

---

# Distributional Adversarial Attacks and Training in Deep Hedging

---

**Guangyi He**

Department of Mathematics  
Imperial College London  
g.he23@imperial.ac.uk

**Tobias Sutter**

Department of Economics  
University of St. Gallen  
tobias.sutter@unisg.ch

**Lukas Gonon**

School of Computer Science  
University of St. Gallen  
lukas.gonon@unisg.ch  
Department of Mathematics  
Imperial College London  
l.gonon@imperial.ac.uk

## Abstract

In this paper, we study the robustness of classical deep hedging strategies under distributional shifts by leveraging the concept of adversarial attacks. We first demonstrate that standard deep hedging models are highly vulnerable to small perturbations in the input distribution, resulting in significant performance degradation. Motivated by this, we propose an adversarial training framework tailored to increase the robustness of deep hedging strategies. Our approach extends pointwise adversarial attacks to the distributional setting and introduces a computationally tractable reformulation of the adversarial optimization problem over a Wasserstein ball. This enables the efficient training of hedging strategies that are resilient to distributional perturbations. Through extensive numerical experiments, we show that adversarially trained deep hedging strategies consistently outperform their classical counterparts in terms of out-of-sample performance and resilience to model misspecification. Additional results indicate that the robust strategies maintain reliable performance on real market data and remain effective during periods of market change. Our findings establish a practical and effective framework for robust deep hedging under realistic market uncertainties.<sup>1</sup>

## 1 Introduction

In the past decades, options and derivatives trading has seen enormous growth, with billions of contracts traded every year [1]. Hedging of financial derivatives refers to taking financial positions in other linked assets to mitigate the risk associated with the derivative. Thereby, hedging is a fundamentally important task in the financial services industry. Deep Hedging [2] introduces a model-free, data-driven framework that employs deep learning to optimize hedging strategies directly from simulated market scenarios. The core idea is to parameterize hedging decisions via neural networks. Optimal hedging is achieved by training the neural networks to minimize a risk measure of the hedged portfolio’s profit and loss (P&L). While this approach has been widely adopted in industry, the performance of learned hedging strategies critically depends on the quality of the training data. For instance, the data-generating distribution during training might differ slightly from the test distribution under which the deep hedging strategy is ultimately deployed and evaluated. This mismatch, known as *model misspecification*, can result in suboptimal or even misleading decisions, a phenomenon that is well documented in various classes of data-driven decision-making problems [3, 4]. This

---

<sup>1</sup>The code is available on <https://github.com/Guangyi-Mira/Distributional-Adversarial-Attacks-and-Training-in-Deep-Hedging>

highlights the fundamental challenge of modeling uncertainty in finance and the importance of designing strategies that remain robust to small changes in the underlying distribution.

To mitigate model uncertainty in decision-making problems, a large variety of different approaches exist. Parametric methods quantify uncertainty on model parameters, allowing strategies to account for potential estimation errors, like in [5, 6]. Moreover, advances in machine learning enable the use of generative models with high-dimensional parameter spaces to simulate complex dynamics [7, 8, 9]. In the field of deep hedging, recent works consider robustness by perturbing the terminal distribution [10], randomized model parameters [11] or penalizing deviations from a reference distribution [12].

On the other hand, model uncertainty can be addressed within the framework of *distributionally robust optimization* (DRO) [13]. DRO seeks to make optimal decisions under the worst-case scenario over a specified set of probability distributions, known as the *ambiguity set*. This problem can be formulated as a two-player game

$$\min_{\theta \in \Theta} \max_{\eta \in B_\delta(\mu)} \mathbb{E}_{x \sim \eta} [l(\theta; x)], \quad (1.1)$$

where  $\theta$  denotes the decision variable optimized over the parameter space  $\Theta$ , representing the first player. The adversarial player selects a distribution  $\eta$  from an ambiguity set  $B_\delta(\mu)$  of plausible distributions centered around a nominal distribution  $\mu$ . The objective is described by the loss function  $l(\theta; x)$ , parameterized by  $\theta$  and evaluated at an input  $x \in \mathbb{R}^d$ , which is modeled as a random variable. The parameter  $\delta > 0$  controls the size of the ambiguity set, with  $B_0(\mu) = \{\mu\}$ . Thus, when  $\delta = 0$ , the DRO problem (1.1) reduces to the nominal stochastic optimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim \mu} [l(\theta; x)]. \quad (1.2)$$

Various functions to encode the difference between probability distributions in  $B_\delta(\mu)$  are popular, such as the  $\phi$ -divergence [14], matching moments [15], and the total variation distance [16]. Among these, the Wasserstein distance is particularly popular and powerful, see the appendix for the detailed definition and see [17, 18, 13] for detailed treatments of Wasserstein-based DRO problems. These existing computational tractability results in Wasserstein DRO rely on structural assumptions about the underlying loss function, such as convexity, which do not hold in the deep hedging setting considered in this work.

The concept of robustness is important across machine learning applications. In image classification tasks, [19] first highlighted that neural networks are vulnerable to adversarial attacks—small, imperceptible changes to input images that cause misclassification. This discovery led to a growing body of research on adversarial attacks and training techniques to improve network robustness against them ([20, 21, 22]). However, most studies have focused on pointwise adversarial perturbations, where attacks modify individual data points. A broader perspective considers distributional attacks, where an adversary perturbs the entire data-generating distribution rather than specific samples ([23, 24, 25, 18, 13, 26, 27]). Specifically, the distributional version of adversarial attacks can be applied to search for the worst-case scenario in the DRO problem.

In this paper, we study the robustness properties of classical deep hedging strategies under distribution shifts by utilizing the concept of adversarial attacks. The main contributions of this paper can be summarized as follows:

- We propose the first framework that unifies distributional adversarial training with deep hedging. Building on tractable reformulations from Wasserstein DRO [26] and sensitivity analysis results [23], our approach derives computationally efficient adversarial attacks within a Wasserstein ball and integrates them into an adversarial training procedure. This framework reveals the vulnerability of classical deep hedging to even small distributional shifts and provides a tractable method to enhance robustness.
- Prior work [13, 17, 18] establishes that data-driven Wasserstein DRO solutions guarantee out-of-sample performance under certain theoretical assumptions. However, in complex dynamic settings such as deep hedging, these assumptions do not generally hold. Through adversarial training with limited data (5,000–100,000 samples), we empirically demonstrate that adversarially trained strategies outperform classical deep hedging methods in terms of out-of-sample and out-of-distribution performance.
- We validate our framework on real market data, demonstrating that adversarially trained strategies remain effective and robust during periods of market stress and distributional shifts, and compare our results against the robust deep hedging approach of [11].

**Related work.** Deep hedging was introduced in [2, 28]. Since its introduction, numerous works have studied deep hedging and related machine learning-based hedging methods from different perspectives. Robustness of deep hedging methods has been studied in [10, 11, 12]. Further directions include tackling complex pricing problems [29, 30, 31], targeting improved efficiency [32, 33] or general initial portfolios [34], alternative reinforcement learning frameworks [35, 36, 37, 38], empirical approaches [39, 40], or implementations on quantum hardware [41]. We refer to the surveys [42, 43] for a comprehensive overview.

Parallel to this, distributionally robust optimization (DRO) has emerged as a framework for decision-making under model uncertainty, with Wasserstein-based methods offering both theoretical guarantees and computational tractability [13, 17, 18]. In machine learning, adversarial training provides another robustness perspective, where worst-case perturbations improve generalization [19, 20, 21, 22]. Recent works extend pointwise adversarial attacks to the distributional setting [23, 24, 25, 26, 27], thereby connecting adversarial training methods with DRO formulations.

## 2 Deep Hedging Framework

**Basic setup.** In Deep Hedging [2], we consider a discrete-time market with trading dates  $\{0, 1, \dots, T\}$  and a set of  $r$  tradable assets. Let  $\mathbf{S} = (S_1, \dots, S_{T+1}) \in \mathbb{R}^{r \times (T+1)}$  denote the mid-price trajectories of the assets. The price process  $\mathbf{S}$  is adapted to a filtration  $(\mathcal{F}_t)_{t=0}^T$  generated by an information process  $\mathbf{I} = (I_1, \dots, I_{T+1})$ , where each  $I_t \in \mathbb{R}^d$  captures market-relevant features at time  $t$  such as asset prices, volatility, risk limits, or trading signals. We denote the price trajectories  $\mathbf{S}(\mathbf{I})$  as a function of  $\mathbf{I}$ . The trajectory  $\mathbf{I}$  is sampled from a known distribution  $\mu$ , which may correspond to a stochastic model or a uniform distribution over simulated or historical scenarios. A trading strategy  $\delta = (\delta_1, \dots, \delta_{T-1}) \in \mathbb{R}^{r \times T}$  represents the portfolio holdings in the  $r$  assets. Each position  $\delta_t$  is determined by a neural network that is parameterized by  $\theta = (\theta_1, \dots, \theta_T)$  and takes the history of available information up to time  $t$  as input,

$$\delta_t = f_{\theta_t}(I_1, \dots, I_t), \quad (2.1)$$

where  $f_{\theta}$  is a function parameterized by network weights  $\theta_t \in \mathbb{R}^n$ . Therefore, the whole trading strategy  $\delta$  depends on the network parameter  $\theta \in \Theta$  and information process  $\mathbf{I}$ , denoted  $\delta(\theta, \mathbf{I})$ .

We consider hedging a contingent claim with payoff  $P(S_T)$  at maturity, where  $P : \mathbb{R}^r \rightarrow \mathbb{R}$  is a given payoff function. The profit and loss (PnL) of the hedging position is defined as

$$\text{PnL}(\theta, \mathbf{I}) = p_0 + \sum_{t=1}^T \delta_t^\top(\theta, \mathbf{I})(S_{t+1}(\mathbf{I}) - S_t(\mathbf{I})) - P(S_T(\mathbf{I})) \quad (2.2)$$

for a given price  $p_0 \geq 0$ . The objective is to optimize the hedging outcome  $\text{PnL}(\theta, \mathbf{I})$  under a convex risk measure  $\rho$ , such as the entropic risk measure or Conditional Value at Risk (CVaR) [44], i.e., solve

$$\min_{\theta \in \Theta} \rho[\text{PnL}(\theta, \mathbf{I})]. \quad (2.3)$$

**Optimized Certainty Equivalents.** We focus on convex risk measures that admit an optimized certainty equivalent (OCE) form. Specifically, consider convex risk measures that can be expressed as

$$\rho(Z) = \inf_{\omega \in \mathbb{R}} \{\omega + \mathbb{E}[\ell(-Z - \omega)]\}, \quad (2.4)$$

for random variables  $Z$  and a fixed loss function  $\ell : \mathbb{R} \rightarrow \mathbb{R}$ . OCE risk measures form an important class of convex risk measures [45], naturally suited for deep hedging [2]. During training, we implement an OCE risk measure by treating  $\omega$  as a trainable parameter. That is, we consider an augmented parameter  $\tilde{\theta} := (\theta, \omega)$  and define the deep hedging loss as

$$l_{\text{DH}}(\tilde{\theta}, \mathbf{I}) = \omega + \ell(-\text{PnL}(\theta, \mathbf{I}) - \omega). \quad (2.5)$$

We can equivalently express the deep hedging problem (2.3) as a standard expected loss minimization

$$\min_{\tilde{\theta}} \mathbb{E}_{\mathbf{I} \sim \mu}[l_{\text{DH}}(\tilde{\theta}, \mathbf{I})]. \quad (2.6)$$

During the evaluation, the optimal  $\omega$  can be obtained by directly solving (2.4).

**DRO Formulation of Deep Hedging.** The reformulated deep hedging problem (2.6) now has the common structure of (1.2), which induces a corresponding DRO formulation

$$\min_{\tilde{\theta}} \max_{\eta \in B_{\delta}(\mu)} \mathbb{E}_{\mathbf{I} \sim \eta} [l_{\text{DH}}(\tilde{\theta}, \mathbf{I})]. \quad (2.7)$$

The OCE framework naturally encompasses common convex risk measures such as entropic risk and Conditional Value-at-Risk (CVaR), whose specific implementations will be demonstrated in subsequent examples. Crucially, the OCE reformulation aligns with standard DRO formulations, enabling direct utilization of established convergence guarantees and computational methods.

**Examples.** We consider a Black-Scholes model and General Affine Diffusion model with entropic risk and a Heston model with CVaR. Detailed formulations of these models and the associated loss functions are provided in the appendix.

