Deep Learning Methods for Optimal Stopping: Replication and Comparison with Traditional Techniques

Sergio Leal, Lana Popović, Pablo Pastor, Matthieu Fisher

May 12, 2025

Abstract

This report implements a deep learning-based approach to optimal stopping problems, following the framework introduced in [1]. The method is applied to the pricing of Bermudan max-call options, callable multi-barrier reverse convertibles, and the optimal stopping of fractional Brownian motion. A comparison with the Longstaff–Schwartz algorithm is also conducted in a one-dimensional setting. The report outlines the theoretical formulation, neural network structure, training procedure, and bound estimation techniques.

1 Introduction

We replicate the deep learning method for optimal stopping problems provided by [1] which learns the optimal stopping rule from Monte Carlo samples. We test the approach on the same problems as the original paper: pricing a Bermudan max-call option, pricing a multi-barrier reverse convertible, and optimally stopping a fractional Brownian motion. We also implement the Longstaff-Schwartz method described in [2] to price a Bermudan call option, which is equivalent to the max-call but in one dimension to assess the efficacy of the deep learning method.

In the first section, we describe the deep optimal stopping algorithm, starting from its mathematical framework, followed by adding a neural network into the problem, providing details on its implementation, and the bound computation. We then present the results of the experiments conducted and compare them with those obtained in [1]. Lastly, we compare the results in a one-dimensional setting with those obtained by the Longstaff-Schwartz method.

2 Deep Learning Approach to Optimal Stopping

2.1 Mathematical Formulation

Our objective is to develop a deep learning method that can, with low computational cost, learn an optimal policy for stopping problems of the form:

$$\sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau) \tag{1}$$

where $g : \{0, 1, \ldots, N\} \times \mathbb{R}^d \to \mathbb{R}$ is a measurable function and $\mathcal{T}$ denotes the set of all X-stopping times. We focus on problems where $X = (X_n)_{n=0}^N$ is an $\mathbb{R}^d$-valued discrete time Markov process on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F})_{n=0}^N, \mathbb{P})$. As a reminder, we say τ is an X-stopping time if $\{\tau = n\} \in \mathcal{F}$, $\forall n \in \{0, 1, \ldots, N\}$.

An important condition that makes 1 well defined is the integrability and square-integrability condition on g, as described in [1]. It is key to remember that problems of the form 1 have an exact solution when the number of stopping opportunities is finite. The optimal value is given by the Snell envelope, which is the smallest supermartingale that dominates the reward process. The corresponding optimal stopping time is the first time when the reward from stopping exceeds the expected reward from continuing (i.e. the continuation value). However, finding this process is not trivial, and therefore, these problems are usually solved numerically using dynamic programming approaches, which fail to scale properly when the dimension of X, d, increases.

The decision to stop at time n depends only on the state X_n, characterized by measurable functions $f_n : \mathbb{R}^d \to \{0,1\}$, $n = 0,1,\ldots,N$. We divide the optimal stopping problem 1 into time n stopping problems:

$$V_n = \sup_{\tau\in\mathcal{T}_n} \mathbb{E}\, g(\tau, X_\tau) \tag{2}$$

where $\mathcal{T}_n$ includes stopping times constrained to the interval $[n,N]$. $\mathcal{T}_N$ contains only the stopping time $\tau_N \equiv N$, thus we set $f_N \equiv 1$. Then, for measurable functions $f_n, f_{n+1},\ldots,f_N : \mathbb{R}^d \to \{0,1\}$ and any given n, we define:

$$\tau_n = \sum_{m=n}^{N} m f_m(X_m) \prod_{j=n}^{m-1} (1 - f_j(X_j)). \tag{3}$$

As proven in [1], the following theorem shows that τ_n is the optimal stopping time at time n. Therefore, to find a solution to the full optimal stopping problem 1, it is sufficient to consider stopping times of the form 3.

Theorem 1. For each $n \in \{0,\ldots,N-1\}$, consider the stopping time $\tau_{n+1} \in \mathcal{T}_{n+1}$ defined by:

$$\tau_{n+1} = \sum_{k=n+1}^{N} k f_k(X_k) \prod_{j=n+1}^{k-1} (1 - f_j(X_j)). \tag{4}$$

Then, there is a measurable function $f_n : \mathbb{R}^d \to \{0,1\}$ ensuring that the stopping time $\tau_n \in \mathcal{T}_n$ meets the condition:

$$\mathbb{E}\, g(\tau_n, X_{\tau_n}) \geq V_n - \left(V_{n+1} - \mathbb{E}\, g(\tau_{n+1}, X_{\tau_{n+1}})\right), \tag{5}$$

with V_n and V_{n+1} satisfying the relation given in 2.

