Stony Brook University

Department of Applied Mathematics and Statistics

Deep Hedging

A project work submitted in fulfilment of the requirements of the AMS517: Quantitative Risk Management course for Spring 2025.

By: Amos Anderson and Siddharth Pampari

May 7, 2025

Abstract

This project presents an implementation of the deep hedging framework introduced by Buehler et al. (2019), which uses neural networks to learn hedging strategies in markets with frictions. Unlike classical approaches that rely heavily on specific pricing models, the deep hedging method is model-agnostic and directly incorporates transaction costs, liquidity constraints, and other market imperfections. In this work, the deep hedging setup is implemented using a vanilla European call option and a call spread under the Heston model. The neural network is trained to minimise a risk measure based on the final PnL. The results obtained are consistent with those in the original study and highlight the effectiveness of deep learning techniques in approximating optimal hedging strategies under realistic market conditions.

Contents

1 Introduction

In financial markets, it is almost impossible to prevent the risk of incurring a loss. However, identifying, controlling and managing risk is well within the abilities of investors and firms. This is the concept of hedging. Hedging is defined as a risk management strategy used to reduce or offset potential losses in investment due to the uncertainties and sudden changes in market conditions. During the hedging process, firms will sacrifice part of their profit to protect themselves against a big potential loss.

Hedging helps investors to manage their exposure to risk associated with stock prices, interest rates, currencies and commodities. Even when a firm seems to be doing well, unexpected market movements may affect the firm's profit and so it is important to undertake hedging strategies to protect the firm's profit. In addition, financial institutions like banks and investment companies carry out hedging practices to meet regulatory capital requirements and keep risk metrics like Value-At-Risk(VaR) and Conditional VaR(CVaR) within acceptable range.

There are several tools for hedging. One of these is the use of financial derivatives like options (put and call), forward contracts, futures and swaps. The actual hedging strategy here is the traditional theoretical model hedge equations from pricing models. We shall discuss some of these models in the next section. Another hedging tool is portfolio diversification which is often used in practice. Here firms spread investments across several asset classes to reduce correlated risk. Finally there is the use of *Deep Reinforcement Learning (RL)* methods for constructing hedging strategies, also called ***Deep Hedging***, which is the focus of this project.

2 Background

We provide a brief introduction to two classical pricing models and their corresponding hedging strategies as well as deep reinforcement learning and its hedging advantages over the traditional methods.

2.1 Traditional Pricing Models and Hedging

Here we consider one dimensional pricing model for European Call Options, also called the Black-Scholes model (1973) and a two dimensional pricing model - the Heston Model (1993).

2.1.1 Black-Scholes Model (1973)

The Black-Scholes model computes the theoretical price of an European Call Option. The model assumes constant volatility and interest rates, no initial dividends and no market friction like transaction costs. Given underlying Stock Price (S_t), Strike Price (K), Time to expiration (T), Risk-free interest rate (r), and a fixed volatility (σ), the price $V(S_t, t)$ of the European Call Option at time t for $(0 \leq t \leq T)$ is giving by

$$V(S_t, t) = S_t \Phi(d_1) - K e^{-r(T-t)} \Phi(d_2)$$

where Φ is the Standard Normal Gaussian Distribution and

$$d_1 = \frac{\log(S_t/K) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}, \qquad d_2 = d_1 - \sigma(T-t)$$

and S_t is assumed to follow a geometric brownian motion

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

where $W = \{W_t\}_{0\leq t\leq T}$ is the standard Brownian Motion defined on a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}\}_{0\leq t\leq T}, \mathbb{P})$

In the Black-Scholes model, *delta* hedging is used, where we hedge against minor price movements in the European call price by calculating

$$\Delta = \Phi(d_1)$$

The *delta*(Δ) of a an option measures the sensitivity of the option price to changes in the stock price S_t. To hedge against the risk associated with stock price, an investor holds Δ shares of the stock to offset the option's price risk.

There are very significant shortcomings of the *delta* hedging strategy. Its assumption that volatility is constant is unrealistic because volatility in markets changes over time and is considered stochastic. Moreover, it fails to include market frictions which are factors that impede the smooth and efficient operation of a market. These includes transaction costs, funding costs and liability constraints. Excluding market frictions reduces the applicability of the Black-Scholes model.

2.1.2 Heston Model (1993)

The Heston Model is used to price and hedge derivatives options. Unlike the Black-Scholes model, the Heston Model relaxes the constant volatility assumption and allows volatility to fluctuate over time, which mimics the real world world behaviour of market volatilty. We say that the Heston Model is a two dimensional model because it explanations the changes in two stochastic processes - the stock price and the volatility.

The Heston Model model is defined by two stochastic differential equations (SDEs):

1. The Stock Price Process

   $$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^1$$

   where S_t is the underlying stock price at time t,μ is the drift rate of the underlying stock, v_t is the instantaneous variance(volatility squared) and W_t^1 is a one dimensional standard Brownian Motion (under a probability measure $\mathbb{Q}$).

2. The Variance Process modelled by the Cox-Ingersoll-Ross (CIR) process.

   $$dv_t = k(\theta - v_t)dt + \xi\sqrt{v_t} dW_t^2$$

   where k is the speed of mean reversion, θ is the long run average variance, ξ is the "volatility of volatility" and dW_t^2 is also a one dimensional standard brownian motion (under a probability measure $\mathbb{Q}$).

Hedging using the Heston model is robust and relatively realistic as compared to the Black-Scholes hedging. However, it possess a challenging and complex task in finding a good hedging strategy. Since volatility is treated as a stochastic variable, a corresponding hedge will be needed. Ideally, hedging under Heston model involves computing *greeks* - delta, vega, gamma - which gives the following hedging strategies for the model:

(A) Delta Hedging
This requires computing Δ from the Heston Model using PDE and Fourier Transfrom methods. An investor holds Δ units of the underlying asset and rebalances his position as S_t and v_t changes.

(B) Vega Hedging
These are hedging strategies that seek to neutralize volatility risk associated with a particular derivative. They involve trading variance swaps or VIX options (or futures) with opposite *vega* as the underlying stock,to offset potential losses of the derivative.

(C) Gamma Hedging
The goal of gamma hedging is to protect a portfolio against unexpected non-linear price jumps. The Heston model allows for fat tails, making it possible to hedge against large movements.

