Advanced topics:
Static replication using ANNs

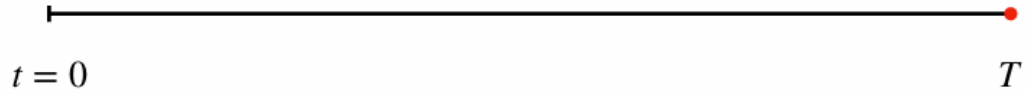
Group 4: Divya Gajera & Nitai Nijholt

Objectives

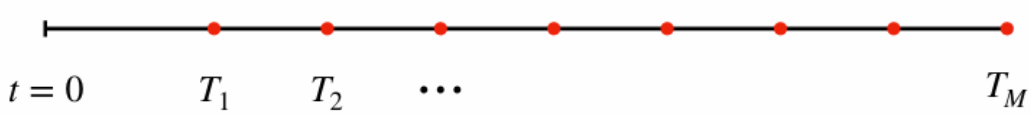
- **Pricing**: Calculate the fair value of the derivative at any timepoint
- **Hedging**: Construct a semi-static hedge using short-maturity European options that replicate the derivative's payoff.

Bermudan Option

- European option:



- Bermudan option:



- American option:

Motivation RLNN

Advantages RLLN:

- (Semi) Static Hedging
- Interpretability
- Lower Computational Cost
- Universal Approximation Theorem

RLNN Algorithm

Main idea: Approximate pay-off of complex derivative using linear combination of simple European Options

Algorithm Steps:

1. Generate simulations of N underlying asset paths using GBM:

$$dS_t = \mu S_t\, dt + \sigma S_t\, dW_t$$

2. Compute option payoffs at maturity for each path:

$$V_T^{(n)} = \max(h(S_T^{(n)}), 0)$$

3. Train Neural Network $G(S)$ each time step to approx. option values:

$$\tilde{G}_{\beta_{t_m}}(S_{t_m}^{(n)}) \approx \tilde{V}_{t_m}^{(n)}$$

4. Compute continuation value using the trained network:

$$Q_{t_{m-1}}^{(n)} = E\left[\tilde{G}_{\beta_{t_m}}(S_{t_m}) \mid S_{t_{m-1}}^{(n)}\right]$$

5. Option Value Update: Update option values:

$$\tilde{V}_{t_{m-1}}^{(n)} = \max(h(S_{t_{m-1}}^{(n)}), Q_{t_{m-1}}^{(n)})$$

6. Repeat step 3-5 backward to the initial time.

$$m = M-1, M-2, \ldots, 1, \text{ moving backward in time}$$

Image credit: Hoencamp, J. (2024, Fall). *Advanced topics: Semi-static replication using ANNs*. Advanced Topics in Computational Finance, University of Amsterdam.

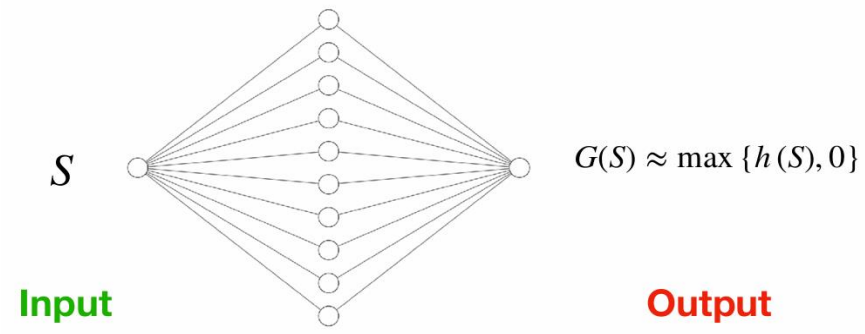
RLNN Algorithm

Main idea: Approximate pay-off of complex derivative using linear combination of simple European Options

NN structure:

- Hidden Layer Uses ReLU activation functions
- Each neuron represents payoff of single European option
- Portfolio weights and biases correspond to strikes and signs imply option type

Regress later because it uses feed forward neural network for the regression at each time step



$$\tilde{G}^{\beta_{t_m}}(\mathbf{x}) = \sum_{i=1}^{p} \omega_i \varphi\left(\mathbf{w}_i^{\top}\mathbf{x} + b_i\right) + b_2$$

Image credit: Hoencamp, J. (2024, Fall). *Advanced topics: Semi-static replication using ANNs*. Advanced Topics in Computational Finance, University of Amsterdam.

Expected value

$$Q_{t_{m-1}}^{(n)} = E\left[\tilde{G}_{\beta_{t_m}}(S_{t_m}) \mid S_{t_{m-1}}^{(n)}\right] \rightarrow \mathbb{E}\left[\varphi\left(\mathbf{w}_i^\top \mathbf{S}_{t_m} + b_i\right) \mid \mathbf{S}_{t_{m-1}}\right] \rightarrow \mathbb{E}[\max\left(w_i S_{t_m} + b_i, 0\right) | S_t]$$

Cases

Case 1: $w_i \geq 0$ and $b_i \geq 0$

The expected value simplifies to the price of a forward contract:

$$\text{Expected Value} = w_i \cdot S_{t_{m-1}} \cdot e^{r \cdot \Delta t} + b_i$$

Case 2: $w_i > 0$ and $b_i < 0$

This case represents a European call option. The strike price is:

$$\text{Strike} = -\frac{b_i}{w_i}$$

$$\text{Expected Value} = w_i \cdot \text{Black-Scholes}(S_{t_{m-1}}, \text{Strike}, r, \sigma, \Delta t, \text{option\_type='call'})$$

Case 3: $w_i < 0$ and $b_i > 0$

This case represents a European put option. The strike price is:

$$\text{Strike} = -\frac{b_i}{w_i}$$

$$\text{Expected Value} = -w_i \cdot \text{Black-Scholes}(S_{t_{m-1}}, \text{Strike}, r, \sigma, \Delta t, \text{option\_type='put'})$$

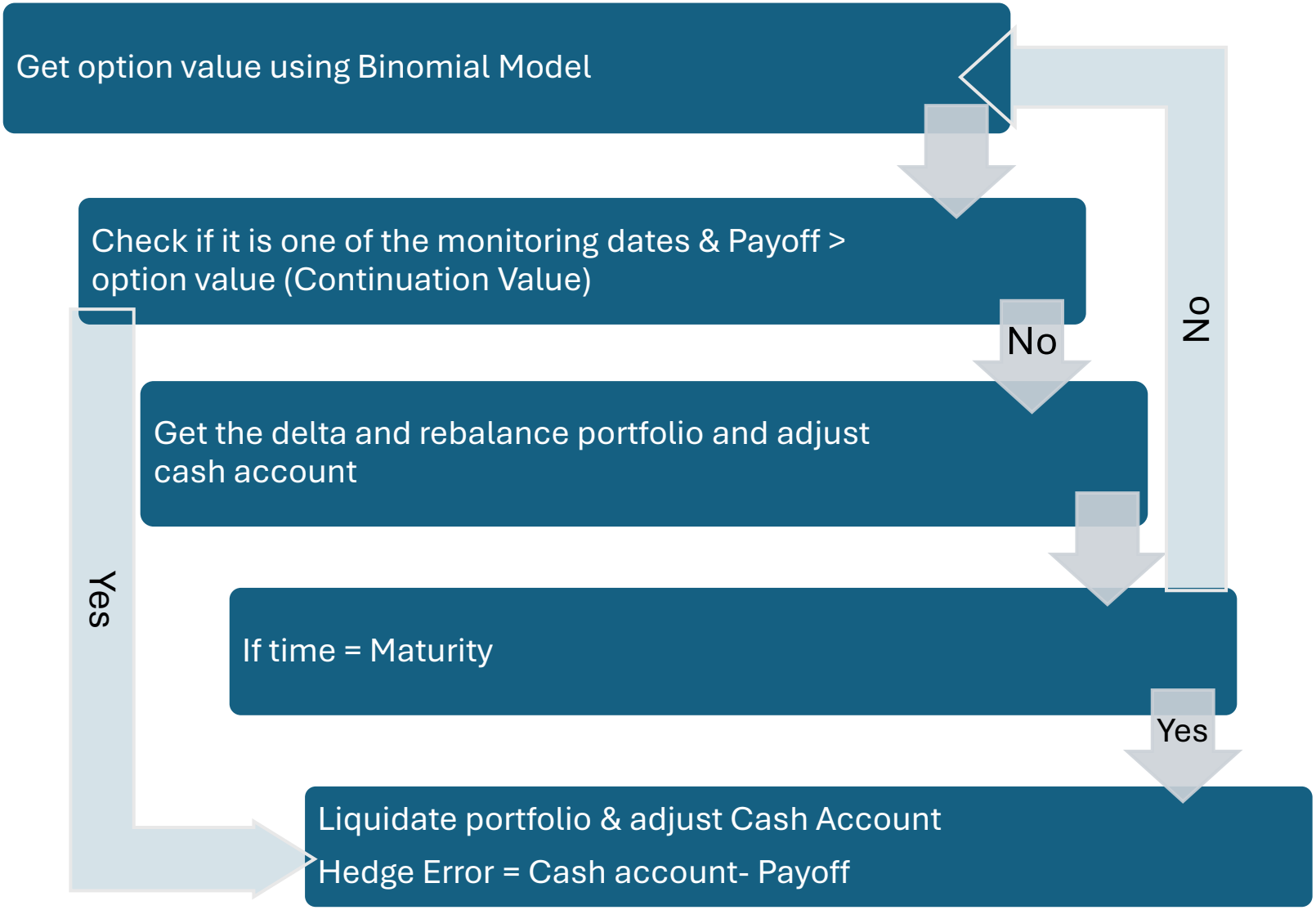
Case 4: $w_i \leq 0$ and $b_i \leq 0$

$$\text{Expected Value} = 0$$

Dynamic Hedging Error Calculation

- Simulate 10000 paths for stock prices
- Calculate Option price at time 0 and get the delta
- Initial Portfolio Setup
  - Short One Bermudan Option
  - Long position in delta shares
  - Remaining in cash account
  - dt = T/M, M = hedging Frequency, j =1

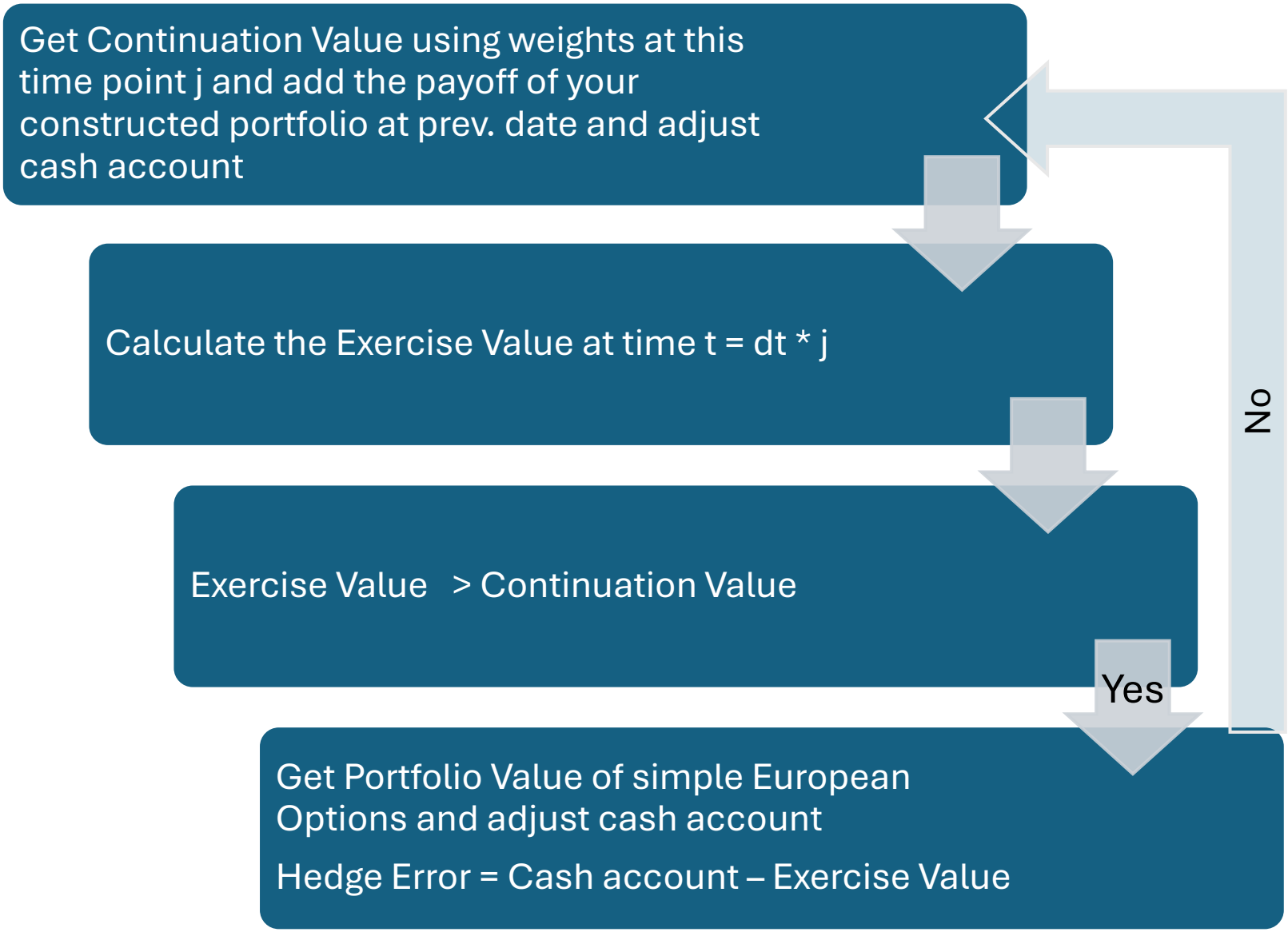
- One iteration of Monte Carlo Simulations



