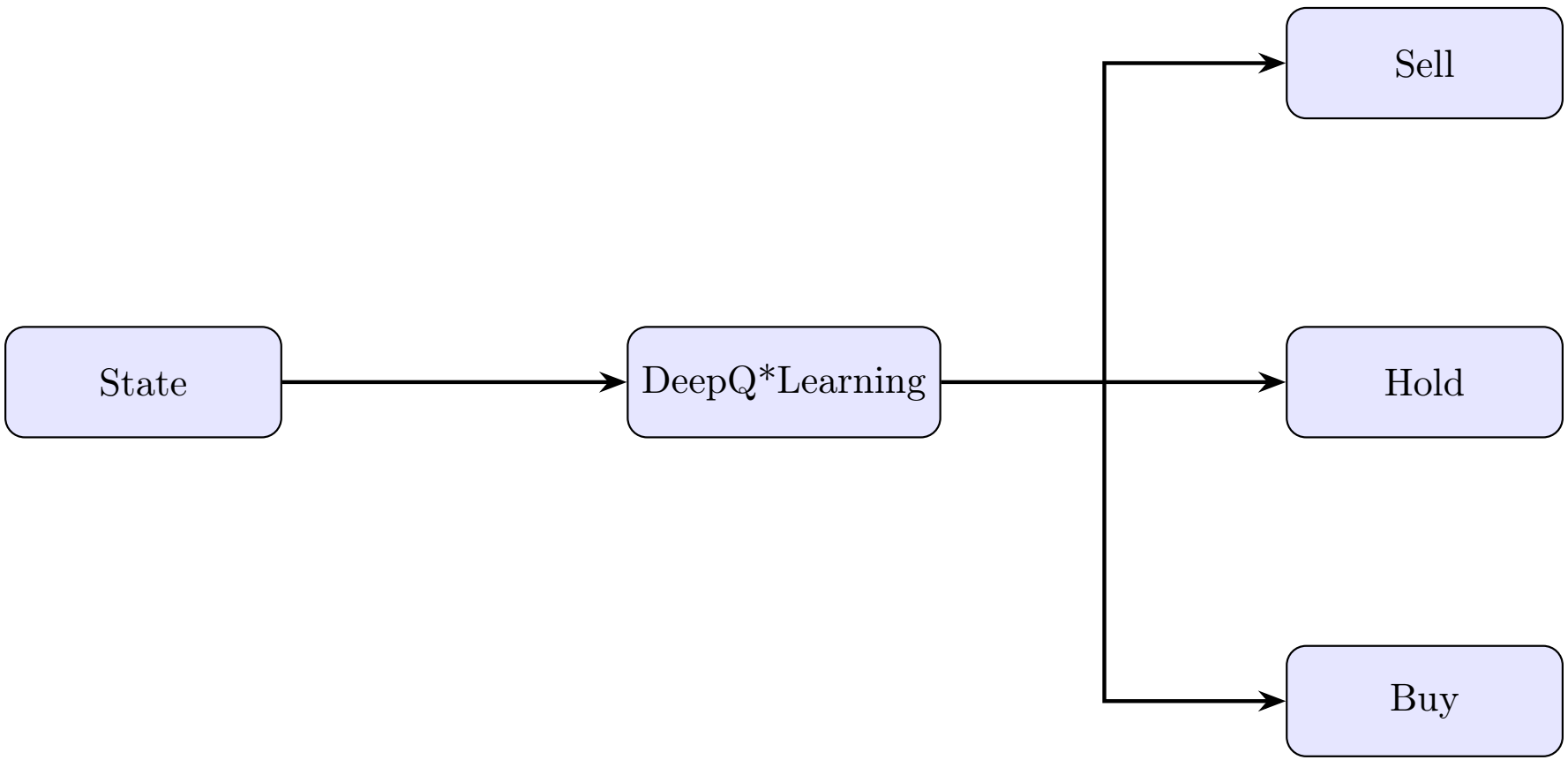
Optimized Kalman Filter

Mathys Vinatier

Seoul National University

September 3, 2025

Deep QLearning model



Deep Q-Learning Model

Layer	Type
`nn.Linear`	Linear Layer
`nn.ReLU`	Activation Function
`nn.Sequential`	Container

