



UNIVERSITÀ CA' FOSCARI VENEZIA
Department of Economics

Master's Degree Programme in
Quantitative Finance and Risk Management

Ambiguity-Averse Deep Hedging of Options under Controlled Distributional Shifts

Supervisor

Prof. Antonella Basso

Graduand

Davide Devetak

Academic Year

2025 / 2026

Acknowledgements

I would like to thank Prof. Dr. Josef Teichmann from ETH Zürich for the generosity with which he discussed this work with me, and for the insight that shaped several of its ideas.

Abstract

Standard deep hedging optimises a risk criterion under a single reference probability measure. Ambiguity-averse extensions try to account for model uncertainty by clustering training paths into market regimes and hedging against the worst of them. But how well this works should depend on the distance between the training distribution and the distribution the hedger is actually tested on. However, that distance is generally not explicit.

We build a simple, controllable way to measure it: a family of mixtures between a Heston and a GARCH generator, parametrised by a single mixture weight, that interpolates continuously between a baseline training regime and a more realistic one. This lets us move the test distribution away from the training distribution by a known amount and watch what happens.

Across this whole range, and on ten years of real index data, the standard deep hedger consistently beats the ambiguity-averse one on both risk and variance, though the ambiguity-averse hedger is reliably cheaper for what concerns transaction costs. This is different from what the paper we build on reports. Tracing the discrepancy, in direct correspondence with its authors, led to a sign error in their volatility model's correlation parameter, which quietly changes what kind of market stress the hedger is trained to survive. We discuss what this means, how the original results should be read, and end with some preliminary findings, left for future work.

Contents

1	Introduction	3
2	Literature Review and Contribution	6
2.1	Literature Review	6
2.1.1	Deep Hedging	6
2.1.2	Model Uncertainty and Robustification	7
2.2	Main Contributions	8
3	Neural Networks	10
3.1	Feedforward Neural Networks	10
3.2	Universal Approximation Theorems	13
3.3	Training a Neural Network	15
3.3.1	Loss Function	15
3.3.2	Mini-Batch Stochastic Gradient Descent	16
3.3.3	Backpropagation	17
4	Standard Deep Hedging	19
4.1	Market Setup	19
4.2	Hedging Strategies	20
4.2.1	Classical Delta Hedging	20
4.2.2	Strategies for Deep Hedging	22
4.3	Deep Hedging	23
4.3.1	From the Ideal Hedging Problem to the Objective Functional	23
4.3.2	Parametrization of the Trading Strategy	24
4.4	Deep Hedging Algorithm	25
5	Ambiguity-Averse Deep Hedging	28
5.1	Ambiguity Aversion	28
5.2	Ambiguity Aversion in Deep Hedging	29

5.2.1	Feature Clustering	30
5.2.2	K-Means Clustering	31
5.3	The Ambiguity-Averse Objective	34
5.4	Ambiguity-Averse Deep Hedgers in Practice	37
5.5	The Ambiguity-Averse Deep Hedging Algorithm	38
6	Data Generation and Distributional Shifts	41
6.1	Synthetic Distributional Shifts	41
6.1.1	The Two Generators	42
6.1.2	The Two Extremal Measures	43
6.1.3	The Mixture Family	43
6.2	Data Generation	46
6.3	All Together: The Distributional-Shift Experiment	47
7	Implementation of the Experiment and Results	49
7.1	Experiment Setup	49
7.1.1	Data Configuration	49
7.1.2	Training Configuration	51
7.1.3	Deep-Hedging Network Architecture	52
7.2	Results	53
7.2.1	In Sample Results	53
7.2.2	Controlled Distributional Shifts Results	59
7.3	Some Considerations on the Results	65
8	Conclusion	69
A	Repository Structure and Code	75

Chapter 1

Introduction

The problem of hedging a derivative lies at the heart of mathematical finance. In Black and Scholes, 1973, a European option written on a single underlying can be replicated exactly by continuous trading in that underlying, provided the market is frictionless and the dynamics of the asset are known. Both assumptions are untenable in practice: trading is discrete and costly, and the true dynamics of the market are never known. The first of these difficulties makes perfect replication impossible and forces the hedger to adopt a risk-minimisation criterion. The second, which is the heart of this thesis, means that any such criterion must be formulated with respect to a probability measure that is itself uncertain, and the sensitivity of hedging to this second issue is well understood even in the classical case (Karoui et al., 1998). Data-driven methods inherit this same problem.