### 3 Adversarial Attacks

**Pointwise adversarial attacks.** Pointwise adversarial attacks aim to perturb the input within a  $\delta$ -ball centered at the input to maximize the corresponding loss, that is,

$$\max_{\hat{x} \in \mathbb{R}^d} l(\theta; \hat{x}) \quad \text{subject to} \quad d(\hat{x}, x) \leq \delta. \quad (3.1)$$

Two cornerstone algorithms in this domain are the Fast Gradient Sign Method (FGSM) [20] and Projected Gradient Descent (PGD) [21]. For input  $x$  and  $L^\infty$  distance  $d(\cdot, \cdot)$ , the FGSM perturbation is computed as

$$\hat{x} = x + \delta \cdot \text{sign}(\nabla_x l(\theta; x)), \quad (3.2)$$

where  $\delta$  is the perturbation magnitude,  $\nabla_x l(\theta; x)$  is the gradient of the loss function  $l$  with respect to the input  $x$ , and  $\text{sign}(\cdot)$  denotes the element-wise sign function. FGSM directly perturbs the input data in the direction of the gradient to the boundary of the  $\delta$ -ball around the original input  $x$ , which is computationally efficient. For a deeper exploration of the worst-case perturbation, PGD is an iterative extension of FGSM that applies multiple small perturbations to the input. At each iteration, the input is updated as

$$\hat{x}^{(k+1)} = \text{Proj}_{d(\cdot, x) \leq \delta}(\hat{x}^{(k)} + \beta \cdot \text{sign}(\nabla_x l(\theta; \hat{x}^{(k)}))), \quad (3.3)$$

where  $\beta$  is the step size, and  $\text{Proj}_{d(\cdot, x) \leq \delta}$  projects the perturbed input back into the  $\delta$ -ball around  $x$  to ensure the perturbation remains bounded.

**Distributional adversarial attacks.** Distributional adversarial attacks aim to find the worst-case data distribution perturbation within an ambiguity set  $B_{\delta}(\mu)$  that maximizes the expected loss of the model. This is formalized as

$$V_{\theta}(\delta) = \max_{\eta \in B_{\delta}(\mu)} \mathbb{E}_{x \sim \eta} [l(\theta; x)], \quad (3.4)$$

where an optimal perturbed distribution  $\eta^*$  achieves this maximum. Solving (3.4) requires to optimize over an infinite-dimensional Wasserstein ball, making direct approaches intractable. Therefore, we provide a reformulation that approximates (3.4) based on the sensitivity results on DRO in Wasserstein balls that are inspired by [46]. The proposed reformulation relies on the following two assumptions.

**Assumption 3.1.** The distance  $d$  on  $\mathbb{R}^d$  is induced by a norm  $\|\cdot\|$ , with corresponding dual norm  $\|\cdot\|_*$  defined as

$$\|y\|_* = \sup_{\|x\| \leq 1} \langle y, x \rangle. \quad (3.5)$$

Furthermore, there exists a function  $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$  such that

$$\|x\|_* = \langle h(x), x \rangle \quad \text{for all } x \in \mathbb{R}^d. \quad (3.6)$$

**Assumption 3.2.** For each  $\theta$ , the map  $x \mapsto l(\theta; x)$  is Lipschitz continuous.

Assumption 3.1 is satisfied, e.g., for  $d$  chosen as the distance induced by the  $\|\cdot\|_p$  norm. In this case, the dual norm is given by  $\|\cdot\|_q$ , where  $q$  is the Hölder conjugate satisfying  $1/p + 1/q = 1$ . The corresponding function  $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is explicitly given (with operations applied component-wise) as

$$h(x) = \text{sign}(x) |x|^{q-1}. \quad (3.7)$$

Additionally, we focus on the data-driven framework, where the input distribution  $\mu$  is the empirical distribution constructed from the training set  $\{X_1, \dots, X_N\}$ , i.e.,  $\mu = \frac{1}{N} \sum_{n=1}^N \delta_{X_n}$ .

Analogously to [23, Theorem 4.1] in the setting of a classification problem, we use [46, Theorem 2.1] and its proof to establish the following theorem. The details are provided in the appendix.

**Theorem 3.3.** *Under Assumption 3.1 and Assumption 3.2,  $V_\theta(\delta)$  can be approximated by*

$$V_\theta(\delta) = \mathbb{E}_{\eta_\delta}[l(\theta, x)] + o(\delta) \quad \text{as } \delta \downarrow 0, \quad (3.8)$$

where  $\eta_\delta = \frac{1}{N} \sum_{n=1}^N \delta_{\hat{X}_n}$ . For  $n = 1, \dots, N$ , each  $\hat{X}_n$  is the perturbation of the original sample

$$\hat{X}_n = X_n + \delta \cdot h(\nabla_x l(\theta; X_n)) \|\nabla_x l(\theta; X_n)\|_*^{q-1} \Upsilon^{1-q}, \quad (3.9)$$

$$\Upsilon = \left( \frac{1}{N} \sum_{n=1}^N \|\nabla_x l(\theta; X_n)\|_*^q \right)^{1/q}. \quad (3.10)$$

Equation (3.9) provides an approximate worst-case perturbation by leveraging the sensitivity expansion of the Wasserstein DRO problem, which directly yields a computationally tractable solution that approximates (3.4). However, this approach may not fully explore the adversarial landscape. Therefore, we instead constrain optimization to a subset of the ambiguity set which contains  $\eta_\delta$  and has a simpler structure. The set  $\hat{B}_\delta(\mu_N)$  is defined as

$$\hat{B}_\delta(\mu_N) = \left\{ \hat{\mu}_N = \frac{1}{N} \sum_{n=1}^N \delta_{\hat{X}_n} : \hat{X}_i \in \mathbb{R}^d, \left( \frac{1}{N} \sum_{i=1}^N d(X_i, \hat{X}_i)^p \right)^{1/p} \leq \delta \right\}. \quad (3.11)$$

**Lemma 3.4.** *Consider the empirical optimization problem*

$$V_\theta^e(\delta) = \max_{\eta \in \hat{B}_\delta(\mu)} \mathbb{E}_{x \sim \eta}[l(\theta; x)], \quad (3.12)$$

obtained by replacing the ambiguity set  $B_\delta(\mu)$  in (3.4) by  $\hat{B}_\delta(\mu)$  as defined in (3.11). Under Assumptions 3.1 and 3.2, we have

$$V_\theta(\delta) = V_\theta^e(\delta) + o(\delta) \quad \text{as } \delta \downarrow 0. \quad (3.13)$$

The set  $\hat{B}_\delta(\mu_N)$  has been previously studied in the literature [26, 27] with focus on the setting where  $N$  grows. Our paper provides a sensitivity analysis as  $\delta \downarrow 0$ . Finally, we write (3.12) into a constrained problem perturbing the dataset directly

$$V_\theta^e(\delta) = \max_{\hat{X}_1, \dots, \hat{X}_N} \frac{1}{N} \sum_{n=1}^N l(\theta; X_n) \quad \text{subject to} \quad \left( \frac{1}{N} \sum_{i=1}^N d(X_i, \hat{X}_i)^p \right)^{1/p} \leq \delta. \quad (3.14)$$

It is worth noticing that, as  $p \rightarrow \infty$ , the constraint in (3.14) becomes an  $L^\infty$ -constraint, i.e.,  $\max_n(d(X_n, \hat{X}_n)) \leq \delta$ , which coincides with the pointwise adversarial attack setting.

## 4 Algorithms for Distributional Adversarial Attacks in Deep Hedging

In this section, we develop a numerical algorithm to solve the distributional adversarial attack problem (3.4) via its tractable approximation (3.14) in the deep hedging framework. Drawing inspiration from adversarial training methods in machine learning, we adapt the Fast Gradient Sign Method (FGSM) [20] and Projected Gradient Descent (PGD) [21] to the distributional setting. For each algorithm, we first propose a one-step FGSM-like perturbation to generate adversarial distributions efficiently. We then extend this to a multi-step PGD-like algorithm, which iteratively refines the perturbation with smaller steps while constraining the perturbation through a projection step.

We first focus on the Black-Scholes model, where the input information is exactly the single price trajectory  $\mathbf{S}$ , as detailed in the appendix. We will then explain how to extend the algorithm to models with more than one trajectory, including the Heston model, where the input is a pair of price and variance trajectories  $\mathbf{S}$  and  $\mathbf{v}$ . For the readers' convenience, we focus on these two cases here and present algorithms for a general setting in the supplementary material.

In deep hedging, with the loss function  $l_{DH}(\theta; \mathbf{S})$  as defined in (2.5), problem (3.14) becomes

$$\max_{\hat{\mathbf{S}}_1, \dots, \hat{\mathbf{S}}_N} \frac{1}{N} \sum_{n=1}^N l_{DH}(\theta; \hat{\mathbf{S}}_n) \quad \text{subject to} \quad \left( \frac{1}{N} \sum_{n=1}^N d(\mathbf{S}_n, \hat{\mathbf{S}}_n)^p \right)^{1/p} \leq \delta. \quad (4.1)$$

We define the distance in the input space  $\mathbb{R}^{T+1}$  as the  $L^\infty$  distance, i.e.,

$$d(\mathbf{S}_n, \hat{\mathbf{S}}_n) = \|\mathbf{S}_n - \hat{\mathbf{S}}_n\|_\infty = \max_{t=0, \dots, T} |S_{n,t} - \hat{S}_{n,t}|. \quad (4.2)$$

The infinity norm has the  $L^1$ -norm as its dual norm, i.e.,  $\|g\|_* = \sum_{t=0}^T |g_t|$ , and  $h(x) = \text{sign}(x)$ , applied component-wise, satisfies  $\langle h(x), x \rangle = \|x\|_*$ , which implies Assumption 3.1.

**Wasserstein Projection Gradient Decent (WPGD).** Following the above setup, equation (3.9) provides an asymptotic approximation of the worst-case distributional shift, leading to a one-step perturbation of each path like FGSM

$$\mathbf{S}_n \mapsto \mathbf{S}_n + \delta \cdot \text{sign}(g_n^S) \|g_n^S\|_*^{q-1} \Upsilon^{1-q}, \quad (4.3)$$

where  $g_n^S$  is the gradient of  $l_{DH}(\theta; \mathbf{S}_n)$  with respect to  $\mathbf{S}_n$  and  $\Upsilon$  is defined as  $(\frac{1}{N} \sum_{n=1}^N \|g_n^S\|_*^q)^{1/q}$ . To propose an analogous PGD-like algorithm, we iteratively apply (4.3) with adjusted step-size  $\beta$  and project the perturbed path back to a ball defined by the constraint in (4.1). During the projection, each sample  $\hat{\mathbf{S}}_n$  is updated to

$$\hat{\mathbf{S}}_n \leftarrow \mathbf{S}_n + \max(1, \delta/\text{dist})(\hat{\mathbf{S}}_n - \mathbf{S}_n), \quad (4.4)$$

where  $\text{dist} = (\frac{1}{N} \sum_{i=1}^N d(\mathbf{S}_i, \hat{\mathbf{S}}_i)^p)^{1/p}$ . The overall procedure, called the Wasserstein Projection Gradient Descent (WPGD), is summarized in Algorithm 1.

---

**Algorithm 1** Wasserstein Projection Gradient Descent (WPGD)

---

```

1: for  $i = 1$  to  $\text{num\_of\_iteration}$  do
2:   Compute  $\hat{\Upsilon} := (\frac{1}{N} \sum_{n=1}^N \|\nabla_x l_{DH}(\theta; \hat{\mathbf{S}}_n)\|_*^q)^{1/q}$ 
3:   for  $n = 1$  to  $N$  do
4:      $\hat{\mathbf{S}}_n \leftarrow \hat{\mathbf{S}}_n + \beta \cdot \text{sign}(\nabla_x l_{DH}(\theta; \hat{\mathbf{S}}_n)) \|\nabla_x l(\theta; \hat{\mathbf{S}}_n)\|_*^{q-1} \hat{\Upsilon}^{1-q}$ ,
5:   end for
6:    $\hat{\mathbf{S}}_1, \dots, \hat{\mathbf{S}}_N \leftarrow \text{Proj}_{\hat{B}_\delta(\mu)}(\hat{\mathbf{S}}_1, \dots, \hat{\mathbf{S}}_N)$ 
7: end for

```

---

**Wasserstein Budget Projection Gradient Descent (WBPGD).** We can interpret  $d(\mathbf{S}_n, \hat{\mathbf{S}}_n)^p$  in (4.1) as a budget allocated to path  $\mathbf{S}_n$  during the attack. This intuition motivates a reparameterization of the perturbed sample as

$$\hat{\mathbf{S}}_n = \mathbf{S}_n + \text{budget}_n \times \text{direction}_n, \quad (4.5)$$

where for each sample  $\mathbf{S}_n$ , the variable  $\text{budget}_n \in \mathbb{R}_{\geq 0}$  represents the magnitude of the perturbation, and  $\text{direction}_n$ , with the same dimension as  $\mathbf{S}_n$ , denotes its direction bounded within  $[-1, 1]$ . Then, the optimization problem (4.1) can be considered as allocating  $\text{budget}_n$  with the restriction  $(\frac{1}{N} \sum_{n=1}^N \text{budget}_n^p)^{1/p} = \delta$  and optimizing the perturbation direction for each sample. We have the following one-step updates on  $\text{budget}_n$  and  $\text{direction}_n$  based on (4.3).