Table 1: Explanation of the MLP layers

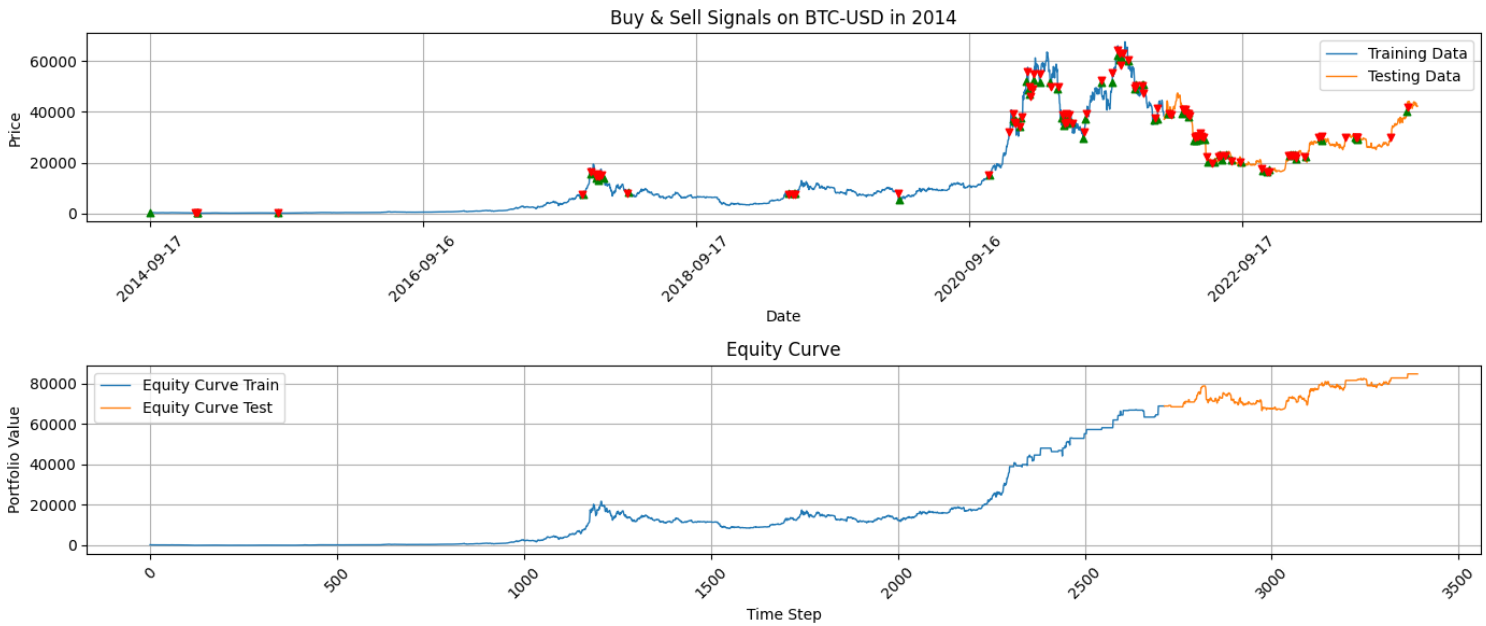
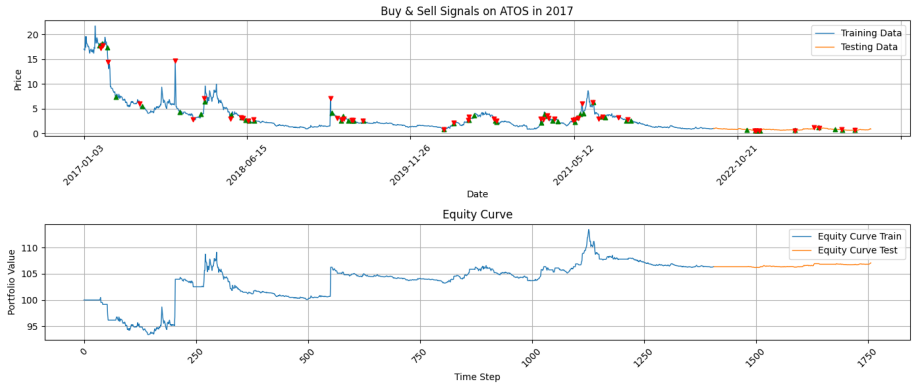
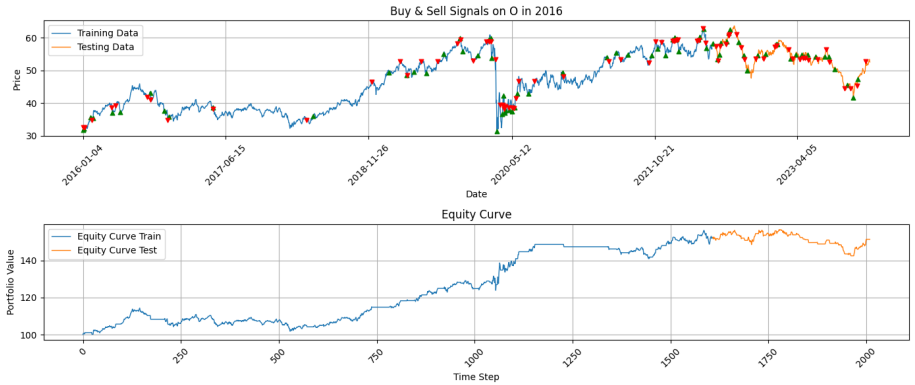