2.2 Neural Network Architecture

Now that optimal stopping times can be seen as sequences of binary decisions, we approximate them using neural networks. Inspired by Becker et al. (2018), we define neural networks $f_n^{\theta_n} : \mathbb{R}^d \to \{0,1\}$ with parameters $\theta_n \in \mathbb{R}^q$. The stopping time τ_{n+1} can then be approximated by:

$$\tau_{n+1} = \sum_{m=n+1}^{N} m f_m^{\theta_m}(X_m) \prod_{j=n+1}^{m-1} (1 - f_j^{\theta_j}(X_j)). \tag{6}$$

Given that f^θ takes values in {0,1}, it does not lend itself directly to a gradient-based optimization method. To ensure differentiability for training via gradient-based methods, we introduce a feedforward neural network

$$F^\theta = \psi \circ a_I^\theta \circ \varphi_{q_{I-1}} \circ a_{I-1}^\theta \circ \cdots \circ \varphi_{q_1} \circ a_1^\theta \tag{7}$$

where $I \geq 2$ denotes the depth of the neural network, the positive constants $q_1, \dots, q_{I-1}$ are the depths of the hidden layers, $a_i^\theta(x) = W_i x + b_i$ are the linear layers, φ_i are component-wise ReLU activations, and ψ is the logistic sigmoid function. This approach intuitively treats network outputs as probabilities of stopping. After training parameters θ_n, we define the binary decision functions f^{θ_n} in terms of the output of the neural networks F^{θ_n}, assigning the values 0 and 1 depending on whether the output is below or above 0.5, respectively.
We set $\theta_N \in \mathbb{R}^q$ such that $f^{\theta_N} \equiv 1$. Then the candidate stopping time

$$\tau^\Theta = \sum_{n=1}^{N} n f^{\theta_n}(X_n) \prod_{j=0}^{n-1} (1 - f_j^{\theta_j}(X_j)) \tag{8}$$

is determined by the vector of the network parameters $\Theta = (\theta_0, \theta_1, \dots, \theta_{N-1}) \in \mathbb{R}^{Nq}$.

2.3 Training and Implementation Details

As a small difference from the paper, we train neural networks of the form 7 with fixed depth $I = 3$ and $q_1 = q_2 = d + 40$. The optimal parameters θ_n are found by approximating expected values with Monte Carlo samples calculated from simulated paths of the Markov process $(X_n)_{n=0}^N$. These sample paths are denoted $(x_n^k)_{n=0}^N$ for $k = 1, 2, \dots, K$. Once we have θ_N such that $f^{\theta_N} \equiv 1$, we determine $\theta_n \in \mathbb{R}^q$ for $n \in \{0, 1, \dots, N-1\}$.

At time step n, after computing the decisions $f^{\theta_{n+1}}, \dots, f^{\theta_N}$, we determine the optimal stopping time at $n+1$ as:

$$\tau_{n+1} = \sum_{m=n+1}^{N} m f^{\theta_m}(X_n) \prod_{j=n+1}^{m-1} (1 - f^{\theta_j}(X_j)). \tag{9}$$

with $\mathbb{E} g(\tau_{n+1}, X_{n+1})$ close to the optimal value (V_{n+1}).
It is clear that if $n = N-1$, $\tau_N \equiv N$ and for $n \leq N-2$ we have $\tau_{n+1} = l_{n+1}(X_{n+1}, \dots, X_N)$ where l_{n+1} is a function that takes values in $\{n+1, n+2, \dots, N\}$.
At step n, the realized reward if stopping occurs according to F^{θ_n}, followed by future decisions, is:

$$r_n^k(\theta_n) = g(n, x_n^k) F^{\theta_n}(x_n^k) + g(l_{n+1}^k, x_{n+1}^k)(1 - F^{\theta_n}(x_n^k)) \tag{10}$$

Thus, for large K, the expected value can be approximated as:

$$\frac{1}{K} \sum_{k=1}^{K} r_n^k(\theta_n) \tag{11}$$

which is smooth in θ a.e. by construction. Therefore, 11 is the function we wish to optimize via a gradient descent algorithm. We use the Adam optimizer for the updates, using mini-batch gradient descent with Xavier initialization and batch normalization in the examples below as it is done in the original paper.

2.4 Upper and Lower Bound Estimation

For the following experiments, we compute lower and upper bounds for the optimal value $V_0 = \sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau)$.
For the lower bound, once the stopping decisions are trained, the stopping time τ^Θ given by 8 yields a lower bound L for the optimal value V_0. To estimate it, we generate K_L new

realizations $(y_n^k)_{n=0}^N$, $k = 1, \dots, K_L$, of the process $(X_n)_{n=0}^N$. τ^Θ is given by $l(y_0^k, \dots, y_{N-1}^k)$ which we denote by l^k. Therefore, the Monte Carlo approximation

$$\hat{L} = \frac{1}{K_L} \sum_{k=1}^{K_L} g(l^k, y_{l^k}^k)$$

converges by the LLN to L when $K_L \to +\infty$.