2.2 Deep Reinforcement Learning and Hedging

We can formulate *hedging* as a reinforcement learning problem. The hedger represents the *Agent*, the *Environment* is the financial market, the *States* (s_t) is the market information (spot price, volatility, past actions), the *Action* (a_t) space includes all trading decisions like adjusting delta/vega positions constitutes and finally the *Reward* (r_t) could be taken as the negative value of the risk measure. The goal is to find an optimal policy $\pi(s, a; \theta)$ that maximizes accumulated reward.

We parametrize the policies $\pi(s, a; \theta)$ by neural-networks. This makes the hedging process a Deep Reinforcement Learning problem. We shall use the term *Deep Hedging* as used by Buehler et. al (2019) to represent a reinforcement learning based hedging process that uses neural network parametrization of value functions $Q(s, a; \theta)$ or $V(s, a; \theta)$ or policies $\pi(s, a; \theta)$.

Unlike the classical pricing models (Black-Scholes and Heston), Deep Hedging does not assume complete and frictionless markets. Instead, it is able to learn an optimized hedging strategy directly from data by constantly adapting to market frictions. Moreover, Deep Hedging does not rely on computing *greeks* which is computationally expensive as a result of the PDEs involved in their computation. In multi-asset options and other high dimensional portfolios, traditional methods fail due to the curse of dimensionality. However, Deep Hedging makes it possible to determine an optimal hedging strategy.

3 Review of Related Literature

This project relies primarily on the article **Deep Hedging** by H. Buehler et al. (2019)[1]. We shall refer to this as the main paper. The paper presents a framework for hedging a portfolio of derivatives in the presence of market frictions such as transaction costs, market impact, liquidity constraints, or risk limits using modern deep reinforcement learning methods.

The proposed deep learning approach is model-free and employs a *policy-based* reinforcement learning (RL) algorithm.

To illustrate the performance and scalability of the proposed algorithm, three research questions were considered:

i. How does neural network hedging (for different risk-preferences) compare to the benchmark in a Heston model without transaction costs?

ii. What is the effect of proportional transaction costs on the exponential utility indifference price?

iii. Is the numerical method scalable to higher dimensions?

To answer these questions, experiments were conducted in synthetic markets driven by the Heston model, incorporating market frictions (specifically, transaction costs). The Heston model serves only as a benchmark for comparison, as the deep learning framework itself is model-independent and learns directly from data. Additionally, the paper builds upon the works of [2] and [3], and focuses on optimizing the hedging of a portfolio under *convex risk measures*.

The technical components of the methodology, including numerical approximations of hedging strategies, are implemented using deep neural networks. These networks are trained via the Adam gradient descent optimizer [4] in a mini-batch setting to address a semi-recurrent reinforcement learning problem. Other state-of-the-art machine learning optimization techniques (see [5]) are also employed to yield a near-optimal *deep hedge*.

All algorithms were implemented in Python using TensorFlow.

4 Objective

This project seeks to apply deep reinforcement learning methods to develop a hedging strategy under market frictions. As we discuss deep reinforcement learning methods, we are interested in finding out whether or not deep reinforcement learning methods are feasible and can work in practice. Moreover, we want to know how deep hedging performance compare to that of traditional hedging strategies like delta hedging. Finally, we wish to study whether deep hedging can scale to multi-dimensional and complex markets. To achieve this goal, we must answer the research questions of the main paper. We shall do this by replicating the results from the experiments in the main paper (with

TensorFlow in Python).

5 Methodology

The core idea of the Deep Hedging strategy as proposed by H. Buehler et al. is to replace dynamic programming that relies on PDE computation and model assumptions with a deep reinforcement learning approach where hedging strategies (δ_k) are parametrized by deep neural networks, optimization is performed directly over a risk measure ρ, and requires no model assumptions (that is, no requirements for risk neutral measures and complete markets are needed).

We shall discuss the methodology of the proposed algorithm by first looking at the problem set-up(Section 5.1), the pricing and hedging via convex risk measures (Section 5.2) and the neural network approximation of the hedging strategies (Section 5.3).

5.1 Problem Set-Up: Discrete Time Market with Frictions

Market Model: Consider finite trading dates $0 = t_0 < t_1 \ldots < t_n = T$ and let $(\Omega, \mathbb{F} = \{\mathcal{F}_k\}_{k=0,\ldots,n}, \mathbb{P})$ be a filtered probability space where $\Omega = \{\omega_1, \ldots, \omega_N\}$ is finite with a probability measure $\mathbb{P}[\omega_i] > 0$. Also, $\mathbb{F}$ denotes the information filtration where $\mathcal{F}_k$ encodes market data $\mathcal{I}_k \in \mathbb{R}^r$ like prices, market signals, etc.

Tradable Instruments: The market contains d assets to be used as hedging instruments with mid-process given by $S = (S_k)_{k=0,\ldots,n}$, adapted to $\mathbb{F}$. The hedging instruments are subject to liquidity restrictions (tradable before a future point in time) which are modelled alongside trading costs.

Trading Strategies and Constraints: Let the random variable Z be a liability. We hedge Z at maturity T by trading in S using an $\mathbb{R}^d$ - valued $\mathbb{F}$-adapted stochastic process $\delta = (\delta_k)_{k=0,\ldots,n}$ with $\delta_k = \left(\delta_k^1, \ldots, \delta_k^d\right)$. Here, δ_k^i denotes the agent's holdings of the ith asset at time t_k. We may also define $\delta_{-1} = \delta_n = 0$. We incorporate liquidity constraints by assuming that δ_k is constrained to a set $\mathcal{H}_k$ via a projection map H_k:

$$\mathcal{H}_k = H_k\left(\mathbb{R}^{d(k+1)}\right)$$

For an unconstrained strategy $\delta^u \in \mathcal{H}^u$, we successively define with $(H \circ \delta^u)_k := H_k((H \circ \delta^u)_0, \ldots, (H \circ \delta^u)_{k-1}, \delta_k^u)$ its constrained projection into $\mathcal{H}_k$. We denote $\mathcal{H} := (H \circ \mathcal{H}^u) \subset \mathcal{H}^u$ as the corresponding non-empty set of restricted trading strategies.