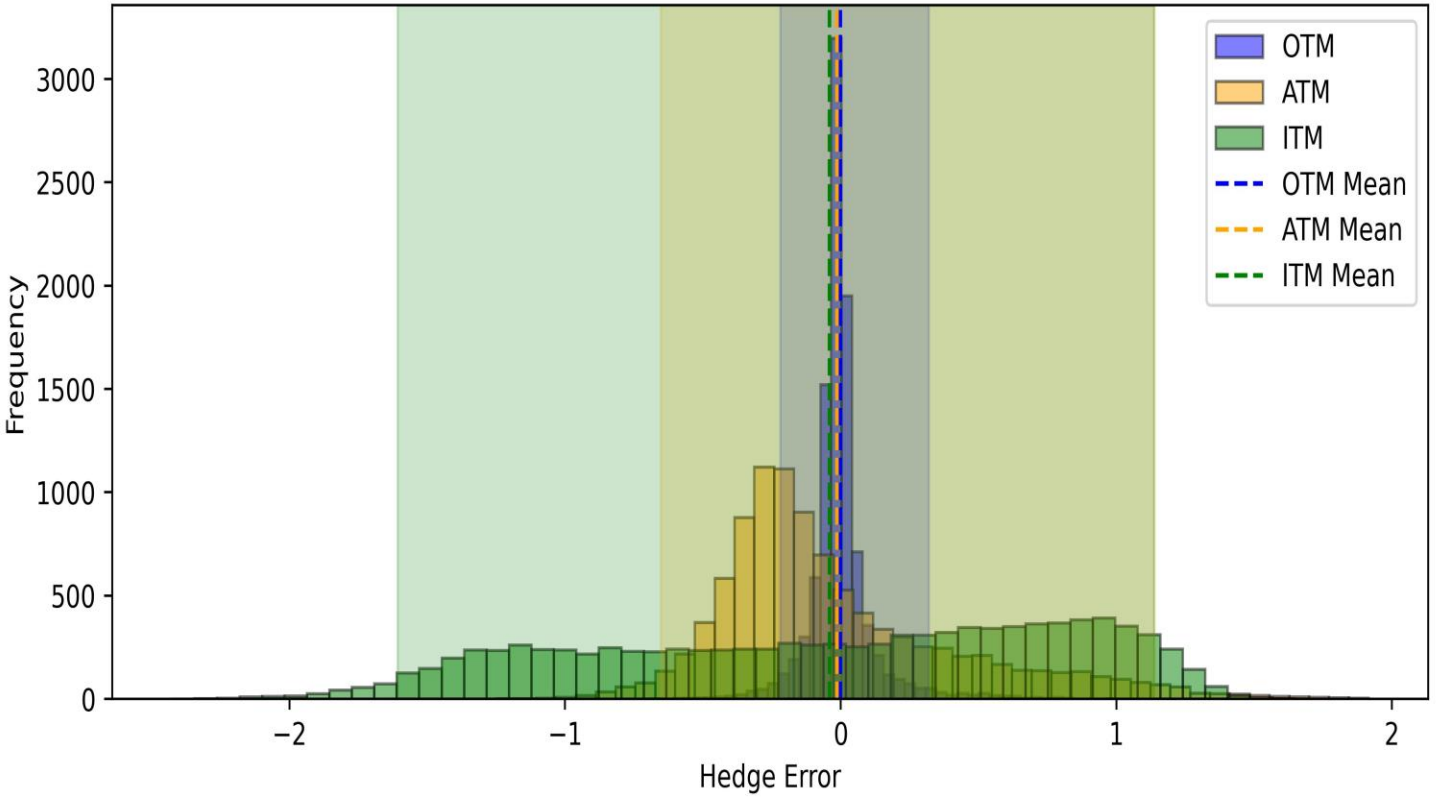
Semi Static Hedging Error Calculation

- Simulate 10000 paths for stock prices
- Calculate Option price at time 0 using RLNN and get weights
- Initial Portfolio Setup
  - Short One Bermudan Option
  - Buy portfoilo based on weights we get from RLNN
  - Remaining in cash account
  - dt = T/M, M = no of monitoring dates, j =1

- One iteration of Monte Carlo Simulations

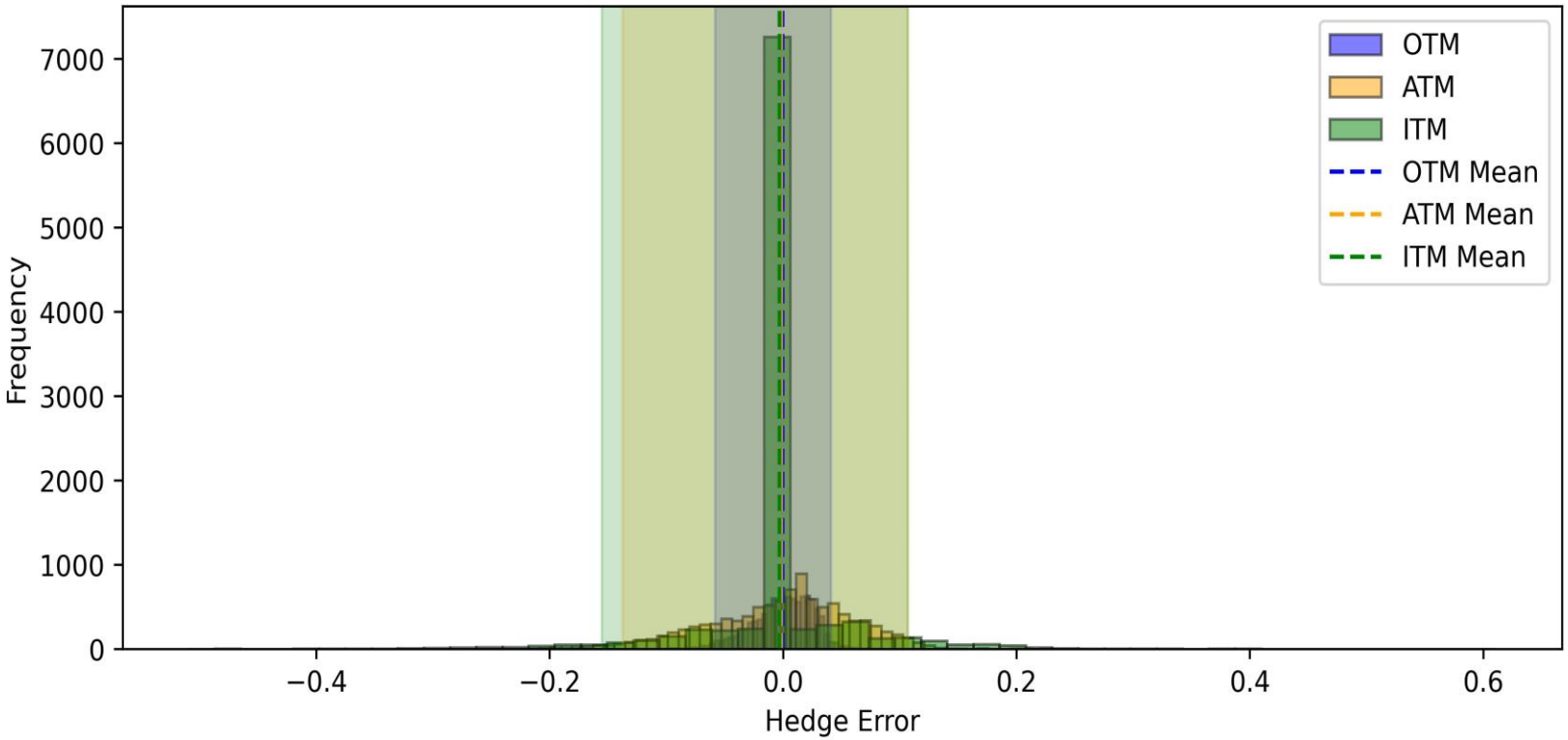


Delta (Dynamic) Hedging Error (Euros) of Bermudan Put Option



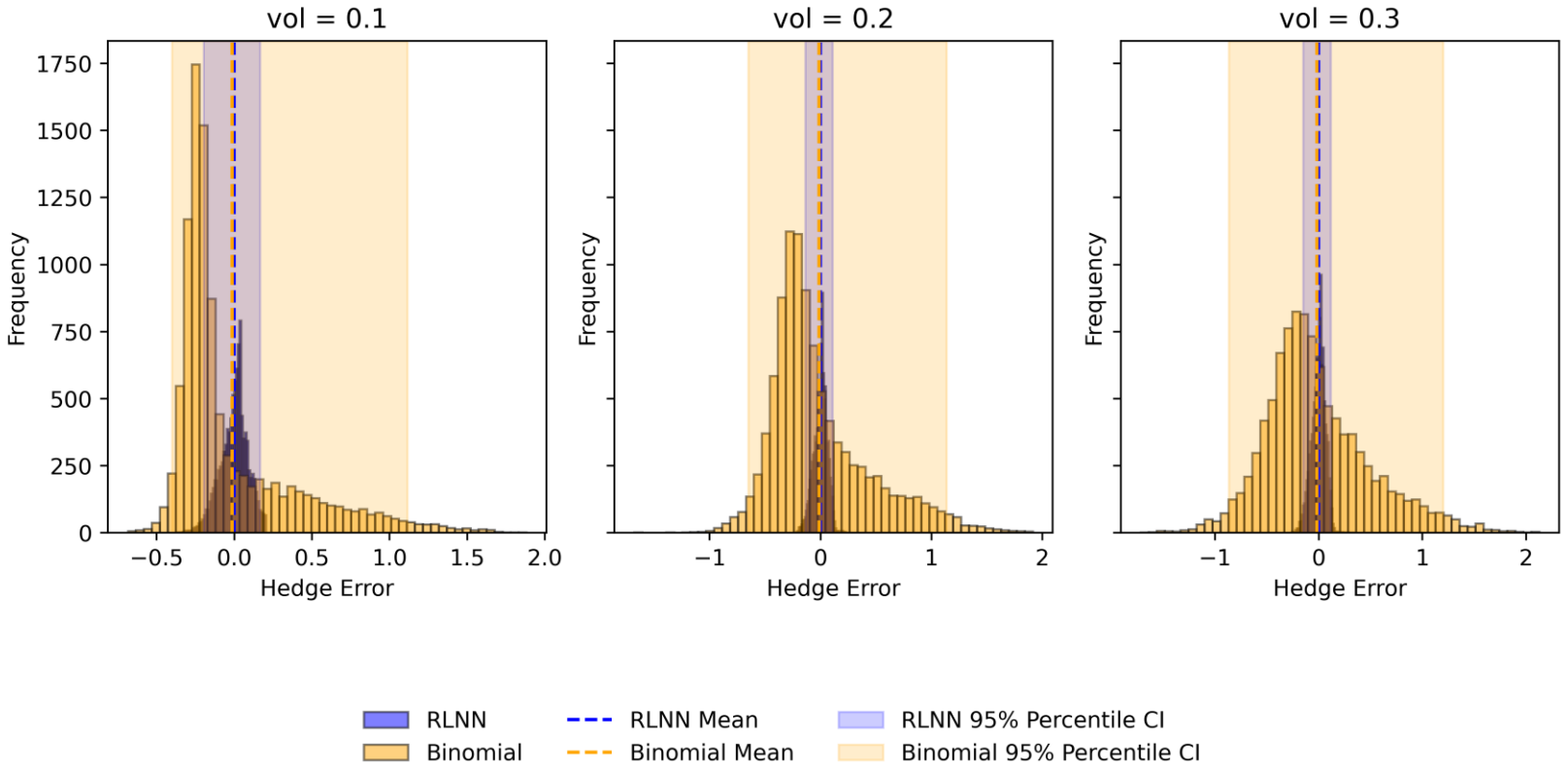
Parameters: S0 = 50, K = 50(ATM), r = 0.06, vol = 0.2, number of simulations = 10000, K = 40(OTM) and K = 60(ITM)

Semi – Static Hedging Error (Euros) of Bermudan Put Option



Parameters: S0 = 50, K = 50(ATM), r = 0.06, vol = 0.2, number of simulations = 10000,
K = 40(OTM) and K = 60(ITM)

Dynamic & Semistatic Hedging Error (Euros) for different volatilities



Fully Static Replication

Step 1: Generate Scenarios

- Simulate N paths for state variables $\{x_{T_0}, x_{T_1}, \ldots, x_{T_{M-1}}\}$ under the risk-neutral measure.