For the upper bound, we use the Snell envelope. Note that all the affirmations below are in the $\mathbb{P}$-almost sure sense. For the reward process $(g(n, X_n))_{n=0}^N$, the Snell envelope is the smallest martingale with respect to $(\mathcal{F}_n)_{n=0}^N$ that dominates $(g(n, X_n))_{n=0}^N$. We denote it by H_n [Pekir], which can be expressed using the Doob-Mayer decomposition as $H_n = H_0 + M_n^H - A_n^H$, where M^H is an $(\mathcal{F}_n)$-martingale and A^H is a non-decreasing (F_n)-predictable process.

We approximate the Snell envelope $(H_n)_{n=0}^N$ by

$$H_n^\Theta = \mathbb{E}\left[g(\tau_n^\Theta, X_{\tau_n^\Theta}) \mid \mathcal{F}_n\right], \quad n = 0, 1, \dots, N, \tag{12}$$

with the corresponding stopping times:

$$\tau_n^\Theta = \sum_{m=n}^{N} m f^{\theta_m}(X_m) \prod_{j=n}^{m-1} (1 - f^{\theta_j}(X_j)). \tag{13}$$

The martingale components satisfy $M_0^\Theta = 0$ and:

$$M_n^\Theta - M_{n-1}^\Theta = H_n^\Theta - \mathbb{E}[H_n^\Theta \mid \mathcal{F}_{n-1}] = f^{\theta_n}(X_n) g(n, X_n) + (1 - f^{\theta_n}(X_n)) C_n^\Theta - C_{n-1}^\Theta, \quad n \geq 1, \tag{14}$$

where C_n^Θ represent continuation values.

$$C_n^\Theta = \mathbb{E}[g(\tau_{n+1}^\Theta, X_{\tau_{n+1}^\Theta}) | \mathcal{F}_n] = \mathbb{E}[g(\tau_{n+1}^\Theta, X_{\tau_{n+1}^\Theta}) | X_n], \quad n = 0, 1, \dots, N-1. \tag{15}$$

To estimate M^Θ, we generate K_U new realizations of $(X_n)_{n=0}^N$,$(z_n^k)_{n=0}^N$, where $k = 1, 2, \dots, K_U$. For every z_n^k we generate J continuation paths $\tilde{z}_{n+1}^{k,j}, \dots, \tilde{z}_N^{k,j}$, $j = 1, \dots, J$. Let $\tau_{n+1}^{k,j}$ denote the stopping times along paths $\tilde{z}_{n+1}^{k,j}, \dots, \tilde{z}_N^{k,j}$. We estimate continuation values as:

$$C_n^k = \frac{1}{J} \sum_{j=1}^{J} g(\tau_{n+1}^{k,j}, \tilde{z}_{\tau_{n+1}^{k,j}}^{k,j}), \quad n = 0, \dots, N-1. \tag{16}$$

This yields noisy martingale increments:

$$\Delta M_n^k = f^{\theta_n}(z_n^k) g(n, z_n^k) + (1 - f^{\theta_n}(z_n^k)) C_n^k - C_{n-1}^k. \tag{17}$$

Summing increments gives:

$$M_n^k = \sum_{m=1}^{n} \Delta M_m^k, \quad n \geq 1, \quad M_0^k = 0, \tag{18}$$

which estimate the martingale part plus noise. Thus, the unbiased upper bound is:

$$\hat{U} = \frac{1}{K_U} \sum_{k=1}^{K_U} \max_{0 \leq n \leq N} (g(n, z_n^k) - M_n^k), \tag{19}$$

converging to the theoretical upper bound U by the LLN as $K_U \to \infty$.

We also show an asymptotically 95% confidence interval for V_0 given by

$$\left[\hat{L} - z_{\alpha/2} \frac{\hat{\sigma}_L}{\sqrt{K_L}}, \hat{U} + z_{\alpha/2} \frac{\hat{\sigma}_U}{\sqrt{K_U}}\right]$$

using $\alpha = 0.05$

3 Numerical Experiments

3.1 Bermudan Max-Call Option

The payoff of this type of option depends on the maximum of d underlying assets. We a multi-dimensional Black–Scholes model describing the dynamics of the assets under a risk-neutral measure:

$$S_t^i = s_0^i \exp\left[(r - \delta_i - \frac{\sigma_i^2}{2})t + \sigma_i W_t^i\right], \quad i = 1, 2, \ldots, d, \tag{20}$$

with initial asset prices $s_0^i \in (0, \infty)$, a constant risk-free interest rate $r \in \mathbb{R}$, dividend yields $\delta_i \in [0, \infty)$, volatilities $\sigma_i \in (0, \infty)$, and a d-dimensional Brownian motion W characterized by constant instantaneous correlations $\rho_{ij} \in [-1, 1]$ between the components W^i and W^j. A Bermudan max-call option on the assets $S^1, S^2, \ldots, S^d$ provides a payoff of the form

$$\left(\max_{1\le i\le d} S_t^i - K\right)^+,$$

which can be exercised at N equispaced points in time defined by $0 = t_0 < t_1 < \cdots < t_N = T$, where $t_i = \frac{iT}{N}$ for $i = 0, \ldots, N$. The value of this Bermudan option is thus expressed as

$$\sup_\tau \mathbb{E}\left[e^{-r\tau}\left(\max_{1\le i\le d} S_\tau^i - K\right)^+\right],$$

where the supremum is taken over all stopping times taking values in $\{t_0, t_1, \ldots, t_N\}$. In our simulations, we use $d \in \{2, 3, 5, 10, 20\}$, $s_0^i \in \{90, 100, 110\}$, $r = 0.05$, $K = 100$, $\delta_i = 0.1$, $T = 3$, $\rho_{ij} = 0\ \forall i \neq j$and $N = 9$.