Over the last decade the response to the first difficulty has been *deep hedging* (Buehler et al., 2019). Rather than solving a partial differential equation for a fixed, tractable model, one trains a neural network directly on simulated paths to minimise a risk measure of the terminal profit and loss. This lets the network absorb whatever frictions, constraints, or path dependence we want to include. However, this flexibility comes at a price: it is optimal, by construction, only for the probability measure used to generate its training data, and that measure is itself a modelling choice, produced by a classical stochastic model or a deep market generator that is periodically recalibrated to the market. Whenever the measure realised once the strategy is deployed differs from the one it was trained on, a *distributional shift*, nothing in the training procedure guarantees that the learned strategy remains sensible.

This is a model uncertainty problem, and the literature has approached it accordingly, by drawing on two related traditions. One is robust and distributionally robust optimisation, where a hedging objective is optimised against the worst measure in

some neighbourhood of the training one. Such approaches are effective but tend to be overly conservative, sometimes forcing the hedger to charge a substantially higher premium to compensate. The other is the decision-theoretic literature on ambiguity aversion, which formalises the older observation of Ellsberg, 1961 that a decision maker who does not fully trust any single model should not evaluate a strategy under one measure alone. The *smooth ambiguity* model of Klibanoff et al., 2005 takes a more moderate stance than the worst-case criterion: it treats a handful of candidate models as unequally trusted and combines the risk computed under each of them through a nonlinear aggregation rule, recovering the fully worst-case criterion only as a limiting case.

The Ambiguity-Averse Deep Hedger (AADH) of Jones et al., 2025 brings this idea into deep hedging in a simple way: rather than requiring an explicit set of candidate models, it clusters the paths already used to train the hedger, according to pathwise features believed relevant to the trading strategy, and treats each cluster as a candidate regime. An inner risk measure is computed within each cluster and an outer ambiguity measure combines them, all without generating any additional data or charging a higher premium. The original paper reports that this construction outperforms the standard deep hedger on a broad range of out-of-distribution test sets, from equity indices to the COVID-19 period, and it is this claim, and this framework, that the present thesis takes as its starting point.

What all of these out-of-distribution evaluations have in common, however, is that they compare a synthetically trained hedger against a handful of fixed, externally given test sets, each at an unknown and uncontrolled distance from the measure used for training. This makes it impossible to separate two different questions: does ambiguity aversion help at all, and, if so, how much of a distributional shift is actually needed before it starts to? Answering the second question requires a way to move continuously, and by a known amount, from the training distribution to a genuinely different one, which is exactly what is missing from the existing empirical evidence.

This thesis builds that missing piece: a one-parameter family of probability measures on path space, obtained by mixing a Heston and a GARCH(1,1) generator, that interpolates continuously between a baseline training regime and a regime that more closely resembles the volatility clustering and autocorrelation of real equity indices. We use this family to retrace, in a controlled way, the empirical comparison between the standard and the ambiguity-averse deep hedger, and we complement it with the same comparison on ten years of real index data. The result we obtain differs significantly to Jones et al., 2025: across the whole range of distributional shifts we consider, and on every equity index tested, the standard deep hedger consistently attains a lower risk and a lower variance than its ambiguity-averse counterpart, al-

though the latter remains reliably cheaper in transaction costs. We traced the source of this discrepancy, in direct correspondence with the authors of the original paper. This uncovered a specific error in how their training data had been generated: a sign error in the correlation parameter of their Heston generator, which quietly changes the kind of market stress the hedger is trained on. This finding does not undermine the case for ambiguity aversion in deep hedging so much as it relocates it, and we discuss at length what it implies for how the original results should be read.

The rest of the thesis is organized in the following way. In Chapter 2 we propose a brief literature review on the topic and we state the contributions of this thesis. In Chapter 3 we introduce feedforward neural networks, the universal approximation results of Hornik, 1991, and the elements of training relevant to our framework: these are the loss function, mini-batch stochastic gradient descent (Kingma & Ba, 2014) and backpropagation (Rumelhart et al., 1986). Chapter 4 sets up the discrete-time market with frictions, recalls classical delta hedging as a benchmark, and presents the standard deep hedging problem and algorithm of Buehler et al., 2019. Chapter 5 develops ambiguity aversion, from the maxmin criterion to the smooth ambiguity model, and derives the ambiguity-averse objective based on feature clustering, together with its two limiting cases. Chapter 6 introduces the two generators, the mixture family and its properties, and formalises the distributional-shift experiment. Chapter 7 describes the implementation and reports the results, both in sample and under controlled shifts, and discusses the origin of the difference with respect to the original paper. Chapter 7 concludes and indicates directions for future work. Finally, Appendix A describes the structure of the code we developed and links to the public Github repository.¹