Figure 1: Performance of the Q-Learning model on a BTC-USD test set

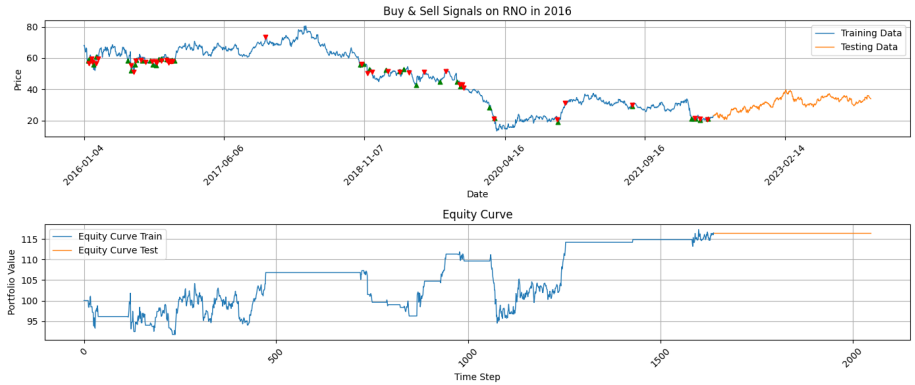
Deep QLearning Model



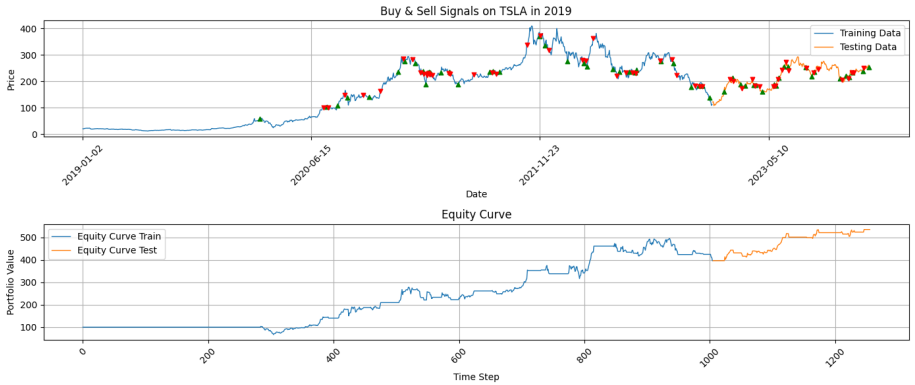
(a) Performance on ATOS data from 2017.



(b) Performance on Realty Income (O) data from 2016.



(c) Performance on Renault (RNO) data from 2016.



(d) Performance on Tesla (TSLA) data from 2019.

Deep QLearning Model

Layer	Type
`DecisionTransformerGPT2Model`	Transformer Backbone
`nn.Linear` (Input)	Linear Layer
`nn.Linear` (Q-Head)	Linear Layer

Table 2: Explanation of the Decision Transformer layers

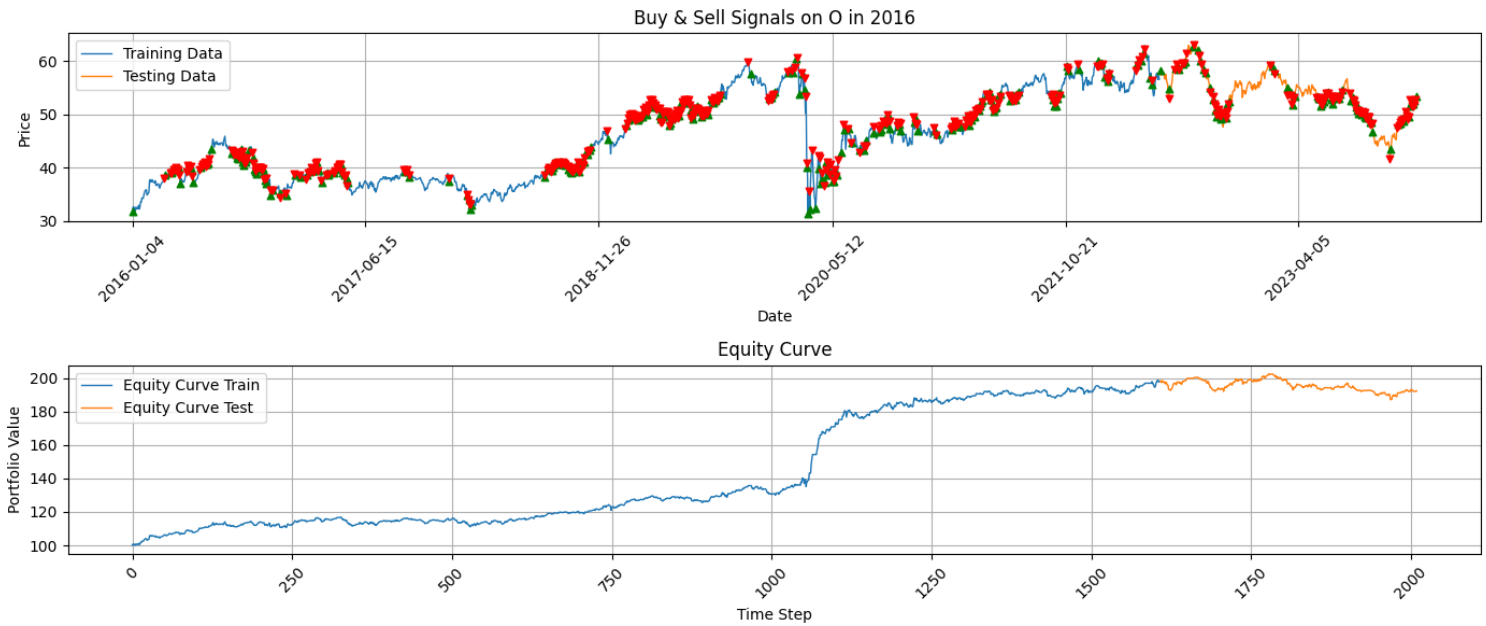


Figure 3: Performance of the Deep Q-Learning Model on Realty Income (O) data from 2016

