x13x16	3x3	ReLU
	Max Pool	11x11x16	16x11x16	3x3	-
Dense Block 3	Conv	16x11x32	12x11x32	1x1	ReLU
	Conv	9x9x32	14x9x32	3x3	ReLU
	Max Pool	7x7x32	12x7x32	3x3	-
MLP	Linear	128	128	-	-
	Linear	64	64	-	-
	Linear	9	9	-	Softmax

Table 3: ViT architectures (Approaches A and C)

Module	**Output (App. A)**	**Output (App. C)**	**Layers** L	**Latent Vector Size**	**Heads**
Embedding	11x512	20x512	-	-	-
Transformer	512	512	4	512	8
MLP	9	9	-	-	-



(a) Aggregated Sharpe Ratio across trading hours.



(b) Aggregated Daily Return of Investment across trading hours.

Figure 6: Training: Aggregated metrics of performance for ViT and CNN models.

4 Results

Figures 6a and 6b show the performance of all four algorithms on each approach as indicated in 1 across a trading window of 731 hours starting from 2022-08-21 at 23:00:00.

4.1 Training Data Results

The first thing to notice is the performance of approach D (red line) which is significantly higher than the rest in terms of Sharpe ratio and daily return of investment. Table 4, shows the average Sharpe ratio over all episodes across the trading window for all four approaches. It can be observed that there is a big difference between taking approach D and taking any other approach. Yet, we can also appreciate that it doesn't reach the end of the training period, this is because of the way the environment is set, it will automatically stop if the sum of the units held value and the amount of cash held is smaller than or equal to 90% the initial cash (More on the algorithm in Appendix, Algorithm 1).

Even though approach D still produces overall negative results and takes a somewhat more volatile strategy, it still produces a much more consistent strategy by looking at its 95% confidence interval in Figure 6a.

This is followed then by approach A, which is the only other approach that appears to be slightly superior to approaches B and C. Both, approaches A and D take x^{ViT} as input, which was initially thought to only work for the ViT model. It may suggest that the input data is making the main difference here. Since they are receiving the exact same type of input as seen in Table 1, this is something worth noticing.

Table 4: Training data. Average ROI and Sharpe Ratio by Approach.

Approach	**Avg. Daily ROI**	**Avg. Sharpe**
A	-0.0369	-0.4214
B	-0.0003	-0.0180
C	-0.0325	-0.4357
D	0.0095	0.0919

Table 5: Training data. Standard Deviation ROI and Sharpe Ratio by Approach.

Approach	**Std. Daily ROI**	**Std. Sharpe**	**Var. Daily ROI**	**Var. Sharpe**
A	0.0309	0.4003	0.0010	0.1602
B	0.0119	0.9073	0.0001	0.8231
C	0.0244	0.3801	0.0006	0.1445
D	0.0621	0.4482	0.0039	0.2009

We can observe some tendencies by looking at Table 4. At first sight, the Average Daily ROI is higher for Approaches D and B and lowest for A and C, similar to their Average Sharpe Ratio. While we cannot run a t-test because the results do not have the same number of datapoints. In Table 5 we can look at the variance and standard deviation of all four approaches during the training process. Interestingly, approaches A and C which both use ViT models produce results with a more similar Variance. Hence, it is possible to run a t-test on them but a Welch's t-test (for different variances) is also possible between approaches B and D. As a comparison point, it would also be interesting to also run t-tests on approaches that used the same input matrices, namely A and D and B and C.

This also may seem to suggest that despite the observation made in Figure 6, the Sharpe Ratio and daily ROI distribution of approaches A and C are more similar than initially thought. Also, as suggested earlier, the Variance of the ROI and Sharpe Ratio for the results of Approach D are higher than those of Approaches A and C. Furthermore, despite Approach B taking a very conservative approach, it still results in a very high variance for the Sharpe Ratio.