Step 2: Initialize Continuation Values

- At the final date , T_{M-1} set continuation values $C_{M-1}(n) = 0.$

Step 3: Backward Induction via Neural Networks

- For m = M−1, … ,1
  - Compute Target Payoff : $f_m(n) = \max(h_m(x_{T_m}(n)) - C_m(n), 0).$
  - Train Neural Network :

$$\beta_m = \arg\min_{\beta} \frac{1}{N} \sum_{n=1}^{N} \left(G_m(z_m(n); \beta) - f_m(n)\right)^2.$$

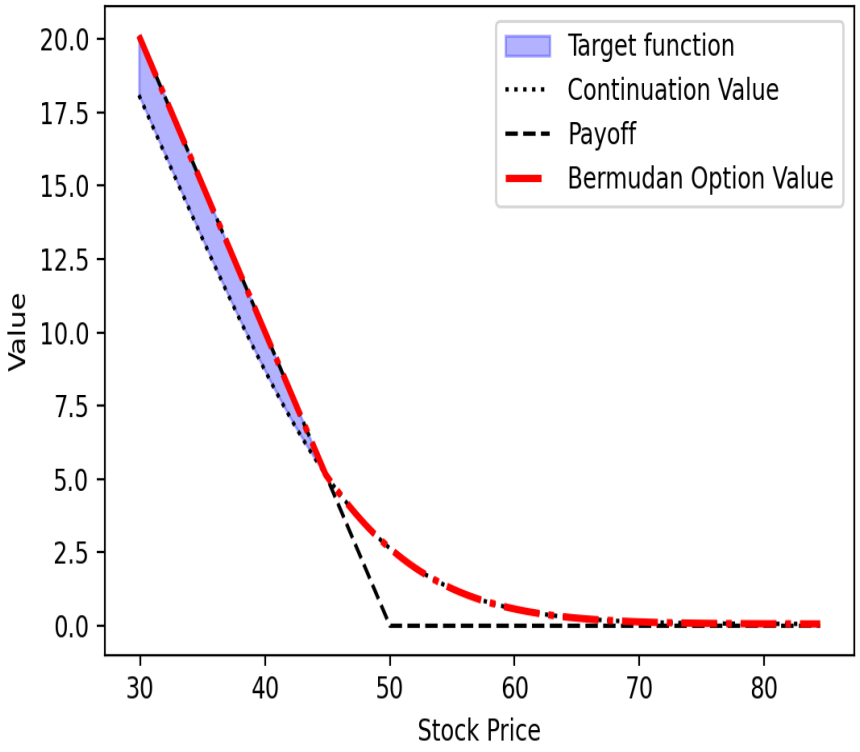
  - Update Continuation Value :

$$C_{m-1}(n) = \mathbb{E}^{\mathbb{Q}} \left[\sum_{j=m}^{M-1} G_j(z_j; \beta_j) \mid x_{T_{m-1}}(n) \right].$$

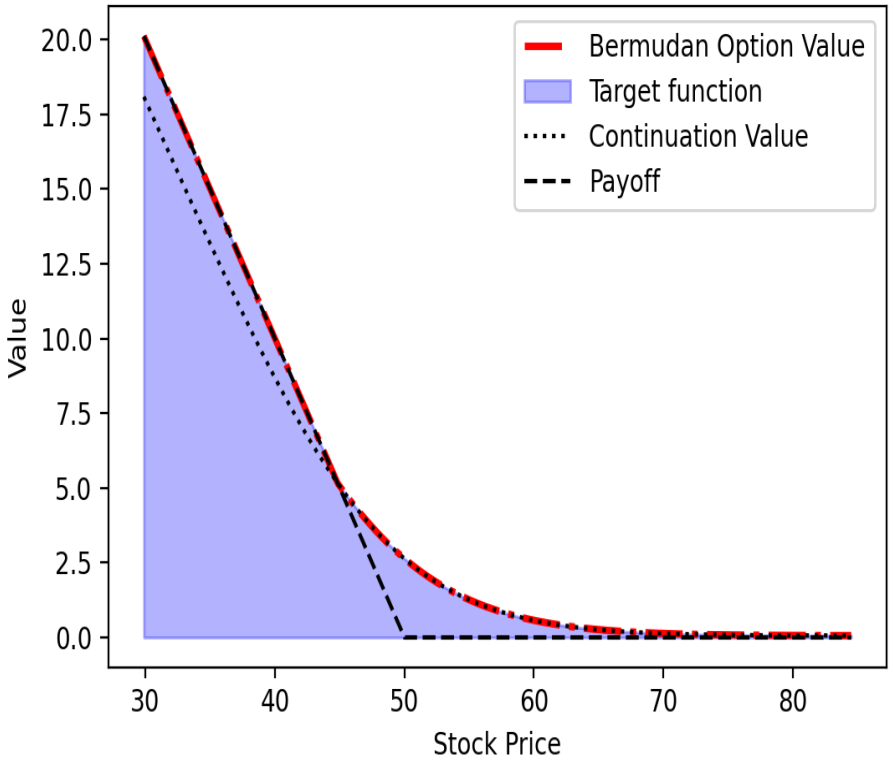
Step 4: Compute Initial Option Value

$$V(0) = \mathbb{E}^{\mathbb{Q}} \left[G_0(z_0; \beta_0) \right].$$

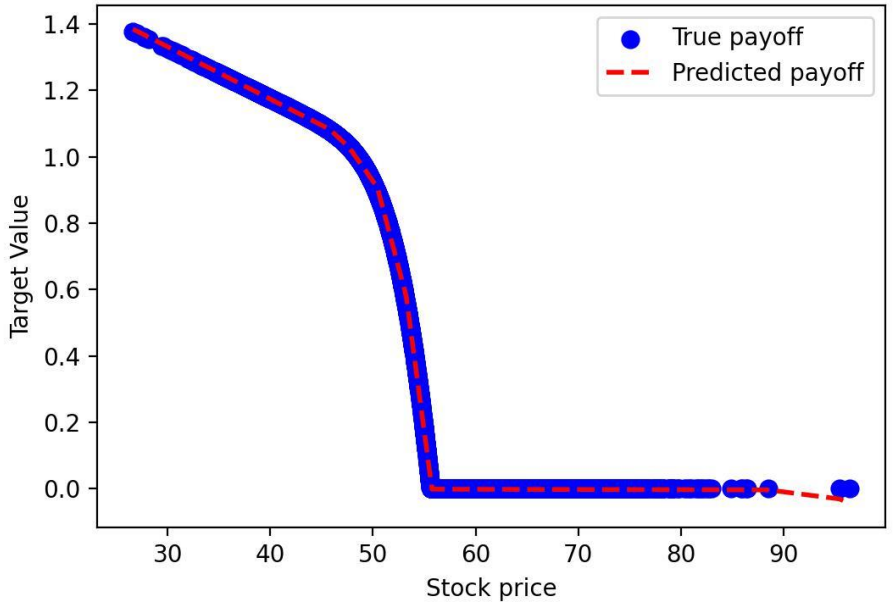
Fully Static Target Function



Semi Static Target Function



NN fit to payoff at prev. Date in Static Case

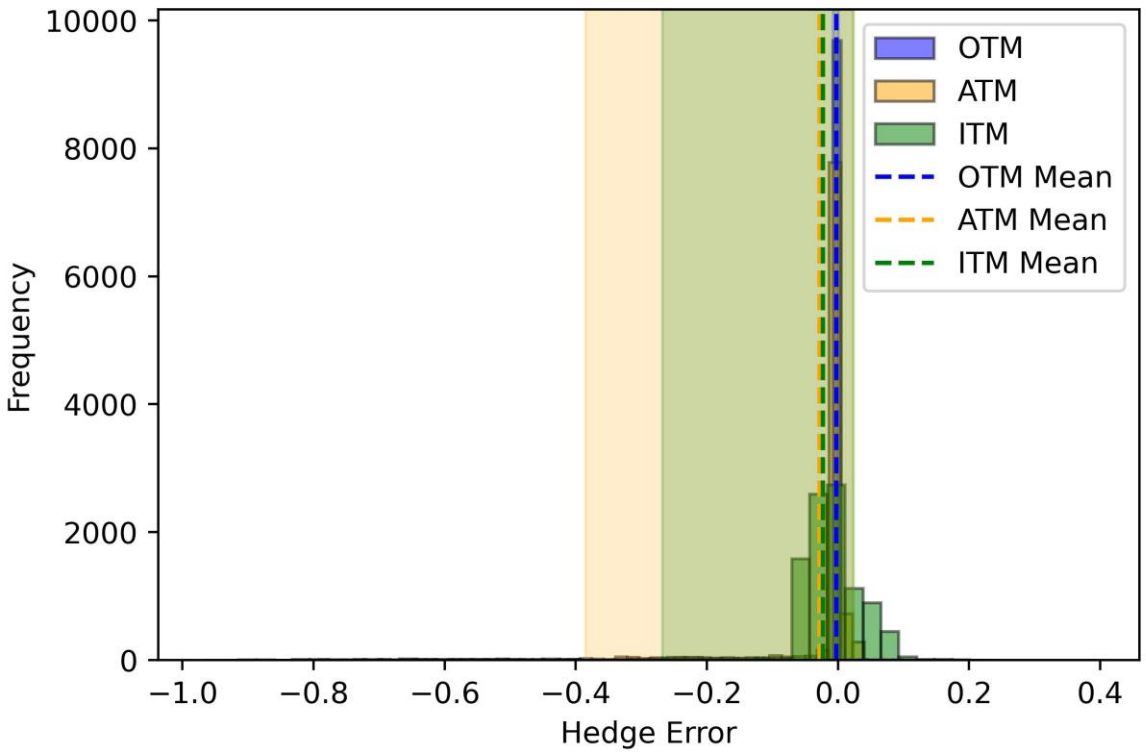


Price of Bermudan Option

S = 50	Static	Semi Static	Binomial
K = 50*1.2	9.4	9.7	9.574
K = 50	2.6764	2.71	2.819
K = 50 *0.8	0.33	0. 307	0.316

Paramaters : no. of nodes = 64, sample size = 25000, learning rate = 5e-4, r = 0.06, T = 1year and no of monitoring dates = 4, vol = 0.2 and batch size = 50 for static case and for other methods same as previously mentioned

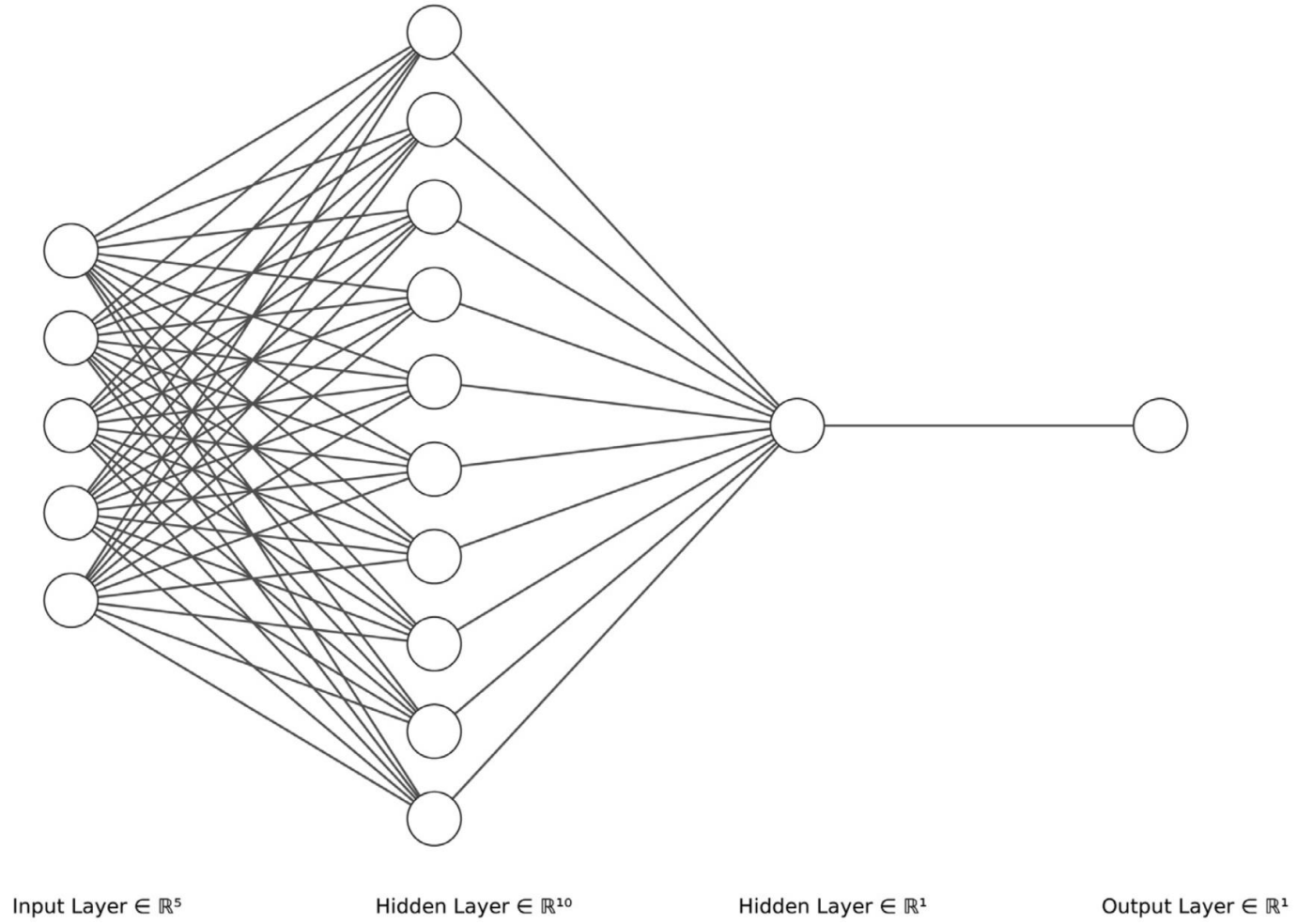
Static Hedging Error (Euros) for Bermudan Put Option



Parameters: S0 = 50, K = 50(ATM), r = 0.06, vol = 0.2, number of simulations = 10000, K = 40(OTM) and K = 60(ITM)

Multi Asset Option Pricing

The neural network architecture chosen for each monitoring date



Continuation Value in Multi Asset Case

- Under the risk-neutral measure, the log-transformed asset price, follows multi variate normal distribution

- With mean vector - $$\mu = \log(S_{t_{m-1}}) + \left(r - \frac{1}{2}\sigma^2\right)\Delta t,$$

- And the covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1d}\sigma_1\sigma_d \\ \rho_{21}\sigma_2\sigma_1 & \sigma_2^2 & \cdots & \rho_{2d}\sigma_2\sigma_d \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d1}\sigma_d\sigma_1 & \rho_{d2}\sigma_d\sigma_2 & \cdots & \sigma_d^2 \end{bmatrix} \Delta t,$$

Contd...