Moreover, we introduce trading costs as follows: if the agent decides to buy a position $n \in \mathbb{R}^d$ in S at time t_k, then this will incur a cost of $c_k(n)$. The total cost of trading a strategy δ up to maturity is therefore

$$C_T(\delta) := \sum_{k=0}^{n} c_k(\delta_k - \delta_{k-1})$$

The agent's portfolio terminal wealth value, which incorporates liabilities Z, initial cash injection p_0, and a trading strategy δ is given by

$$\mathrm{PL}_T(Z, p_0, \delta) := -Z + p_0 + (\delta \cdot S)_T - C_T(\delta)$$

where

$$(\delta \cdot S)_T = \sum_{k=0}^{n} \delta_k (S_{k+1} - S_k)$$

5.2 Pricing and Hedging via Convex Risk Measures

It is impossible to compute prices of derivatives, and corresponding hedging strategies without a risk measure. Traditional pricing and hedging models like the Black-Scholes and Heston models rely on the following: *risk neutral pricing* where derivatives are priced as $\mathbb{E}^{\mathbb{Q}}[\text{Payoff}]$ given the martingale $\mathbb{Q}$ and *perfect replication* which eliminates all risk via continuous trading. In such idealized complete market with continuous-time trading, no transaction costs, and unconstrained hedging, for any liabilities Z there exists a unique replication strategy δ and a fair price $p_0 \in \mathbb{R}$ such that

$$-Z + p_0 + (\delta \cdot S)_T - C_T(\delta) = 0$$

However, in reality, markets are incomplete and have frictions.

The paper proposed using convex risk measures for pricing and hedging. Convex risk measures incorporates risk aversion, handles frictions and provide robust prices for derivatives.

Definition of Convex Risk Measures: Let $X_1, X_2, X_3 \in \mathcal{X}$ be asset positions ($-X$ is liability). A functional $\rho : \mathcal{X} \to \mathbb{R}$ is a convex risk measure if:

1. Monotonicity: If $X_1 \geq X_2$, then $\rho(X_1) \leq \rho(X_2)$.
   Less liability requires less capital.

2. Convexity: $\rho(\lambda X_1) + (1 - \lambda)X_2 \leq \lambda\rho(X_1) + (1 - \lambda)\rho(X_2)$ for $\lambda \in (0, 1)$
   Diversification reduces risk.

3. Cash invariance: $\rho(X_1 + c) = \rho(X_1) - c$.
   Adding cash reduces risk linearly.

Optimized Certainty Equivalents (OCEs) as Convex Risk Measures: Assume a continuous, non-decreasing and convex loss function $l : \mathbb{R} \to \mathbb{R}$, then the optimized certainty equivalents are defined as:

$$\rho(X) = \inf_{w \in \mathbb{R}} \{w + \mathbb{E}[l(-X - w)]\}, \qquad X \in \mathcal{X} \tag{5.1}$$

It was shown that all optimized certainty equivalents of the form of Equation (5.1) are convex risk measures. The paper considered two special cases of OCE risk measures:

- The *entropic risk measure* which is related to the Exponential Utility Indifference Pricing discussed in subsequent sections and is given by given by

$$\rho(X) = \frac{1}{\lambda} \log \mathbb{E}[\exp(-\lambda X)] \tag{5.2}$$

- *Conditional Value-At-Risk (CVaR)* or Expected Shortfall (ES) given by setting $l(x) := \dfrac{\max(x,0)}{1-\alpha}$ in Equation (5.1) which gives

$$\rho(X) = \inf_{w\in\mathbb{R}} \left\{ w + \mathbb{E}\left[\frac{\max(-X-w,0)}{1-\alpha}\right]\right\}, \qquad X \in \mathcal{X} \tag{5.3}$$

Duality Property of Convex Risk Measures: The duality theory for convex risk measures connects risk minimization with robust pricing. The duality theorem states that every convex risk measure ρ has a dual representation

$$\rho(X) = \sup_{\mathbb{Q}\in\mathcal{P}} \left(\mathbb{E}^{\mathbb{Q}}[-X] - \alpha(\mathbb{Q})\right) \tag{5.4}$$

where $\mathcal{P}$ is the set of probability measures on Ω and $\alpha(\mathbb{Q}) = \sup_{X\in\mathcal{X}}(\mathbb{E}_{\mathbb{Q}}[-X] - \rho(X))$ is the penalty term for deviating from ideal scenarios. This robust representation measures risk as the worst-case expected loss over all possible models ($\mathbb{Q}$) not just historical paths, and penalized by $\alpha(\mathbb{Q})$ which encode different risk preferences. This justifies the use of ρ for pricing or hedging in incomplete markets. For OCE risk measures, duality simplifies the training of neural networks to a single minimization problem.

5.2.1 Pricing using Exponenential Utility Indifference Pricing

Indifference pricing generalizes classical pricing to incomplete markets with frictions. Unlike risk-neutral pricing (which assumes perfect hedging), indifference pricing incorporates risk preferences (via convex risk measures ρ) and real-world constraints like (transaction costs $C_T(\delta)$, liquidity constraints $\mathcal{H}$).

Several papers have used exponential utility indifference pricing to derive prices consistent with a trader's risk aversion under market frictions. Examples include [6],[7], [8] and [9]. We define the exponential utility function as

$$U(x) = -\exp(-\lambda x)$$

where $\lambda > 0$ is the risk aversion parameter.

The indifference price $q(Z) \in \mathbb{R}$ is the cash amount that makes the trader indifferent between selling the claim Z and hedging it, or not selling Z at all.

Mathematically, we define $q(Z)$ as:

$$\sup_{\delta\in\mathcal{H}} \mathbb{E}[U(q(Z) - Z + (\delta\cdot S)_T - C_T(\delta))] = \sup_{\delta\in\mathcal{H}} \mathbb{E}[U((\delta\cdot S)_T - C_T(\delta))]$$

It was shown by H. Buehler et al. that the exponential utility indifference pricing used above is equivalent to pricing via the entropic risk measure

$$\rho(X) = \frac{1}{\lambda}\log\mathbb{E}[\exp(-\lambda X)], \quad \text{for} \quad X \in \mathcal{X}$$

The indifference price $q(Z)$ simplifies to:

$$q(Z) = \pi(-Z) - \pi(0), \quad \text{where} \quad \pi(X) = \inf_{\delta \in \mathcal{H}} (\rho(X + (\delta \cdot S)_T - C_T(\delta))) \tag{5.5}$$

It was deduced that such a market with indifference price $q(-Z)$ is still relevant when $\pi(0)$=0.
Later on, by choosing a convex risk measure ρ like CVaR or entropic risk measure, we shall train neural networks to minimize $\rho(-Z + (\delta \cdot S)_T - C_T(\delta))$ and use the trained strategy to evaluate the price $q(Z) = \pi(-Z) - \pi(0)$.

