XCS229ii Literature Review

Ammar Husain

Below are four papers I read and summarized. Please also find how they compare and contrast with each other along with possible extensions of their work.

FinRL: Deep Reinforcement Learning Framework to Automate Trading in Quantitative Finance

This paper introduces a new library for deep reinforcement learning algorithms in the domain of quantitative finance. Algorithmic trading is essentially making dynamic decisions in a highly stochastic and complex financial market. In several other fields, deep neural networks and specifically reinforcement learning techniques have shown promising results in estimating and thereby maximizing the expected reward of taking a certain action in a certain state.

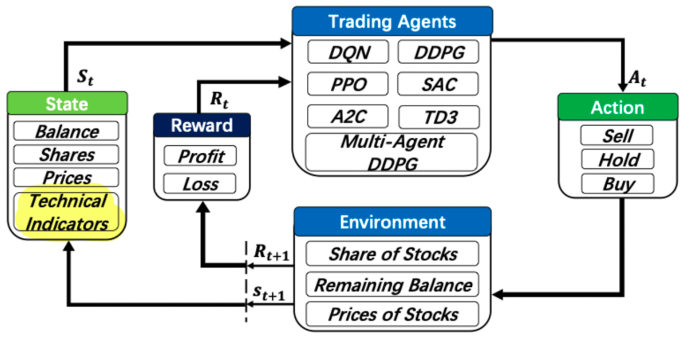


Figure 1: Overview of automated trading in FinRL, using deep reinforcement learning.

Similarly for quantitative finance the state would be the agent state such as the account balance, number of shares held for a certain security etc and the market state such as OHCLV (Open, High, Low, Close & Volume), Technical Indicators, NLP market sentiment features etc. The action involves a simple [-1,0,1] for a certain security where -1 is sell, 0 is hold and 1 is buy. This action space could be extended to [-k....+k] for multi stock trading applications. The reward function could be as simple as the change in portfolio value or more complex measure such as the Shape ratio. The DRL trading agent operates in an environment that simulates the financial market as DOW-30, NASDAQ-100 etc.

The main contribution of this paper is in creating a modular three-layer framework as shown in the figure below.