For the volatility parameters, we use two cases as in the paper, a symmetric case where $\sigma_i = 0.2$ and an asymmetric case where $\sigma_1 < \sigma_2 < \ldots < \sigma_d$. If $d \le 5$ then we use $\sigma_i = 0.08 + 0.32(i-1)/(d-1)$, otherwise we set $\sigma_i = 0.1 + i/2d$.

In order to simulate the model, we run $300 + d$ training steps for every network and at each step we use a batch size of $8,192$. To estimate the lower bound, we generate $K_L = 4096000$ trial paths. For the upper bound estimate, we produce $K_U = 1024$ paths $(z_n^k)_{n=0}^N$ and $J = 2048$ continuation paths for each of the paths. For this we generate K_UxJ independent realizations $(v_n^{k,j})_{n=1}^N$, $k = 1, \ldots, K_U$, $j = 1, \ldots, J$ of $(W_{t_n} - W_{t_{n-1}})_{n=1}^N$. For all n and $k,$we generated the i-th component o
continuation path departing from z_n^k according to

$$\tilde{z}_m^{i,k,j} = z_n^{i,k} \exp\left(\left[r - \delta_i - \sigma_i^2/2\right](m-n)\Delta t + \sigma_i[v_{n+1}^{i,k,j} + \cdots + v_m^{i,k,j}]\right), \quad m = n+1, \ldots, N. \tag{21}$$

Table 1 shows the results for the symmetric case, and Table 2 shows the results for the asymmetric case. Compared to the results obtained in [1], we achieve very similar estimates for the lower bounds. However, as d increases, the estimates for the upper bounds become less precise, which can be due to the number of continuation paths J. In the paper, the point estimates are given by

$$\frac{\hat{L} + \hat{U}}{2}$$

but given that K_L and K_U differ significantly in our implementation, we consider a weighted point estimate given by

$$\frac{K_L}{K_L + K_U}\hat{L} + \frac{K_U}{K_L + K_U}\hat{U}$$

that matches more closely with the point estimates given in the paper.

d	s_0	$\hat{L}$	t_L	$\hat{U}$	t_U	t_R	Point est.	95% CI
2	90	8.037	123.95	8.456	30.76	112.27	8.037	[8.025, 8.555]
2	100	13.825	119.62	14.612	29.14	107.82	13.825	[13.810, 14.722]
2	110	21.280	115.67	22.445	28.83	104.19	21.280	[21.262, 22.562]
3	90	11.203	157.01	12.693	50.51	140.46	11.203	[11.189, 12.864]
3	100	18.605	154.72	20.559	49.64	138.64	18.605	[18.589, 20.730]
3	110	27.411	194.01	30.295	52.49	176.00	27.412	[27.392, 30.492]
5	90	16.487	1597.18	19.880	324.12	1569.64	16.488	[16.472, 20.145]
5	100	25.885	361.88	32.399	74.78	339.67	25.887	[25.865, 32.738]
5	110	36.473	342.54	44.460	80.42	317.44	36.475	[36.451, 44.823]
10	90	25.920	592.37	36.369	118.70	508.58	25.923	[25.901, 36.848]
10	100	38.033	1074.80	53.372	1105.16	836.79	38.037	[38.011, 53.888]
10	110	50.461	1231.17	69.569	1100.60	994.28	50.466	[50.437, 70.131]
20	90	37.477	1529.40	57.712	1501.34	1077.25	37.482	[37.457, 58.309]
20	100	51.262	1595.02	77.638	1496.84	1147.76	51.269	[51.239, 78.267]
20	110	65.227	1697.00	96.595	1498.54	1248.91	65.235	[65.203, 97.297]

Table 1: Summary results for max-call options on d symmetric assets for parameter values $r = 5\%$, $\delta_i = 0.1$, $T = 3$, $N = 9$, $J = 2048$, batch size 8192, $K_L = 4096000$ and $K_U = 1024$. t_L is the total time in seconds to train τ^{Θ} and to compute the lower bound, $\hat{L}$,. t_R is the training time and t_U is the computation time for the upper bound, $\hat{U}$.