• To compute the conditional expectation, $\mathbb{E}\left[\phi\left(w_i \log(S_{t_m}) + b_i\right) \middle| S_{t_{m-1}}\right]$ the weighted sum of log asset prices we consider

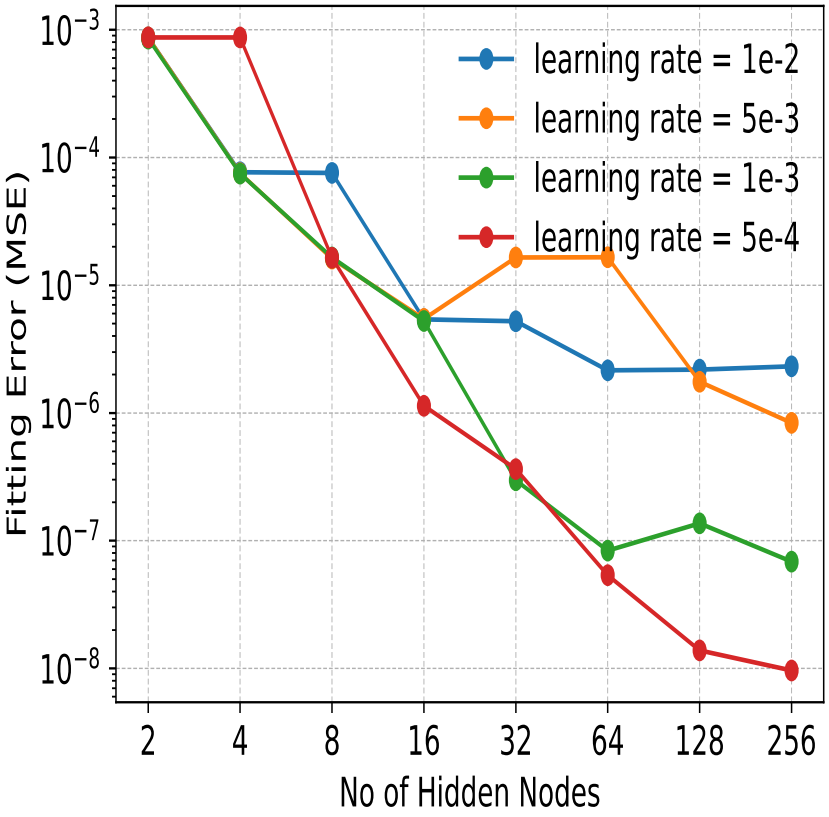
$$Y = w_i^\top \log(S_{t_m}) + b_i$$

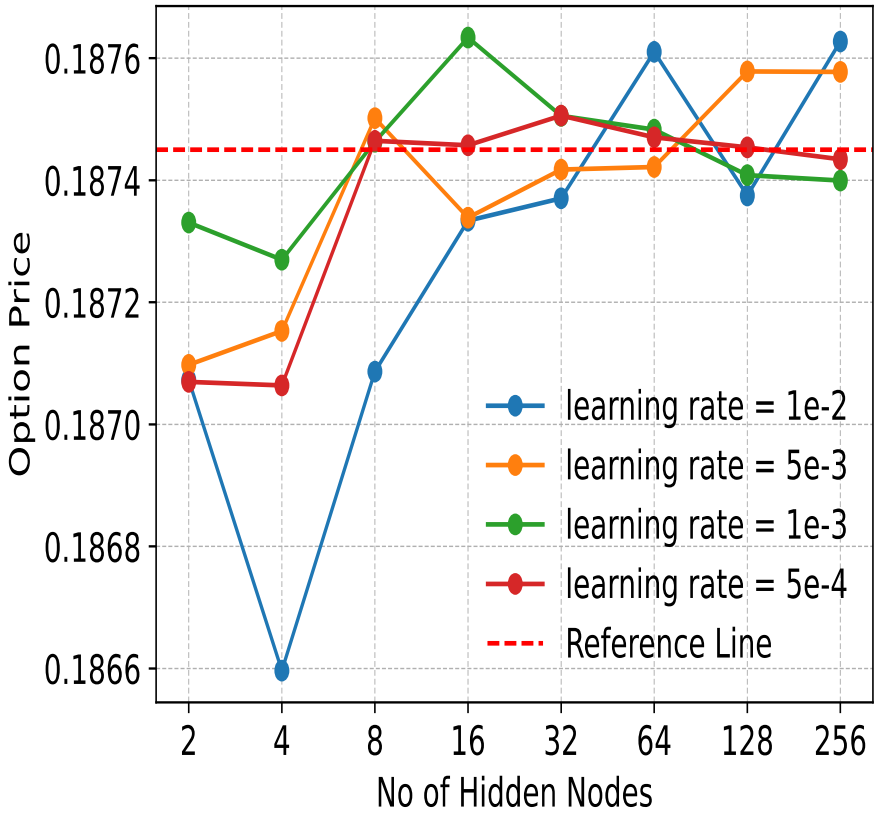
is Random Variable with mean - $\mu_Y = w_i^\top \mu + b_i$ and variance - $\sigma_Y^2 = w_i^\top \Sigma w_i$

$$\mathbb{E}[\max(Y, 0)] = \sigma_Y \frac{1}{\sqrt{2\pi}} e^{-\frac{\mu_Y^2}{2\sigma_Y^2}} + \mu_Y \left[1 - \Phi\left(-\frac{\mu_Y}{\sigma_Y}\right)\right]$$

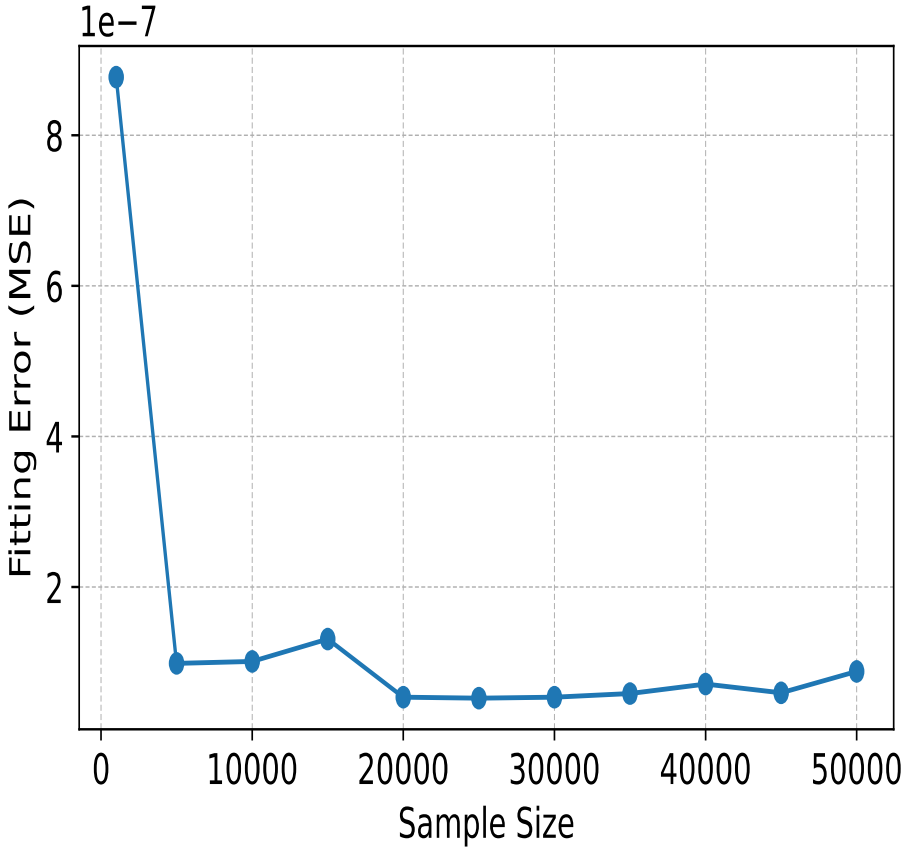
Arithmetic Basket Option European style

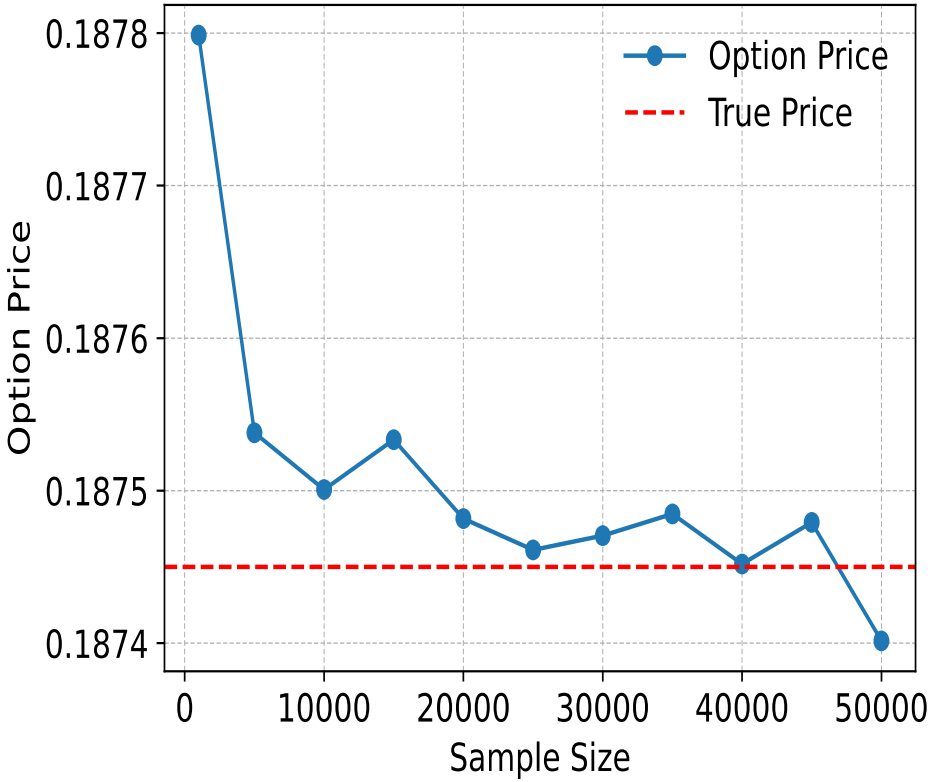
Hyper Parameter Tunning



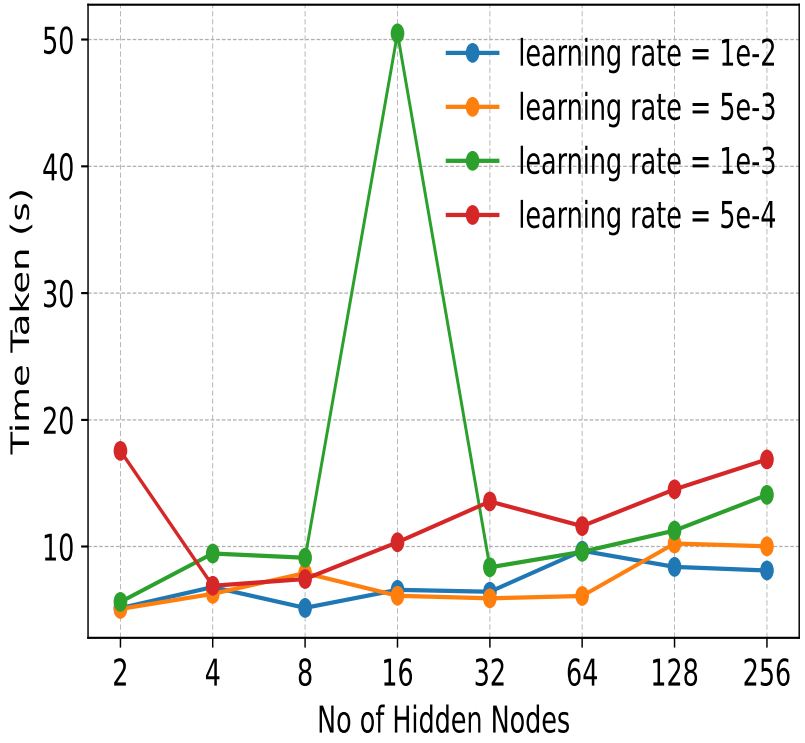


Contd…


1e−7
8
6
4
2
Fitting Error (MSE)
0
10000
20000
30000
40000
50000
Sample Size


0.1878
0.1877
0.1876
0.1875
0.1874
Option Price
Option Price
True Price
0
10000
20000
30000
40000
50000
Sample Size

Contd…



S0 = 1	**RLNN Method**		**Monte Carlo**	
Moneyness	Option Price	Std. Error	Option Price	Std. Error
OTM (K = 0.9)	0.2452	1.30 * 1e-6	0.2454	0.00030
ATM (K = 1.0)	0.1875	1.59 *1e-6	0.1872	0.00026
ITM (K = 1.1)	0.1365	1.46 *1e-6	0.1366	0.00021

Paramaters : no. of nodes = 64, sample size = 25000, learning rate = 5e-4, r = 0.05, T = 1year used to price option in the table

Max Call Bermudan Option with 2 assets

K = 1	RLNN	Binomial
Moneyness	Option Price & Std. Error	Option Price
ITM (S0 = 1.1)	$ 0.24694 (+- 0.000122)	$ 0.24606
OTM (S0 = 0.9)	$ 0.08095 (+- 0.000138)	$ 0.08103
ATM (S0 = 1.0)	$ 0.15510 (+- 0.000129)	$ 0.15382

Arithmetic Basket Bermudan Put Option with 5 Assets

K = 1	RLNN	SGBM*
Moneyness	Option Price & Std. Error	Option Price
OTM (S0 = 1.1)	0.143151372 (+- 6.801e-05)	0.1463
ITM (S0 = 0.9)	0.2157810 (+- 0.0001008)	0.2220
ATM (S0 = 1.0)	$ 0.175693 (+- 0.0001362)	0.1803

Paramaters : same as prev. option and correlation between two assets = 0.5 in max call option

*Ref. Lokeshwar et al (2022) and parameteres same as paper

Towards constructing a tradable hedge RLNN

Outline of this part

- **Recap:** Neural network structure and parameter interpretation.
- **Algorithmic Steps:**
  - Fitting the unconstrained neural network.
  - Mapping neurons to portfolio logic.
  - Constructing theoretical and real-instrument portfolios.
- **Benchmarking:** Min–max linear programming for portfolio replication.
- **Experimental Setup:** Simulation, data & parameters
- **Results and Discussion:** Comparisons of unconstrained and tradable portfolios.
- **Conclusion:** Insights on RLNN-based hedging and practical implications.

Step 1: Fit the Neural Network and Interpret Each Neuron (2/2)

Neuron Interpretation:

After solving the above problem, extract each hidden neuron's parameters $\big(w_{1,i}, b_{1,i}, w_{2,i}\big)$.

- Use the sign of $w_{1,i}$ and $b_{1,i}$ to classify the neuron as forward-like, call-like, or put-like.
- Compute the implied strike (if $w_{1,i} \neq 0$):

$$K_i = -\frac{b_{1,i}}{w_{1,i}}.$$

- Define the effective units:

$$u_i = w_{2,i}\, w_{1,i}.$$

Then form theoretically optimal portfolio:

$$\Pi_{\text{theoretical}}(S) = b_2 + \sum_{i=1}^{H} \big[u_i\, h_i(S)\big],$$

Note yields a set of payoffs $h_i(\cdot)$ that may have continuous or non-standard strikes.