¹<https://github.com/DavideDevetak24/Ambiguity-Averse-Deep-Hedging-of-Options-Under-Controlled-Distributional-Shifts>

Chapter 2

Literature Review and Contribution

2.1 Literature Review

2.1.1 Deep Hedging

The use of neural networks for pricing and hedging has expanded rapidly over the last decade, supported by the general developments in deep learning (Goodfellow et al., 2016; LeCun et al., 2015). The term *deep hedging* was introduced by Buehler et al., 2019, who parametrise the trading strategy by a neural network and optimise a convex risk measure of the terminal profit and loss directly, by stochastic gradient descent. The approach is essentially a policy gradient method applied to a hedging objective, and its main strength is that it accommodates realistic trading environments, including transaction costs, position limits and market impact, which are out of reach for analytic model-based approaches. Related formulations of discrete-time hedging in incomplete markets by deep learning are given in Fecamp et al., 2020, while Mikkilä and Kanninen, 2023 study the behaviour of deep hedgers trained directly on empirical data.

A parallel and closely related literature approaches the same problem through reinforcement learning, in the tradition of Sutton et al., 1998. A survey of the use of these methods in economics and finance is given in Charpentier et al., 2023. Early contributions include the Q-learning formulations of Halperin, 2017 and Halperin, 2019 and the dynamic replication approach of Kolm and Ritter, 2019, followed by Giurca and Borovkova, 2021, Cao et al., 2021, Cannelli et al., 2023, and by Cao et al., 2023, who extend the framework to gamma and vega hedging. The deep

hedging formulation adopted in this thesis, however, follows primarily the framework of Buehler et al., 2019.

The strategies produced by all these methods are only as good as the data used to train them. Since a single historical trajectory is available for any given underlying, the training set is in practice generated synthetically, either by a classical model or by a deep market generator. This makes the resulting hedger model-dependent in a way that is easy to overlook: it is optimal for the measure induced by the generator, and a priori for that measure only. The limitations of generative models of financial markets are discussed in Cetingoz and Lehalle, 2026, and the difficulty of estimating the parameters of a market model at all is the subject of Duembgen and Rogers, 2014. A further difficulty specific to this setting is documented by Horikawa and Nakagawa, 2024, who show that a deep hedger trained to minimise a convex risk measure will tend to learn a statistical arbitrage strategy alongside the hedge, exploiting any residual drift in the training measure.

The choice of generator also determines which stylised facts (Cont, 2001) the hedger is exposed to. Stochastic volatility models such as Heston, 1993 capture the leverage effect and volatility clustering only partially, and discrete-time conditional heteroskedasticity models in the tradition of Engle, 1982 and Bollerslev, 1986 capture persistence in volatility but not its continuous-time structure. More recently, the evidence for rough volatility reported in Livieri et al., 2018 has motivated hedging approaches under such dynamics, both in a classical setting (Fukasawa et al., 2021) and in the deep hedging setting (Horvath et al., 2021). Each of these choices induces a different measure on path space, and hence a different hedger.

2.1.2 Model Uncertainty and Robustification

The mismatch between the training measure and the measure realised during live hedging, which we refer to throughout as a distributional shift, is the motivation for the robustification of deep hedging. The available approaches largely instantiate the categories of robust and distributionally robust optimisation (DRO) reviewed in Rahimian and Mehrotra, 2022, where the objective is optimised against the worst case over an ambiguity set of measures, for instance a Wasserstein ball around the training measure (Bartl et al., 2021). In the hedging context, Lütkebohmert et al., 2022 randomise the parameters of the market generator during training, Limmer and Horvath, 2024 train an adversarial market generator alongside the hedger, and Wu and Jaimungal, 2023 formulate a robust risk-aware hedging objective. These methods are effective, but they inherit the drawbacks of the worst-case criterion: they can require charging a substantially higher option premium to compensate for

an overly conservative hedge, or they add a further layer of complexity to a strategy that is already criticised for its black-box character.