5.2.2 Hedging as Risk Minimization

The paper defines the fundamental hedging problem as the following constrained optimization problem:

$$\pi(X) = \inf_{\delta \in \mathcal{H}} (\rho(X + (\delta \cdot S)_T - C_T(\delta))) \tag{5.6}$$

Given a liability $-Z$, the optimal hedge is the strategy δ^* that solves:

$$\delta^* = \operatorname*{argmin}_{\delta \in \mathcal{H}} \rho(X + (\delta \cdot S)_T - C_T(\delta))$$

That is, the optimal hedge is the strategy that minimizes the risk of the residual P&L.

5.3 Neural Network Parametrization

This section discusses the approximation of hedging strategies under market frictions by neural network, as outlined by H. Buehler et al. It builds on the universal approximation capabilities of neural networks, the well-founded theoretical results of neural network approximations and the fact that hedging strategies built from neural networks can numerically be calculated very efficiently.

5.3.1 Universal Approximation by Neural Networks

The universal approximation theorem [10] illustrates that the feed forward neural network $\mathcal{NN}_{M,d_0,d_1}$ with bounded activation function σ can approximated any measurable function. Motivated by the universal approximation theorem, we can consider neural network hedging strategies.

We rewrite the constrained risk minimization problem as

$$\pi(X) = \inf_{\delta \in \mathcal{H}^u} (\rho(X + (H \circ \delta \cdot S)_T - C_T(H \circ \delta)))$$

By incoporating market information described by the observed maximal feature set $I_0, \ldots, I_k$, and setting $\mathcal{H}^u = \mathcal{H}_M$ where

$$\mathcal{H}_M = \left\{ \left(\delta_k^\theta\right)_{k=0,\ldots,n-1} \in \mathcal{H}^u : \delta_k^\theta = F^{\theta_k}\left(_0, \ldots, I_k, \delta_{k-1}\right), \theta_k \in \Theta_{M,r(k+1)+d,d} \right\}$$

we get the following optimization problem for our unconstrained trading strategies:

$$\begin{aligned}\pi^M(X) &:= \inf_{\delta\in\mathcal{H}_M}(\rho(X+(H\circ\delta\cdot S)_T - C_T(H\circ\delta)) \\ &= \inf_{\theta\in\Theta_M}(\rho(X+(H\circ\delta^\theta\cdot S)_T - C_T(H\circ\delta^\theta))\end{aligned}$$

where $\Theta_M = \prod_{k=0}^{n-1}\Theta_{M,r(k+1)+d,d}$. Thus, the infinite-dimensional problem of finding an optimal hedging strategy is reduced to the finite-dimensional constraint problem of finding optimal parameters for our neural network.

5.3.2 Architecture Design

The paper proposes a *semi-recurrent feed-forward neural network* architecture where at each time step t_k, the hedging strategy depends on two inputs: market data $I_0,\ldots,I_k$ and prior position δ_{k-1} and is computed as:

$$\delta_k^\theta = F_k^\theta\left(I_0,\ldots,I_k,\delta_{k-1}^\theta\right), \quad F_k^\theta\in\mathcal{NN}_{M,r(k+1)+d,d} \tag{5.12}$$

The hedging strategies δ_k at each time t_k are determined by solving the optimization problem

$$\pi^M(X) = \inf_{\theta\in\Theta_M}(\rho(X+(H\circ\delta^\theta\cdot S)_T - C_T(H\circ\delta^\theta)) \tag{5.13}$$

The network structure consists of two hidden layers with ReLU activation $\sigma(x)=\max(x,0)$, width $= d+15$ where d is the number of hedging instruments and a batch normalization before activation.

The key parameters of the algorithm are θ which denotes the weights optimized per time step t_k and M which controls network complexity.

5.3.3 Training via Stochastic Gradient Descent

The network is trained to find the optimal or close-to optimal parameter θ that minimizes the convex risk measure ρ of the terminal P&L. This is given by:

$$\min_\theta \rho(-Z+(\delta^\theta\cdot S)_T - C_T(\delta^\theta))$$

We consider when ρ is an OCE risk measure. Inserting the definition of ρ, see Equation(5.1), into Equation (5.13), the optimization problem can be rewritten as

$$\pi^M(-Z) = \inf_{\bar\theta\in\Theta_M}\inf_{w\in\mathbb{R}}\left\{w+\mathbb{E}[l(Z-(\delta^{\bar\theta}\cdot S)_T - C_T(\delta^{\bar\theta})-w)]\right\} = \inf_{\theta\in\Theta}J(\theta)$$

where $\Theta=\mathbb{R}\times\Theta_M$ and for $\theta=(w,\bar\theta)\in\Theta$,

$$J(\theta) := w+\mathbb{E}[l(Z-(\delta^{\bar\theta}\cdot S)_T - C_T(\delta^{\bar\theta})-w)] \tag{5.14}$$

To determine the optimal θ that minimizes the risk measure $J(\theta)$ we use the *stochastic gradient descent* and the (error) *backpropagation* algorithm. Starting with an initial guess $\theta^{(0)}$, we iteratively define

$$\theta^{(j+1)} = \theta^{(j)} - \eta_j\nabla J_j\left(\theta^{(j)}\right) \tag{5.15}$$

where the J_j used to update Equation (5.15) is given as

$$J_j(\theta) = w + \sum_{m=1}^{N_{\text{batch}}} l\left(Z\left(w_m^{(j)}\right) - (\delta^{\bar{\theta}} \cdot S)_T\left(w_m^{(j)}\right) + C_T(\delta^{\bar{\theta}})\left(w_m^{(j)}\right) - w\right) \frac{N}{N_{\text{batch}}} \mathbb{P}\left[\{w_m^{(j)}\}\right]$$

for some $w_1^{(j)}, \dots w_{N_{\text{batch}}}^{(j)} \in \Omega$ and a convex loss function l.