Neuron-Level Payoff

For each neuron H_i, define its single-neuron payoff $h_i(S)$ as follows:

1. Forward-like: If $w_{1,i} \geq 0$ and $b_{1,i} \geq 0$, then

$$h_i(S) := S.$$

2. Call-like: If $w_{1,i} > 0$ and $b_{1,i} < 0$, then

$$K_i := -\frac{b_{1,i}}{w_{1,i}}, \quad h_i(S) := \max\big(S - K_i,\, 0\big).$$

3. Put-like: If $w_{1,i} < 0$ and $b_{1,i} > 0$, then

$$K_i := -\frac{b_{1,i}}{w_{1,i}}, \quad h_i(S) := \max\big(K_i - S,\, 0\big).$$

4. Otherwise: Skip neuron H_i.

Target Payoff Π_{target}

Let $\Pi_{\text{target}}(S)$ be the payoff we ultimately wish to replicate in practice (e.g. an exotic derivative or a portfolio to hedge). For this experiment we limit ourselves to a simple European call.

Naive Mapped Portfolio $\Pi_{\text{naive-Tradeable}}$

Given the unconstrained network's neuron-level strikes $\{K_i\}$ and units $\{u_i\}$, assign each neuron H_i to its nearest traded strike $\tilde{K}_\ell$ by $\min |K_i - \tilde{K}|$, without changing its units:

$$\Pi_{\text{naive-Tradeable}}(S) = b_2 + \sum_{i=1}^{H} \big[u_i \, g_i(S) \big],$$

where

$$g_i(S, \tilde{K}_\ell) := h_i(S, K_i)$$

hence:

$$g_i(S) = \begin{cases} \max(S - \tilde{K}_\ell, 0), & \text{if } w_{1,i} > 0 \text{ and } b_{1,i} < 0, \\ \max(\tilde{K}_\ell - S, 0), & \text{if } w_{1,i} < 0 \text{ and } b_{1,i} > 0, \\ S, & \text{if } w_{1,i} \geq 0 \text{ and } b_{1,i} \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

Real-Instrument Portfolio $\Pi_{\text{refitted\_tradable}}$

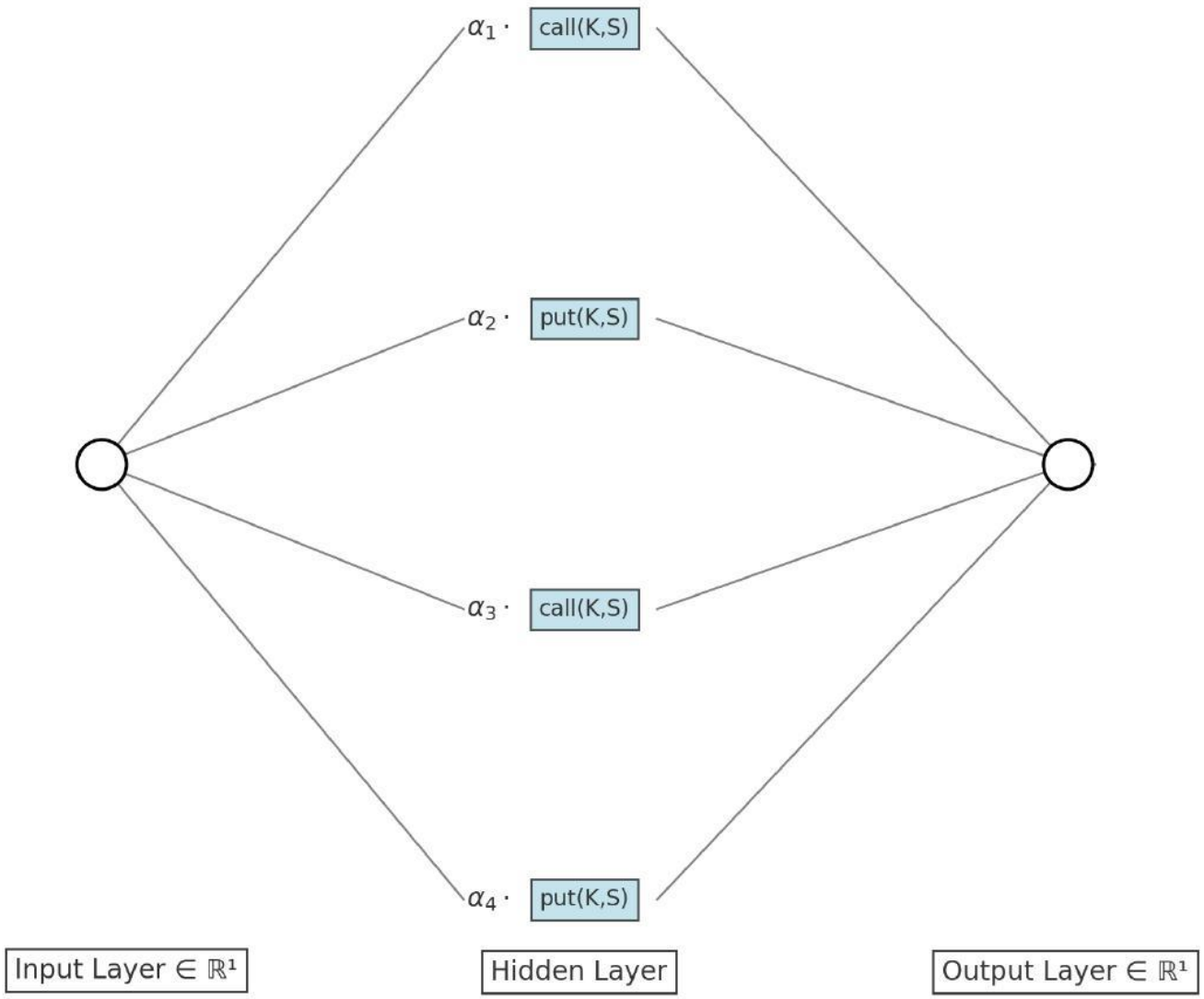
Suppose we have a finite set $\{g_\ell(S)\}_{\ell=1}^{L}$ of real, exchange-traded instruments (calls/puts/forwards at discrete strikes $\{\tilde{K}_\ell\}$). Then

$$\Pi_{\text{refitted\_tradable}}(S) = \alpha_0 + \sum_{\ell=1}^{L} \alpha_\ell \, g_\ell(S),$$

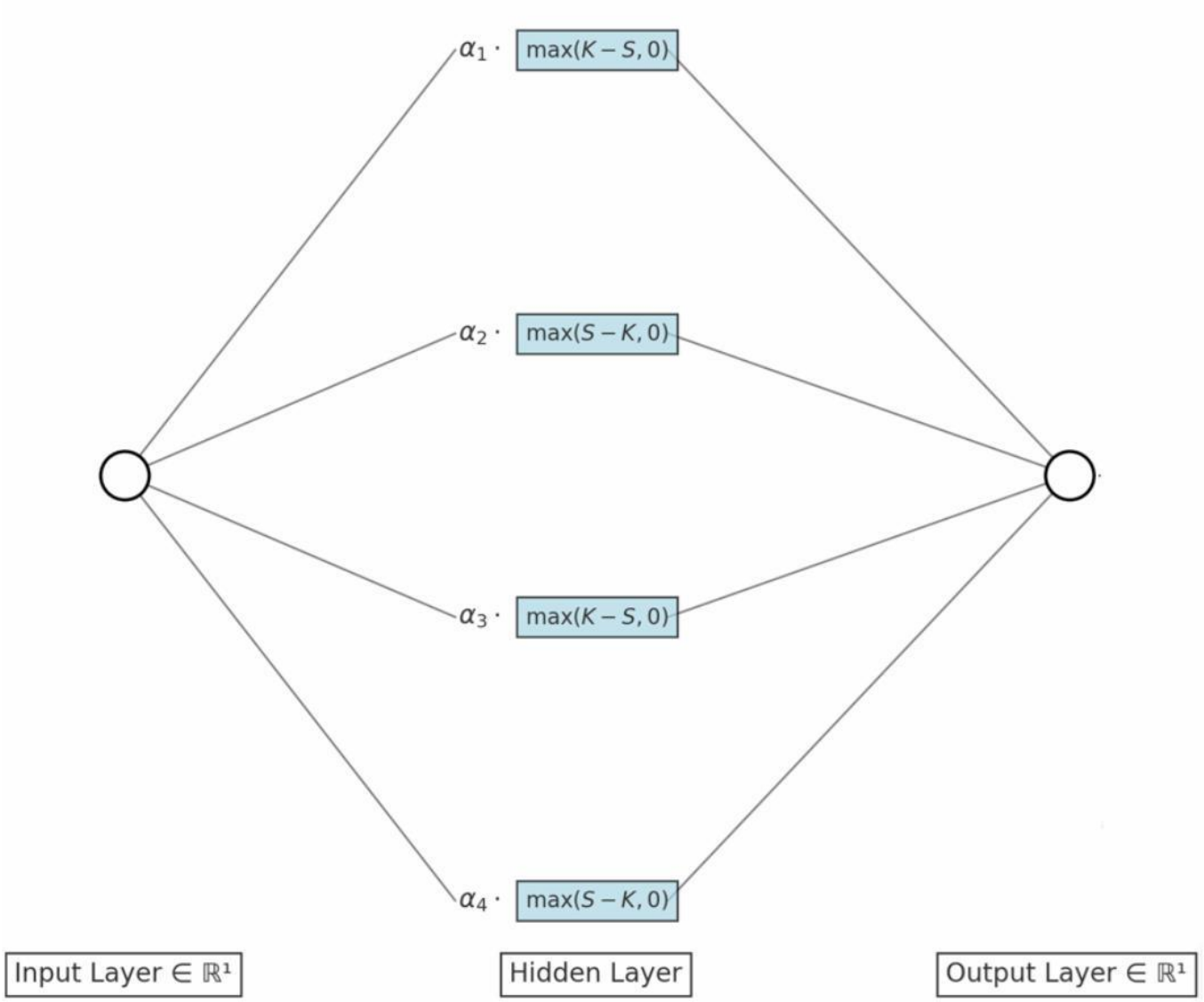
where α_ℓ are new weights in units of each real instrument. A least-squares fit would choose $\{\alpha_\ell\}$ to minimize

$$\sum_{m=1}^{M} \Big[\Pi_{\text{target}}(S_m) - \Big(\alpha_0 + \sum_{\ell=1}^{L} \alpha_\ell \, g_\ell(S_m) \Big) \Big]^2.$$

Real-Instrument Portfolio $\Pi_{\text{refitted\_tradable}}$ NN structure



Real-Instrument Portfolio $\Pi_{\text{refitted\_tradable}}$ NN structure



Benchmark: Min–Max Linear Programming Approach (1/2)

In some applications, one may wish to minimize the *maximum* replication error (rather than the mean-squared error). Let $\Pi_{\text{target}}(S_m)$ be the target payoff values at $\{S_m\}_{m=1}^{M}$, and let $\{g_\ell(S)\}_{\ell=1}^{L}$ be the same finite set of real instruments (calls, puts, or forwards). We seek α_ℓ that solve:

$$\min_{\alpha_0,\,\{\alpha_\ell\},\,z} \quad E \quad \text{subject to} \quad \left|\Pi_{\text{target}}(S_m) - \left(\alpha_0 + \sum_{\ell=1}^{L} \alpha_\ell\, g_\ell(S_m)\right)\right| \le E, \quad m = 1, \ldots, M$$

Benchmark: Min–Max Linear Programming Approach (2/2)

This is a linear program in $\alpha_0, \{\alpha_\ell\}, z$ when each $g_\ell(\cdot)$ is affine or piecewise-linear in S. In practice:

1. **Define linear constraints:** For each S_m,

$$-E \leq \Pi_{\text{target}}(S_m) - \left(\alpha_0 + \sum_{\ell=1}^{L} \alpha_\ell \, g_\ell(S_m)\right) \leq E.$$

2. **Solve for** $\{\alpha_\ell\}$: Minimizing E under these constraints yields the smallest maximum error over $\{S_m\}$.

3. **Construct** $\Pi_{\text{LP-direct}}$:

$$\Pi_{\text{LP-direct}}(S) = \hat{\alpha}_0 + \sum_{\ell=1}^{L} \hat{\alpha}_\ell \, g_\ell(S).$$

Optionally select a subset of instruments obtained by interpreting the weights of the unconstrained neural network to do the fit, yielding $\Pi_{\text{LP-refitted}}$.

Experimental setup

(A) Fit an unconstrained neural network on data $\left\{(S^{(n)}, \Pi_{\text{target}}(S^{(n)}))\right\}$ using the least-squares objective.

(B) Interpret each hidden neuron's sign and bias to get $\{K_i\}$ and $\{u_i\}$. Build $\Pi_{\text{theoretical}}(S)$.

(C) Naive mapped portfolio: Snap each strike K_i to $\tilde{K}_\ell$ without altering the weights u_i, yielding $\Pi_{\text{naive-Tradeable}}$.

(D) Perform a second least-squares refit to obtain $\Pi_{\text{refitted-Tradable}}$.

(E) *Benchmark: Min–Max LP approach*: Solve for $\Pi_{\text{LP-direct}}$ directly. Optionally refit or reduce instruments, yielding $\Pi_{\text{LP-refitted}}$. Thus, we can generate multiple approximate portfolios:
$\Pi_{\text{naive-Tradeable}}, \Pi_{\text{refitted-Tradeable}}, \Pi_{\text{LP-direct}}, \Pi_{\text{LP-refitted}}$.