**Lemma 4.1.** *For the perturbation  $\hat{\mathbf{S}}_n$  satisfying (4.3), we have the equivalent representation*

$$\hat{\mathbf{S}}_n = \mathbf{S}_n + \text{budget}_n \times \text{direction}_n. \quad (4.6)$$

Let  $g_n^b$  and  $g_n^d$  be the gradient of  $l_{DH}(\theta; \mathbf{S}_n + b \times d)$  with respect to  $b$  and  $d$  when  $b = 0$  and  $d = \text{sign}(\nabla_x l_{DH}(\theta; \mathbf{S}_n))$ , then

$$\text{budget}_n = \delta \cdot (g_n^b)^{q-1} \Upsilon^{1-q}, \quad \text{direction}_n = \text{sign}(g_n^d). \quad (4.7)$$

Furthermore,  $\Upsilon$  in (4.3) satisfies  $\Upsilon = (\frac{1}{N} \sum_{n=1}^N (g_n^b)^q)^{1/q}$ .

By applying the above updates iteratively with step size and projection, we propose a new PGD-like algorithm that optimizes the budget allocation and perturbation direction for each sample independently. The full procedure is summarized in Algorithm 2.

---

**Algorithm 2** Wasserstein Budget Projection Gradient Descent (WBPGD)

---

```
1: for  $i = 1$  to  $\text{num\_of\_iteration}$  do
2:   Compute  $g_n^b$  and  $g_n^d$  through back-propagation for  $n = 1, \dots, N$ 
3:   Compute  $\hat{\Upsilon} := (\frac{1}{N} \sum_{n=1}^N (g_n^b)^q)^{1/p}$ 
4:   for  $n = 1$  to  $N$  do
5:      $\text{budget}_n \leftarrow \text{budget}_n + \beta \cdot (g_n^b)^{q-1} \hat{\Upsilon}^{1-q}$ 
6:      $\text{direction}_n \leftarrow \text{direction}_n + \beta/\delta \cdot \text{sign}(g_n^d)$ 
7:     Clamp  $\text{direction}_n$  to  $[-1, 1]$ 
8:      $\hat{\mathbf{S}}_n \leftarrow \mathbf{S}_n + \text{budget}_n \times \text{direction}_n$ 
9:   end for
10:   $\hat{\mathbf{S}}_1, \dots, \hat{\mathbf{S}}_N \leftarrow \text{Proj}_{\hat{B}_\delta(\mu)}(\hat{\mathbf{S}}_1, \dots, \hat{\mathbf{S}}_N)$ 
11: end for
```

---

**Extension to Heston model.** We now consider how to extend the above algorithms to models with several series of data as input, such as the Heston model. In this case, the network strategy uses two input series: the price process  $\mathbf{S}$  and the variance process  $\mathbf{v}$ . During the attack phase, we perturb either the price process  $\mathbf{S}$  alone (referred to as S-Attack) or both the price and variance processes simultaneously (referred to as SV-Attack). For S-Attack, we can directly apply the above algorithms. For SV-attack, we first need to define, for a weight  $\lambda > 0$ , the distance on  $(\mathbf{S}, \mathbf{V})$  as

$$d((\mathbf{S}, \mathbf{V}), (\hat{\mathbf{S}}, \hat{\mathbf{v}})) = ((\max_t |S_t - \hat{S}_t|)^p + (\lambda \cdot \max_t |v_t - \hat{v}_t|)^p)^{1/p}. \quad (4.8)$$

The price and variance are weighted differently as they are on different scales. Under this distance, we have  $d((\mathbf{S}, \mathbf{v}), (\hat{\mathbf{S}}, \hat{\mathbf{v}}))^p = d(\mathbf{S}, \hat{\mathbf{S}})^p + \lambda^p d(\mathbf{v}, \hat{\mathbf{v}})^p$  and the following result by direct calculation.

**Corollary 4.2.** *Under Assumption 3.1 and Assumption 3.2, for input  $x = (\mathbf{S}, \mathbf{v})$  with distance defined as (4.8),  $\Upsilon$  becomes*

$$\Upsilon = (\sum_{n=1}^N \|\nabla_{\mathbf{S}} l(\theta; \mathbf{S}_n, \mathbf{v}_n)\|_1^q + \|1/\lambda \cdot \nabla_{\mathbf{v}} l(\theta; \mathbf{S}_n, \mathbf{v}_n)\|_1^q)^{1/q} \quad (4.9)$$

and the perturbation (3.9) can be written as

$$\mathbf{S} \mapsto \mathbf{S} + \text{sign}(\nabla_{\mathbf{S}} l(\theta; \mathbf{S}, \mathbf{v})) \|\nabla_{\mathbf{S}} l(\theta; \mathbf{S}, \mathbf{v})\|_*^{q-1} \Upsilon^{1-q} \quad (4.10)$$

$$\mathbf{v} \mapsto \mathbf{v} + 1/\lambda \cdot \text{sign}(\nabla_{\mathbf{v}} l(\theta; \mathbf{S}, \mathbf{v})) \|1/\lambda \cdot \nabla_{\mathbf{v}} l(\theta; \mathbf{S}, \mathbf{v})\|_*^{q-1} \Upsilon^{1-q}. \quad (4.11)$$

This corollary reveals that applying an adversarial perturbation jointly to the price and variance processes  $(\mathbf{S}, \mathbf{v})$  under the defined metric (4.8) is equivalent to independently perturbing the price series  $\mathbf{S}$  and the scaled variance series  $\lambda \mathbf{v}$  separately using the  $l_\infty$ -distance. Consequently, in practical implementation, we can conveniently treat  $\mathbf{S}$  and  $\lambda \mathbf{v}$  as separate input sequences, by transforming the original sample set  $\{\mathbf{S}_1, \dots, \mathbf{S}_N\}$  into  $\{\mathbf{S}_1, \lambda \mathbf{v}_1, \dots, \mathbf{S}_N, \lambda \mathbf{v}_N\}$ .

## 5 Experimental Results

### 5.1 Attacks on classical deep hedging strategies

We start with applying distributional adversarial attacks to neural network strategies trained in the classical deep hedging [2] setting, i.e., on the Heston model with CVaR loss function. As detailed in Section 4, both S-attack and SV-attack are implemented using the WPGD and WBPGD algorithms introduced in Algorithms 1 and 2. The resulting hedging loss across varying perturbation magnitudes  $\delta$  is summarized in Table 1. The base case ( $\delta = 0$ ) corresponds to the unperturbed hedging loss. Table 1 shows that the adversarial loss significantly increases as the attack magnitude ( $\delta$ ) increases. Additionally, the WBPGD method consistently outperforms the WPGD method, especially at larger perturbation magnitudes. Therefore, WBPGD will be used as the main attack method in subsequent experiments.

To understand practical implications and quantify the extent of distortion in the path space due to the attack, we assessed perturbation impacts through covariance matrix comparisons, as shown in Table 2. The Frobenius distance between the covariance matrices of perturbed and original paths,

given the original covariance matrix norm ( $\approx 386$ ), indicates minor covariance distortions, especially for small perturbations ( $\delta < 0.1$ ). Specifically, simultaneous perturbations on both price and variance processes (SV-Attack) result in less pronounced covariance changes compared to perturbations only on the price process (S-Attack). In the appendix we also report autocorrelation function comparisons.

Overall, together with with Table 1, these analyses show that adversarial attacks may significantly deteriorate the hedging strategy’s performance, even under seemingly modest perturbations, thereby underscoring the necessity for robust neural network models in financial applications.

Table 1: Robustness of classical deep hedging strategy under different attack methods and magnitudes

$\delta$	0	0.01	0.03	0.05	0.1	0.3	0.5
S-WBPGD	1.9280	1.9642	2.0432	2.1356	2.4466	4.5771	8.0745
SV-WBPGD	1.9280	1.9659	2.0485	2.1451	2.4660	4.5898	7.7391
S-WPGD	1.9280	1.9642	2.0431	2.1343	2.4369	4.5148	7.5411
SV-WPGD	1.9280	1.9652	2.0457	2.1378	2.4393	4.4802	7.4678

Table 2: Distance between covariance matrices of perturbed and original paths

$\delta$	0.01	0.03	0.05	0.1	0.3	0.5
S-Attack	0.1421	0.4038	0.6295	1.0248	2.3864	4.5057
SV-Attack	0.1355	0.3836	0.5929	0.9548	2.1787	3.8617

## 5.2 Adversarial training

We introduce adversarial training, which aims to enhance the robustness of strategies against distributional adversarial attacks and hence solve the DRO problem. We adopt the standard deep hedging methodology [2] as baseline and expand it to incorporate adversarial examples during training. Experimental details (e.g., network architecture, hyperparameters) are provided in the appendix.

**Loss functions.** In standard deep hedging, the network parameters  $\theta$  are optimized using the loss function defined by  $\mathcal{L}_{clean}(\theta) = \sum_{n=1}^N l_{DH}(\theta; X_n)$ . In adversarial training, we separate the optimization problem into two parts: the inner maximization problem and the outer minimization problem. During the inner maximization part, the network parameters  $\theta$  are fixed and we apply a distributionally adversarial attack to find the worst-case perturbation. With the adversarial perturbation  $\hat{X}_n$  obtained from the distributional adversarial attack, we can then minimize the expected loss function with respect to the network parameters  $\theta$  in the outer minimization part. Following [20, 22], our training uses an enhanced loss function

$$\mathcal{L}_{adv}(\theta) = \alpha \cdot \sum_{n=1}^N l_{DH}(\theta; X_n) + \sum_{n=1}^N l_{DH}(\theta; \hat{X}_n), \quad (5.1)$$

where  $\{\hat{X}_1, \dots, \hat{X}_n\}$  are adversarially perturbed versions of the original samples and  $\alpha$  balances the importance of clean versus adversarial samples. This leads to an iterative process of adversarial training, alternating between generating adversarial samples and optimizing the network parameters.

**Dataset.** We conduct our experiments using three well-established financial models: Black–Scholes, Heston, and the General Affine Diffusion (GAD) model. Here we present results for the Heston model in Section 5.3, while model details and results for the other models are provided in the appendix. Results for real market data are reported in Section 5.4 below. For each model, we generate extensive training datasets of 100,000 sample paths. To examine robustness across varying dataset sizes, we partition each dataset into smaller subsets with sizes  $N$  ranging from 5,000 to 100,000 samples. Neural networks are independently trained on these subsets, and the average performance is assessed and reported on a fixed test set, which contains 1 million paths, generated from the same distribution. In addition, we generate a validation set of 100,000 paths, but only  $N$  paths will be used for validation, so that the training is exposed to only a limited number of data depending on  $N$ .

**Robustness of adversarial training.** Strategies trained with our adversarial procedure achieve lower losses under distributional perturbations than classical deep hedging strategies, indicating that



the method effectively approximates the desired distributionally robust solution. Detailed experiments and results are provided in the appendix.

### 5.3 Adversarial training improves out-of-sample and out-of-distribution performance

**Out-of-sample performance.** Given robust strategies alongside the corresponding clean strategies, Figure 1a evaluates hedging performance on a test set of 1 million simulated paths approximating the true data-generating distribution. The midline presents the average performance of strategies trained on partitioned datasets of size  $N$ , while the shaded area shows the performance range. From the plot, adversarial training significantly outperforms conventional methods, especially when data is scarce: at  $N = 5,000$ , SV-Attack achieves a 54% lower mean hedging loss than clean training (2.86 vs 6.21) while reducing worst-case outcomes by 66% (max loss 3.60 vs 10.62). When the data size becomes larger, the empirical distribution becomes closer to the true underlying distribution, and all strategies approach near-identical performance ( $\sim 1.95$ ). However, the robust strategy still outperforms the clean one even at relatively smaller scales. In addition, though S-Attack and SV-Attack show comparable average performance, SV-Attack exhibits tighter variance across all  $N$ , indicating the advantage of allowing perturbation in the variance process.