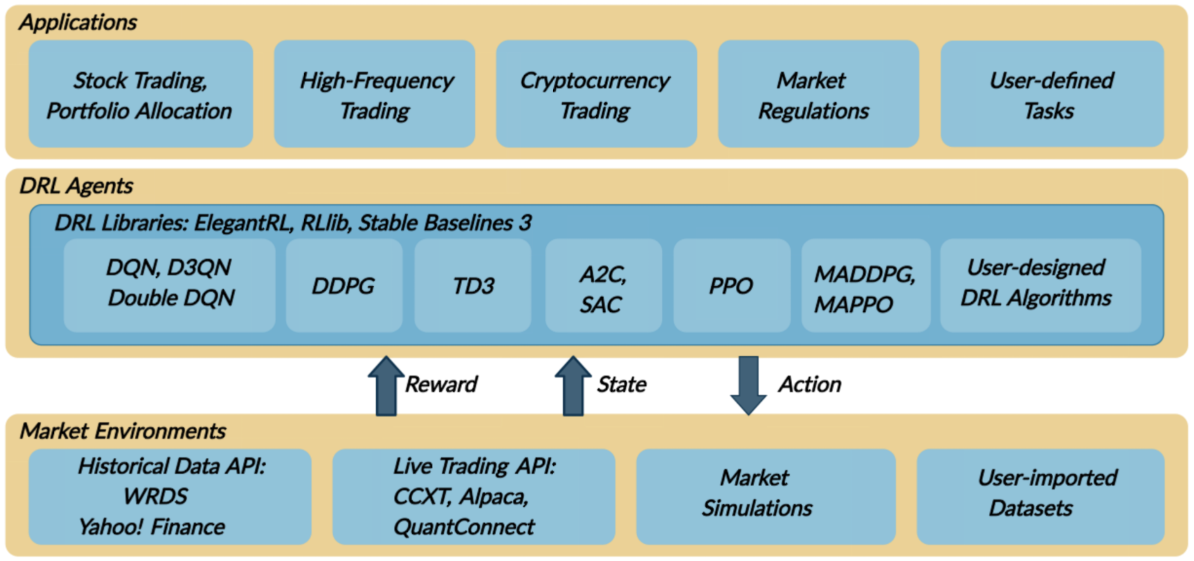


Figure 2: Overview of FinRL: application layer at the top, agent layer in the middle and environment layer at the bottom.

The bottom layer is an environment layer that simulates financial markets using actual historical data, such as closing price, shares, trading volume, and technical indicators. The middle layer is the agent layer that implements fine-tuned DRL algorithms and common reward functions. These have in turn been imported from three different libraries Stable Baselines 3, RLLib, ElegantRL. The agent interacts with the environment through properly defined reward functions on the state space and action space. The top layer includes several applications in automated trading. The RL training process involves observing price change, taking an action and calculating a reward. By interacting with the environment, the agent updates iteratively and eventually obtains a trading strategy to maximize the expected return.

Another contribution of this paper is that it creates standardized OpenAI gym-style environments using both historical market data and live trading APIs. This enables training the RL agent to train on historical data and seamlessly translate the policy for real time trading. This is achieved by implementing preprocessors that utilize standard APIs to download data and create their corresponding Pandas DataFrame. The bottom layer of the library also implements a step() like functionality with a (state, action) input and returns a (next_state, reward) output.

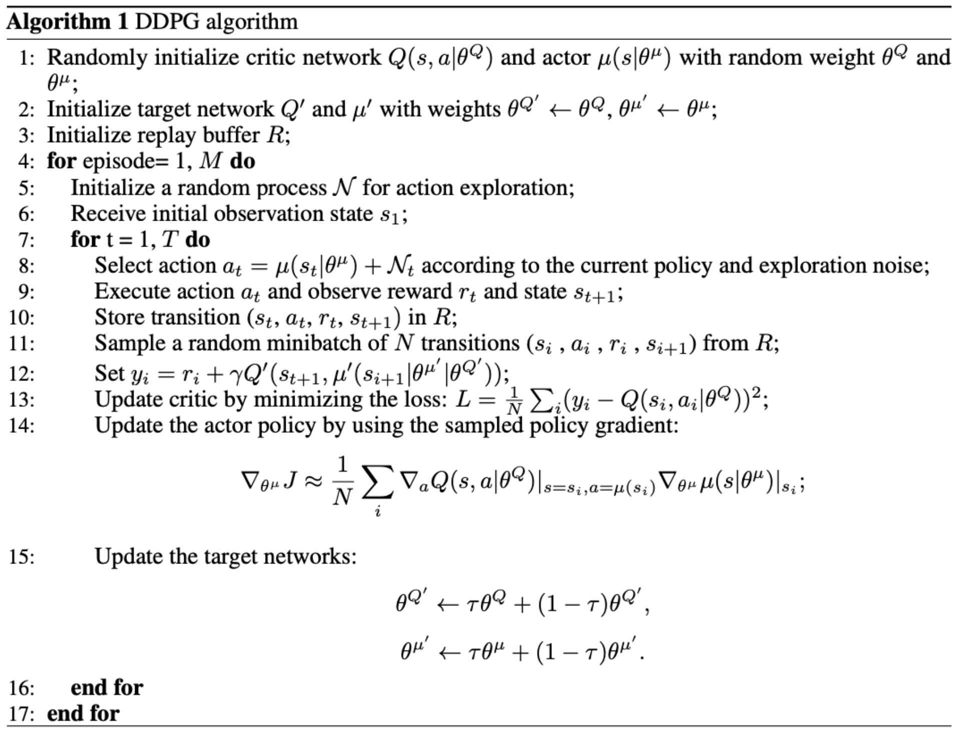
Given that this paper (and the corresponding) library introduces a modular framework for applying DRL to algorithmic trading, some logical next steps would be to add more datasets as well as algorithms that can be used in a plug and play manner. Example datasets would be ones that include earnings call statements, transcripts, historical news sources etc. This would facilitate further research in incorporating NLP techniques such as sentiment analysis to expand the feature set of the existing market state spaces.