The explicit forms of $J(\theta)$ for the special cases of OCE risk measures considered in the article, which are the Entropic Risk Measure and CVaR could are given by:

(a) Entropic Risk (Exponential Utility)

$$J(\theta) := \mathbb{E}\left[\exp\left(-\lambda(-Z + (\delta^\theta \cdot S)_T - C_T(\delta^\theta))\right)\right] \tag{5.16}$$

The goal is to find the parameter θ^* that minimizes the exponential loss, which heavily penalizes large losses (tail risk).

(b) CVaR (as an OCE Risk)

From the OCE CVaR in Equation (5.3), we have that

$$J(\theta) := w + \mathbb{E}\left[\frac{\max\left(Z + (\delta^\theta \cdot S)_T - C_T(\delta^\theta), 0\right)}{1-\alpha}\right] \tag{5.17}$$

Here, we seek to find θ^* which minimizes the average of the worst $(1-\alpha)\%$ losses (tail conditional expectation).

5.3.4 Theoretical Guarantees

The paper provides two critical theoretical guarantees for the neural network based hedging approach. These are convergence guarantee and duality property guarantee.

Convergence Guarantee: As the neural network complexity $M \to \infty$, the neural-approximated risk $\pi^M(X)$ defined by Equation (5.13) converges to the true minimal risk $\pi(X)$ in Equation (5.6). That is,

$$\lim_{M\to\infty} \pi^M(X) = \pi(X)$$

This justifies using neural networks to approximate otherwise intractable stochastic control problems and holds for any convex risk measure.

Duality Property Guarantee: The duality property for convex risk measures is given by Equation (5.4). The primal problem of minimizing $J(\theta)$ in Equation (5.14) is equivalent to

$$\rho^M(X) := \sup_{\substack{\theta\in\Theta_{M,r(n+1),1} \\ \mathbb{E}[\exp(F^\theta\cdot\bar{I}=1}} \left(\mathbb{E}[-X\exp(F^\theta\cdot\bar{I})] - \bar{\alpha}(F^\theta\cdot\bar{I})\right) \tag{5.18}$$

such that

$$\lim_{M\to\infty} \rho^M(X) = \rho(X)$$

So we are able to find a duality formulation for the neural network parametrization of the risk measure. The dual formulation shows that hedging optimizes against the worst-case scenario, that is, the optimal hedge protects against adversarial market dynamics. The duality property being satisfied also implies the neural network strategy generalizes across different market models.

6 Numerical Experiments and Results

In this section, we perform 3 experiments to illustrate the performance and scalability of the deep hedging algorithm.

Setting and Implementation: We choose $T = 30$ business days with daily rebalancing $(t_i = i/365, i = 0, \ldots, 30)$. The market paths S is generated on the fly by a Monte-Carlo simulation(using the Heston model). We state again that the framework remains model-agnostic, that is, any stochastic price generator can replace the Heston model. At each step the agent observes $I_k = \Phi(S_0, \ldots, S_k)$ which represent the full price history plus observed derived features (like realised volatility).

We use the policy architecture design described in Section 5.3.2. The algorithm has been implemented in Python using TensorFlow to build and train the neural networks. To allow for a larger learning rate, the technique of batch normalization (see [11] and [5]) is used in each layer of each network right before applying the activation function. The network parameters are initialized randomly (drawn from uniform and normal distribution). For network training the Adam algorithm (see [4], [5]) with a learning rate of 0005 and a batch size of 256 has been used. Finally, the model hedge for the benchmark in Section 6.1 has been calculated using Quantlib.

6.1 Experiment 1: Deep Hedging vs. Classical Heston Delta Hedging under Zero Transaction Costs

The goal of this experiment is to compare the performance of deep hedging strategies to the classical Heston delta-hedging strategy in a frictionless market (without transaction cost and liquidity contraint).

Setting: We work within the framework of the discretized Heston model, where the asset price S_t^1 and its stochastic variance V_t evolve according to

$$dS_t^1 = \sqrt{V_t} S_t^1 dB_t, \qquad dV_t = \alpha(b - V_t)d_t + \sigma\sqrt{V_t}dW_t \tag{6.1}$$

where B_t and W_t are correlated Brownian motions with $\rho \in [-1, 1]$. In our setup, we have chosen $\alpha = 1$,$b = 004$,$\rho = -0.7$,$\sigma = 2$,$v_0 = 0.04$ and $s_0 = 100$, reflecting a typical situation in an equity market.

No transaction costs are considered ($c = 0$). The sample paths of (S_t^1, V_t) are generated using an exact sampling method for the Cox-Ingersoll-Ross (CIR) process (see [12]) and the simplified Brodie-Kaya scheme for S_t^1 (see [13] and [14]). In our framework V_t is modeled by an idealized variance swap with maturity T, i.e, we set $\mathcal{F}_t := \sigma((S_s^1, V_s) : s \in [0, t])$ and we define

$$S_t^2 = \int_0^t V_s ds + \frac{u - v}{\alpha}(1 - e^{-\alpha(T-t)}) + b(T - t) \tag{6.2}$$

The pair (S^1, S^2) are the prices of the liquidly tradeable assets.

The aim is to hedge an European option with payoff $g(S_T^1)$ at maturity $T = 30$ for some $g : \mathbb{R} \to \mathbb{R}$. By the Markov property of (S_1, V), one may write the option price at t as $H_t = u(t, S_t^1, V_t)$ for some $u : [0, T] \times [0, \infty)^2 \to \mathbb{R}$. The portfolio value under the risk neutral measure $\mathbb{Q}$ was shown to be given by

$$g(S_T^1) = q + \int_0^T \delta_t^1 dS_T^1 + \int_0^T \delta_t^2 dS_T^2 \tag{6.3}$$

where $q = \mathbb{E}_{\mathbb{Q}}[g(S_T^1)]$ and

$$\delta_t^1 = \partial_s u(t, S_t^1, V_t) \quad \delta_t^2 = \frac{\partial_v u(t, S_t^1, V_t)}{\partial_v L(t, V_t)} \tag{6.4}$$

where $L(t, V_t) = \dfrac{u - v}{\alpha}(1 - e^{-\alpha(T-t)}) + b(T - t)$

The benchmark strategy, known as the **model hedge**, is $\delta_k^H = (\delta_k^1, \delta_k^2)$ for $k = 0, \ldots, n-1$ with δ^1, δ^2 given by (6.4) and the corresponding risk neutral price is q.