Data, Simulation, & Experimental parameters

Stock and option data retrieved via `yfinance` API (expiration: 17-01-2025). The initial stock price S_0 corresponds to the closing price 1 month before expiration. Stock prices are simulated as Geometric Brownian Motion (GBM) using:

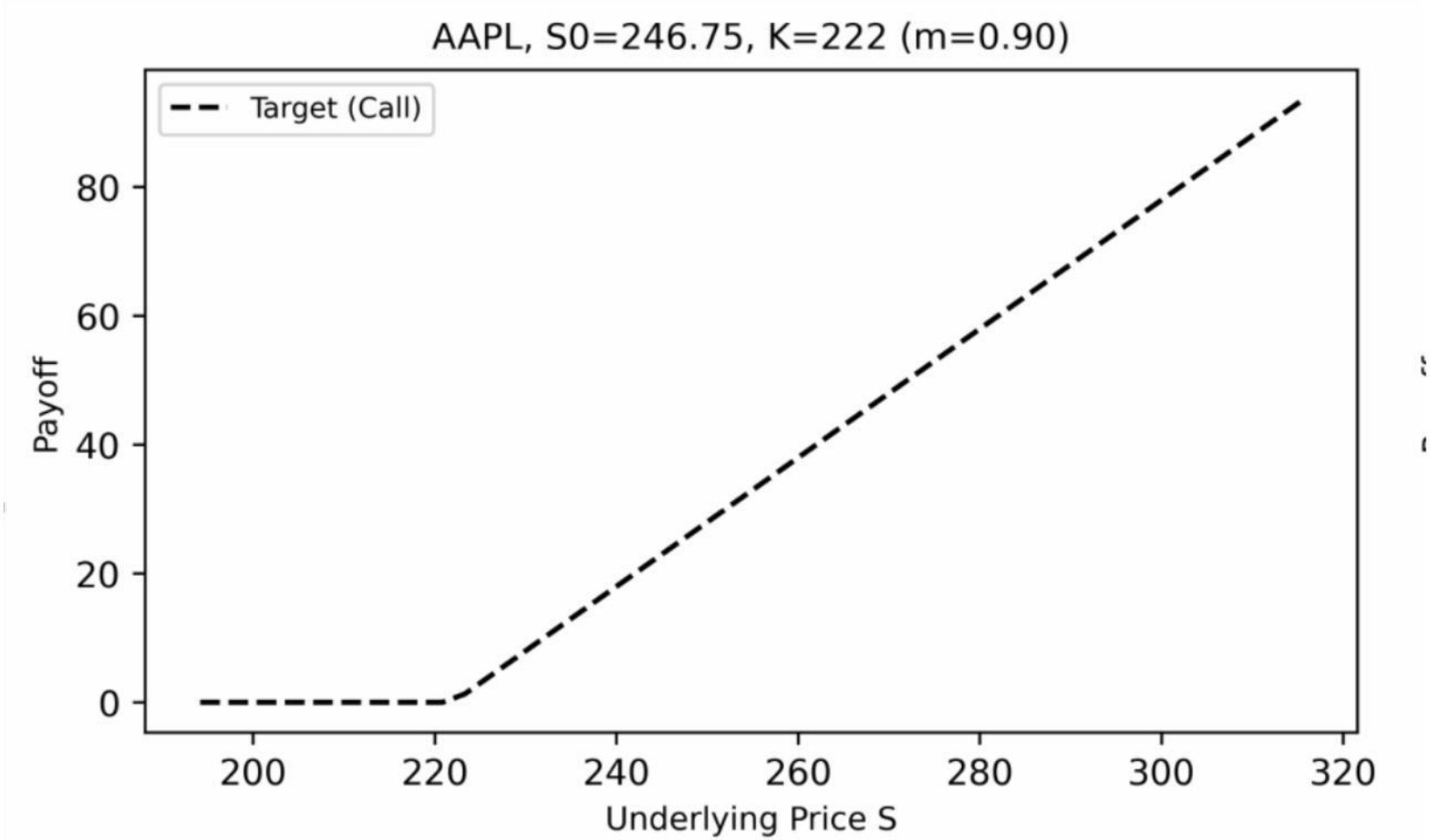
$$dS_t = \mu S_t \, dt + \sigma S_t \, dW_t,$$

where μ is drift, σ is volatility, and W_t is a Wiener process.

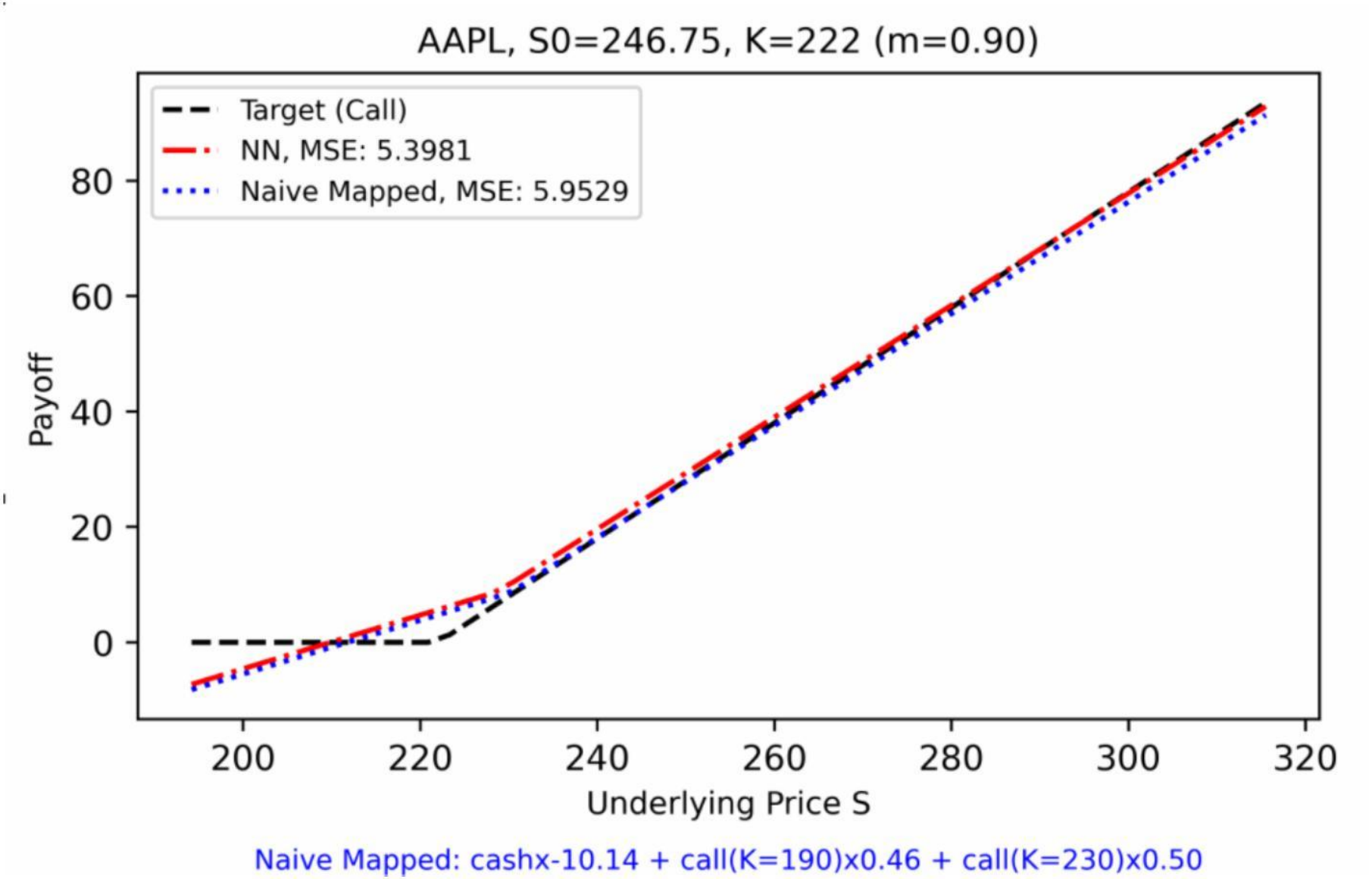
Parameters:

- Instrument: Simple European Call
- Moneyness: [0.9, 1.0, 1.1]
- Ticker List: ['AAPL', 'AMZN', 'SPY']
- Period: '1mo', Paths: 3000
- Learning Rate: 0.01, Hidden Units: 4
- Training Epochs: 300 (Real Instrument: 500)
- Risk-Free Rate (r): 0.05, Volatility (σ): 0.2

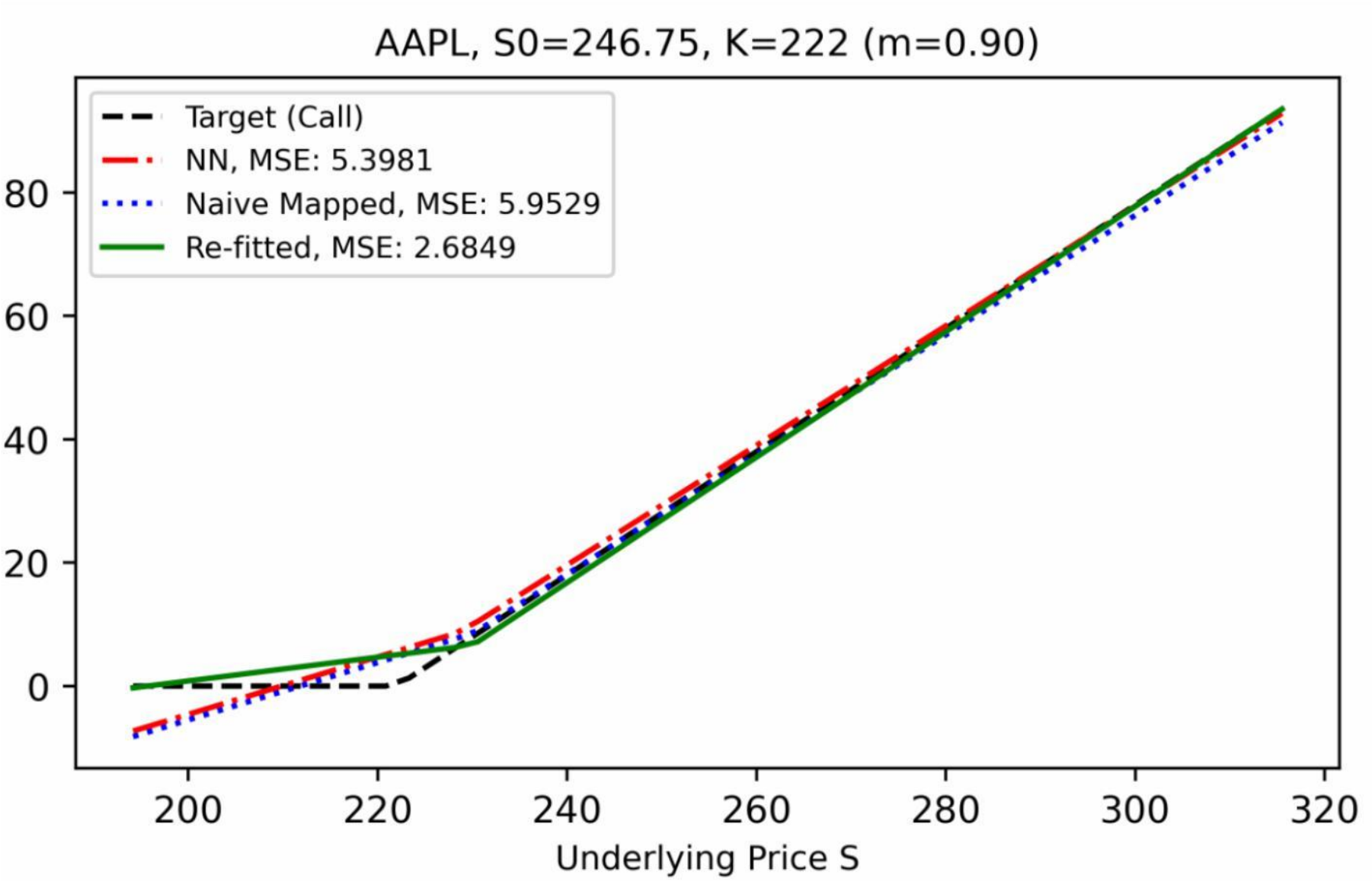
Results


AAPL, S0=246.75, K=222 (m=0.90)
Target (Call)
Payoff
0
20
40
60
80
200
220
240
260
280
300
320
Underlying Price S