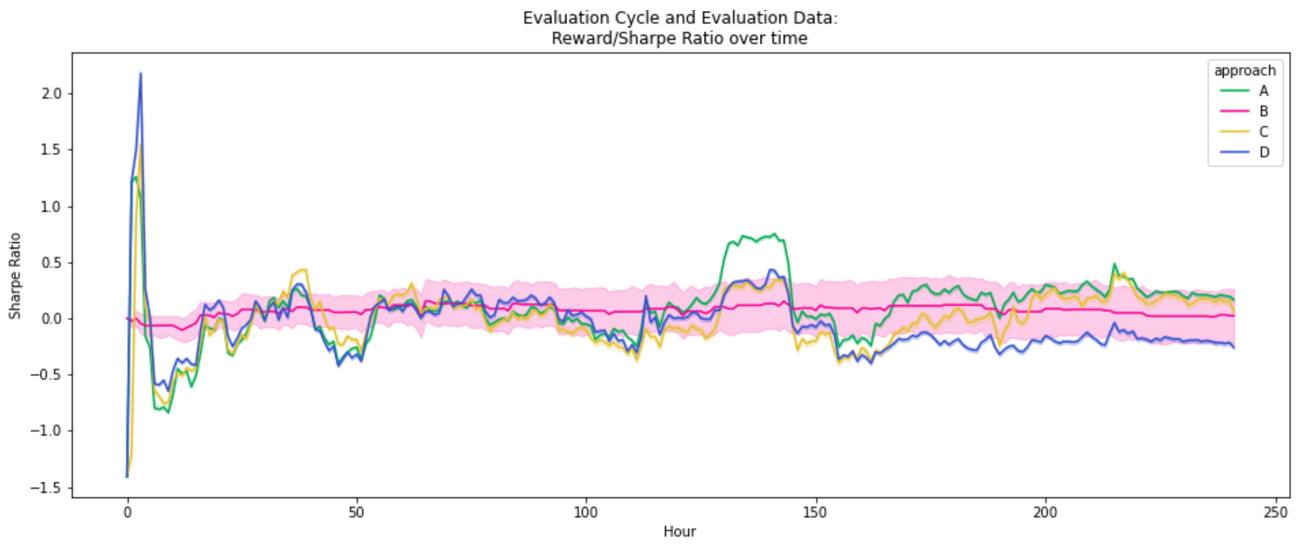
This suggests that despite the radically different strategies followed by the models in approaches B and D. The results seem to indicate that ViTs do take more steady strategies than CNNs in terms of the Sharpe Ratio.

4.2 Evaluation Data Results

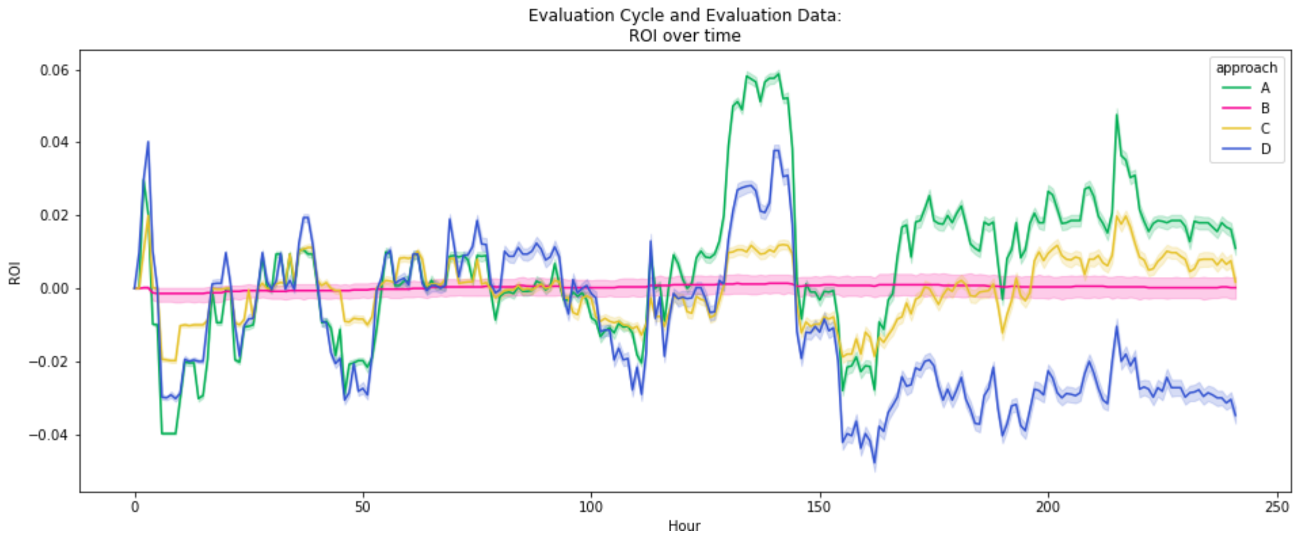
While the training results may indicate that Approach D would yield the best results. The truth is that Evaluation results show something different.