The neural network strategy, also called the **deep hedging** strategy, is computed using the network input at each time step as $I_k = (\log(S_t^1), V_k)$. The algorithm is as described in Section 5. The networks are trained to minimize Conditional Value at Risk (CVaR) at various confidence levels.

Risk Measure: The Conditional Value at Risk (CVaR) is used to quantify the hedging performance. The CVaR at level α is defined as:

$$\rho(X) = \frac{1}{1 - \alpha} \int_0^{1-\alpha} \mathrm{VaR}_\gamma(X) d\gamma$$

where $\mathrm{VaR}_\gamma(X)$ denotes the Value-At-Risk at level γ.

Training Details: The deep hedging models are trained using the **Adam optimizer** with a learning rate of 0.005 and a mini-batch size of 256. Batch normalization is applied to accelerate convergence.

Evaluation Procedure: After training, another set of 10^6 out-of-sample test trajectories is simulated. Hedging errors are evaluated for both the model hedge and the deep hedging strategy

- Model Hedge Error: $q - Z + (\delta_H \cdot S)_T$
- Deep Hedge Error: $p_{\theta_0} - Z + (\delta_\theta \cdot S)_T$

where q is the risk-neutral and $p_{\theta_0}$is the risk-adjusted price obtained from training (approximation of $q(Z)$ in (5.5)).

6.1.1 Hedging a Vanilla Call Option

In this first example, we consider hedging a plain vanilla European call option written on the underlying asset S^1, whose dynamics are governed by the Heston stochastic volatility model (6.1). It's value at maturity is

$$Z = (S_T^1 - K)^+ \quad \text{with} \quad K = s_0$$

Key Results: Below are the price results for different risk preference levels:

Table 1: Comparison of Prices under Different Risk Preferences

Risk Preference Level	**Model Price** (q)	**Deep Hedge Price** (p_0^θ)
$\alpha = 0.5$	1.6918	1.8337
$\alpha = 0.99$	1.6918	22.7549

The following figure illustrate the comparisons between the model hedge δ^H and the deep hedging strategies δ^θ corresponding to CVaR of 50%.

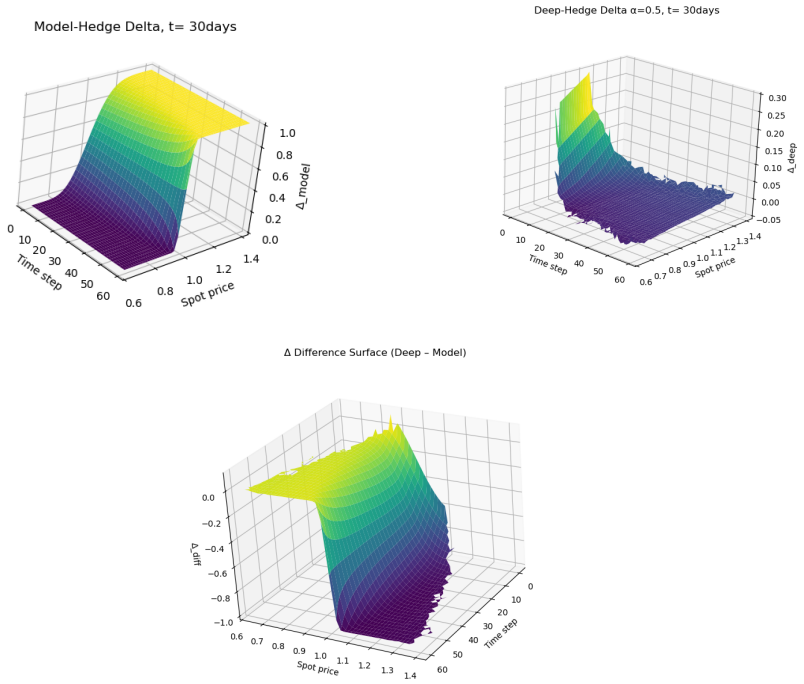

Figure 1: Call Option Delta from Heston Model and Deep Hedging (no transaction costs)

6.1.2 Hedging a Call Spread

Next, we demonstrate the flexibility of the deep hedging algorithm in handling complex derivatives like a call spread in a stochastic volatility model. The payoff of the call spread is given by

$$Z = \frac{(S_T^1 - K_1)^+ - (S_T^1 - K_2)^+}{K_2 - K_1} \quad \text{with} \quad K_1 = 100 \text{ and } K_2 = 101$$

We aim to study the performance of the deep hedge algorithm relative to the model hedge under extreme market scenarios. We do this by comparing the abilities of the deep hedge and model hedge in capturing tail events by plotting the distributions of deep hedge error for different risk aversion levels as well as the model hedge errors. This is illustrated in Figure 2. We choose CVaR of 95% and 99% respectively to achieve this goal.

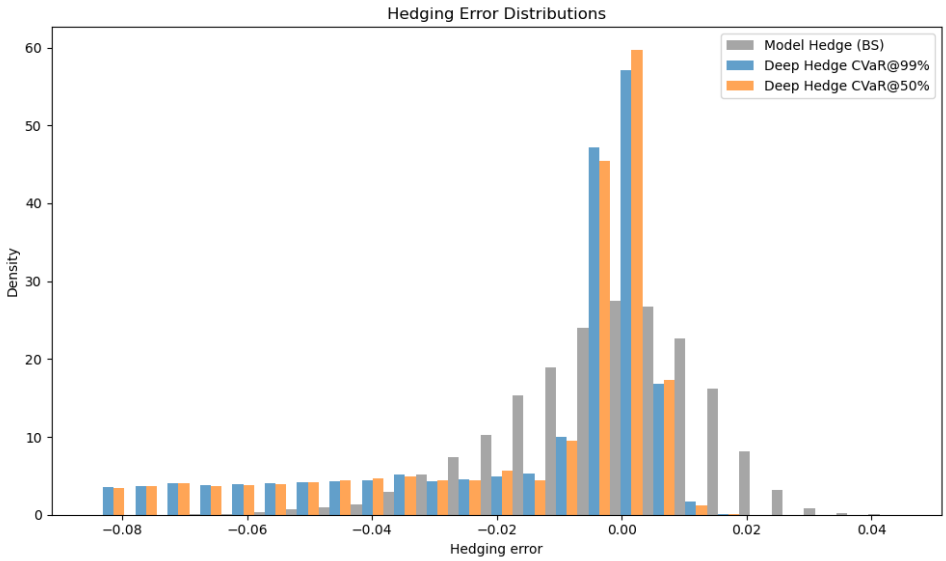

Figure 2: Comparison of Model Hedge Errors vs. Deep Hedge Errors

From Figure 2, we see that the deep hedge method is efficient in capturing tail events and outperforms the model hedge in this regard.