Deep QLearning Model

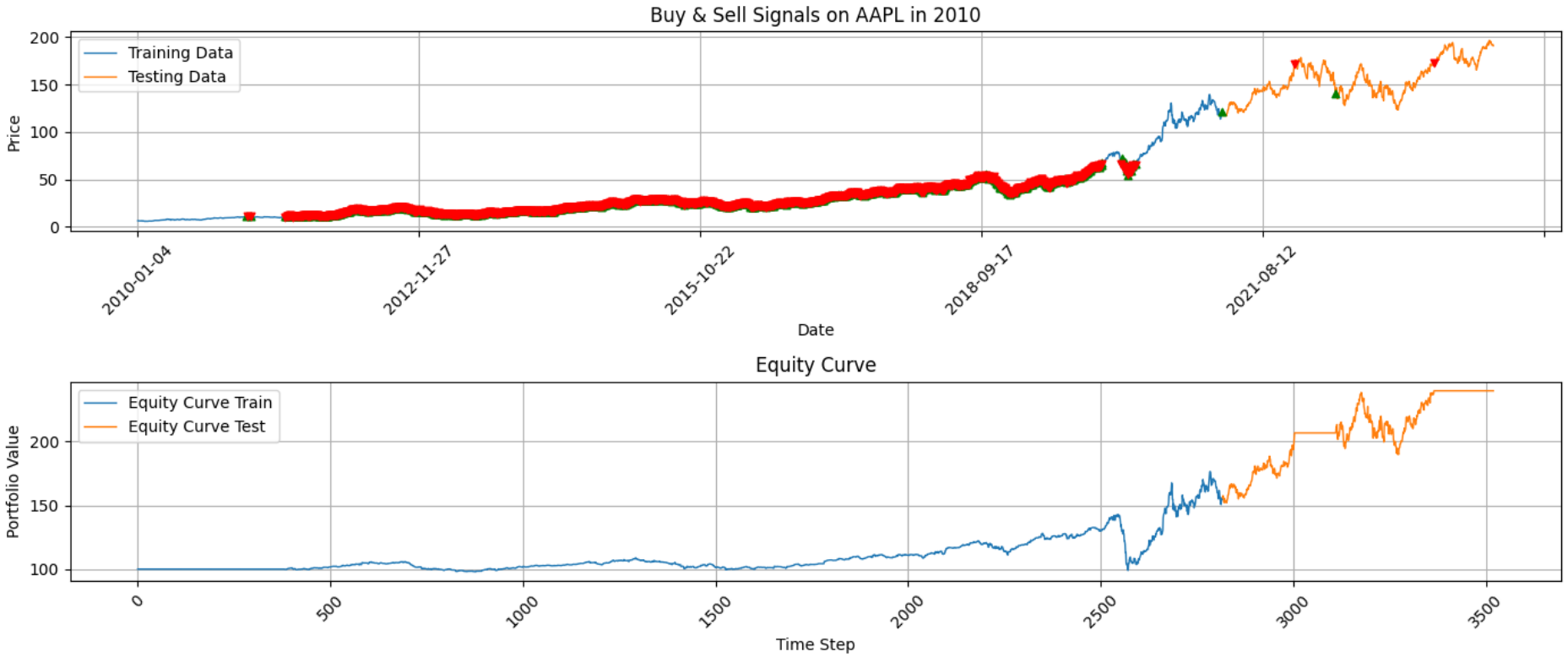
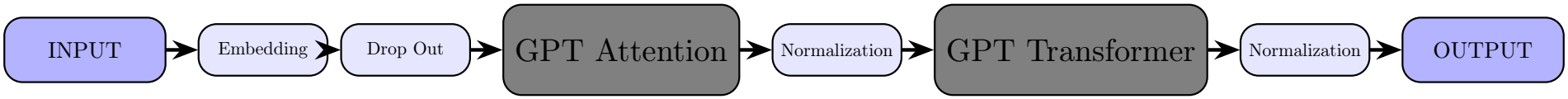
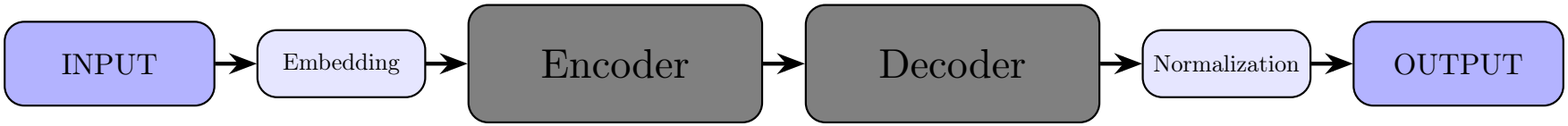