Contrary to the results found in the Training Results section, Figures 7a and 7b show that the best-performing algorithm is Approach A which uses a ViT architecture and takes a snapshot of daily returns as input x^{ViT_t}.

Approach D on the other hand seems to have chosen a riskier strategy than the rest, which, while performing well at the beginning of the training period, dropped significantly at the end of it and the



(a) Evaluation Data. Aggregated Sharpe Ratio across trading hours.



(b) Evaluation Data. Aggregated ROI across trading hours.

Figure 7: Evaluation: Aggregated metrics of performance for ViT and CNN models.

evaluation period as well. Contrary to approach A which choose a more steady option and, while not performing extremely well during the training period, it performed much better during the evaluation period.

Table 6: Evaluation data. Average ROI and Sharpe Ratio by Approach

Approach	**Avg. Daily ROI**	**Avg. Sharpe**
A	0.0060	0.0892
B	0.0003	0.0717
C	-0.0003	-0.0061
D	-0.0110	-0.0583

Table 7: Evaluation data. Standard Deviation ROI and Sharpe Ratio by Approach

Approach	**Std. Daily ROI**	**Std. Sharpe**	**Var. Daily ROI**	**Var. Sharpe**
A	0.0199	0.3132	0.0004	0.0981
B	0.0091	0.7418	0.0001	0.5502
C	0.0097	0.2933	0.0001	0.0860
D	0.0201	0.3025	0.0004	0.0915

The results show, however, that while the average daily ROI and Sharpe ratio are significantly higher in Approach B, the volatility is also higher compared to the rest of the Approaches, which implies that is a higher-risk strategy.

Then, similar to the results obtained in the Training Results section, models A and C–which both use the ViT architecture–are much more stable throughout time–their volatility is not far from that of Approach D (which is the lowest of all)–but is also very similar to each other 0.3348 and 0.3700 for A and C respectively.

From here, it is possible to test Approaches A and C using a t-Test, while the rest need to be compared using Welch's t-Test for distribution with different variances. Such results are shown in Table 8.

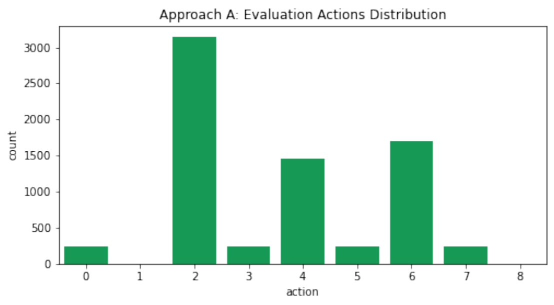
Table 8: Evaluation Data. T-test results.

Test	**Similarity**	**t-value**	**p-value**
$\mathbb{E}[r_t^{\text{Approach A}}] = \mathbb{E}[r_t^{\text{Approach C}}]$	ViT model	24.43	0.0
$\mathbb{E}[r_t^{\text{Approach B}}] = \mathbb{E}[r_t^{\text{Approach D}}]$	CNN model	17.85	0.0
$\mathbb{E}[r_t^{\text{Approach A}}] = \mathbb{E}[r_t^{\text{Approach D}}]$	input x^{ViT_t}	37.26	0.0
$\mathbb{E}[r_t^{\text{Approach B}}] = \mathbb{E}[r_t^{\text{Approach C}}]$	input x^{CNN_t}	10.72	0.0

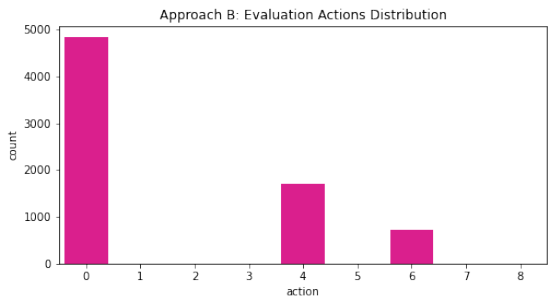
4.3 Actions

To have a better picture of why the models perform the way they do is better to have a look at the actions they take. Since the number of possible actions is very narrow (only 9), this is feasible and useful to understand the performance of the models, such is shown in Figures 8 and 9.