In experiment 1, we have obtained the following results:

- Model-delta hedge yields accurate pricing but poor risk control under high quantile losses.
- Deep hedging adapts to market uncertainty and risk preference via CVaR.
- For $\alpha = 0.5$, deep hedge price is close to model price but improves hedging error distribution.
- For $\alpha = 0.99$, deep hedge leads to significantly higher price, reflecting risk aversion to tail losses.

6.2 Experiment 2: Price Asymptotics under Proportional Transaction Costs

Here, our goal is to find out whether an introduction of transaction cost into the deep hedge algorithm will change the deep hedge price by a known theoretical factor.

Known Theoretical Results: We define proportional transaction cost as

$$c_k(n) = \sum_{i=1}^{d} \varepsilon |n^i| S_k^i \tag{6.5}$$

where ε is the transaction cost parameter. Also let:

- p^0 be the indifference price without transaction cost.
- p^ε be the indifference price with transaction cost ε.

It is known that the change in utility indifference price $p^\varepsilon - p^0$ satisfies

$$p^\varepsilon - p^0 = \mathcal{O}(\varepsilon^{2/3}) \quad \text{as} \quad \varepsilon \downarrow 0 \tag{6.6}$$

This is a non-trivial result (not obvious dimensionally, but it comes from optimizing trade frequency – essentially optimal policy trades less frequently as costs go up, leaving some risk, leading to that particular scaling of price with ε).

Experiment Setup: We now consider a Heston model with two hedging instruments, i.e. $d = 2$ and the setting is precisely as in Section (6.1) except that ρ here is chosen as (5.2) and proportional transaction costs (6.5) are incurred. Choosing $\lambda = 1$ and $Z = (S_T^1 - K_1)^+$ and $\varepsilon_i = 2^{-i+5}$ for $i = 1, \cdots, 5$, we emphasize the results derived by the paper in this experiment. The calculated exponential utility indifference prices p^{ε_i} as well as show the differences to p^0 are shown in a log-log plot (see Figure 3). These are shown as red dots in Figure 3. Here the blue line in Figure 3 is the regression line, i.e. the least squares fit of the red dots. The rate is very close to 2/3 and so it appears that the relation (6.6) also holds in this case.

Next, we try to find out if the deep hedge algorithm can adjust its hedging strategy as transaction cost is introduced. We use the same setup here but choose ρ as CVaR with $\alpha = 0.5$. That is, we introduce transaction costs to the setup in (Section 6.1.1) and plot the model hedge and deep hedge delta strategies. The plots are shown in Figure 4.
In experiment 2, we have obtained the following results:

- Neural networks trained under transaction cost environments accurately recovered the asymptotic growth behavior.
- Log-log plots of price difference vs. transaction cost show slopes close to 2/3, validating theory.
- Demonstrates that Deep Hedging can internalize cost sensitivity and produce correct pricing and hedging strategies without explicit PDE modeling.

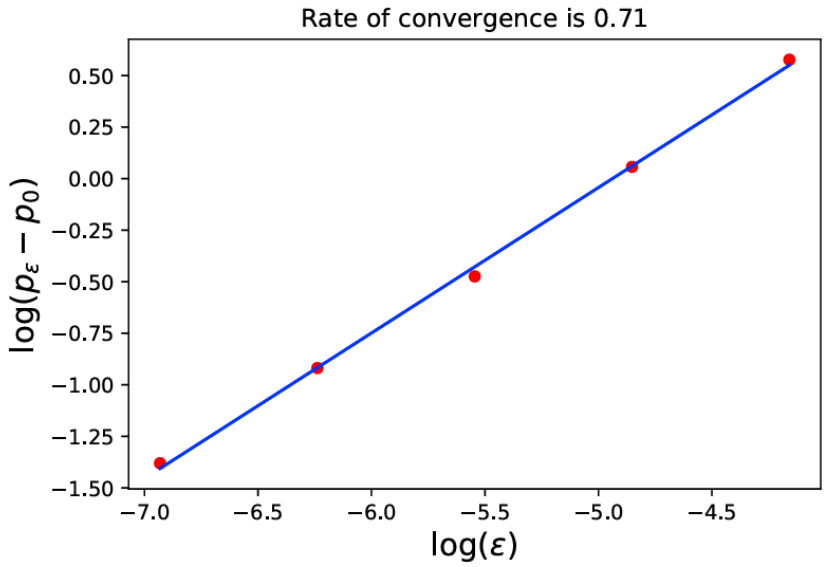

Figure 3: Comparison of Model Hedge Errors vs. Deep Hedge Errors

6.3 Experiment 3: High Dimensional Hedging

In this experiment, we evaluate the scalability of the deep hedging framework in a high-dimensional market by simulating multiple independent sources of risk and corresponding hedging instruments.

Setup. We consider a market consisting of 5 independent Heston stochastic volatility models, each representing a distinct underlying asset. For each asset, we assume access to two hedging instruments: the stock itself and a variance swap. This results in a total of 10 tradable instruments across the 5 assets.
The liability to be hedged is a portfolio of 5 European call options, one written on each of the 5 underlying stocks, all with identical maturities and strikes. The payoff function is:

$$Z = \sum_{h=1}^{5} \left(S_T^{(h)} - K \right)^+,$$

where $S_T^{(h)}$ denotes the terminal price of the h-th stock and K is the strike price (assumed to be 100 for all options).
The goal is to construct a hedging strategy that minimizes the variance of the terminal profit and loss (P&L), using the 10 available hedging instruments. This experiment is conducted under a zero transaction cost setting to isolate the effect of dimensionality on performance.