**Out-of-distribution performance.** In prior experiments, test data were drawn from the same distribution as the training data. To further assess generalizability, we now evaluate the strategy on out-of-distribution (OOD) samples generated under perturbed parameter regimes. Specifically, we generate 100 new parameter configurations by scaling the original values by factors uniformly sampled from  $[0.9, 1.1]$ , introducing bounded deviations of  $\pm 10\%$ . For each perturbed configuration, 10,000 sample paths are simulated, resulting in a comprehensive OOD dataset of 1 million paths. Figure 1b illustrates the strategy’s performance on this OOD dataset. Despite the models never encountering these perturbed parameter regimes during training, the observed performance trends align closely with the out-of-sample results in Figure 1a, underscoring the robustness of the approach under parameter distribution shifts.

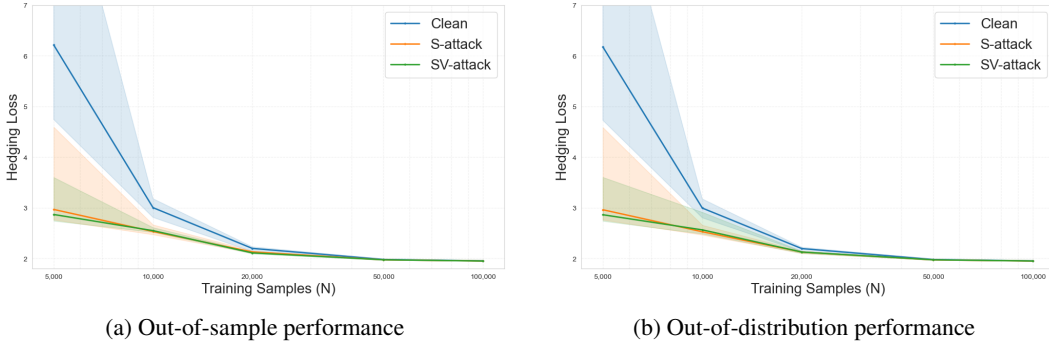


Figure 1: Comparative hedging performance under Heston models. Shaded regions indicate min-max ranges across training partitions.

### 5.4 Experiments on Market Data

In this section, we evaluate the performance of adversarial training on real market data. Specifically, we train hedging strategies using historical daily closing prices from leading companies in the S&P 500 index from [47], covering the period from 26 September 2008 to 8 March 2020. The time period is chosen in line with the benchmark method [11] with which we compare our method.

For this evaluation, we constructed two synthetic datasets based on an additional model introduced in the appendix. The **FIX** dataset is simulated using the General Affine Diffusion (GAD) model with fixed parameters estimated from a 250-day period prior to 8 March 2020. The **ROBUST** dataset, following [11], is also based on the GAD model but incorporates parameter robustness by sampling parameters uniformly from intervals determined by the extreme values across 26 rolling estimates. These estimates are obtained from 250-day windows updated every 100 days between 26 September 2008 and 8 March 2020. For each dataset (FIX and ROBUST), we generate 100,000 paths for training,

validation, and testing. The starting price is set as the closing price of the respective company on 8 March 2020, and each trajectory is scaled to begin from 10.

Directly training on the FIX and ROBUST datasets corresponds to the methods proposed in Deep Hedging [2] and Robust Deep Hedging [11], respectively. Building on this foundation, we implement the adversarial framework described in Section 5.2, resulting in two adversarial variants of the strategies. Similar to [11], we evaluate out-of-sample performance on real data by computing profit and loss (PnL) for hedging strategies on the tested price trajectories starting from 9 March 2020. Table 3 summarizes results for five companies.<sup>2</sup>, where bold entries indicate top-performing results per company. To address the limitations of using a single price trajectory, we refer the reader to the Appendix for details of the evaluation protocol and an additional evaluation result.

Several notable patterns emerge from this analysis. The clean strategy trained on the ROBUST dataset generally outperforms the clean strategy trained on the FIX dataset. This aligns with the conclusion of [11], which highlights the benefit of incorporating parameter uncertainty into the training data. However, our proposed adversarial training framework consistently achieves better performance overall. When trained on the FIX dataset, adversarial training yields notable improvements over clean training across all companies. In cases where clean training on the ROBUST dataset performs better than adversarial training on the FIX dataset, applying adversarial training to the ROBUST dataset further enhances performance. These findings underscore the effectiveness of the adversarial training framework we propose in addressing model misspecification and enhancing strategy robustness.

Table 3: Testing performance (P&L) on tested price trajectory

Method	AAPL	AMZN	BRK-B	GOOGL	MSFT
Clean training on FIX [2]	-2.261	-0.981	-2.086	-1.275	-4.341
Clean training on ROBUST [11]	-0.830	-0.849	-0.281	0.026	-0.223
Adversarial training on FIX	<b>-0.579</b>	<b>-0.291</b>	<b>-0.127</b>	-0.459	-0.564
Adversarial training on ROBUST	-0.739	-0.860	-0.294	<b>0.199</b>	<b>-0.144</b>

## 6 Conclusion

We presented a robust deep hedging framework that leverages adversarial training under Wasserstein ambiguity to address the challenges of model misspecification and distributional shifts. By formulating the distributionally robust optimization problem as a minimax objective and approximating it through a tractable reformulation, we demonstrated how deep hedging strategies can be trained adversarially. Empirical results across a variety of widely used synthetic models and data regimes show that adversarially trained strategies achieve improved out-of-sample and out-of-distribution performance, especially under structural changes or limited data availability. Further experiments on real market data suggest that these strategies also generalize well beyond simulated settings, maintaining robustness in periods of market stress. These findings underscore the practical value of robust training methods in financial environments characterized by uncertainty and instability.

**Limitations and future work.** Our adversarially trained deep hedging framework enhances robustness, but its performance remains sensitive to the choice of the Wasserstein radius, which may differ across models. Future work could explore potentially stronger distributional adversarial attack methods and explore extensions to alternative ambiguity set geometries beyond Wasserstein balls.

**Acknowledgment** GH’s research is supported by the Department of Mathematics at Imperial College London through the Roth Scholarship. We also thank G-Research for the travel support to attend NeurIPS.

<sup>2</sup>Including Apple (AAPL), Amazon (AMZN), Microsoft (MSFT), Google (GOOGL), and Berkshire Hathaway (BRK-B)

## References

- [1] Cboe Global Markets, Inc. Cboe Global Markets Reports Trading Volume for December and Full Year 2023. Technical report, Cboe Global Markets, Inc., 2024.
- [2] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.
- [3] Tobias Sutter, Andreas Krause, and Daniel Kuhn. Robust generalization despite distribution shift via minimum discriminating information. In *Advances in Neural Information Processing Systems*, 2021.
- [4] Zijian Liu, Qinxun Bai, Jose Blanchet, Perry Dong, Wei Xu, Zhengqing Zhou, and Zhengyuan Zhou. Distributionally robust  $q$ -learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13623–13643. PMLR, July 2022. ICML 2022, 17–23 July.
- [5] Qi Feng and J. George Shanthikumar. The framework of parametric and nonparametric operational data analytics. *Production and Operations Management*, 32(9):2685–2703, 2023.
- [6] Liwan H. Liyanage and J. George Shanthikumar. A practical inventory control policy using operational statistics. *Operations Research Letters*, 33(4):341–348, 2005.
- [7] Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-wasserstein gans for time series generation. *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–8, 11 2021.
- [8] Tianlin Xu, Li Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.
- [9] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.
- [10] David Wu and Sebastian Jaimungal. Robust risk-aware option hedging. *Applied Mathematical Finance*, 30(3):153–174, 2023.
- [11] Eva Lütkebohmert, Thorsten Schmidt, and Julian Sester. Robust deep hedging. *Quantitative Finance*, 22(8):1465–1480, 2022.
- [12] Yannick Limmer and Blanka Horvath. Robust hedging gans: Towards automated robustification of hedging strategies. *Applied Mathematical Finance*, pages 1–38, 2024.
- [13] Daniel Kuhn, Soroosh Shafiee, and Wolfram Wiesemann. Distributionally Robust Optimization. *arXiv preprint, arXiv:2411.02549*, 2024.
- [14] Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.
- [15] Eric Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, 2010.
- [16] Hamed Rahimian, Güzin Bayraksan, and Tito Homem-De-Mello. Identifying effective scenarios in distributionally robust stochastic programs with total variation distance. *Mathematical Programming*, 173(1–2):393–430, 2019.
- [17] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1–2):115–166, 2018.
- [18] Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. *Wasserstein Distributionally Robust Optimization: Theory and Applications in Machine Learning*, chapter 6, pages 130–166. 2019.

- [19] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
- [20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint*, arXiv:1412.6572, 2014.
- [21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.
- [22] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7472–7482. PMLR, May 2019.
- [23] Xingjian Bai, Guangyi He, Yifan Jiang, and Jan Obloj. Wasserstein distributional robustness of neural networks. *Advances in Neural Information Processing Systems*, 36:26322–26347, 2023.
- [24] Xingjian Bai, Guangyi He, Yifan Jiang, and Jan Obloj. Wasserstein distributional adversarial training for deep neural networks. *arXiv preprint arXiv:2502.09352*, 2025.
- [25] Junhao Dong, Xinghua Qu, Z Jane Wang, and Yew-Soon Ong. Enhancing adversarial robustness via uncertainty-aware distributional adversarial training. *arXiv preprint arXiv:2411.02871*, 2024.
- [26] Matthew Staib and Stefanie Jegelka. Distributionally robust deep learning as a generalization of adversarial training. In *NIPS workshop on Machine Learning and Computer Security*, volume 3, page 4, 2017.
- [27] Rui Gao and Anton Kleywegt. Distributionally robust stochastic optimization with wasserstein distance. *Mathematics of Operations Research*, 48(2):603–655, 2023.
- [28] Hans Buehler, Lukas Gonon, Josef Teichmann, Ben Wood, Baranidharan Mohan, and Jonathan Kochems. Deep hedging: hedging derivatives under generic market frictions using reinforcement learning. *Swiss Finance Institute Research Paper*, (19-80), 2019.
- [29] Alexandre Carbonneau and Frédéric Godin. Equal risk pricing of derivatives with deep hedging. *Quantitative Finance*, 21(4):593–608, 2021.
- [30] Alexandre Carbonneau. Deep hedging of long-term financial derivatives. *Insurance: Mathematics and Economics*, 99:327–340, 2021.
- [31] Francesca Biagini, Lukas Gonon, and Thomas Reitsam. Neural network approximation for superhedging prices. *Mathematical Finance*, 33(1):146–184, 2023.
- [32] Shota Imaki, Kentaro Imajo, Katsuya Ito, Kentaro Minami, and Kei Nakagawa. No-transaction band network: A neural network architecture for efficient deep hedging. *arXiv preprint arXiv:2103.01775*, 2021.
- [33] Konrad Mueller, Amira Akkari, Lukas Gonon, and Ben Wood. Fast deep hedging with second-order optimization. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF ’24, page 319–327, New York, NY, USA, 2024. Association for Computing Machinery.
- [34] Phillip Murray, Ben Wood, Hans Buehler, Magnus Wiese, and Mikko S. Pakkanen. Deep Hedging: Continuous Reinforcement Learning for Hedging of General Portfolios across Multiple Risk Aversions, July 2022. *arXiv:2207.07467*.
- [35] Petter N Kolm and Gordon Ritter. Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, 1(1):159–171, 2019.
- [36] Jiayi Du, Muyang Jin, Petter N Kolm, Gordon Ritter, Yixuan Wang, and Bofei Zhang. Deep reinforcement learning for option replication and hedging. *The Journal of Financial Data Science*, 2(4):44–57, 2020.
- [37] Jay Cao, Jacky Chen, John Hull, and Zissis Poulos. Deep hedging of derivatives using reinforcement learning. *arXiv preprint arXiv:2103.16409*, 2021.

- [38] Edoardo Vittori, Michele Trapletti, and Marcello Restelli. Option hedging with risk averse reinforcement learning. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–8, 2020.
- [39] Johannes Ruf and Weiguan Wang. Hedging with linear regressions and neural networks. *Journal of Business & Economic Statistics*, 40(4):1442–1454, 2022.
- [40] Oskari Mikkilä and Juho Kanniainen. Empirical deep hedging. *Quantitative Finance*, 23(1):111–122, 2023.
- [41] Snehal Raj, Iordanis Kerenidis, Abhishek Shekhar, Ben Wood, Jon Dee, Shouvanik Chakrabarti, Richard Chen, Dylan Herman, Shaohan Hu, Pierre Minssen, et al. Quantum deep hedging. *Quantum*, 7:1191, 2023.
- [42] Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3):437–503, 2023.
- [43] Johannes Ruf and Weiguan Wang. Neural networks for option pricing and hedging: a literature review. *J. Comput. Finance*, 24(1):1–46, 2020.
- [44] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*. Walter de Gruyter, 4th rev. ed. edition, 2016.
- [45] Aharon Ben-Tal and Marc Teboulle. An old-new concept of convex risk measures: The optimized certainty equivalent. *Mathematical Finance*, 17(3):449–476, 2007.
- [46] Daniel Bartl, Samuel Drapeau, Jan Obłój, and Johannes Wiesel. Sensitivity analysis of wasserstein distributionally robust optimization problems. *Proceedings of the Royal Society A*, 477(2256):20210176, 2021.
- [47] Ran Aroussi. yfinance: Yahoo! finance market data downloader. <https://github.com/ranaroussi/yfinance>, 2019.
- [48] Alexander J McNeil, Rüdiger Frey, and Paul Embrechts. *Quantitative risk management: concepts, techniques and tools-revised edition*. Princeton university press, 2015.