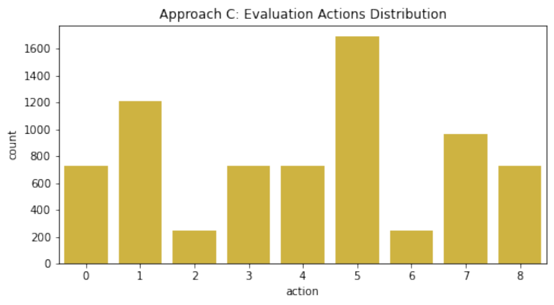
Remembering that each action maps to a different option from 1 to 8, except for 0 which maps to holding. It is worth noticing that three out of four approaches have a near-uniform distribution for



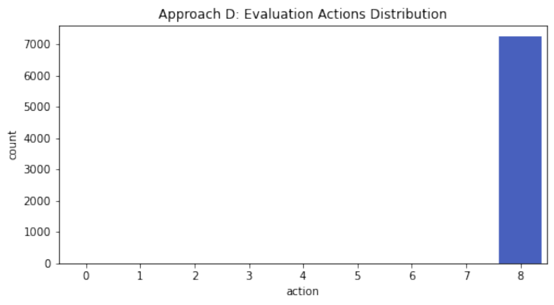
(a) Actions taken in training data using Approach A.



(b) Actions taken in training data using Approach B.



(c) Actions taken in training data using Approach C.

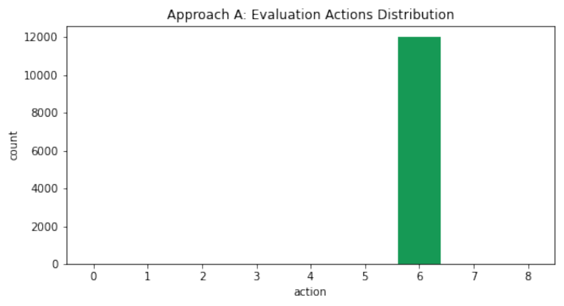


(d) Actions taken in training data using Approach D.

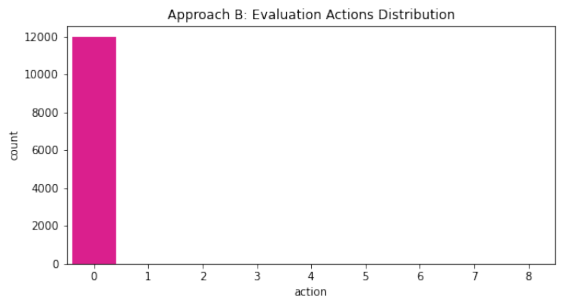
Figure 8: Training Data: Actions distribution across all four approaches.

a handful of their options. Instead of trying to focus on a single or few options, they seem to rather ignore just a few of all available options.

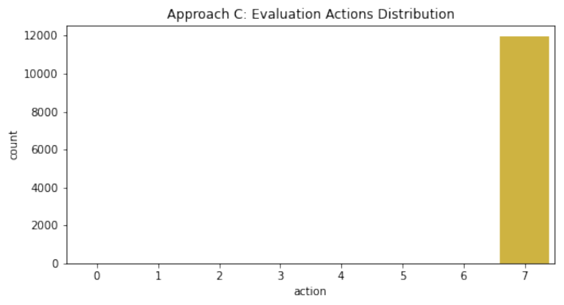
In Approach A (Figures 8a and 9a), the ViT model seems to have narrowed its options to mainly three (options 2, 4, and 6) in the Training data, however, for the evaluation data it opted for option 6 only. The CNN model in Approach B (Figures 8b and 9b) is seemingly opting to rather play safe and stay neutral in both the training and the evaluating data. Then the ViT model in Approach C (Figures 8c and 9c) seems to have a more even distribution over the training data, however, it converges to option 7 in the evaluation data. Lastly, in Approach D (Figures 8d and 9d) seems to have reached a local minimum very early since it converged to taking option 8 in both the training and evaluation data.