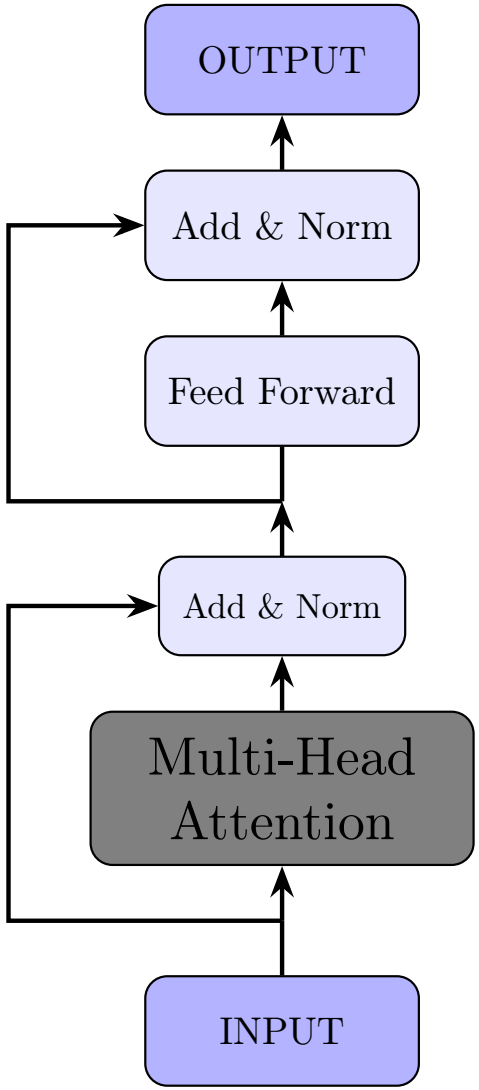
Figure 4: Performance of the Transofrmer Model on Apple data from 2010.

Creating Market Model





Creating Market Model



Creating Market Model

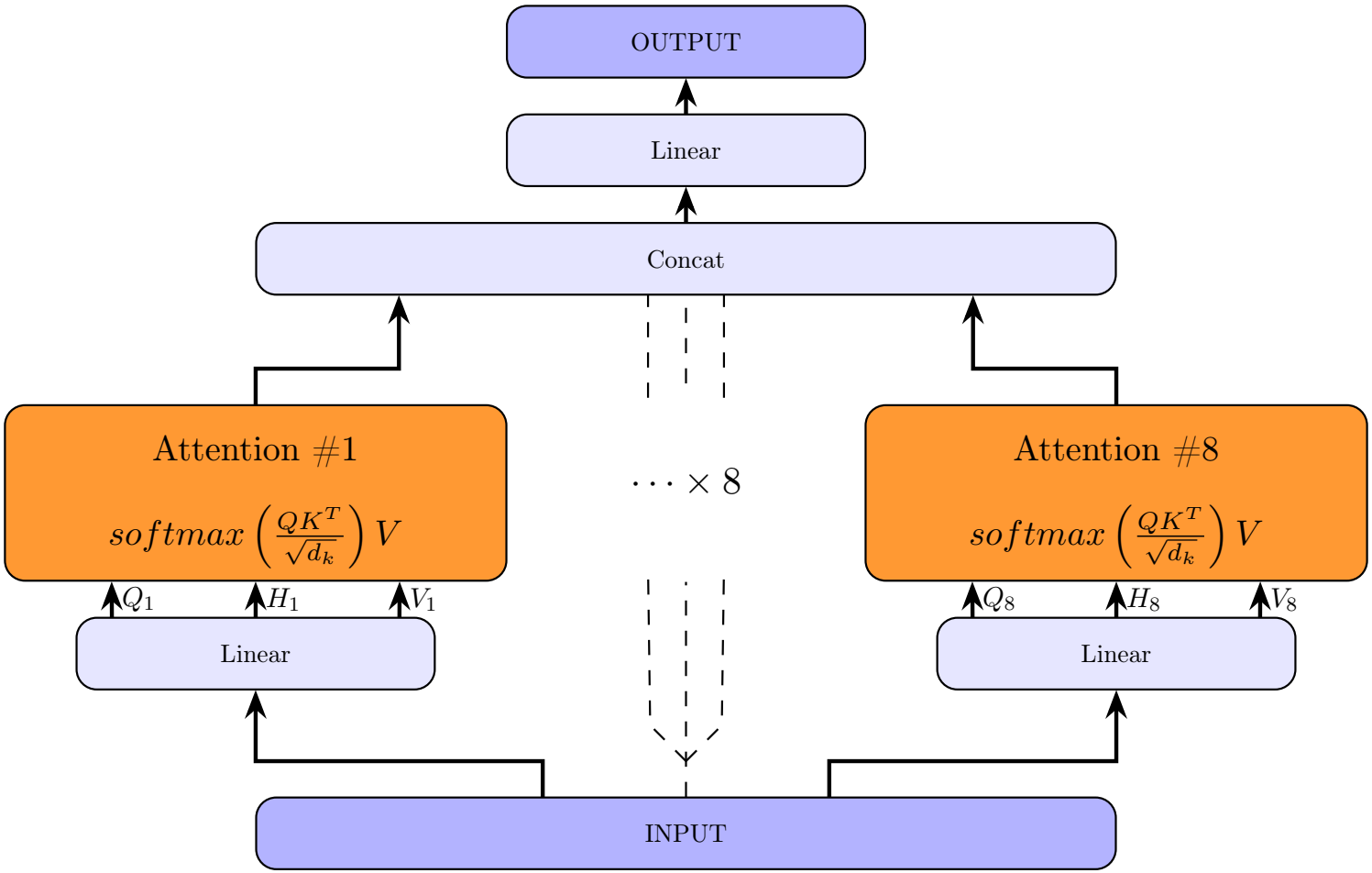


Figure 5: Multi-Head Attention stack (x8 heads)