Practical Deep Reinforcement Learning Approach for Stock Trading

This paper is a precursor to the FinRL library paper described above. It explores the potential of deep reinforcement learning to optimize stock trading strategy and thus maximize investment return. Given the extremely stochastic and complex nature of financial markets and the various technical indicators that must be incorporated in valuing a particular stock in order to make a trading decision it is challenging for analysts to take all relevant factors into consideration.
Motivated by these challenges, the primary hypothesis of this paper is that reinforcement learning algorithms are better suited to handle such a sequential decision making process. The Deep Deterministic Policy Gradient (DDPG) algorithms is implemented to find the best trading strategy. This algorithm consists of three key components: (i) actor-critic framework that models large state and action spaces; (ii) target network that stabilizes the training process; (iii) experience replay that removes the correlation between samples and increases the usage of data. The authors choose this policy gradient based algorithm over a value based such as Deep Q-network (DQN) due to the large size of the action spaces required in optimal portfolio allocation.
The key contribution of this paper is the formulation of the MDP such that it may effectively be solved using DDPG. The authors chose the following formulation:

- State s: [p, h, b] where p includes the prices of D stocks in the market, h is the amount of holdings in the portfolio and b is the cash balance.
- Action a: [-1,0,1] on all D stocks in the market where -1 is sell, 0 is hold, and 1 is buy.
- Reward r(s,a,s’): sum of equities in all the held stocks as well as the balance.
- Policy is essentially the probability distribution of a at state s.
- Action-value function Q(s,a): expected & estimated reward achieved by action a at state s by following the learned policy.

The algorithm implementation itself is a very standard one as shown in the figure below.

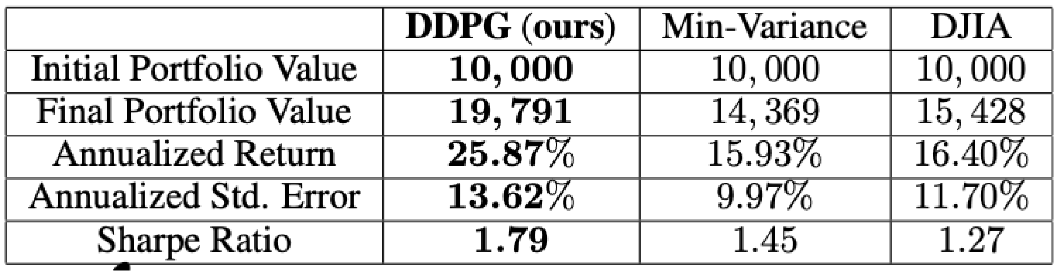


Algorithm 1 DDPG algorithm

```
1: Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with random weight $\theta^Q$ and $\theta^\mu$;
2: Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$;
3: Initialize replay buffer $R$;
4: for episode= 1, $M$ do
5:    Initialize a random process $\mathcal{N}$ for action exploration;
6:    Receive initial observation state $s_1$;
7:    for t = 1, $T$ do
8:       Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise;
9:       Execute action $a_t$ and observe reward $r_t$ and state $s_{t+1}$;
10:      Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$;
11:      Sample a random minibatch of $N$ transitions $(s_i , a_i , r_i , s_{i+1})$ from $R$;
12:      Set $y_i = r_i + \gamma Q'(s_{t+1}, \mu'(s_{i+1}|\theta^{\mu'}|\theta^{Q'}))$;
13:      Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i(y_i - Q(s_i, a_i|\theta^Q))^2$;
14:      Update the actor policy by using the sampled policy gradient:
             $$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i};$$
15:      Update the target networks:
             $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'},$$
             $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}.$$
16:   end for
17: end for
```