d	s_0	$\hat{L}$	t_L	$\hat{U}$	t_U	t_R	Point est.	95% CI
2	90	14.320	66.06	14.859	19.82	33.96	14.320	[14.294, 14.970]
2	100	19.733	62.73	20.591	19.17	31.96	19.733	[19.704, 20.602]
2	110	27.131	70.12	27.676	20.19	35.70	27.131	[27.099, 27.987]
3	90	19.043	94.66	20.264	23.98	37.78	19.043	[19.015, 20.620]
3	100	26.623	101.26	28.346	32.94	37.90	26.623	[26.590, 28.758]
3	110	35.740	92.65	36.301	22.63	41.47	35.740	[35.704, 36.689]
5	90	27.578	136.74	31.740	51.86	43.27	27.579	[27.545, 32.315]
5	100	37.851	142.48	43.395	109.50	52.37	37.852	[37.815, 44.042]
5	110	49.285	152.69	57.117	61.83	58.63	49.287	[49.245, 57.789]
10	90	85.477	148.33	117.937	38.77	52.15	85.485	[85.395, 119.977]
10	100	104.249	148.43	142.907	38.75	52.43	104.259	[104.158, 144.985]
10	110	123.043	148.67	168.160	38.74	52.89	123.054	[122.942, 170.461]
20	90	125.407	209.10	187.237	60.28	70.03	125.422	[125.306, 190.274]
20	100	149.184	207.98	215.712	60.04	69.83	149.201	[149.074, 218.845]
20	110	172.740	208.88	252.759	59.95	70.71	172.760	[172.617, 256.261]

Table 2: Summary results for max-call options on d asymmetric assets for parameter values $r = 5\%$, $\delta_i = 0.1$, $T = 3$, $N = 9$, $J = 2048$, batch size 8192, $K_L = 4096000$ and $K_U = 1024$. t_L is the total time in seconds to train τ^{Θ} and to compute the lower bound, $\hat{L}$,. t_R is the training time and t_U is the computation time for the upper bound, $\hat{U}$.

3.2 Callable Multi Barrier Reverse Convertibles

A Multi-Barrier Reverse Convertible (MBRC) is a coupon-paying financial instrument that converts into shares of the worst-performing asset among a basket of d underlying assets if certain barrier or strike conditions are met. The dynamics of the i-th asset, expressed as a percentage of its initial value, are given under the risk-neutral measure by:

$$S_t^i = \begin{cases} 100 \exp\left[(r - \frac{\sigma_i^2}{2})t + \sigma_i W_t^i\right], & t \in [0, T_i), \\ 100(1-\delta_i) \exp\left[(r - \frac{\sigma_i^2}{2})t + \sigma_i W_t^i\right], & t \in [T_i, T] \end{cases} \tag{22}$$

where r is the risk-free rate, σ_i the volatility, δ_i the dividend yield, T_i the dividend time, and W a d-dimensional Brownian motion with correlations ρ_{ij}.

The MBRC pays fixed coupons c at N time points $t_n = nT/N$, and a terminal payoff G at maturity T defined as:

$$G = \begin{cases} F, & \text{if } \min_{i,m} S_{u_m}^i > B \text{ or } \min_i S_T^i > K, \\ \min_i S_T^i, & \text{if } \min_{i,m} S_{u_m}^i \leq B \text{ and } \min_i S_T^i \leq K \end{cases}$$

where F is the notional, B the barrier, K the strike, and u_m the end of day m.
In the callable version, the issuer can redeem the MBRC early at any t_n, avoiding future coupon payments. To minimize cost, the issuer selects a stopping time τ to solve:

$$\min_\tau \mathbb{E}\left[\sum_{n=1}^{\tau} e^{-rt_n}c + 1_{\{\tau<T\}}e^{-r\tau}F + 1_{\{\tau=T\}}e^{-rT}G\right].$$

Letting $X_n = (S_{t_n}^1, \dots, S_{t_n}^d, X_n^{d+1})$ with $X_n^{d+1} = 1$ if the barrier has been breached by t_n and 0 otherwhise, the problem becomes:

$$\inf_{\tau\in\mathcal{T}} \mathbb{E}[g(\tau, X_\tau)],$$

where $\mathcal{T}$ is the set of stopping times for X, and

$$g(n,x) = \begin{cases} \sum_{m=1}^{n} e^{-rt_m}c + e^{-rt_n}F, & \text{if } n < N \text{ and } x^{d+1} = 0, \\ \sum_{m=1}^{N} e^{-rt_m}c + e^{-rT}h(x), & \text{if } n = N \text{ and } x^{d+1} = 1. \end{cases}$$

where

$$h(x) = \begin{cases} F & \text{if } \min_{1\leq i\leq d} x^i > K, \\ \min_{1\leq i\leq d} x^i & \text{if } \min_{1\leq i\leq d} x^i \leq K. \end{cases}$$