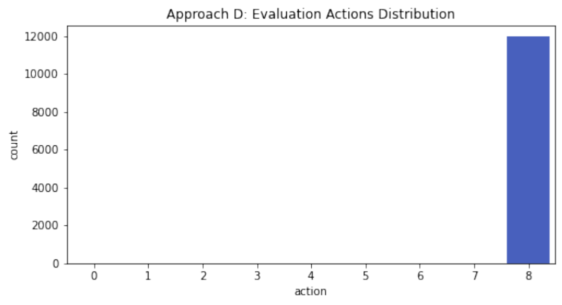
(a) Actions taken in training data using Approach A.



(b) Actions taken in training data using Approach B.



(c) Actions taken in training data using Approach C.



(d) Actions taken in training data using Approach D.

Figure 9: Eval. Data: Actions distribution across all four approaches.

5 Discussion

5.1 Using ViT vs CNN

Given the results shown in Tables 4 and 6, it is not quite clear which strategy is superior, since Approach D shows to be superior to the rest of the training data, however, in the evaluation data, it performs the worst out of all models. However, by looking at the actions taken and the performance in Figure 6, it seems to be taking a high-risk strategy that has relatively high volatility but also grows faster than other options at given times. On the other extreme, we find Approach B, which chooses to have a very conservative strategy, it mainly chooses to stay neutral and therefore it does not lose nor earn a lot of money. Interestingly, both approaches used a CNN-based architecture. It might be worth investigating why exactly are they choosing very different decisions.

On the other side, we have both ViT-based architectures. Different from the results shown for Approaches B and D. Approaches A and C both have a very similar Average Daily ROI and Sharpe Ratio in both datasets. Their Variances and Standard Deviations are very similar to each other as well. This might indicate that the options that they choose are very similar to each other, in this case, actions 6 and 7. Figures 7b and 7a show that both options show similar behavior when computing their daily ROI and Sharpe Ratio.

To unveil why similar architectures choose different approaches, we need to look at the input data.

5.2 Input: Raw Data vs Covariance

In the previous subsection, we already found that the CNN model is either taking very conservative or very risky decisions, however, the difference lies in their input. The CNN model in Approach D takes x_t^{ViT} which is considered the raw data because it doesn't have any other transformation or treatment. Because the model doesn't have direct information about the correlations between different assets with each other, it is likely missing some information that would make it take more safe decisions. Furthermore, it can be observed in Figures 6 that this option doesn't make it to the end of the training trading period since the Environment is set to restart the game if the total value of units plus cash is below 90% of the initial amount of cash.

Then we have the case where the ViT models both converged towards choosing similar strategies. This is an interesting observation since one (Approach A) received just the raw data x_t^{ViT} while the other one received the covariance matrix x_t^{CNN}. Then, when comparing both approaches, one can observe that Approach C is slightly better in the training dataset since it is able to reach the end of the trading window. However, in the evaluation data, the ViT model used in Approach A is performing slightly better than the rest. This result seems to be contradictory but it could also just be because the threshold to stop the game was too high.

6 Conclusion and Future Work

6.1 The Advantage of Attention

Attention mechanisms were a huge revolution in the AI community, however, not as many papers have made use of it in Q-Learning. This work shows that Transformers (the building blocks of ViT) can manage assets with middle-risk and more stability than CNNs which tend to either choose one extreme or the other.

While the results still don't show a positive net return of investment in both the train and evaluation data, the results are very promising to explore similar options in the future.

6.2 Future Work

This project had its limitations that need to be mentioned and could be improved in future studies. These are the following:

- **Limited Randomness:** The environment could be improved to avoid overfitting by rearranging the assets at random every time.
- **More Information in the Observation x_t:** The environment might be benefited by introducing more information in the observation such as the current options held. Or information about other tokens in the market or other metrics such as the Open Price.
- **Model Complexity:** There were some hardware limitations that made it unfeasible to introduce more complex (larger) models.
- **Attention-based Models:** ViT is a solution designed for images and not observing prices. There might be other architectures better suited for this task.
- **Replay Memory:** The learning process could be improved by introducing a prioritized replay memory.
- **Modeling Transaction Fees:** The current model assumes that gas prices follow a normal distribution while in reality gas prices are much more volatile. Being able to model gas prices in a more precise way can result in better performance and strategies.