## NeurIPS Paper Checklist

### (i) Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and Section 1 accurately reflect our paper's contributions and scope on adversarial attack and training in deep hedging framework.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### (ii) Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discuss the limitations of the work and future expectations at the end of the paper.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### (iii) Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Complete proofs of all theoretical results are provided in the Appendix C, with assumptions clearly stated.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

(iv) **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We disclose all the information needed to reproduce all the results of the paper in Section 5.2 and Appendix D. The codes to reproduce the results are provided in supplemental material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

(v) **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide codes in the supplemental material with clear instructions to reproduce all experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

(vi) **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We describe the experimental setup in detail in Section 5.2 and Appendix D, with all necessary information provided.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

(vii) **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the out-of-sample and out-of-distribution performance in Fig 1 containing min-max ranges across training partitions.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.



- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

(viii) **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide the details of computer resources we use to train the neural network in Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

(ix) **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our work does not violate NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

(x) **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: In Section 1, we emphasize that potential issues of model misspecification, which may result in suboptimal or even misleading decisions. This highlights the practical importance of designing strategies that remain robust to small changes in the underlying distribution.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

(xi) **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper does not pose a risk of potential misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

(xii) **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: Our paper does not use any existing assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

(xiii) **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: Our paper does not release any new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

(xiv) **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

(xv) **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

(xvi) **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our paper does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

## Appendix

### A Wasserstein Distance

For a Wasserstein distributionally robust optimization problem (1.1), the ambiguity set  $B_\delta(\mu)$  is defined as a Wasserstein ball centered at  $\mu \in \mathcal{P}(\mathbb{R}^d)$ , i.e.,

$$B_\delta(\mu) = \{\eta \in \mathcal{P}(\mathbb{R}^d) : W_p(\mu, \eta) \leq \delta\}. \quad (\text{A.1})$$

For any  $p \in (1, \infty)$ , the Wasserstein distance  $W_p(\mu, \eta)$  between two distributions  $\mu$  and  $\eta$  is defined as

$$W_p(\mu, \eta) = \left( \inf_{\gamma \in \Pi(\mu, \eta)} \int_{\mathbb{R}^d \times \mathbb{R}^d} d(x, y)^p d\gamma(x, y) \right)^{1/p}, \quad (\text{A.2})$$

where  $\Pi(\mu, \eta)$  denotes the set of all joint distributions on  $\mathbb{R}^d \times \mathbb{R}^d$  with marginals  $\mu$  and  $\eta$  and  $d$  denotes a metric on  $\mathbb{R}^d$ .

For  $p = \infty$ , the Wasserstein distance becomes the minimal maximal displacement between two distributions

$$W_\infty(\mu, \eta) = \inf_{\gamma \in \Pi(\mu, \eta)} \{\gamma\text{-ess sup } d(x, y)\}. \quad (\text{A.3})$$

Here,  $\gamma\text{-ess sup}$  denotes the essential supremum with respect to the measure  $\gamma$  over  $\mathbb{R}^d \times \mathbb{R}^d$ .

### B Deep Hedging Examples

**Case 1: Black–Scholes Model with Entropic Risk Measure.** In this scenario, we assume that the asset price follows the classical Black–Scholes model

$$dS_t = mS_t dt + \sigma S_t dW_t, \quad (\text{B.1})$$

where  $m$  is the drift,  $\sigma$  is the volatility, and  $W_t$  is a standard Brownian motion. The process is discretized in time for training the neural network.

Here, as the process is Markovian, we define the information process directly as the price process  $S_t$ , which provides all the necessary information to make decisions at time  $t$ . The network strategy in (2.1) is simplified to

$$\delta_t = f_{\theta_t}(S_t). \quad (\text{B.2})$$

We consider hedging a European call option with terminal payoff

$$P(S_T) = \max(S_T - K, 0), \quad (\text{B.3})$$

where  $K$  is the strike price. To account for risk aversion in the objective function, we adopt the entropic risk measure

$$\rho(Z) = \frac{1}{\lambda} \log \mathbb{E} [e^{-\lambda Z}], \quad (\text{B.4})$$

where  $\lambda > 0$  is the risk aversion parameter. This risk measure is commonly used in finance to model the risk preferences of investors [44].

By [2, Example 3.8], the the entropic risk measure admits the OCE form

$$\rho(Z) = \inf_{\omega \in \mathbb{R}} \left\{ \omega + \mathbb{E} \left[ \exp(-\lambda(Z + \omega)) - \frac{1 + \log \lambda}{\lambda} \right] \right\}. \quad (\text{B.5})$$

Moreover, the corresponding optimal  $\omega$  in (B.5) is given by

$$\omega^* = \frac{1}{\lambda} \log \mathbb{E} [\lambda \cdot \exp(-\lambda Z)]. \quad (\text{B.6})$$

The corresponding deep hedging loss is then defined as

$$l_{DH}(\theta, \omega, \mathbf{S}) = \omega - \frac{1 + \log \lambda}{\lambda} + \exp(-\lambda(\text{PnL}(\theta, \mathbf{S}) + \omega)). \quad (\text{B.7})$$

**Case 2: Heston Model with Conditional Value-at-Risk (CVaR).** In this scenario, we assume that the asset price follows the Heston stochastic volatility model:

$$dS_t^1 = mS_t^1 dt + \sqrt{v_t}S_t^1 dW_t^1, \quad dv_t = a(b - v_t) dt + \sigma\sqrt{v_t} dW_t^2, \quad (\text{B.8})$$

where  $v_t$  is the stochastic variance process, and the Brownian motions  $W_t^1$  and  $W_t^2$  satisfy

$$\mathbb{E}[dW_t^1 dW_t^2] = \rho dt. \quad (\text{B.9})$$

The parameters  $a$ ,  $b$ , and  $\sigma$  control the mean reversion speed, long-run variance level, and volatility of volatility, respectively.

As  $v_t$  itself is not directly tradable, to hedge the volatility risk, we introduce a second price process representing a variance swap corresponding to the tradable asset.

The variance swap at time  $t$  is given by

$$S_t^2 = \int_0^t v_s ds + \frac{v_t - b}{a}(1 - e^{-a(T-t)}) + b(T-t), \quad (\text{B.10})$$

where the integral is approximated by the trapezium rule in practice.

We then hedge through trading both the underlying asset and the variance swap, i.e., we define the combined price process as  $S_t = (S_t^1, S_t^2)$ . Moreover, since the network needs both the price of the underlying and variance to make decisions, the information process is defined as  $I_t = (S_t^1, v_t)$ . The Heston model is Markovian with respect to this information process, so the network strategy becomes:

$$\delta_t = f_{\theta_t}(S_t^1, v_t). \quad (\text{B.11})$$

We again consider hedging a European call option with the same terminal payoff as in (B.3).

To evaluate hedging performance under downside risk, we adopt the Conditional Value-at-Risk (CVaR) risk measure at confidence level  $\alpha \in [0, 1]$

$$\text{CVaR}_\alpha(Z) = \frac{1}{1-\alpha} \int_0^{1-\alpha} \text{VaR}_\gamma(Z) d\gamma \quad (\text{B.12})$$

$$\text{VaR}_\gamma(Z) = \inf \{z \in \mathbb{R} : \mathbb{P}(Z \leq -z) < \gamma\}. \quad (\text{B.13})$$

This risk measure captures the expected loss in the worst  $1 - \alpha$  fraction of outcomes and is widely used in risk management [48]. The CVaR can be written in OCE form

$$\text{CVaR}_\alpha(Z) = \inf_{\omega \in \mathbb{R}} \left\{ \omega + \frac{1}{1-\alpha} \mathbb{E}[\max(-Z - \omega, 0)] \right\}, \quad (\text{B.14})$$

where the optimal  $\omega$  is attained at the  $\alpha$ -quantile of  $Z$ .

We then define the corresponding deep hedging loss as

$$l_{DH}(\theta, \omega, \mathbf{S}^1, \mathbf{v}) = \omega + \frac{1}{1-\alpha} \max(-\text{PnL}(\theta, \mathbf{S}^1, \mathbf{v}) - \omega, 0). \quad (\text{B.15})$$

**Case 3: General Affine Diffusion (GAD) Model with Entropic Risk Measure.** In this scenario, we assume the asset price follows a General Affine Diffusion (GAD) process:

$$dS_t = (b_0 + b_1 S_t) dt + (a_0 + a_1 S_t)^\gamma dW_t, \quad (\text{B.16})$$

where  $W_t$  is a standard Brownian motion and the parameters  $b_0$ ,  $b_1$ ,  $a_0$ ,  $a_1$ , and  $\gamma$  control the drift and diffusion characteristics.

To discretize the process for numerical implementation, we apply the Euler–Maruyama scheme. Following the robust approach of [11], we introduce parameter uncertainty through intervals, where at each path and each time step, parameters are drawn uniformly from their respective intervals:

$$\begin{aligned} & \text{for } t = 1, \dots, T : \\ & \Delta W_t \sim \mathcal{N}(0, \Delta t), \\ & a_0 \sim U[\underline{a}_0, \bar{a}_0], \quad a_1 \sim U[\underline{a}_1, \bar{a}_1], \quad b_0 \sim U[\underline{b}_0, \bar{b}_0], \quad b_1 \sim U[\underline{b}_1, \bar{b}_1], \\ & S_t = S_{t-1} + (b_0 + b_1 S_{t-1}) \Delta t + (a_0 + a_1 S_{t-1})^\gamma \Delta W_t. \end{aligned} \quad (\text{B.17})$$

By taking intervals as a point value, the above process approximates the classical GAD process.

As in the Black–Scholes model, the network strategy is defined using only the current asset price:

$$\delta_t = f_{\theta_t}(S_t). \quad (\text{B.18})$$

Following the same setup as in [11], we consider hedging an Asian at-the-money put option with the terminal payoff

$$P(\mathbf{S}) = \max \left( S_0 - \frac{1}{T} \sum_{t=1}^T S_t, 0 \right). \quad (\text{B.19})$$

We use the same entropic risk measure as in the Black-Scholes model (see Case 1 above).

## C Proofs

*Proof of Theorem 3.3.* Our proof follows the approach of [23, Theorem 4.1], adapted to the classification setting. Specifically, we build on [46, Theorem 2.1] and its proof, which we restate below as a theorem.