Creating Market Model

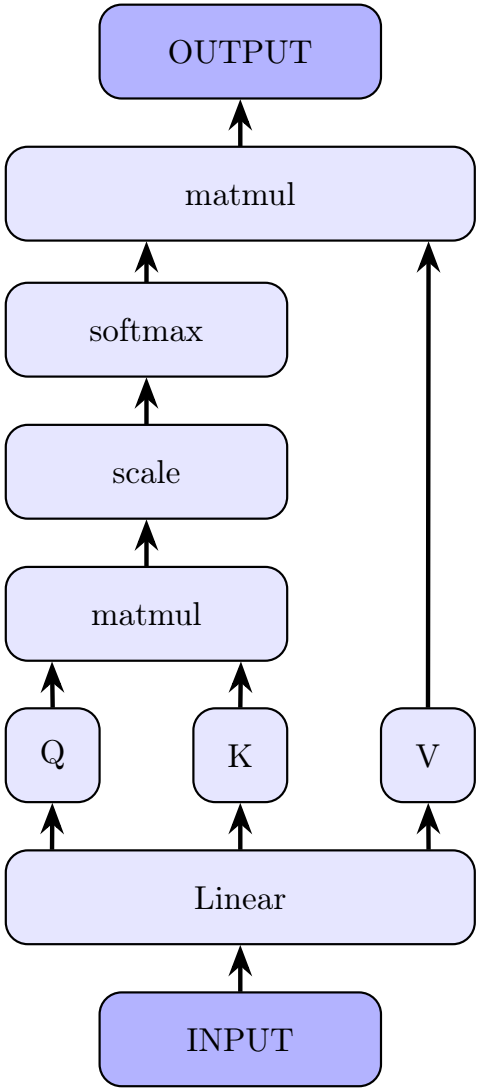


Figure 6: Attention stack

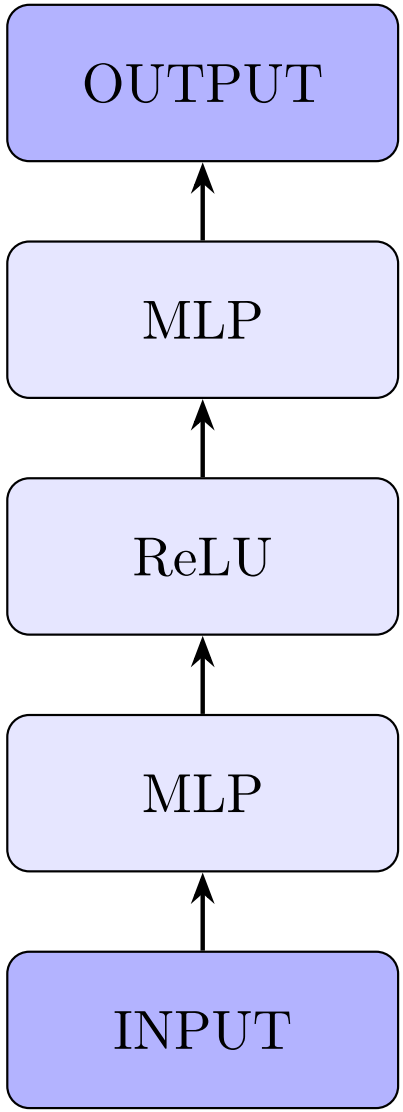


Figure 7: Feed-Forward stack

Adding commission

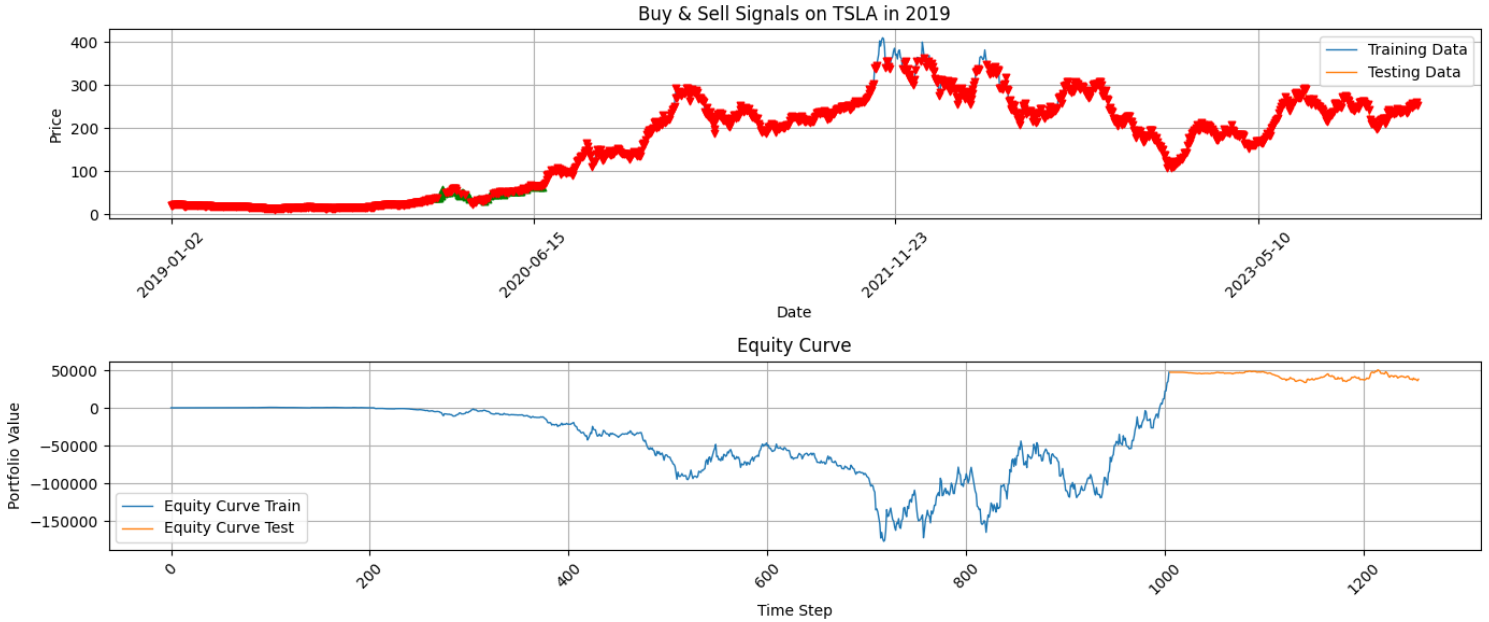


Figure 8: Performance of the Deep Q-Learning Model with commission on Tesla data from 2019.