The experiments use the Dow Jones 30 stocks from Jan 1, 2009 through Sept 30, 2018 to train the agent and test the performance. To better exploit the trading data, the authors chose to continue training the agent while in the trading stage as this improves the agent to better adapt the market dynamics. Four metrics were used to evaluate results: final portfolio value, annualized return, annualized standard error and the Sharpe ratio. The final results are summarized in the table below and as can be seen the DDPG strategy achieved an annualized return of 22.4% compared with the index average 16.40% over that time period.

Table 1: Trading Performance.



	DDPG (ours)	Min-Variance	DJIA
Initial Portfolio Value	**10,000**	10,000	10,000
Final Portfolio Value	**19,791**	14,369	15,428
Annualized Return	**25.87%**	15.93%	16.40%
Annualized Std. Error	**13.62%**	9.97%	11.70%
Sharpe Ratio	**1.79**	1.45	1.27

This paper demonstrated the great potential of applying DeepRL and could be extended by comparing these results against other algorithms, for ex: DQN. This could be achieved by simplifying the application from portfolio allocation to single stock trading that would simplify the action space making value based methods tractable. Another extension of this paper would be to expand the state space from its current simplistic representation to one that incorporates other “Technical Indicators”, both for the market and the company, as well. The dataset used in this paper already contains that so is a very easy extension. More NLP driven complex indicators could also be incorporated that the FinRL (paper above) has already built support for.

FinBERT: Financial Sentiment Analysis with Pre-trained Language Models

This paper introduces a language model based on BERT that tackles NLP tasks, more specifically sentiment analysis, in the financial domain. Large language models have become the recent trend in NLP and has ushered in an era of “transfer learning” where models pre-trained on huge language datasets (such as BooksCorpus + Wikipedia articles in the case for BERT) are either directly used or fine tuned for various specialized tasks. OpenAI demonstrated this in the paper Language Models are Few-Shot Learners by Brown et al. The core contribution of this paper is that it provides an open sourced model, FinBERT, that has been pre-trained to perform sentiment analysis (classifying whether a sentence is positive, negative or neutral) on financial text. The authors also compare the results for their fine tuned BERT based model with three different NLP baselines: LSTM classifier with GLoVe embeddings, LSTM classifier with ELMo embeddings and ULMFit classifier. In addition to these language model based techniques the authors evaluate FinBERT’s performance against specialized classifiers that were built and trained for the Financial PhraseBank dataset. Proving the strength of large language models, it can be seen in the results below that FinBERT does outperform all other techniques in all categories.