Since this is a minimization problem, the lower bound method yields an upper bound and vice versa. The parameters used are $F = K = 100$, $B = 70$, $T = 1$, $N = 12$, $c = 7/12$, $\delta_i = 5\%$, $T_i = 1/2$, $r = 0.05$, $\sigma_i = 0.2$ and $\rho_{ij} = \rho$ for $i \neq j$. We observed that this problem requires a high number of training epochs to converge, primarily because the barrier value is very low and in most cases the stock prices do not reach the barrier, particularly for lower values of d. This is apparent as the values obtained for $B = 70$ are very close to the prices for the non-callable MBRC provided in [1]. With this in mind, we increased the barrier to $B = 85$, and we trained each network for $1500+d$ epochs using a batch size of 8192. We used $K_L = 4096000$ realizations of the process to compute the lower bound estimator and $K_U = 1024$, $J = 4096$ for the upper bound estimation. With a more optimized code and better GPUs to run the training of the networks, we would be able to detect the very low probability event of the stocks going under lower barriers. Another technique to test in future research would be implementing importance

sampling inside the networks, being careful not to produce an overfitted model. This research is out of the scope of the project, and therefore, it is left for future research.
The results are shown in Table 3. In this case, we can see how the lower bound and upper bounds are closer, given that the number of training steps used is much higher.

d	ρ	$\hat{L}$	t_L	$\hat{U}$	t_U	t_R	95% CI
2	0.6	101.96	305.141	102.92	19.47	200.718	[101.456, 103.422]
2	0.1	101.79	305.254	102.07	19.47	199.341	[101.382, 102.87]
3	0.6	101.80	332.011	101.99	24.30	214.718	[101.348, 102.23]
3	0.1	101.76	327.654	101.96	24.12	210.287	[101.212, 102.141]
5	0.6	101.14	343.855	101.60	33.31	228.488	[101.382, 102.02]
5	0.1	100.99	337.911	101.48	33.13	214.386	[100.96, 101.59]

Table 3: Summary results for max-call options on d asymmetric assets for parameter values $r = 5\%$, $\delta_i = 0.05$, $T = 1$, $N = 12$, $J = 4096$, batch size 8192, $K_L = 4096000$ and $K_U = 1024$. t_L is the total time in seconds to train τ^{Θ} and to compute the lower bound, $\hat{L}$,. t_R is the training time and t_U is the computation time for the upper bound, $\hat{U}$.

3.3 Optimally Stopping Fractional Brownian Motion

In this experiment, we optimally stop a FBM with Hurst parameter H, which is a continuous Gaussian process $(W_t^H)_{t\geq 0}$ that has correlated increments if $H \neq 0.5$. It is known that for $H = 0.5$, W^H is a standard Brownian motion, however, for $H \neq 0.5$, the process is neither a martingale nor a Markov process. In order for our method to work, we construct a Markov process with its observations. We approximate the supremum

$$\sup_{0\leq\tau\leq 1} \mathbb{E}W_\tau^H \tag{23}$$

over all W^H-stopping times $0 \leq \tau \leq 1$. We denote $t_n = n/100$, $n = 0, 1, 2, \ldots, 50$, and introduce the 50-dimensional Markov process $(X_n)_{n=0}^{50}$ given by

$$\begin{aligned} X_0 &= (0, 0, \ldots, 0) \\ X_1 &= (W_{t_1}^H, 0, \ldots, 0) \\ X_2 &= (W_{t_2}^H, W_{t_1}^H, 0, \ldots, 0) \\ &\vdots \\ X_{50} &= (W_{t_{50}}^H, W_{t_{49}}^H, \ldots, W_{t_1}^H). \end{aligned}$$

The discretized stopping problem becomes

$$\sup_{\tau\in\mathcal{T}} \mathbb{E}g(X_\tau),$$

where $\mathcal{T}$ is the set of all X-stopping times and $g : \mathbb{R}^{50} \to \mathbb{R}$ is the projection $(x^1, \ldots, x^{50}) \mapsto x^1$ that approximates 23 from below. We compute estimates for $H \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$ using $d = n = 50$ to train the networks. Memory complexity was a clear problem in this experiment and that is why we used $n = 50$ instead of $n = 100$, which is used in the paper. We trained the networks for 500 epochs using a batch size of 1024 and to provide the estimations, we used $K_L = 4096$. Table 4 shows the estimates of the lower bound and the standard deviations of the estimates. The results are very similar to the ones obtained in the paper and match the fact that for $H = 1/2$, $W^{1/2}$ is a Brownian motion and therefore $\mathbb{E}W_\tau^{1/2} = 0$ for every $(W_\tau^{1/2})_{n=0}^{50}$-stopping time τ. The results are also shown graphically in Figure 1, where our results are very similar to those obtained in the paper despite our much lower number of epochs and simulations.

H	$\hat{L}$	$\sigma(\hat{L})$
0.01	1.2993	0.0191
0.05	1.0764	0.0204
0.1	0.8977	0.0195
0.2	0.5661	0.0193
0.3	0.3293	0.0191
0.5	0.0004	0.0124
0.7	0.1657	0.0149
0.8	0.2844	0.0145
0.9	0.3284	0.0137
0.95	0.3541	0.0134
0.99	0.3741	0.0129

Table 4: Lower bound estimates for different values of H. Number of epochs is 500, $K_L = 4096$, $n = 50$, batch size is 1024. It took around 980 seconds to train and compute the values of the lower bounds for every value of H.

