**GIT LINK:** [**https://github.com/raj-013/Blockhouse-worktrial**](https://github.com/raj-013/Blockhouse-worktrial)

**Analysis of existing code:**

**1. Installing required packages**

# Install required system packages

!apt-get install -y wget make gcc

# Download the TA-Lib source code

!wget http://prdownloads.sourceforge.net/ta-lib/ta-lib-0.4.0-src.tar.gz

# Extract the source code

!tar -xzf ta-lib-0.4.0-src.tar.gz

# Build and install the library

%cd ta-lib/

!./configure --prefix=/usr

!make

!make install

# Go back to the initial directory

%cd ..

# Install the Python wrapper for TA-Lib

!pip install ta-lib

#Restart required

!pip install stable-baselines3

!pip install 'shimmy>=0.2.1'

**Key Changes to Implement a Transformer-Based PPO Model:**

**1. Transformer Model Integration:**

* Introduced a custom feature extractor that uses a transformer encoder. This replaces the traditional MLP policy with a transformer-based model, enabling it to process sequence data effectively.
* The transformer model was implemented to handle the time-series data for trades, ensuring that the input data is appropriately processed by the transformer architecture.

**2. Modify PPO to Use Transformer Outputs:**

* The PPO algorithm was modified to incorporate the transformer's output as the basis for the policy distribution, with the policy network using the transformer's extracted features.
* The PPO loss function was utilized to optimize the policy network, which now includes the transformer-based feature extractor.

**3. Training and Fine-Tuning:**

* The training loop was adapted to accommodate the transformer-based model, ensuring that the rewards and policy gradients are properly backpropagated through the transformer network.
* The code was structured to allow for proper backpropagation of rewards and policy gradients through the transformer network.

**4. Evaluation:**

* The evaluation process was conducted by comparing the trading performance of the Transformer with PPO model against the original PPO model using key metrics such as rewards, transaction costs, and slippage.

**Result Analysis:**

**A) Improvements of Transformer (with PPO) over PPO after training.**

* Introduction of Transformer Feature Extractor: Replaced the traditional MLP policy with a Transformer-based feature extractor, allowing the model to capture more complex patterns in the data.
* Entropy Coefficient Decay: Implemented a decay for the entropy coefficient to gradually shift focus from exploration to exploitation as training progresses.
* Learning Rate Scheduler: Added a learning rate scheduler to gradually reduce the learning rate, ensuring stability as the model converges.
* Stable Policy Updates: Achieved more stable and consistent policy updates, with low approx\_kl and no need for clipping, indicating that the model maintained well-controlled learning throughout training.
* Improved Value Function Learning: Significant reduction in value loss over time, indicating better learning and prediction of returns, leading to more accurate policy decisions.
* Stable and Lower Policy Gradient Loss: Achieved smaller and more stable policy gradient loss, reducing the risk of oscillations in policy updates and contributing to overall model stability.
* Reduced Exploitation-Exploration Trade-off: By reducing entropy loss more effectively, the Transformer with PPO model balanced exploration and exploitation more effectively compared to the original PPO.

**B) PPO vs. Transformer with PPO Based on Trading Blotter Data**

* **Risk Management:** The Transformer with PPO model exhibited higher transaction costs, slippage, and more significant negative rewards, indicating potentially riskier trading decisions.
* **Introduction of Transformer Feature Extractor:** Enabled the model to capture more complex patterns, resulting in more frequent and varied trading actions.
* **Learning Rate Scheduling and Entropy Decay:** Introduced mechanisms to stabilize learning and shift focus from exploration to exploitation, contributing to more aggressive trading decisions.

**Conclusion:**

The Transformer with PPO model is more responsive and aggressive compared to the original PPO model, resulting in higher trade frequency, higher transaction costs, and more significant negative rewards. While it demonstrates the capability of capturing complex patterns, this model may benefit from additional tuning to manage risk and reduce transaction costs and slippage. The increased complexity and responsiveness may offer advantages in specific market conditions but at the cost of higher risk.

**Future Works:**

**A) Improvements in current model**

* Hyperparameter Optimization
* Dynamic Entropy Coefficient: explore adaptive methods where the entropy coefficient changes in response to the agent's performance.
* Transformer Model Enhancements: Experiment with architectures like Temporal Fusion Transformers (TFT) or Long-Short Term Memory (LSTM).

**B) Other Suggestions:**

* Multi-Agent Reinforcement Learning (MARL): Implement a multi-agent version of the transformer with PPO, where multiple agents interact in the market environment.
* Real-Time Trading Simulation
* Comparison with Other Algorithms

The Transformer with PPO model has shown promise in capturing complex trading patterns and making informed decisions. However, there is ample room for further research and refinement. By exploring the outlined future work, the model could be enhanced to achieve better performance, greater stability, and broader applicability in diverse trading environments.

**References:**

[1] <https://github.com/datvodinh/ppo-transformer?tab=readme-ov-file>

[2] <https://huggingface.co/docs/trl/en/index>