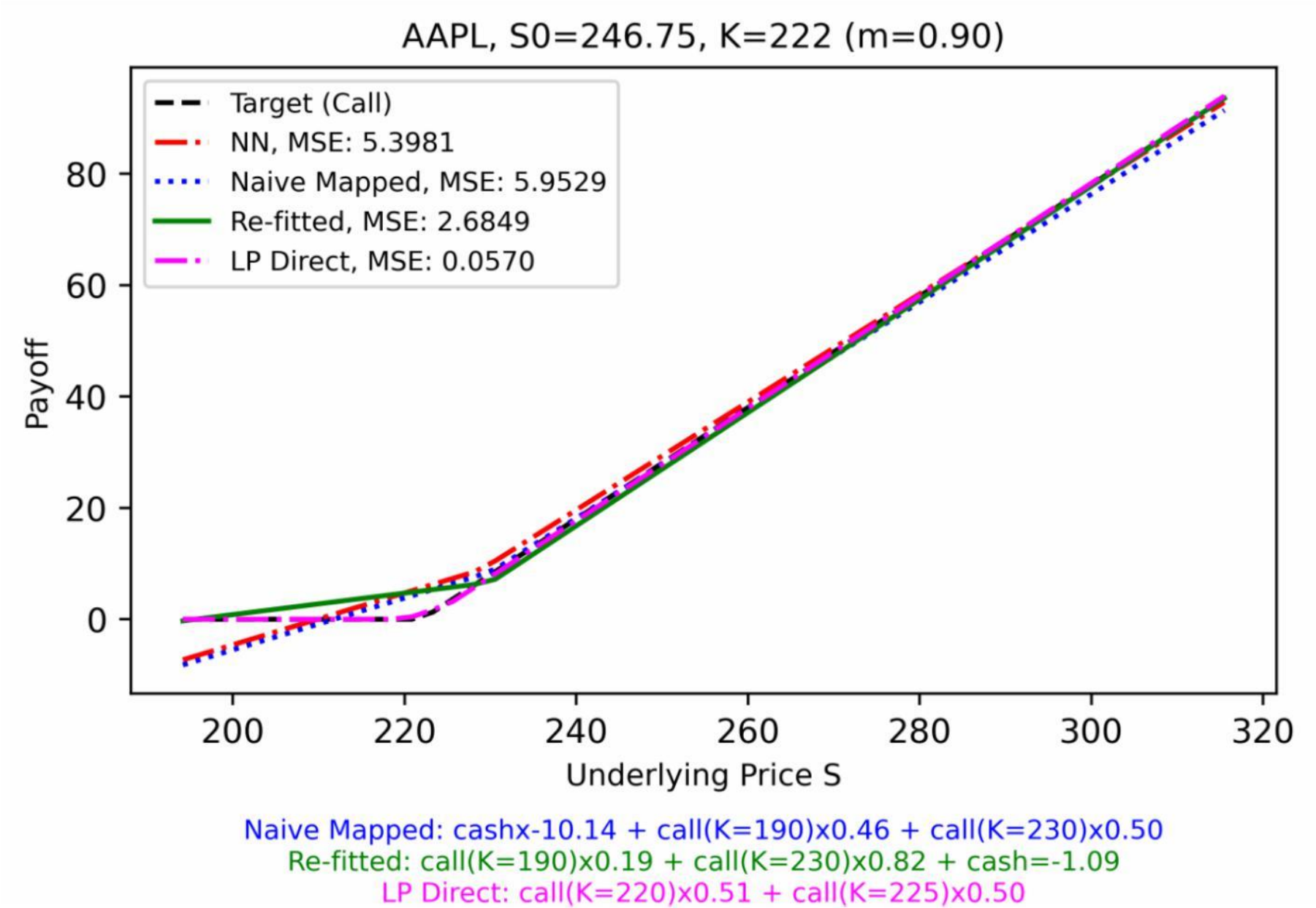
Results


AAPL, S0=246.75, K=222 (m=0.90)
Target (Call)
NN, MSE: 5.3981
Naive Mapped, MSE: 5.9529
Payoff
0
20
40
60
80
200
220
240
260
280
300
320
Underlying Price S
Naive Mapped: cashx-10.14 + call(K=190)x0.46 + call(K=230)x0.50

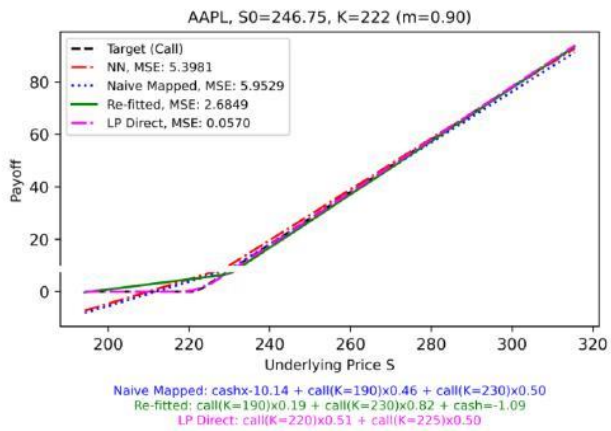
Results

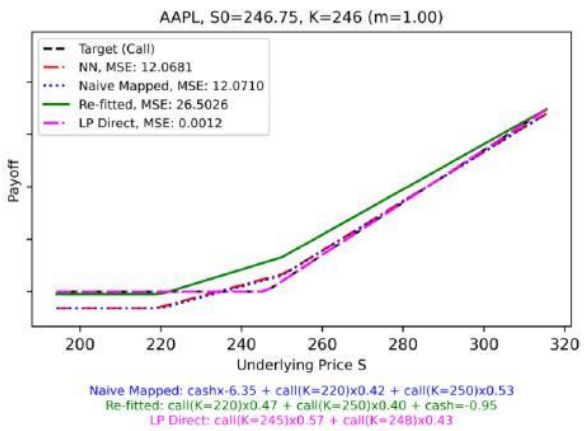

AAPL, S0=246.75, K=222 (m=0.90)
Target (Call)
NN, MSE: 5.3981
Naive Mapped, MSE: 5.9529
Re-fitted, MSE: 2.6849
80
60
40
20
0
200
220
240
260
280
300
320
Underlying Price S
Naive Mapped: cashx-10.14 + call(K=190)x0.46 + call(K=230)x0.50
Re-fitted: call(K=190)x0.19 + call(K=230)x0.82 + cash=-1.09

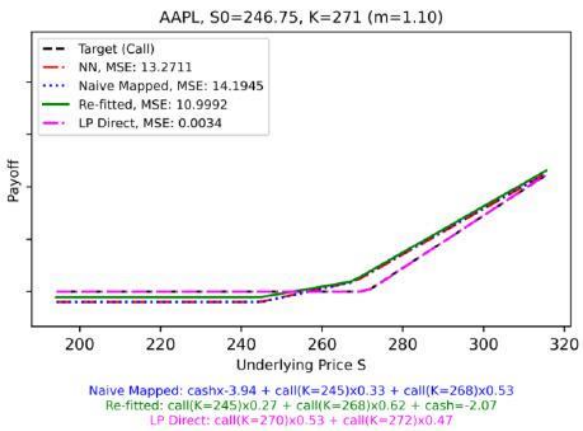
Results


AAPL, S0=246.75, K=222 (m=0.90)
Target (Call)
NN, MSE: 5.3981
Naive Mapped, MSE: 5.9529
Re-fitted, MSE: 2.6849
LP Direct, MSE: 0.0570
Payoff
0
20
40
60
80
200
220
240
260
280
300
320
Underlying Price S
Naive Mapped: cashx-10.14 + call(K=190)x0.46 + call(K=230)x0.50
Re-fitted: call(K=190)x0.19 + call(K=230)x0.82 + cash=-1.09
LP Direct: call(K=220)x0.51 + call(K=225)x0.50

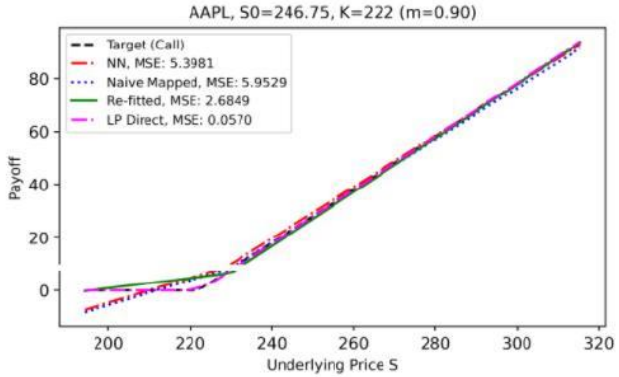
Results

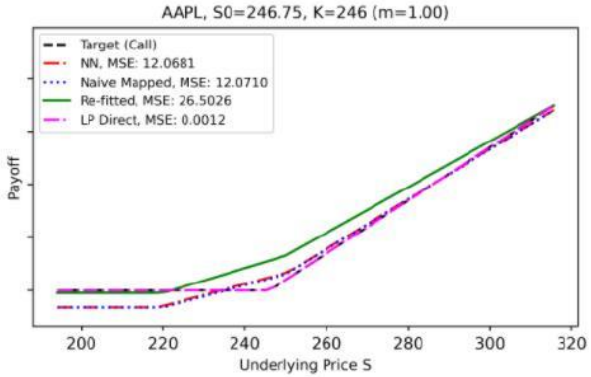

AAPL, S0=246.75, K=222 (m=0.90)
Target (Call)
NN, MSE: 5.3981
Naive Mapped, MSE: 5.9529
Re-fitted, MSE: 2.6849
LP Direct, MSE: 0.0570
Payoff
Underlying Price S
Naive Mapped: cashx-10.14 + call(K=190)x0.46 + call(K=230)x0.50
Re-fitted: call(K=190)x0.19 + call(K=230)x0.82 + cash=-1.09
LP Direct: call(K=220)x0.51 + call(K=225)x0.50

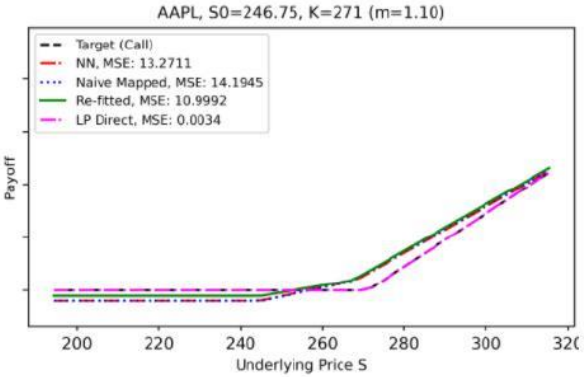

AAPL, S0=246.75, K=246 (m=1.00)
Target (Call)
NN, MSE: 12.0681
Naive Mapped, MSE: 12.0710
Re-fitted, MSE: 26.5026
LP Direct, MSE: 0.0012
Payoff
Underlying Price S
Naive Mapped: cashx-6.35 + call(K=220)x0.42 + call(K=250)x0.53
Re-fitted: call(K=220)x0.47 + call(K=250)x0.40 + cash=-0.95
LP Direct: call(K=245)x0.57 + call(K=248)x0.43

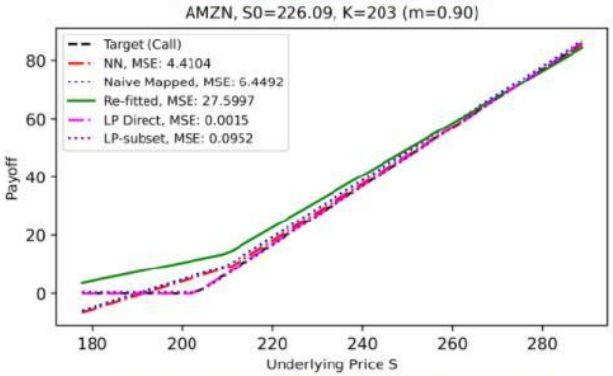

AAPL, S0=246.75, K=271 (m=1.10)
Target (Call)
NN, MSE: 13.2711
Naive Mapped, MSE: 14.1945
Re-fitted, MSE: 10.9992
LP Direct, MSE: 0.0034
Payoff
Underlying Price S
Naive Mapped: cashx-3.94 + call(K=245)x0.33 + call(K=268)x0.53
Re-fitted: call(K=245)x0.27 + call(K=268)x0.62 + cash=-2.07
LP Direct: call(K=270)x0.53 + call(K=272)x0.47

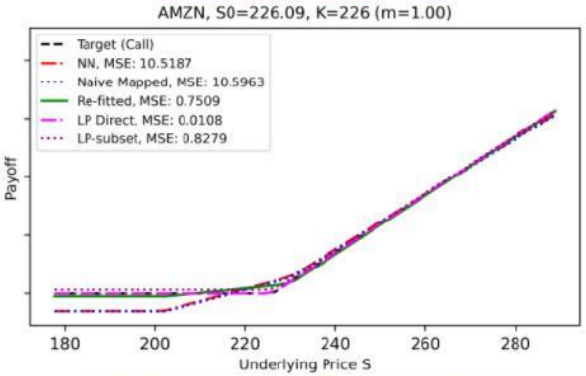
Results

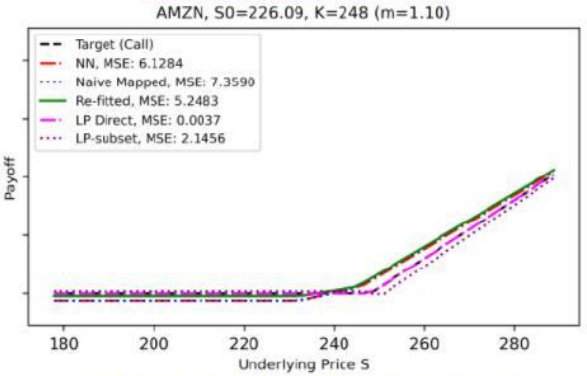

AAPL, S0=246.75, K=222 (m=0.90)
Target (Call)
NN, MSE: 5.3981
Naive Mapped, MSE: 5.9529
Re-fitted, MSE: 2.6849
LP Direct, MSE: 0.0570
Payoff
Underlying Price S
Naive Mapped: cashx-10.14 + call(K=190)x0.46 + call(K=230)x0.50
Re-fitted: call(K=190)x0.19 + call(K=230)x0.82 + cash=-1.09
LP Direct: call(K=220)x0.51 + call(K=225)x0.50

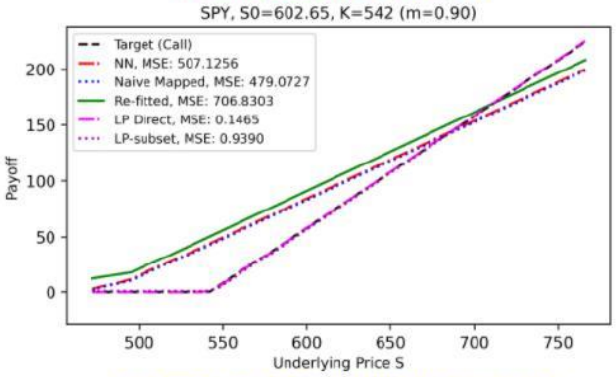

AAPL, S0=246.75, K=246 (m=1.00)
Target (Call)
NN, MSE: 12.0681
Naive Mapped, MSE: 12.0710
Re-fitted, MSE: 26.5026
LP Direct, MSE: 0.0012
Payoff
Underlying Price S
Naive Mapped: cashx-6.35 + call(K=220)x0.42 + call(K=250)x0.53
Re-fitted: call(K=220)x0.47 + call(K=250)x0.40 + cash=-0.95
LP Direct: call(K=245)x0.57 + call(K=248)x0.43