**Theorem C.1** (Adapted from [46, 23]). *Under Assumption 3.1 and Assumption 3.2, the following statements hold.*

(i) *The first-order sensitivity expansion as  $\delta \downarrow 0$  ensures*

$$V(\delta) = V(0) + \delta \Upsilon + o(\delta), \quad \text{where} \quad \Upsilon := \mathbb{E}_{x \sim \mu} [\|\nabla_x l(\theta; x)\|_*^q]^{1/q} \quad (\text{C.1})$$

*and  $q$  is the conjugate exponent of  $p$ , satisfying  $1/q + 1/p = 1$ .*

(ii) *Furthermore,  $V(\delta)$  can be approximated by*

$$V(\delta) = \mathbb{E}_{\eta_\delta} [l(\theta, x)] + o(\delta) \quad \text{as} \quad \delta \downarrow 0 \quad (\text{C.2})$$

*where the perturbed distribution  $\eta_\delta$  is explicitly given by*

$$\eta_\delta = [x \mapsto x + \delta \cdot h(\nabla_x l(\theta; x)) \|\nabla_x l(\theta; x)\|_*^{q-1} \Upsilon^{1-q}]_{\#} \mu. \quad (\text{C.3})$$

In the data-driven framework we choose  $\mu = \frac{1}{N} \sum_{n=1}^N \delta_{X_n}$ . Then  $\Upsilon$  in (C.1) becomes the average

$$\Upsilon = \left( \frac{1}{N} \sum_{n=1}^N \|\nabla_x l(\theta; X_n)\|_*^q \right)^{1/q} \quad (\text{C.4})$$

and the perturbation  $\eta_\delta$  in (C.3) becomes a uniform distribution on the perturbed dataset  $\{\hat{X}_1, \dots, \hat{X}_N\}$ , where each  $\hat{X}_n$  satisfies

$$\hat{X}_n = X_n + \delta \cdot h(\nabla_x l(\theta; X_n)) \|\nabla_x l(\theta; X_n)\|_*^{q-1} \Upsilon^{1-q}. \quad (\text{C.5})$$

□

*Proof of Lemma 3.4.* By Theorem 3.3,  $\eta_\delta$  is of the form  $\frac{1}{N} \sum_{n=1}^N \delta_{\hat{X}_n}$  and satisfy

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N d(X_i, \hat{X}_i)^p &= \frac{1}{N} \sum_{i=1}^N \|\delta \cdot h(\nabla_x l(\theta; X_n)) \|\nabla_x l(\theta; X_n)\|_*^{q-1} \Upsilon^{1-q}\|^p \\ &= \frac{\delta^p}{\Upsilon^{(q-1)p}} \cdot \frac{1}{N} \sum_{i=1}^N \|h(\nabla_x l(\theta; X_n)) \|\nabla_x l(\theta; X_n)\|_*^{(q-1)p}\| \\ &= \frac{\delta^p}{\Upsilon^q} \cdot \frac{1}{N} \sum_{i=1}^N \|\nabla_x l(\theta; X_n)\|_*^q \\ &= \delta^p \end{aligned} \quad (\text{C.6})$$

where we use  $\|h(x)\| = \sup_{\|x\|_* \leq 1} \langle h(x), x \rangle = \sup_{\|x\|_* \leq 1} \|x\|_* = 1$ ,  $(q-1)p = (1-1/q)pq = q$  for exponent conjugate and (C.4). Therefore, we have  $\eta_\delta \in \hat{B}_\delta(\mu)$ .

Moreover, for  $\mu = \frac{1}{N} \sum_{n=1}^N \delta_{X_n}$  and  $\hat{\mu} = \frac{1}{N} \sum_{n=1}^N \delta_{\hat{X}_n}$ , we have

$$\pi = \frac{1}{N} \sum_{n=1}^N \delta_{(X_n, \hat{X}_n)} \in \Pi(\mu, \hat{\mu}). \quad (\text{C.7})$$

Therefore, by the definition of the Wasserstein distance,

$$W_p(\mu, \hat{\mu}) \leq \left( \int d(x, y)^p d\pi(x, y) \right)^{1/p} = \left( \frac{1}{N} \sum_{i=1}^N d(X_i, \hat{X}_i)^p \right)^{1/p}. \quad (\text{C.8})$$

If  $(\frac{1}{N} \sum_{i=1}^N d(X_i, \hat{X}_i)^p)^{1/p} < \delta$ , so is  $W_p(\mu, \hat{\mu})$ , hence  $\hat{B}_\delta(\mu) \subset B_\delta(\mu)$ .

Overall, we proved that  $\eta_\delta \in \hat{B}_\delta(\mu) \subset B_\delta(\mu)$ . Therefore,  $\mathbb{E}_{\eta_\delta}[l(\theta; x)] \leq V_\theta^e(\delta) \leq V_\theta(\delta)$  and

$$0 \leq \frac{1}{\delta} (V_\theta(\delta) - V_\theta^e(\delta)) \leq \frac{1}{\delta} (V_\theta(\delta) - \mathbb{E}_{\eta_\delta}[l(\theta; x)]). \quad (\text{C.9})$$

By Theorem 3.3, RHS  $\rightarrow 0$  as  $\delta \rightarrow 0$ , so  $\frac{1}{\delta} (V_\theta(\delta) - V_\theta^e(\delta))$  also converges to 0. In other words,

$$V_\theta(\delta) = V_\theta^e(\delta) + o(\delta) \quad \text{as } \delta \downarrow 0. \quad (\text{C.10})$$

□

*Proof of Lemma 4.1.* For  $g_n^S = \nabla_S l(\theta; \mathbf{S}_n)$ , the updates (4.3) can be separated into updates on  $\text{budget}_n$  and  $\text{direction}_n$

$$\Upsilon = \left( \frac{1}{N} \sum_{n=1}^N \|g_n^S\|_* \right)^{1/q}, \quad \text{budget}_n = \|g_n^S\|_*^{q-1} \Upsilon^{1-q}, \quad \text{direction}_n = \text{sign}(g_n^S). \quad (\text{C.11})$$

By the chain rule, we have  $g_n^b = \langle g_n^S, \text{direction}_n \rangle = \|g_n^S\|_*$  and  $g_n^d = g_n^S / \text{budget}_n$ . The update above becomes

$$\Upsilon = \left( \frac{1}{N} \sum_{n=1}^N (g_n^b)^q \right)^{1/q}, \quad \text{budget}_n = (g_n^b)^{q-1} \Upsilon^{1-q}, \quad \text{direction}_n = \text{sign}(g_n^d), \quad (\text{C.12})$$

which proves the lemma. □

Corollary 4.2 is a special case of the following more general result, which we will prove next.

**Corollary C.2.** Consider the setting of Theorem 3.3 with inputs of the form  $x = (x^1, \dots, x^d) \in \mathbb{R}^{d \times (T+1)}$  representing  $d$ -dimensional sequences of length  $T+1$ . Set the norm as

$$\|x\| := \left( \sum_{i=1}^d (\lambda_i \|x^i\|_\infty)^p \right)^{1/p}, \quad (\text{C.13})$$

where  $\|\cdot\|_\infty$  is the infinity norm defined in the space of the trajectories  $\mathbb{R}^{T+1}$ . Over the input samples  $\{X_1, \dots, X_N\}$ , where each sample contains  $d$  trajectories  $X_n = (X_n^1, \dots, X_n^d)$ , we define  $g_n = (g_n^1, \dots, g_n^d) = \nabla_x l(\theta; (X_n^1, \dots, X_n^d))$  to be the gradient with respect to the input. In this setting, the perturbation (3.9) can be written as

$$\hat{X}_n^i = X_n^i + \frac{1}{\lambda_i} \text{sign}(g_n^i) \left\| \frac{1}{\lambda_i} g_n^i \right\|_1^{q-1} \Upsilon^{1-q} \quad (\text{C.14})$$

for  $i = 1, \dots, d$  and  $n = 1, \dots, N$ . Moreover,  $\Upsilon$  becomes

$$\Upsilon = \left( \sum_{n=1}^N \sum_{i=1}^d \left\| \frac{1}{\lambda_i} g_n^i \right\|_1^q \right)^{1/q} \quad (\text{C.15})$$

where  $\|\cdot\|_1$  is the  $l_1$ -norm defined on  $\mathbb{R}^{T+1}$ .

*Proof.* We first show that the norm in (C.13) has dual norm defined as

$$\|y\|_* = \left( \sum_{i=1}^d \left( \frac{1}{\lambda_i} \|y^i\|_1 \right)^q \right)^{1/q}. \quad (\text{C.16})$$



With the standard pairing  $\langle x, y \rangle = \sum_i \langle x^i, y^i \rangle$ , we estimate

$$|\langle x, y \rangle| \leq \sum_{i=1}^d \|x^i\|_\infty \|y^i\|_1 \leq \left( \sum_{i=1}^d (\lambda_i \|x^i\|_\infty)^p \right)^{1/p} \left( \sum_{i=1}^d \left( \frac{1}{\lambda_i} \|y^i\|_1 \right)^q \right)^{1/q} = \|x\| \|y\|_*, \quad (\text{C.17})$$

where the inequalities hold by Hölder's inequality.

By setting  $\lambda_i \|x^i\|_\infty \propto \frac{1}{\lambda_i} \|y^i\|_1$  and  $x^i = \|x^i\|_\infty \text{sign}(y^i)$  for each  $i$ , for any  $y$  there exists  $x$  such that  $\|x\| = 1$  and  $|\langle x, y \rangle| = \|x\| \|y\|_*$ . Hence (C.16) indeed defines the dual norm by definition. The corresponding function  $h(y)$  such that  $\langle h(y), y \rangle = y$  is defined as

$$h(y) = \frac{1}{\|y\|_*^{q-1}} \left( \frac{1}{\lambda_1} \text{sign}(y^1) \left\| \frac{1}{\lambda_1} y^1 \right\|_1^{q-1}, \dots, \frac{1}{\lambda_d} \text{sign}(y^d) \left\| \frac{1}{\lambda_d} y^d \right\|_1^{q-1} \right). \quad (\text{C.18})$$

Recall that the perturbation in (3.9) is

$$\hat{X}_n = X_n + \delta \cdot h(g_n) \|g_n\|_*^{q-1} \Upsilon^{1-q}. \quad (\text{C.19})$$

By (C.18),  $h(g_n)$  becomes,

$$h(g_n) = \frac{1}{\|g_n\|_*^{q-1}} \left( \frac{1}{\lambda_1} \text{sign}(g_n^1) \left\| \frac{1}{\lambda_1} g_n^1 \right\|_1^{q-1}, \dots, \frac{1}{\lambda_d} \text{sign}(g_n^d) \left\| \frac{1}{\lambda_d} g_n^d \right\|_1^{q-1} \right). \quad (\text{C.20})$$

Therefore, bringing (C.20) into (C.19), each trajectory in each sample  $X_n$  is perturbed to

$$\begin{aligned} \hat{X}_n^i &= X_n^i + \delta \cdot \left( \frac{1}{\|g_n\|_*^{q-1}} \frac{1}{\lambda_i} \text{sign}(g_n^i) \left\| \frac{1}{\lambda_i} g_n^i \right\|_1^{q-1} \right) \cdot \|g_n\|_*^{q-1} \Upsilon^{1-q} \\ &= X_n^i + \frac{\delta}{\lambda_i} \text{sign}(g_n^i) \left\| \frac{1}{\lambda_i} g_n^i \right\|_1^{q-1} \Upsilon^{1-q} \end{aligned} \quad (\text{C.21})$$

for  $i = 1, \dots, d$  and  $n = 1, \dots, N$ . Finally,

$$\Upsilon = \left( \sum_{n=1}^N \|g_n\|_*^q \right)^{1/q} = \left( \sum_{n=1}^N \sum_{i=1}^d \left\| \frac{1}{\lambda_i} g_n^i \right\|_1^q \right)^{1/q}. \quad (\text{C.22})$$

□

## D Experimental Details

Here we provide additional experimental details regarding the adversarial training introduced in Section 5.2. Readers can refer to the code for a comprehensive implementation.

**Network Architecture.** The neural network architecture remains consistent with the standard deep hedging framework [2], characterized by decision-making at each time step  $t$  through:

$$\delta_t = f_t^{\theta_t}(I_t), \quad (\text{D.1})$$

where  $I_t$  encapsulates all relevant information available at step  $t$ . In line with [2], each  $f_t^{\theta_t}$  comprises two hidden layers, each with 20 neurons, batch normalization, and ReLU activation.

**Training Procedure.** Our training procedure begins with a preliminary phase of clean training to establish stable initial parameters. Specifically, this phase lasts 100 epochs for the BS model and 300 epochs for the more complex Heston model. Subsequently, the network undergoes adversarial training for an additional 200 epochs (BS) or 400 epochs (Heston), alternating adversarial example generation and optimization of Eq. (5.1). For comparison, we train baseline networks (clean strategies) exclusively with clean training for an equivalent total duration (300 epochs for BS, 700 epochs for Heston).

**Optimizer and Learning rate.** Optimization utilizes the Adam optimizer, with decaying learning rate—initially set to 0.005 for BS and 0.05 for Heston. The batch size is set to 10,000 unless the dataset size  $N$  is smaller, in which case the entire dataset is utilized per batch.

**Hyperparameters.** Critical adversarial training hyperparameters include  $\alpha$ , tested at 0, 1, 10 to gauge the relative influence of clean versus adversarial loss, and perturbation magnitude  $\delta$ , explored across  $\{0.001, 0.003, 0.005, 0.01, 0.03, 0.05, 0.1, 0.3, 0.5\}$ . Hyperparameter selection is performed by evaluating performance on a validation set of size  $N$  and selecting the hyperparameters yielding the best validation results.

**Adversarial attack.** During the experiment, we employ the WBPGD algorithm detailed in Algorithm 2 for adversarial attacks. We execute this algorithm for 20 iterations, setting the step-size as  $\beta = \frac{4}{20}\delta$ , which is dependent on the perturbation magnitude  $\delta$ . Additionally, for the two models considered, input trajectories have identical initial values across all samples. Consequently, we avoid perturbing the initial values by explicitly setting both the perturbation and corresponding gradient components to zero.