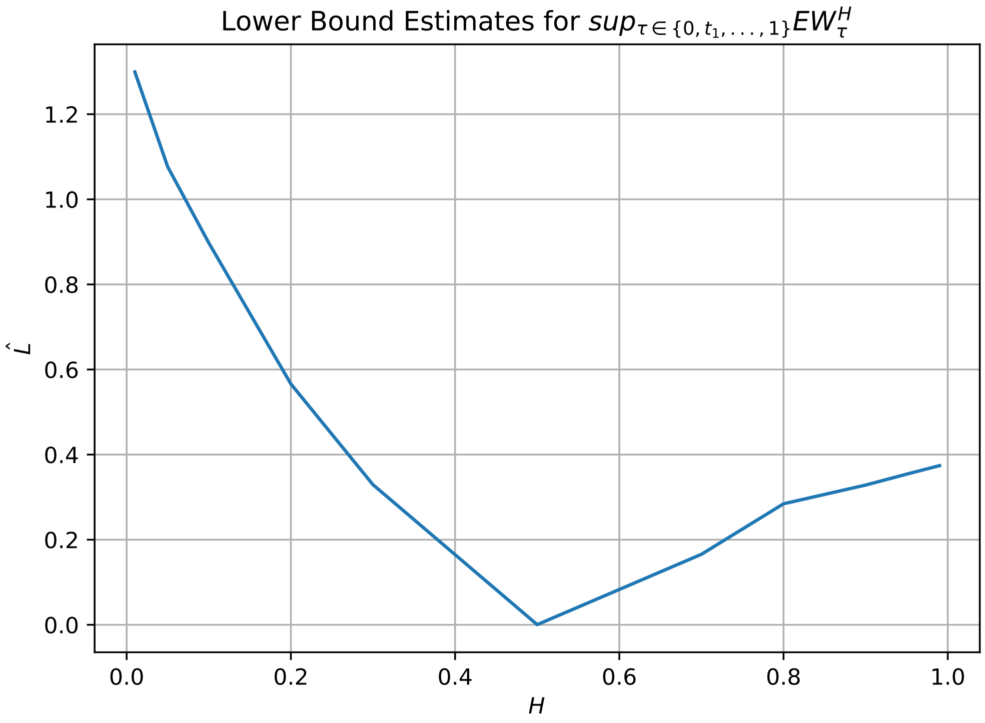

Figure 1: Estimates of a lower bound to $\sup_{\tau\in\{0,t_1,\ldots,1\}} \mathbb{E}W_\tau^H$ for different values of H.

4 Benchmarking with Longstaff-Schwartz Method

The Longstaff-Schwartz Method (LSM) is a Monte Carlo simulation method for pricing American options. It uses a simple least-squares regression to estimate the continuation values. The methods extension to Bermudan options is straightforward: simply fix the dates when the holder can exercise or continue holding the option.

4.1 Methodology

Starting immediately before the terminal time at $T - \Delta_t^n$, the algorithm runs backward iteratively by comparing at each time step the immediate payoff from exercising now, $f(S^j_{t^n_i})$, with the estimated payoff from not exercising (the continuation value), $\hat{C}_{t^n_i}$. The continuation value is estimated by regressing (across the simulated paths) the payoff at the future time step, $V^j_{t+\Delta^n_t}$ against a polynomial basis function of the underlying $g(S^j_{t^n_i})$, for in-the-money simulated paths. If the fitted continuation value is inferior to the immediate payoff at that time step, then the option value at that time step, $V^j_{t^n_i}$, becomes the immediate payoff. Otherwise, it becomes the discounted value of the payoff at the future time step $e^{-r\Delta_t}V^j_{t+\Delta^n_t}$. The algorithm runs this procedure iteratively up to, and including, Step 0. The average payoff across simulations $\frac{1}{J}\sum_j^J V_0^j$ is the estimated value of the option.

The iterative algorithm is described below.

1. Simulate M paths of the underlying asset S_t under the risk-neutral measure.
2. At maturity t_N, set the option value to the payoff:
$$V^{(m)}_{t_N} = h(S^{(m)}_{t_N}), \quad \text{for all } m = 1, \dots, M$$
3. For each earlier time t_n (working backward from t_{N-1} to t_0):
   (a) Identify in-the-money paths: $\mathcal{I}_n = \left\{ m \mid h(S^{(m)}_{t_n}) > 0 \right\}$
   (b) Estimate the continuation value $C^{(m)}_{t_n}$ using least squares regression:
$$C^{(m)}_{t_n} \approx \mathbb{E}[V^{(m)}_{t_{n+1}} \mid S^{(m)}_{t_n}] \approx \sum_{k=1}^{K} \beta_k \phi_k(S^{(m)}_{t_n})$$
   where $\{\phi_k\}$ are basis functions (e.g., polynomials) and β_k are regression coefficients.
   (c) Make the exercise decision:
$$V^{(m)}_{t_n} = \begin{cases} h(S^{(m)}_{t_n}) & \text{if } h(S^{(m)}_{t_n}) > C^{(m)}_{t_n} \\ e^{-r\Delta t} V^{(m)}_{t_{n+1}} & \text{otherwise} \end{cases}$$