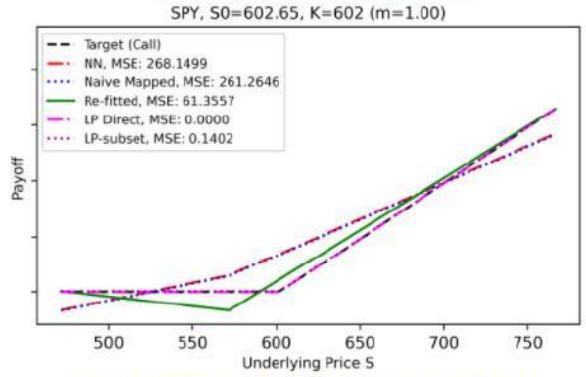

AAPL, S0=246.75, K=271 (m=1.10)
Target (Call)
NN, MSE: 13.2711
Naive Mapped, MSE: 14.1945
Re-fitted, MSE: 10.9992
LP Direct, MSE: 0.0034
Payoff
Underlying Price S
Naive Mapped: cashx-3.94 + call(K=245)x0.33 + call(K=268)x0.53
Re-fitted: call(K=245)x0.27 + call(K=268)x0.62 + cash=-2.07
LP Direct: call(K=270)x0.53 + call(K=272)x0.47

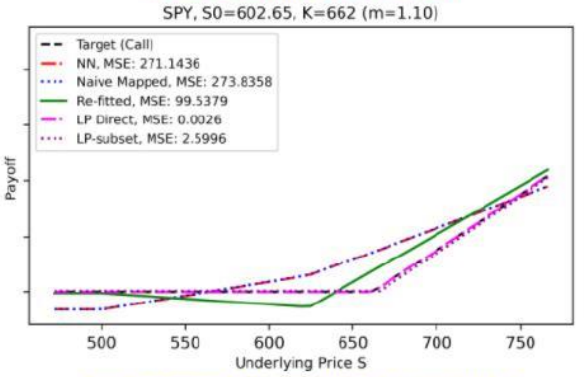

AMZN, S0=226.09, K=203 (m=0.90)
Target (Call)
NN, MSE: 4.4104
Naive Mapped, MSE: 6.4492
Re-fitted, MSE: 27.5997
LP Direct, MSE: 0.0015
LP-subset, MSE: 0.0952
Payoff
Underlying Price S
Naive Mapped: cashx-9.45 + call(K=170)x0.48 + call(K=210)x0.50
Re-fitted: call(K=170)x0.32 + call(K=210)x0.58 + cash=-1.09
LP Direct: call(K=202)x0.77 + call(K=205)x0.23


AMZN, S0=226.09, K=226 (m=1.00)
Target (Call)
NN, MSE: 10.5187
Naive Mapped, MSE: 10.5963
Re-fitted, MSE: 0.7509
LP Direct, MSE: 0.0108
LP-subset, MSE: 0.8279
Payoff
Underlying Price S
Naive Mapped: cashx-5.95 + call(K=202)x0.41 + call(K=230)x0.54
Re-fitted: call(K=202)x0.15 + call(K=230)x0.86 + cash=-0.94
LP Direct: call(K=225)x0.51 + call(K=228)x0.50


AMZN, S0=226.09, K=248 (m=1.10)
Target (Call)
NN, MSE: 6.1284
Naive Mapped, MSE: 7.3590
Re-fitted, MSE: 5.2483
LP Direct, MSE: 0.0037
LP-subset, MSE: 2.1456
Payoff
Underlying Price S
Naive Mapped: cashx-2.45 + call(K=232)x0.35 + call(K=245)x0.57
Re-fitted: call(K=232)x0.25 + call(K=245)x0.67 + cash=-0.88
LP Direct: call(K=248)x0.73 + call(K=250)x0.27


SPY, S0=602.65, K=542 (m=0.90)
Target (Call)
NN, MSE: 507.1256
Naive Mapped, MSE: 479.0727
Re-fitted, MSE: 706.8303
LP Direct, MSE: 0.1465
LP-subset, MSE: 0.9390
Payoff
Underlying Price S
Naive Mapped: cashx-18.84 + call(K=410)x0.34 + call(K=495)x0.35
Re-fitted: call(K=410)x0.23 + call(K=495)x0.47 + cash=-2.25
LP Direct: call(K=540)x0.45 + call(K=545)x0.56


SPY, S0=602.65, K=602 (m=1.00)
Target (Call)
NN, MSE: 268.1499
Naive Mapped, MSE: 261.2646
Re-fitted, MSE: 61.3557
LP Direct, MSE: 0.0000
LP-subset, MSE: 0.1402
Payoff
Underlying Price S
Naive Mapped: cashx-17.58 + call(K=470)x0.31 + call(K=572)x0.34
Re-fitted: call(K=470)x-0.17 + call(K=572)x1.10 + cash=0.75
LP Direct: call(K=602)x1.00


SPY, S0=602.65, K=662 (m=1.10)
Target (Call)
NN, MSE: 271.1436
Naive Mapped, MSE: 273.8358
Re-fitted, MSE: 99.5379
LP Direct, MSE: 0.0026
LP-subset, MSE: 2.5996
Payoff
Underlying Price S
Naive Mapped: cashx-15.83 + call(K=500)x0.25 + call(K=624)x0.31
Re-fitted: call(K=500)x-0.10 + call(K=624)x0.97 + cash=-1.24
LP Direct: call(K=660)x0.57 + call(K=665)x0.43

Discussion

1. RLNN enables semi-static hedging of complex Bermudan options, offering interpretable hedges with low rebalancing frequency using simple European options in single asset case.
2. Static hedging is achievable and allows for even lower rebalancing frequency than semi-static.
3. RLNN can price Multi-asset derivatives. However, a low number of experiments and less interpretable network structures (in the case of fully connected networks) and reduce benefit of method (i.e. hedge complex derivative with complex derivative instead of complex with simple)
4. In selected high-volume stocks using real data, real payoffs can be approximated effectively through both naive mapping and constrained real NN fitting. Results suggest tradable constrained RLNN hedging could be feasible, although only simple Eeuropeans were hedged and Naive and Constrained NN methods did n'ot outperform the benchmark min-max LP solution.
5. Interpreting the neural network fit could provide an initial subset of options for the LP problem. This reduces arbitrariness/opaqueness in selecting initial subset of instruments for LP problem to hedge complex derivatives and can lowers the LP problem's dimensionality.

Conclusion

We extended the RLNN algorithm to multi-asset and fully static settings, demonstrating accurate pricing and hedging for complex derivatives. Additionally we presented a method for inferring tradable replicating portfolio for simple European options from NN weights for selected high-volume stocks.

Thank you! Questions?

Appendix: Bounds Computation

- **Lower Bound:**
  - Simulate new paths.
  - Use RLNN continuation values to determine stopping times.
  - Compute discounted payoffs:

$$\text{Lower Bound} = \frac{1}{N_L} \sum_{n=1}^{N_L} \frac{h(S_{\tau(n)})}{B_{\tau(n)}}$$

- **Upper Bound:**
  - Use dual formulation with martingales:

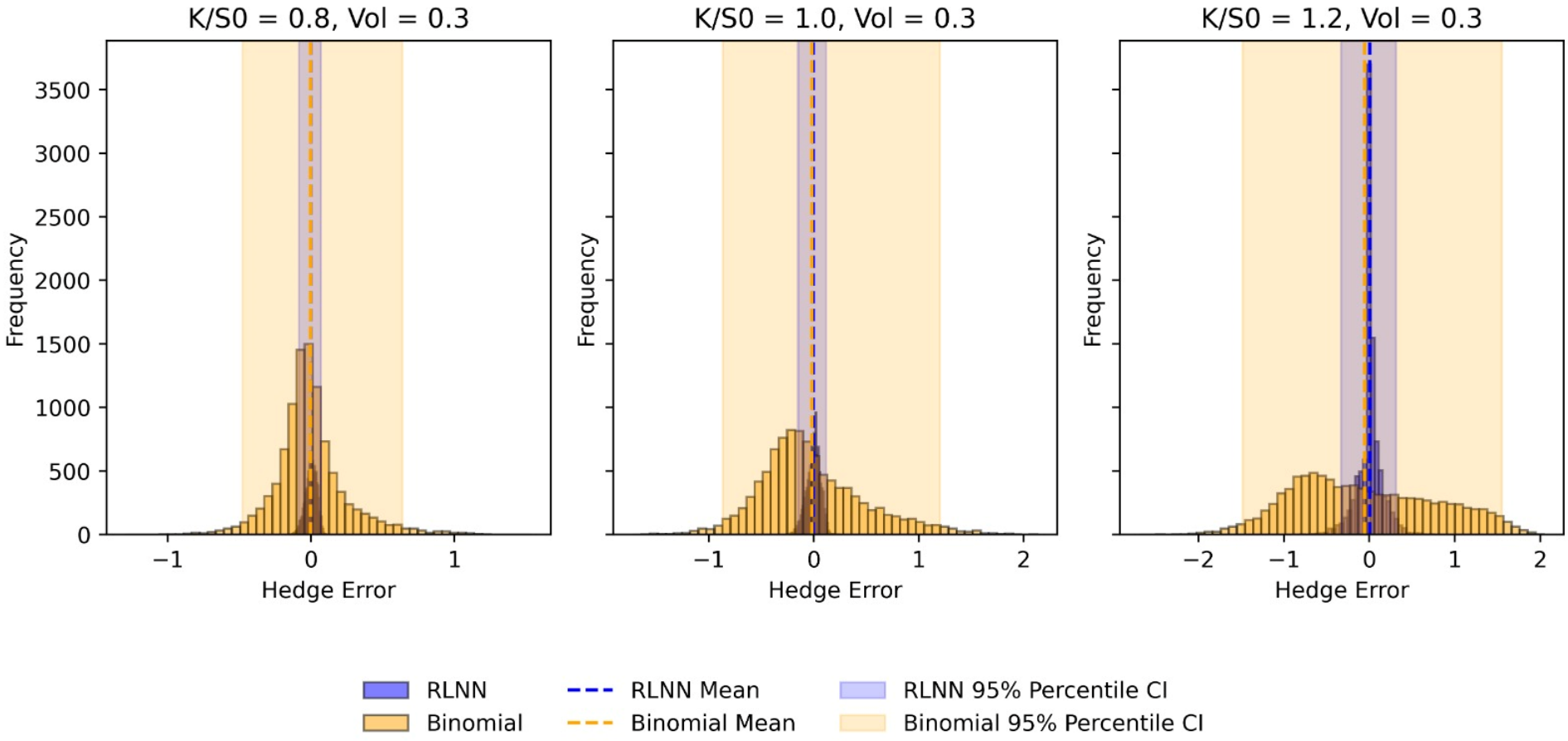
$$M_t = \sum_{i=0}^{m-1} \left(\frac{\tilde{G}_{\beta_{t_i+1}}(S_{t_i+1})}{B_{t_i+1}} - \frac{Q_{t_i}(S_{t_i})}{B_{t_i}} \right)$$

  - Construct the upper bound:

$$\text{Upper Bound} = \mathbb{E}\left[\max_t \left(\frac{h(S_t)}{B_t} - M_t \right) \right]$$

Advantages:

- **Efficient:** Avoids costly sub-simulations.
- **Accurate:** Tighter bounds for Bermudan-style derivatives.
- **Versatile:** Handles multi-asset, path-dependent claims effectively.


K/S0 = 0.8, Vol = 0.3
K/S0 = 1.0, Vol = 0.3
K/S0 = 1.2, Vol = 0.3
Frequency
Hedge Error
RLNN
Binomial
RLNN Mean
Binomial Mean
RLNN 95% Percentile CI
Binomial 95% Percentile CI

Min–Max LP: Setup (1/2)

Goal: Replicate a target payoff $\Pi_{\text{target}}(S)$ by a linear combination of simpler instrument payoffs $\{g_\ell(S)\}_{\ell=1}^{L}$ *while minimizing the maximum replication error* over sampled prices $\{S_i\}_{i=1}^{N}$.

Mathematical Problem:

$$\min_{\{\alpha_\ell\},\, E} \; E \quad \text{subject to} \quad -E \;\le\; \Pi_{\text{target}}(S_i) - \sum_{\ell=1}^{L} \alpha_\ell \, g_\ell(S_i) \;\le\; E, \quad i = 1, \ldots, N.$$

- $\{\alpha_\ell\}$ are the weights (or quantities) of each tradable payoff g_ℓ.
- E is a scalar bounding the *worst-case* (i.e. maximum) absolute error.

Min–Max LP: Setup (2/2)

Matrix Formulation:

- Let $p_{i,\ell} := g_\ell(S_i)$.
- Build the payoff matrix $\mathbf{P} \in \mathbb{R}^{N \times L}$ where $\mathbf{P}(i, \ell) = p_{i,\ell}$.
- Let $\Pi_{\text{target}} \in \mathbb{R}^N$ be $\big(\Pi_{\text{target}}(S_1), \ldots, \Pi_{\text{target}}(S_N)\big)^\top$.

Then the problem is

$$\min_{\alpha,\, E} \; E \quad \text{subject to} \quad -E \;\le\; \big[\mathbf{P}\,\alpha\big]_i - \Pi_{\text{target},\,i} \;\le\; E, \quad i = 1, \ldots, N,$$

$$\alpha \in \mathbb{R}^L, \quad E \ge 0.$$

Interpretation:

- We force the portfolio payoff $\mathbf{P}\,\alpha$ to stay within a band of width $2E$ around the target payoff for all i.
- Minimizing E shrinks this band as much as possible.

Min–Max LP Algorithmic Steps

Step 1: Sample $\{S_i\}$.

- Choose a grid or set of simulated prices $\{S_1, \ldots, S_N\}$.
- Evaluate $\Pi_{\text{target}}(S_i)$.

Step 2: Build Payoff Matrix.

- For each instrument g_ℓ and each S_i, compute $p_{i,\ell} = g_\ell(S_i)$.
- Form $\mathbf{P} \in \mathbb{R}^{N \times L}$.

Step 3: Formulate Min–Max LP.

- Introduce variables $\{\alpha_\ell\}_{\ell=1}^{L}$ and $E \geq 0$.
- Impose constraints $-E \leq \sum_\ell \alpha_\ell p_{i,\ell} - \Pi_{\text{target}}(S_i) \leq E$.

Step 4: Solve LP.

- Standard linear solvers find α^* and E^*.

Step 5: Construct Final Portfolio.

$$\Pi_{\text{LP}}(S) = \sum_{\ell=1}^{L} \alpha_\ell^* \, g_\ell(S).$$