Results. The deep hedging framework successfully learns an effective hedging strategy in this high-dimensional setup. Notably, the trained strategy naturally decomposes into independent hedges for each asset. This emergent decoupling aligns with intuition, given the independence of the underlying processes.

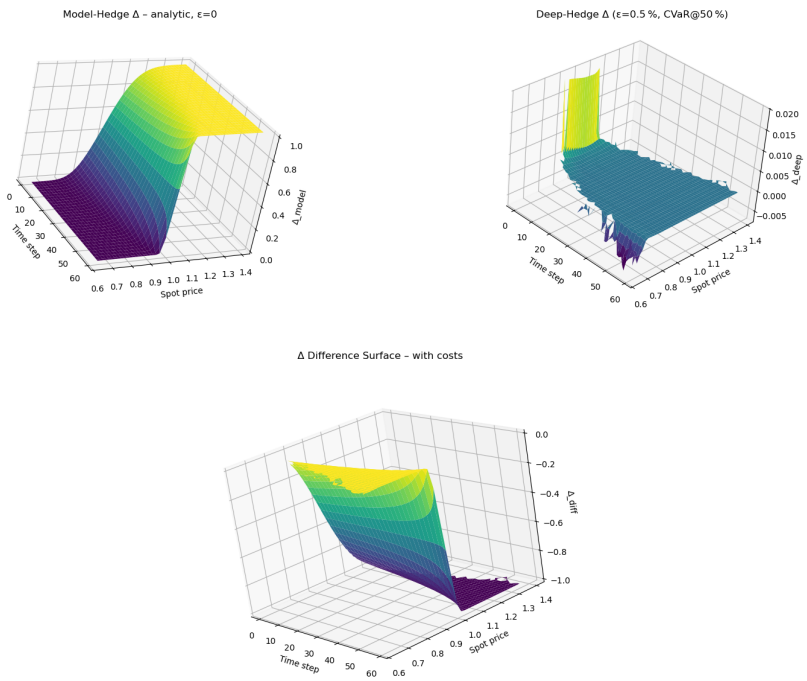

Figure 4: Call Option Delta from Heston Model and Deep Hedging under transaction costs

The network learns to hedge each option using its associated stock and variance swap, without cross-hedging across assets. This demonstrates that the method can not only handle high-dimensional inputs, but also infer and exploit problem structure without explicit programming.

Summary of Experiment 3. This experiment illustrates that deep hedging scales effectively to markets with many tradable instruments and independent sources of risk. The approach can discover and leverage structural separability when present, enabling efficient learning even in 10-dimensional hedging problems where classical methods would be computationally infeasible.

7 Conclusion

This project demonstrates the practical applicability and effectiveness of the deep hedging framework proposed by Buehler et al. (2019). Through a series of carefully designed experiments, we validated the method's ability to generate optimal hedging strategies under a variety of market conditions, risk measures, and cost structures.
Our key findings are summarized as follows:

- **Feasibility:** Deep hedging is not only conceptually sound but also computationally feasible. Using standard optimization tools (e.g., Adam optimizer in TensorFlow), we were able to train neural network-based policies that closely matched or outperformed classical strategies.

- **Flexibility:** The framework adapts to a wide range of risk preferences, including Conditional Value-at-Risk (CVaR) and exponential utility. It also accommodates market frictions such as proportional transaction costs and can recover known theoretical pricing asymptotics.

- **Accuracy:** In a frictionless Heston environment, deep hedging reproduced benchmark results, yielding competitive pricing and reduced hedging errors. Under transaction costs, the learned policies displayed robust sensitivity and conformed to expected asymptotic behavior.

- **Scalability:** The deep hedging method scales efficiently to high-dimensional settings. In our final experiment, a 10-instrument market composed of 5 independent Heston models was successfully hedged using a semi-recurrent neural network, with the resulting strategy exhibiting an intuitive decoupled structure.

While computational constraints limited us from fully replicating every scenario in the original paper, we were able to reproduce all major experiments and theoretical insights. Overall, this project confirms that deep hedging is a powerful and scalable alternative to traditional hedging techniques, especially in realistic financial environments characterized by frictions, risk aversion, and high dimensionality.

8 References

1. Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance, 19*(8), 1271–1291.

2. Jonsson, M., Ilhan, A., & Sircar, R. (2009). Optimal static-dynamic hedges for exotic options under convex risk measures. *Stochastic Processes and their Applications, 119*(10), 3608–3632.

3. Föllmer, H., & Leukert, P. (2000). Efficient hedging: Cost versus shortfall risk. *Finance and Stochastics, 4*(2), 117–146.

4. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR).*

5. Bengio, Y., Goodfellow, I., & Courville, A. (2016). *Deep learning.* MIT Press. http://www.deeplearningbook.org

6. Hodges, S., & Neuberger, A. (1989). Optimal replication of contingent claims under transaction costs. *The Review of Futures Markets, 8*(2), 222–239.

7. Panas, V. G., Davis, M. H. A., & Zariphopoulou, T. (1993). European option pricing with transaction costs. *SIAM Journal on Control and Optimization, 31*(2), 470–493.

8. Whalley, A. E., & Wilmott, P. (1997). An asymptotic analysis of an optimal hedging model for option pricing with transaction costs. *Mathematical Finance, 7*(3), 307–324.

9. Kallsen, J., & Muhle-Karbe, J. (2015). Option pricing and hedging with small transaction costs. *Mathematical Finance, 25*(4), 702–723.

10. Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4*(2), 251–257.

11. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 37, 448–456.

12. Glasserman, P. (2004). *Monte Carlo methods in financial engineering*. Applications of mathematics: Stochastic modelling and applied probability. Springer.

13. Jackel, P., Andersen, L. B. G., & Kahl, C. (2010). Simulation of square-root processes. *Encyclopedia of Quantitative Finance*. John Wiley & Sons, Ltd.

14. Broadie, M., & Kaya, O. (2006). Exact simulation of stochastic volatility and other affine jump diffusion processes. *Operations Research, 54*(2), 217–231.