**Computation time.** All computational runs are conducted without GPU on AMD EPYC 7742 or Intel Icelake Xeon Platinum 8358 processors equipped with less than 64GB of memory. For standard adversarial training involving 100,000 sample paths, the computation time is approximately 3 hours for the Black-Scholes model and around 10 hours for the Heston model. In contrast, classical deep hedging without adversarial training requires roughly one-tenth of this computational effort. The increased computational demand for adversarial training is reasonable, as each network update includes an additional 20 iterations of adversarial perturbations, enhancing the network’s robustness.

**Cash-invariant property of convex risk measure.** By the cash-invariance property [44] of the convex risk measure  $\rho$ , we have  $\rho(Z + c) = \rho(Z) - c$  for any random variable  $Z$  and constant  $c$  (representing cash injection). Therefore, in practice, we directly set  $p_0$  in (2.2) to 0 as it does not affect the optimization problem. Note that we are only interested in hedging here; for pricing the an appropriate choice of  $p_0$  could be determined as described in [2].

## E Additional Experimental Results

In this section, we provide supplementary experimental results to further validate and contextualize the analyses presented in Section 5.

### E.1 Autocorrelation function comparison

Building upon the analysis presented in Section 5.1, we further examine the impact of adversarial perturbations by comparing the autocorrelation functions (ACFs) of the adversarially perturbed trajectories against those of the original trajectories. For a path  $\{x_t\}$ , the ACF is defined as:

$$\text{ACF}(x, \text{lag}) = \frac{1}{\sigma^2} \sum_{i=0}^{\text{lag}} \frac{1}{N-i} \sum_{t=1}^{N-i} (x_t - \bar{x})(x_{t+i} - \bar{x}), \quad (\text{E.1})$$

where  $\bar{x}$  is the empirical mean of the path  $x$ , and  $\sigma^2$  is its empirical variance.

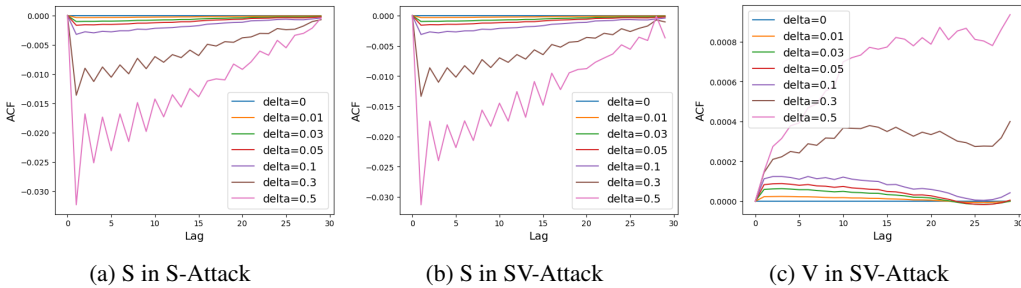


Figure 2: Difference in Autocorrelation function (ACF) of perturbed paths for different  $\delta$  values and original paths.

Fig. 2 illustrates the difference in ACF of perturbed paths and original paths. It reveals minimal deviations ( $< 0.005$ ) in autocorrelation for perturbations with magnitude  $\delta < 0.1$ . Despite these

modest ACF differences, the corresponding hedging errors are notably large (see Table 1), thereby undermining further that an adversarially perturbed sample path distribution may lead to significant hedging losses, even though comparisons of ACF and covariance matrices suggest that the perturbed distribution is close to the original distribution.

## E.2 Robustness of adversarial training

For the robust strategy trained using adversarial training detailed in Section 5.2, we examine the adversarial loss to confirm training effectiveness. We evaluate robust strategies trained under distributional adversarial attacks with perturbation magnitudes  $\delta = 0.01$  and  $0.1$ , comparing their performance against a clean strategy trained on an identical dataset of 100,000 samples. Table 4 illustrates the adversarial loss for each strategy under varying perturbation levels. The robust strategies consistently yield significantly lower adversarial losses compared to the clean strategy, clearly demonstrating that our robust training framework successfully provides a computationally tractable approximation of the theoretically optimal distributionally robust optimization (DRO) solution, enhancing model resilience against adversarial perturbations. Additionally, there is an expected trade-off observed in the loss: for the robust strategy trained with  $\delta = 0.1$ , the loss at no or small perturbation levels is higher, but the adversarial loss at larger perturbation magnitudes is substantially reduced, aligning with the designed training objective.

Table 4: Comparison of robust and clean strategies under different perturbation magnitudes

$\delta$	0	0.01	0.03	0.05	0.1	0.3	0.5
clean	1.9162	1.9598	2.0530	2.1560	2.4659	4.6069	8.0379
robust ( $\delta = 0.01$ )	1.9186	1.9541	2.0298	2.1130	2.3675	4.1626	7.1445
robust ( $\delta = 0.1$ )	1.9796	2.0035	2.0534	2.1064	2.2549	3.1161	4.3156

## E.3 Results on adversarial training

**Optimal hyperparameter choice.** As detailed in Section 3 and Appendix D, our robust training framework introduces two critical hyperparameters: the attack radius  $\delta$  controlling perturbation magnitude, and the balance weight  $\alpha$  modulating between nominal and adversarial losses. Through grid search across  $(\delta, \alpha) \in [0.001, 0.5] \times \{0, 1, 10\}$ , we identify optimal configurations that maximize validation performance for each training set size  $N$ . The optimal hyperparameters for the Heston model are presented in Table 5a where we can observe that the optimal  $\delta$  decreases from 0.5 at  $N = 5,000$  to 0.005 at  $N = 100,000$ . This phenomenon arises because smaller training sizes induce greater divergence between the empirical distribution  $\mu_N$  and the true data-generating distribution  $\mu$ , thus larger adversarial perturbations ( $\delta$ ) are required to bridge this distributional gap. The optimal parameters for the Black-Scholes model show a similar pattern, see Table 5b.

**Detailed Heston results.** Table 6 provides detailed out-of-sample performance results, supplementing the information shown in Figure 1a. Specifically, the table shows improvements in robust strategy performance as the sample size becomes large and demonstrates that the SV-Attack strategy exhibits lower variance compared to the S-Attack strategy.

**Black-Scholes results.** Figure 3 and Table 7 show comprehensive results for the Black-Scholes model, revealing patterns analogous to the Heston model. However, the gap between robust and clean strategies is relatively smaller than in Heston. This aligns with expectations - the simpler Black-Scholes model offers fewer exploitable gaps for adversarial training to mitigate, particularly in volatility dynamics.

Table 5: Hyperparameter Choices for Heston and Black-Scholes Models

(a) Heston Model Hyperparameters					(b) Black-Scholes Model Hyperparameters		
Training Samples (N)	S-Attack		SV-Attack		Training Samples (N)	$\delta$	$\alpha$
	$\delta$	$\alpha$	$\delta$	$\alpha$			
5,000	0.3	1	0.5	1	5,000	0.01	10
10,000	0.1	10	1.0	10	10,000	0.005	10
20,000	0.05	1	0.1	1	20,000	0.003	10
50,000	0.03	0	0.03	0	50,000	0.001	1
100,000	0.01	0	0.005	0	100,000	0.001	0

Table 6: Detailed out-of-sample performance across sample sizes (N) for SV-Attack, S-Attack, and Clean strategies on Heston model

Strategy	N	Avg Loss	Min Loss	Max Loss	Variance
SV-Attack	5,000	2.8644	2.7386	3.5975	0.0346
SV-Attack	10,000	2.5460	2.5173	2.6087	0.0008
SV-Attack	20,000	2.1063	2.0928	2.1282	0.0002
SV-Attack	50,000	1.9706	1.9669	1.9742	2.6e-5
SV-Attack	100,000	1.9469	1.9469	1.9469	–
S-Attack	5,000	2.9646	2.7605	4.5912	0.1574
S-Attack	10,000	2.5287	2.4694	2.6629	0.0028
S-Attack	20,000	2.1259	2.1027	2.1489	0.0004
S-Attack	50,000	1.9705	1.9665	1.9745	3.2e-5
S-Attack	100,000	1.9472	1.9472	1.9472	–
Clean	5,000	6.2095	4.7379	10.6187	2.0887
Clean	10,000	3.0000	2.8068	3.1755	0.0129
Clean	20,000	2.1955	2.1329	2.2266	0.0014
Clean	50,000	1.9773	1.9705	1.9841	9.2e-5
Clean	100,000	1.9503	1.9503	1.9503	–

Table 7: Detailed BS model performance metrics across sample sizes (N) for robust and Clean strategies

Strategy	N	Avg Loss	Min Loss	Max Loss
Robust	5,000	2.4109	2.4040	2.4195
Robust	10,000	2.3947	2.3920	2.3976
Robust	20,000	2.3855	2.3852	2.3861
Robust	50,000	2.3798	2.3794	2.3802
Robust	100,000	2.3769	2.3769	2.3769
Clean	5,000	2.4136	2.4055	2.4208
Clean	10,000	2.3976	2.3947	2.3993
Clean	20,000	2.3860	2.3854	2.3866
Clean	50,000	2.3798	2.3794	2.3801
Clean	100,000	2.3772	2.3772	2.3772

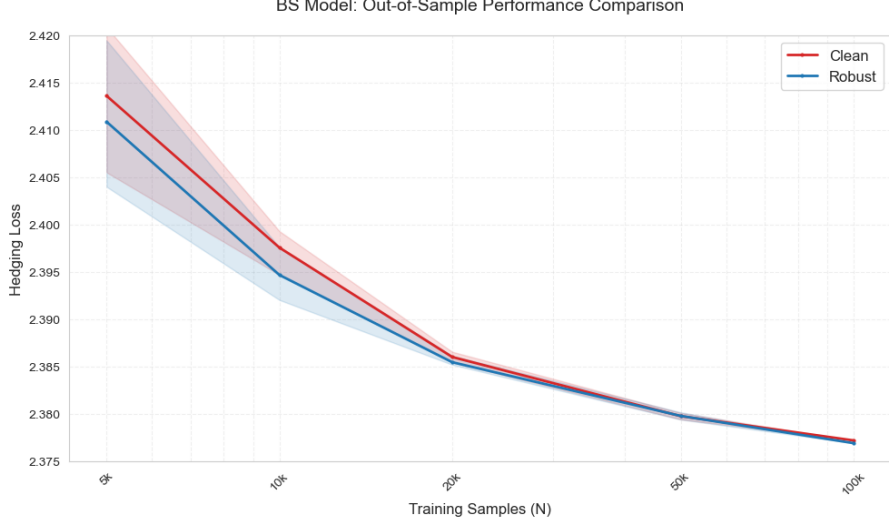


Figure 3: Out-of-sample hedging performance comparison under Black-Scholes dynamics, comparing robust training with clean strategies. Shaded regions indicate min-max ranges across training partitions.

#### E.4 Results on Heston models with transaction costs

In this section, we provide an additional example demonstrating the effectiveness of adversarial training by considering the Heston model with transaction costs. Following Equation (2.2), we modify the profit and loss (P&L) by incorporating a transaction cost term, which is proportional to the value of the assets traded:

$$\text{PnL}(\theta, \mathbf{I}) = p_0 + \sum_{t=1}^T \left( \delta_t^\top(\theta, \mathbf{I})(S_{t+1}(\mathbf{I}) - S_t(\mathbf{I})) + \epsilon S_t^\top(\mathbf{I})|\delta_t(\theta, \mathbf{I}) - \delta_{t-1}(\theta, \mathbf{I})| \right) - P(S_T(\mathbf{I}))$$

where  $\epsilon$  denotes the transaction cost rate, which we set to 0.005 in this experiment.

Apart from adding an additional transaction cost term to the loss function, the experimental setup follows the same procedure described in Sections 5.2 and Appendix D. To account for transaction costs, we also consider a recurrent neural network architecture, in which the portfolio holding at each time step is passed as input to the next. In this setting, the update rule in (B.11) becomes

$$\delta_t = f_{\theta_t}(S_t^1, v_t, \delta_{t-1}). \quad (\text{E.2})$$

We refer to the original feedforward network as NetSim, and the recurrent version as NetRec.

As in Figure 1, we report the out-of-sample performance of both adversarial and clean training in Figure 4 and Figure 5 on NetSim and NetRec, respectively. The results show that the adversarially trained strategy consistently outperforms the clean strategy. However, this behavior differs from the case without transaction costs: instead of plateauing as the sample size increases, the hedging loss decreases more rapidly. We also observe that although NetRec is more complex, its performance does not outperform NetSim. Both findings may indicate that even with 100,000 samples, the data does not fully capture the underlying distribution in the more complex setting. Under such conditions, adversarial training demonstrates clear advantages over clean training. These findings further highlight the potential of adversarial methods to enhance robustness in data-limited or distributionally uncertain environments.