6.3 Conclusion

This study shows the capabilities of Attention-based Q-approximators for Q-learning. ViT models show a clear superiority when tested on the evaluation dataset since they produce a higher Sharpe Ratio and Daily ROI than CNN-based models. There is room for improvement in the environment and the model to achieve an efficient and effective asset management algorithm via Reinforcement Learning such as using more indicators, introducing more layers (core complexity) on each model, adapting transformer architectures, using prioritized replay memory, testing more diverse observations to learn more complex patterns and avoid overfitting, and modeling Transaction costs.

References

[1] L. W. Cong, K. Tang, J. Wang, and Y. Zhang, "Alphaportfolio: Direct construction through deep reinforcement learning and interpretable ai," *Available at SSRN*, vol. 3554486, 2021.

[2] Z. Wang, B. Huang, S. Tu, K. Zhang, and L. Xu, "Deeptrader: a deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 643–650, 2021.

[3] S. M. Sarmento and N. Horta, "Enhancing a pairs trading strategy with the application of machine learning," *Expert Systems with Applications*, vol. 158, p. 113490, 2020.

[4] J.P.Morgan, "Bitcoin, blockchain and digital finance: Fintech goes mainstream in the covid-19 era," Apr 2021.

[5] J. Street, "What we do jane street," 2023.

[6] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland, *et al.*, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[7] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[10] E. F. Fama, "Two pillars of asset pricing," *American Economic Review*, vol. 104, no. 6, pp. 1467–85, 2014.

[11] E. F. Fama, "The behavior of stock-market prices," *The journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.

[12] E. F. Fama, "Random walks in stock market prices," *Financial analysts journal*, vol. 51, no. 1, pp. 75–80, 1995.

[13] E. F. Fama and K. R. French, "The cross-section of expected stock returns," *the Journal of Finance*, vol. 47, no. 2, pp. 427–465, 1992.

[14] E. F. Fama and K. R. French, "A five-factor asset pricing model," *Journal of financial economics*, vol. 116, no. 1, pp. 1–22, 2015.

[15] coingecko.com, "Cryptocurrency prices, charts, and crypto market cap," Jan 2023.

[16] P. Chaim and M. P. Laurini, "Volatility and return jumps in bitcoin," *Economics Letters*, vol. 173, pp. 158–163, 2018.

[17] A. H. Dyhrberg, "Bitcoin, gold and the dollar–a garch volatility analysis," *Finance Research Letters*, vol. 16, pp. 85–92, 2016.

[18] J. Chu, Y. Zhang, and S. Chan, "The adaptive market hypothesis in the high frequency cryptocurrency market," *International Review of Financial Analysis*, vol. 64, pp. 221–231, 2019.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, p. 84–90, May 2012.

[20] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[21] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[22] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz, *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[23] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[25] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[26] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

[27] Tesla, "Tesla ai day 2022," Oct 2022.

[28] A. Fridman, "Andrej karpathy: Tesla ai, self-driving, optimus, aliens, and agi | lex fridman podcast #333," Oct 2022.

[29] G. Jeong and H. Y. Kim, "Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning," *Expert Systems with Applications*, vol. 117, pp. 125–138, 2019.

[30] S. Carta, A. Ferreira, A. S. Podda, D. R. Recupero, and A. Sanna, "Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting," *Expert systems with applications*, vol. 164, p. 113820, 2021.

[31] T. Théate and D. Ernst, "An application of deep reinforcement learning to algorithmic trading," *Expert Systems with Applications*, vol. 173, p. 114632, 2021.

[32] H. Park, M. K. Sim, and D. G. Choi, "An intelligent financial portfolio trading strategy using deep q-learning," *Expert Systems with Applications*, vol. 158, p. 113573, 2020.

[33] J. B. Chakole, M. S. Kolhe, G. D. Mahapurush, A. Yadav, and M. P. Kurhekar, "A q-learning agent for automated trading in equity stock markets," *Expert Systems with Applications*, vol. 163, p. 113761, 2021.

Appendix

Algorithm 1 Deep Q-Network Algorithm

Input: Environment, Hyperparameters
Output: Trained Deep Q-Network

```
 1: Initialize replay memory $\mathcal{D}$ with capacity $10,000$
 2: Initialize action-value function $Q$ with random weights $\theta$
 3: Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
 4: Initialize state $s_1$ and preprocessed history $x_1$
 5: for episode $= 1$ to $M$ do
 6:     Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
 7:     for timestep $= 1$ to $T$ do
 8:         With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
 9:         Execute action $a_t$ in the environment and observe reward $r_t$ and next state $s_{t+1}$
10:         Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
11:         Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
12:         Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
13:         Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to $\theta$
14:         Every $C$ steps reset $\hat{Q} = Q$
15:     end for
16: end for
```

Table 9: All Available Cryptocurrencies.