Reworking the Reward

Figure 9: Reward based on portfolio

Reworking the Reward

Figure 10: Reward based on portfolio differences

Reworking the Reward

Figure 11: Reward based on portfolio times the velocity

Reworking the Reward

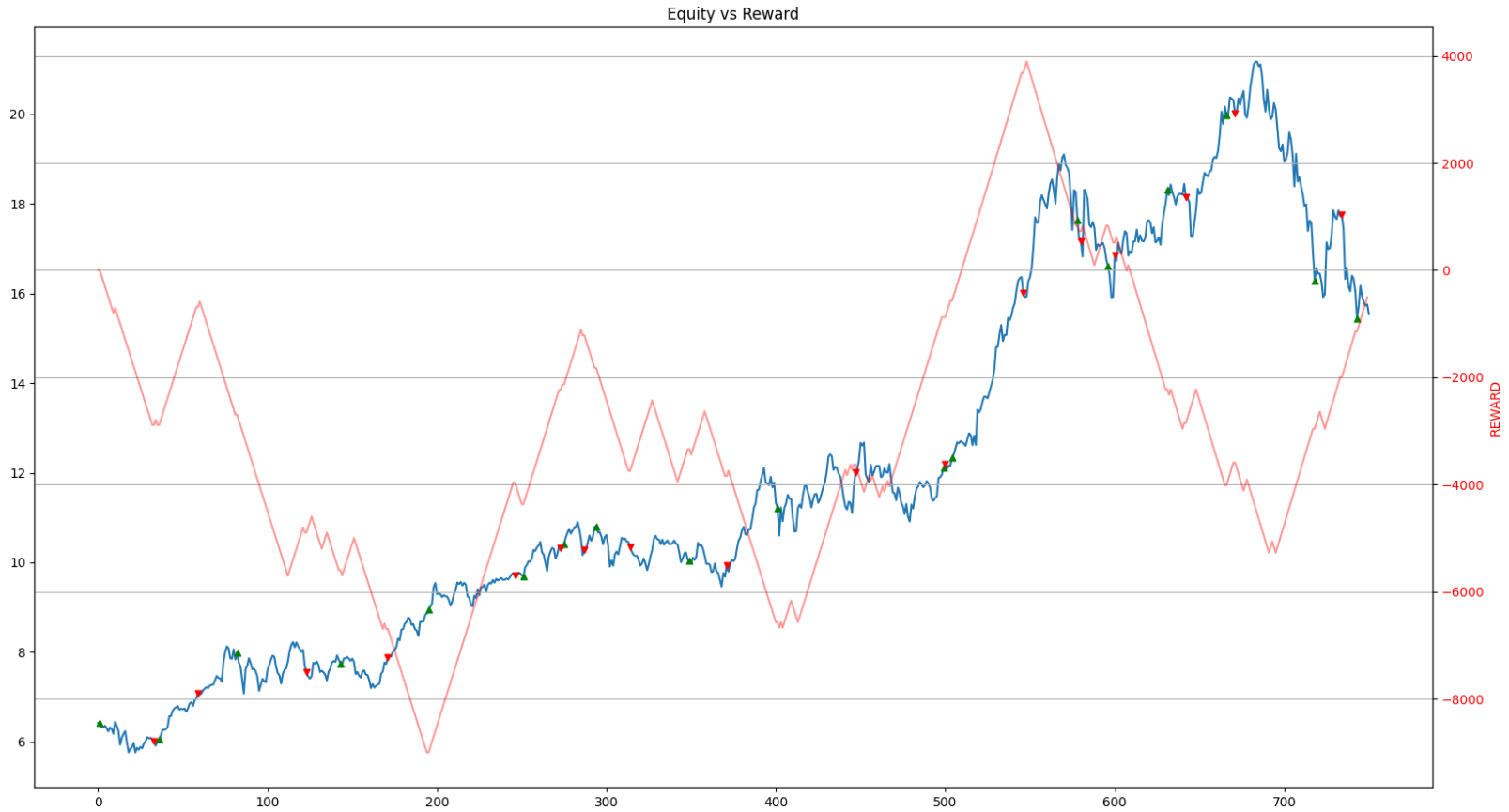


Figure 12: Reward based on added portfolios times the velocity

Reworking the Reward

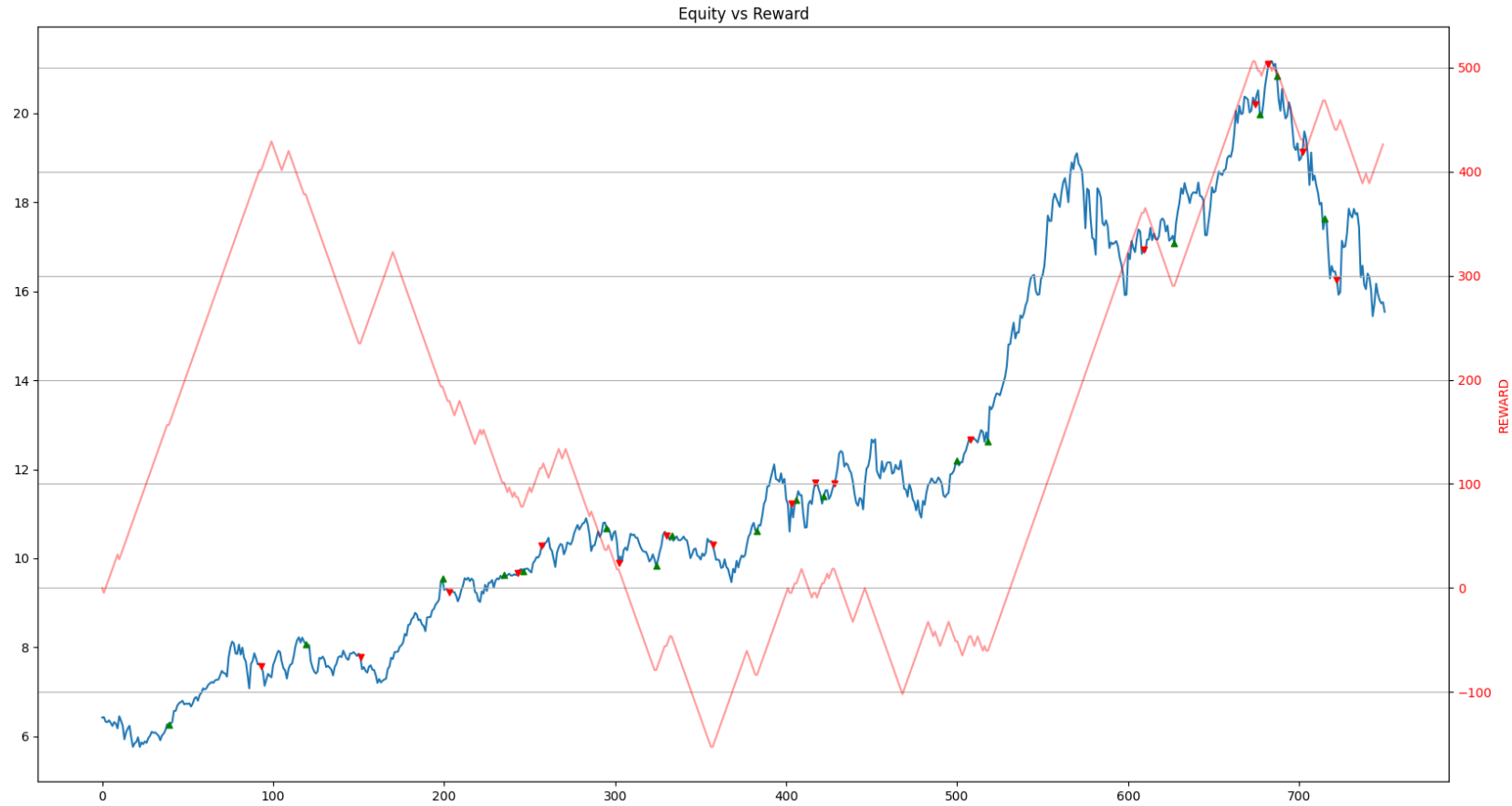


Figure 13: Reward based on added portfolios times the velocity with logarithmic function

Thank You