#### E.5 Results on real market data

In Section 5.4, we evaluated performance on a single real price trajectory, following the approach of [11]. However, relying on a single path leads to the absence of validation data and exposure to training randomness. Hence, we adopt the following evaluation protocol. For adversarial training, we use  $N = 100,000$  training samples and report the average out-of-sample performance across six

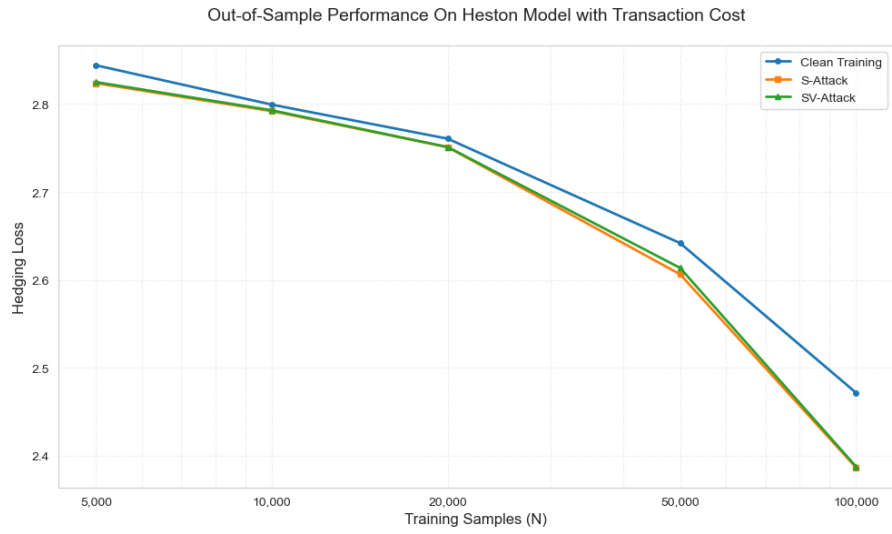


Figure 4: Out-of-sample hedging performance comparison under Heston model with transaction cost on NetSim, comparing among clean strategies and robust strategies under S-attack and SV-attack.



Figure 5: Out-of-sample hedging performance comparison under Heston model with transaction cost on NetRec, comparing among clean strategies and robust strategies under S-attack and SV-attack.

strategies corresponding to  $(\alpha, \delta) \in \{0, 1\} \times \{0.03, 0.05, 0.1\}$ . For clean training, we average the results over three independent runs with identical settings.

For a more thorough assessment of the model performance, we construct a more comprehensive REAL dataset by applying a rolling window of length 30 over the period from 7 March 2020 to 30 September 2021. Each path is normalized to start at a price of 10, yielding a total of 300 sample trajectories. We adopt the entropic risk measure to evaluate strategy performance, thereby avoiding the need to compute option prices—one of the key motivations for using this loss function. Performance results are summarized in Table 8, where lower values indicate better outcomes, and bold entries highlight the best performance in each column. We observe that the adversarial strategy trained on the FIX dataset consistently outperforms all others across all stocks. In contrast, clean training on the ROBUST dataset does not offer notable improvements over FIX dataset, and applying adversarial training to the ROBUST dataset further degrades performance. These findings suggest that explicitly incorporating robust parameter intervals into the data generation process may be unnecessary—or even counterproductive. Since adversarial training already mitigates the impact of model misspecification, the use of parameter intervals estimated from historical data that may no longer reflect current market conditions can introduce outdated constraints, ultimately diminishing the relevance and effectiveness of the strategy.

Table 8: Testing performance (entropic loss on REAL dataset)

Method	AAPL	AMZN	BRK-B	GOOGL	MSFT
Clean training on FIX [2]	0.2920	0.2971	0.1993	0.2608	0.3269
Clean training on ROBUST [11]	0.3053	0.3005	0.2410	0.3797	0.2692
Adversarial training on FIX	<b>0.2517</b>	<b>0.2408</b>	<b>0.1692</b>	<b>0.2401</b>	<b>0.1891</b>
Adversarial training on ROBUST	0.3077	0.3148	0.2437	0.3965	0.2776

Finally, we complement that analysis by reporting the performance on the FIX and ROBUST datasets in Tables 9 and 10, respectively. Similar to Black-Scholes and Heston model, neural networks are independently trained on subsets of size  $N$ , and we compute the average performance on the test set. For each adversarial training result, the hyperparameters, including the attack radius  $\delta$  and the balance weight  $\alpha$ , are selected using a validation set of size  $N$ .

When models are trained and evaluated on the FIX dataset, we observe trends consistent with those seen in the Black-Scholes and Heston experiments: adversarial training consistently outperforms clean training, with the advantage especially pronounced when the sample size is small, suggesting particular effectiveness in data-scarce regimes. The ROBUST dataset, which generalizes the FIX dataset by incorporating model misspecification, further highlights the benefits of adversarial training—strategies trained adversarially on FIX significantly outperform their clean-trained counterparts, demonstrating enhanced robustness to distributional shifts. When models are trained directly on the ROBUST dataset, the pattern becomes more nuanced: performance does not always improve with larger sample sizes, yet adversarial training still generally provides a performance edge. Together, these results underscore the practical value of adversarial training in improving generalization under complex and uncertain market conditions.

## F Extension to other models

In Section 3, we introduced distributional adversarial attack algorithms specifically for the Black-Scholes model, with input  $\mathbf{I} = \mathbf{S}$ , and the Heston model, with input  $\mathbf{I} = (\mathbf{S}^1, \mathbf{v})$ . In this section, we generalize these approaches to an arbitrary model characterized by input trajectories  $\mathbf{I} = (\mathbf{I}^1, \dots, \mathbf{I}^d)$ .

We begin by defining a distance measure on the general input space

$$d(\mathbf{I}, \hat{\mathbf{I}}) = \left( \sum_{i=1}^d (\lambda_i \cdot \max_t |I_t^i - \hat{I}_t^i|)^p \right)^{1/p}, \quad (\text{F.1})$$

analogous to the distance measure introduced for the Heston model in (4.8). This definition allows distinct perturbation scales for each trajectory, consistent with the Heston model framework.

Table 9: Performance comparison on FIX dataset

Company	Method	N=5000	10000	20000	50000	100000
<i>Trained on FIX dataset</i>						
AAPL	Adversarial	0.1737	0.1752	0.1706	0.1667	0.1643
	Clean	0.1960	0.1943	0.1782	0.1734	0.1745
	Improvement	<b>0.0223</b>	<b>0.0191</b>	<b>0.0076</b>	<b>0.0067</b>	<b>0.0101</b>
AMZN	Adversarial	0.1927	0.1785	0.1763	0.1753	0.1737
	Clean	0.2401	0.1970	0.1867	0.1782	0.1770
	Improvement	<b>0.0474</b>	<b>0.0185</b>	<b>0.0104</b>	<b>0.0029</b>	<b>0.0033</b>
BRK-B	Adversarial	0.1412	0.1368	0.1353	0.1353	0.1352
	Clean	0.1867	0.1649	0.1523	0.1408	0.1363
	Improvement	<b>0.0455</b>	<b>0.0281</b>	<b>0.0169</b>	<b>0.0055</b>	<b>0.0011</b>
GOOGL	Adversarial	0.2258	0.2025	0.1985	0.1981	0.1976
	Clean	0.2955	0.2224	0.2065	0.2005	0.1984
	Improvement	<b>0.0696</b>	<b>0.0199</b>	<b>0.0080</b>	<b>0.0023</b>	<b>0.0008</b>
MSFT	Adversarial	0.1549	0.1491	0.1431	0.1436	0.1418
	Clean	0.1878	0.1789	0.1551	0.1479	0.1473
	Improvement	<b>0.0329</b>	<b>0.0298</b>	<b>0.0120</b>	<b>0.0043</b>	<b>0.0055</b>

Utilizing Corollary C.2, a direct generalization of Corollary 4.2, the update rule for each scaled trajectory is given by:

$$\lambda_i \hat{\mathbf{I}}_n^i = \lambda_i \mathbf{I}_n^i + \delta \cdot \text{sign}\left(\frac{1}{\lambda_i} g_n^i\right) \left\| \frac{1}{\lambda_i} g_n^i \right\|_1^{q-1} \Upsilon^{1-q}, \quad (\text{F.2})$$

which closely parallels the update step for the complete sample set

$$\hat{\mathbf{I}}_n = \mathbf{I}_n + \delta \cdot h(g_n) \|g_n\|_*^{q-1} \Upsilon^{1-q}. \quad (\text{F.3})$$

Furthermore, the total distance  $\sum_{n=1}^N d(\mathbf{I}_n, \hat{\mathbf{I}}_n)^p$  and the term  $\Upsilon^p$  used during the iterative update process can naturally be decomposed into sums across both trajectory dimensions ( $i = 1, \dots, d$ ) and individual samples ( $n = 1, \dots, N$ ).

Consequently, similar to the approach in the Heston model, each trajectory will be independently perturbed according to an  $l_\infty$ -norm metric, effectively transforming the original set of samples into a structured set of scaled trajectories  $\{\lambda_i \mathbf{I}_n^i\}_{n=1, \dots, N}^{i=1, \dots, d}$ .



Table 10: Performance comparison on ROBUST dataset

Company	Method	N=5000	10000	20000	50000	100000
<i>Trained on FIX dataset</i>						
AAPL	Adversarial	0.4346	0.5023	0.4748	0.4404	0.4490
	Clean	0.4387	0.5354	0.5679	0.5328	0.4831
	Improvement	<b>0.0041</b>	<b>0.0331</b>	<b>0.0931</b>	<b>0.0924</b>	<b>0.0341</b>
AMZN	Adversarial	0.4983	0.5342	0.5875	0.5958	0.5994
	Clean	0.5859	0.6509	0.6677	0.6743	0.6737
	Improvement	<b>0.0876</b>	<b>0.1167</b>	<b>0.0802</b>	<b>0.0785</b>	<b>0.0743</b>
BRK-B	Adversarial	0.5548	0.5330	0.5416	0.4872	0.5029
	Clean	0.6408	0.6733	0.6876	0.6441	0.6351
	Improvement	<b>0.0860</b>	<b>0.1403</b>	<b>0.1459</b>	<b>0.1569</b>	<b>0.1322</b>
GOOGL	Adversarial	0.5276	0.6236	0.6563	0.6054	0.6055
	Clean	0.5378	0.6707	0.6814	0.6775	0.6844
	Improvement	<b>0.0102</b>	<b>0.0471</b>	<b>0.0252</b>	<b>0.0721</b>	<b>0.0788</b>
MSFT	Adversarial	0.5859	0.6357	0.5716	0.5505	0.5678
	Clean	0.6358	0.6870	0.6993	0.6847	0.7124
	Improvement	<b>0.0499</b>	<b>0.0513</b>	<b>0.1276</b>	<b>0.1342</b>	<b>0.1446</b>
<i>Trained on ROBUST dataset</i>						
AAPL	Adversarial	0.3936	0.3607	0.3463	0.3703	0.4325
	Clean	0.4229	0.3705	0.3699	0.3875	0.4510
	Improvement	<b>0.0293</b>	<b>0.0098</b>	<b>0.0237</b>	<b>0.0172</b>	<b>0.0185</b>
AMZN	Adversarial	0.4276	0.3525	0.2701	0.2401	0.2470
	Clean	0.4505	0.3627	0.2633	0.2647	0.3065
	Improvement	<b>0.0229</b>	<b>0.0102</b>	-0.0068	<b>0.0246</b>	<b>0.0595</b>
BRK-B	Adversarial	0.3702	0.2849	0.1912	0.2123	0.2467
	Clean	0.4117	0.3618	0.2265	0.1800	0.3076
	Improvement	<b>0.0415</b>	<b>0.0769</b>	<b>0.0354</b>	-0.0323	<b>0.0609</b>
GOOGL	Adversarial	0.3748	0.2584	0.1434	0.0985	0.1427
	Clean	0.4222	0.3351	0.2100	0.1276	0.2207
	Improvement	<b>0.0474</b>	<b>0.0766</b>	<b>0.0667</b>	<b>0.0291</b>	<b>0.0780</b>
MSFT	Adversarial	0.3885	0.2800	0.1945	0.2170	0.2399
	Clean	0.4115	0.3473	0.2535	0.2674	0.2944
	Improvement	<b>0.0230</b>	<b>0.0673</b>	<b>0.0590</b>	<b>0.0504</b>	<b>0.0544</b>