CODE	Name
ADA	Cardano
AION	Aion
ALGO	Algorand
ARDR	Ardor
ATOM	Cosmos
AVAX	Avalanche
BCC	BitConnect
BCH	Bitcoin Cash
BNB	Binance Coin
BTC	Bitcoin
BTG	Bitcoin Gold
BTS	BitShares
DASH	Dash
DCR	Decred
DGB	DigiByte
DOGE	Dogecoin
DOT	Polkadot
EGLD	Elrond
EOS	EOS
ETC	Ethereum Classic
ETH	Ethereum
FIL	Filecoin
GXS	GXChain
HBAR	Hedera Hashgraph
ICX	ICON
IOST	IOST
IOTA	IOTA
IOTX	IoTeX
KLAY	Klaytn
KMD	Komodo
KSM	Kusama
LSK	Lisk
LTC	Litecoin
MATIC	Polygon
NANO	Nano
NEO	NEO
NULS	Nuls
ONT	Ontology
QTUM	Qtum
SC	Siacoin
SOL	Solana
STEEM	Steem
STRAT	Stratis
STX	Stacks
SYS	Syscoin
TRX	TRON
VET	VeChain
WAN	Wanchain
WAVES	Waves
WTC	Waltonchain
XEM	NEM
XLM	Stellar
XMR	Monero
XRP	Ripple
XTZ	Tezos
XVG	Verge
XZC	Zcoin
ZEC	Zcash
ZEN	Horizen
ZIL	Zilliqa

Table 10: All Available Tokens.

CODE	Name
AAVE	Aave
ADX	AdEx
AMP	Amp
ANC	Anchor Protocol
ANT	Aragon
ATM	ATMChain
BAND	Band Protocol
BAT	Basic Attention Token
BLZ	Bluzelle
BNT	Bancor Network Token
BTT	BitTorrent
CAKE	PancakeSwap
COMP	Compound
CTXC	Cortex
CVC	Civic
DAR	Dare
DATA	Streamr
DENT	Dent
DNT	district0x
ELF	Aelf
ENJ	Enjin Coin
FTT	FTX Token
FUN	FunFair
GNO	Gnosis
GRT	The Graph
GTC	Game.com
GTO	Gifto
KNC	Kyber Network
LEND	Aave
LINK	Chainlink
LRC	Loopring
LUNA	Terra
MANA	Decentraland
MCO	Crypto.com
MITH	Mithril
MKR	Maker
MLN	Enzyme
NMR	Numeraire
NPXS	Pundi X
OMG	OMG Network
ORN	Orion Protocol
POLY	Polygon
POWR	Power Ledger
QUICK	Quickswap
REP	Augur
REQ	Request Network
RLC	iExecRLC
RUNE	THORChain
SHIB	Shiba Inu
SNX	Synthetix
STORJ	Storj
STORM	StormX
THETA	Theta Network
TRIBE	Tribe
UNI	Uniswap
UTK	Utrust
ZRX	0x

Table 11: All Available Stablecoins.

CODE	Name
NBT	NuBits
BUSD	Binance USD
TUSD	TrueUSD
UST	TerraUSD

Table 12: Digital Assets Used.

CODE	Name	Type
AAVE	Aave	Token
AVAX	Avalanche	Cryptocurrency
BAT	Basic Attention Token	Token
BNB	Binance Coin	Cryptocurrency
BTC	Bitcoin	Cryptocurrency
BTG	Bitcoin Gold	Cryptocurrency
CAKE	PancakeSwap	Token
ETC	Ethereum Classic	Cryptocurrency
FTT	FTX Token	Token
FUN	FunFair	Token
MANA	Decentraland	Token
MATIC	Polygon	Cryptocurrency
NEO	NEO	Cryptocurrency
ONT	Ontology	Cryptocurrency
QTUM	Qtum	Cryptocurrency
RUNE	THORChain	Token
SOL	Solana	Cryptocurrency
TRX	TRON	Cryptocurrency
VET	VeChain	Cryptocurrency