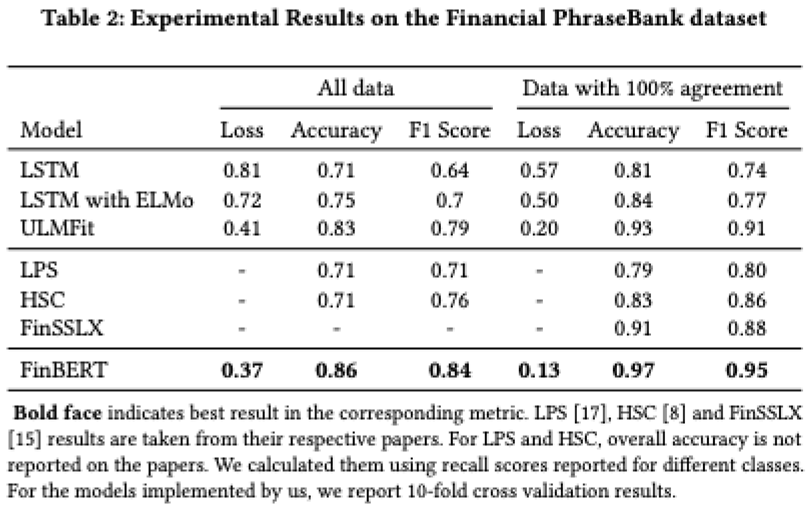


Table 2: Experimental Results on the Financial PhraseBank dataset

	All data			Data with 100% agreement		
Model	Loss	Accuracy	F1 Score	Loss	Accuracy	F1 Score
LSTM	0.81	0.71	0.64	0.57	0.81	0.74
LSTM with ELMo	0.72	0.75	0.7	0.50	0.84	0.77
ULMFit	0.41	0.83	0.79	0.20	0.93	0.91
LPS	-	0.71	0.71	-	0.79	0.80
HSC	-	0.71	0.76	-	0.83	0.86
FinSSLX	-	-	-	-	0.91	0.88
FinBERT	**0.37**	**0.86**	**0.84**	**0.13**	**0.97**	**0.95**

Bold face indicates best result in the corresponding metric. LPS [17], HSC [8] and FinSSLX [15] results are taken from their respective papers. For LPS and HSC, overall accuracy is not reported on the papers. We calculated them using recall scores reported for different classes. For the models implemented by us, we report 10-fold cross validation results.

The experimental methodology employed by the authors was two fold. They first took the pre-trained BERT model as provided by Google (phase 1 in the image below) and fine tuned it using the Reuters TRC2-financial text corpus. This training (phase 2 in image below) was done using the standard BERT objectives of MLM (masked language modeling, aka hiding tokens for the model to predict) and NSP (nest sentence prediction, aka model predicts whether two sentences passed for inference are related or not). The idea behind this training was to provide the model context that is more specific for the financial domain. Finally (phase 3 in image below) only the last layer of the model was fine tuned using the softmax objective to predict whether the sentiment of the sentence contextualized in the CLS token is positive, negative or neutral.

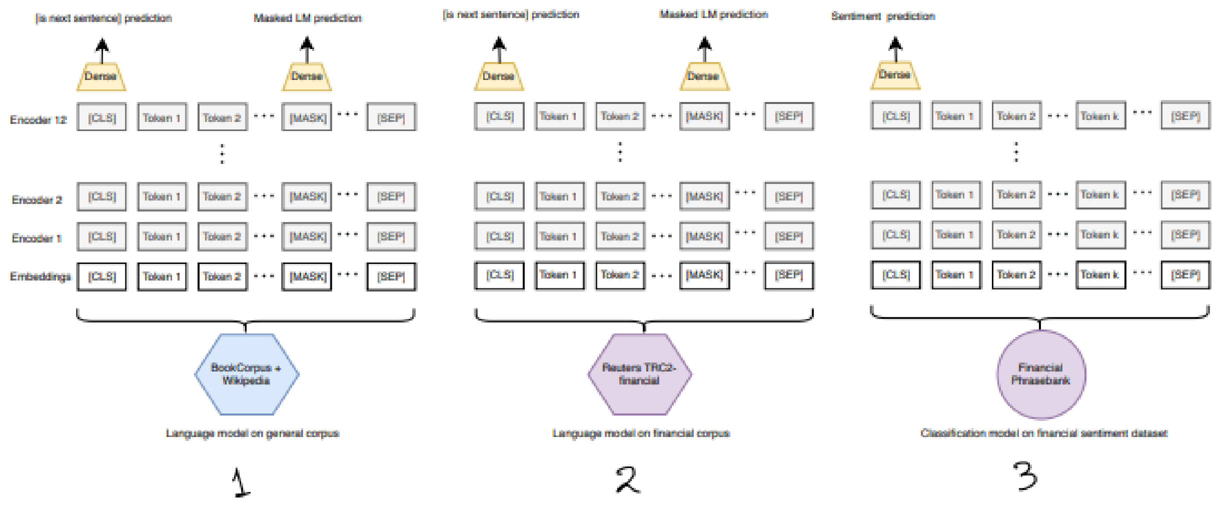


Figure 1: Overview of pre-training, further pre-training and classification fine-tuning