After training the exercise policy using LSM on one set of simulated paths (training set), we estimate the lower bound on a separate set of out-of-sample paths (test set) by applying the previously learned stopping strategy. Let $\tau^{(m)}$ be the stopping time on path m, determined by the learned policy. Then the lower bound is:

$$\underline{V_0} = \frac{1}{M} \sum_{m=1}^{M} e^{-r\tau^{(m)}} h(S^{(m)}_{\tau^{(m)}}).$$

To obtain an upper bound estimate, we construct a martingale M_t and define the dual estimator

$$\overline{V}_0 = \mathbb{E}\left[\max_{0\le t\le T}\left(h(S_t) - M_t\right)\right]$$

where M_t is estimated from the difference between the estimated continuation values at two different time-steps. The upper bound is then computed as

$$\overline{V}_0 \approx \frac{1}{M}\sum_{m=1}^{M}\max_{0\le n\le N}\left(h(S_{t_n}^{(m)}) - \sum_{k=0}^{n-1}\Delta M_{t_k}^{(m)}\right)$$

where $\Delta M_{t_k}^{(m)}$ is the increment in the estimated martingale along path m.

4.2 Results and Comparison with Deep Learning

In this section, we provide the results for the different values of s_0 and compare them with those obtained using the deep learning method. In Table 5 we show the comparison between both methods, where the subindex LSM indicates the results for the Longstaff-Schwartz method.

s_0	$\hat{L}$	$\hat{L}_{LSM}$	t_L	$t_{L,LSM}$	$\hat{U}$	$\hat{U}_{LSM}$	t_U	$t_{U,LSM}$	t_R	$t_{R,LSM}$
90	4.335	4.357	4.26	0.90	4.3636	6.604	100.02	0.01	18.858	1.19
100	7.955	7.794	3.99	0.95	7.898	9.596	99.144	0.01	18.71	1.19
110	13.126	13.111	4.01	1.13	13.252	13.243	99.07	0.02	18.387	1.19

Table 5: Summary results for Bermudan call options on $d = 1$ asset with $r = 5\%$, $T = 3$, $N = 9$, $q = 0.1$, $\sigma = 0.2$, $K = 100$, $n_{\text{sim}} = 100000$, $n_{\text{steps}} = 9$, $n_{\text{eval}} = 9$. $K_L = 4096000$, $K_U = 1024$, $J = 2048$ were used for the deep learning method (training 301 epochs with batch size of 8192). t_L and t_U are computation times (in seconds) for lower and upper bounds, respectively. For LSM, K_L and K_U were set to 100000 and 1500000, respectively. In-sample training and out-of-sample training are relatively quick, as they depend only on the simulation sample size.

Compared to recent deep learning-based approaches for pricing Bermudan options, the Longstaff-Schwartz regression method retains its advantage in interpretability and computational efficiency for low-dimensional problems ($d = 1$ in our case). While NN's often yield tighter confidence intervals by capturing complex continuation value surfaces, they typically require significantly more training time, hyperparameter tuning, and computational resources. In contrast, the estimates presented in Table 5 are obtained with modest simulation sizes and still yield tight bounds and accurate point estimates. Notably, the confidence intervals in our setup are relatively narrow, with upper and lower bounds within a few bps of the point estimates. As another example, a plain vanilla in-sample and out-of-sample LSM exercise with 100000 simulations yields 1.19 seconds and 0.9 seconds to estimate in-sample and out-of-sample, respectively. For higher-dimensional problems, however, deep learning methods may outperform LSM by avoiding the curse of dimensionality.

5 Conclusions

This study explores the use of deep learning for optimal stopping problems, particularly in pricing financial derivatives such as Bermudan max-call options and multi-barrier reverse convertibles, and also for optimally stopping a fractional Brownian motion. By replicating methods from prior research, we applied deep learning to these problems and obtained results closely aligned with theoretical expectations. The findings demonstrate that deep learning is effective at approximating the optimal stopping rule, especially in high-dimensional settings where traditional methods like dynamic programming face scalability issues.

While deep learning proved advantageous for complex problems, it also presented challenges, mainly in computational cost. As the problem's dimensionality increased, so did the training time and resource demands, highlighting the need for improved algorithm efficiency. Future research could focus on refining neural network architectures to reduce training time while maintaining accuracy. One potential route for this is developing hybrid models that combine the interpretability of traditional methods like LSM with the scalability of deep learning. Reducing the number of training epochs and enhancing convergence rates could also address some of the computational challenges encountered in this study.

References

[1] Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. "Deep Optimal Stopping". In: (2020).

[2] Francis A Longstaff and Eduardo S Schwartz. "Valuing American options by simulation: a simple least-squares approach". In: *The review of financial studies* 14.1 (2001), pp. 113–147.