The authors find that the classifier that were further pre-trained on financial domain (phase 2) corpus performs best among the three, though the difference is not very high as shown in the table below:

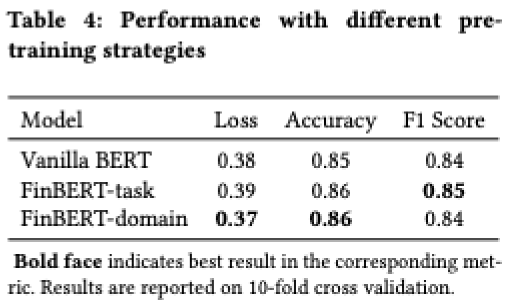


Table 4: Performance with different pre-training strategies

Model	Loss	Accuracy	F1 Score
Vanilla BERT	0.38	0.85	0.84
FinBERT-task	0.39	0.86	**0.85**
FinBERT-domain	**0.37**	**0.86**	0.84

Bold face indicates best result in the corresponding metric. Results are reported on 10-fold cross validation.

This is most likely because the performance of BERT itself is already so good that there is not much room for improvement.

Finally the authors highlight several instances where the model fails. The most common failure modes for the FinBERT model are:

- It does not know how to do math and simply relies on the connotations of the words present, such as “loss”:
    - Example: Pre-tax loss totaled euro 0.3 million , compared to a loss of euro 2.2 million in the first quarter of 2005 .
      True value: Positive ; Predicted: Negative
- Distinguishing between neutral and positive/negative. A lot of financial jargon is usually given a positive spin where the objective reality might actually be neutral and the model struggles to distinguish that.
    - Example: This implementation is very important to the operator , since it is about to launch its Fixed to Mobile convergence service in Brazil,
      True value: Neutral ; Predicted: Positive

Overall this paper presented interesting overall results on financial NLP analysis. As a possible extension, some of these results could be tied in to the previous two papers (FinRL and Practical DeepRL for stock trading) as potential “Technical Indicators” to the state space defined for a certain security. In other words if there were datasets that contained several news articles, earnings reports, twitter feeds related to certain stocks at different timestamps, those could be correlated to provide sentiment data on how the market feels about a certain stock. This could potentially be a powerful indicator as such analysis is extremely difficult, if not impossible, to perform manually.

This work could also be extended in several ways. One possible extension can be using FinBERT for other natural language processing tasks such as named entity recognition or question answering in the financial domain. Another possible extension could be using FinBERT directly with stock market return data to meas

Stock Values and Earnings Call Transcripts: Dataset Suitable for Sentiment Analysis

This paper introduces a new dataset that reports a collection of earnings call transcripts, the related stock prices of 10 popular stocks on the Nasdaq index, along with the sector index. The purpose of this dataset is to promote research into NLP techniques that can exploit possible correlation between market sentiments and stock prices. The dataset contains a total of 188 transcripts, 11970 stock prices, and 1196 NASDAQ sector index values in the period between January 1st, 2016 through October 1st, 2020. This dataset was collected by merging the Yahoo Finance dataset that offers daily stock prices with the Thomas Reuters Eikon data repository to find the time correlated earnings call transcripts. The dataset introduced in this paper would tie in very well with the FinBERT model introduced in the paper above to produce sentiment indicators that could potentially be used with the FinRL library (paper above) as an augmented signal in the state space of the stock. While this paper introduces a novel dataset, the authors could have compared it against other related datasets for broader context. The major limitation of this paper, that could also be future extensions of this work, is the limited amount of data available. It would serve to be more useful if the dataset was increased to cover a wider variety of stocks in a longer time span.