The decision-theoretic counterpart of this discussion begins with Ellsberg, 1961, whose observations undermine the Savage axioms underlying the expected utility representation of preferences whenever the decision maker does not know the randomness to which they are subject. The maxmin expected utility representation of Gilboa and Schmeidler, 1989 rationalises this behaviour by evaluating a position under the worst measure in a set of priors, and is therefore the decision-theoretic image of worst-case robustification. Klibanoff et al., 2005 generalise it: their smooth ambiguity model separates an inner utility, which captures risk within a given model, from an outer utility, which captures ambiguity across models, and recovers maxmin expected utility as the limiting case of infinite ambiguity aversion.

Jones et al., 2025 builds from this. Their Ambiguity-Averse Deep Hedger (AADH) partitions the existing training data into clusters using pathwise features relevant for the hedging strategy. Then, they treat each cluster as a candidate model, and evaluates an inner risk measure per cluster together with an outer ambiguity measure across clusters. The construction requires no additional data, no auxiliary or adversarial generator, and no increase in the premium charged. According to the authors, this outperforms the standard deep hedger on a broad range of out-of-distribution test sets. It is this framework that the present thesis takes as its starting point.

2.2 Main Contributions

The results reported in Jones et al., 2025 are obtained on fixed and externally given test sets. Each of these lies at an unknown and uncontrolled distance from the training measure, which leaves open the more basic question of when ambiguity aversion begins to pay: as the test distribution is moved continuously away from the training distribution, at what point, and how quickly, does the advantage of the AADH appear?

The main contribution of this thesis is a framework designed to answer that question. Specifically:

1. We give a self-contained mathematical treatment of the ambiguity-averse objective, extending the analysis of Jones et al., 2025 and highlighting the departure of Proposition 5.3.2 from the practical implementation.
2. We formalize mathematically the concept of distributional shift. Here all the mathematics should be seen as an original contribution on the topic (Definition

6.1.1, Lemma 6.1.1 and Proposition 6.1.1). More specifically, we construct a one-parameter family of probability measures on path space, obtained as mixtures of a Heston and a GARCH(1,1) generator. We show that this family is a rigorous proxy for the severity of the distributional shift, and we derive a closed-form decomposition of the variance under the mixture.

3. We evaluate both hedgers over a grid of training and test mixture weights, finding that the standard deep hedger attains a lower risk and a lower variance than the ambiguity-averse hedger at every point of this surface, while the latter is consistently cheaper in transaction costs. We also show that the theoretical mixture family reproduces both the risk levels observed on real indices and the difference between the two hedgers.
4. Finally, we document the origin of the divergence with the results of Jones et al., 2025, through a direct exchange with the authors. An error was identified in the generation of their training paths, which effectively reversed the sign of the Heston correlation parameter. We discuss the consequences of this and formulate a conjecture, supported by some preliminary analysis, pointing to possible future work.

Chapter 3

Neural Networks

In this chapter, we present the fundamental mathematical objects we will work with throughout the entire paper: Neural Networks. We define the formal mathematical structure, the properties, and the way in which they work.

3.1 Feedforward Neural Networks

We start by defining the feedforward neural network, one of the most fundamental architectures of a neural network.

Definition 3.1.1 (Feedforward Neural Network). Let $L \in \mathbb{N}$, let $d_0, d_1, \dots, d_L \in \mathbb{N}$, and let $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ be an activation function (for a more detailed explanation, see Definition 3.1.2 below). A feedforward neural network (FNN) of depth L and architecture $(d_0, d_1, \dots, d_L)$ is a function $f_\theta: \mathbb{R}^{d_0} \rightarrow \mathbb{R}^{d_L}$ defined recursively by

$$\begin{aligned} h^{(0)} &:= x, \\ h^{(\ell)} &:= \varphi(W_\ell h^{(\ell-1)} + b_\ell), \quad \ell = 1, \dots, L-1, \end{aligned} \tag{3.1}$$

$$f_\theta(x) := W_L h^{(L-1)} + b_L, \tag{3.2}$$

where, for each $\ell = 1, \dots, L$, $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ is the weight matrix and $b_\ell \in \mathbb{R}^{d_\ell}$ is the bias vector. The tuple

$$\theta := ((W_1, b_1), \dots, (W_L, b_L)) \in \prod_{\ell=1}^L (\mathbb{R}^{d_\ell \times d_{\ell-1}} \times \mathbb{R}^{d_\ell}) = \Theta$$

is called the parameter vector of the network. We denote by $\mathcal{NN}(d_0, \dots, d_L; \varphi)$ the set of all feedforward neural networks of architecture $(d_0, \dots, d_L)$ and activation function φ . For a schematic representation see Figure 3.1

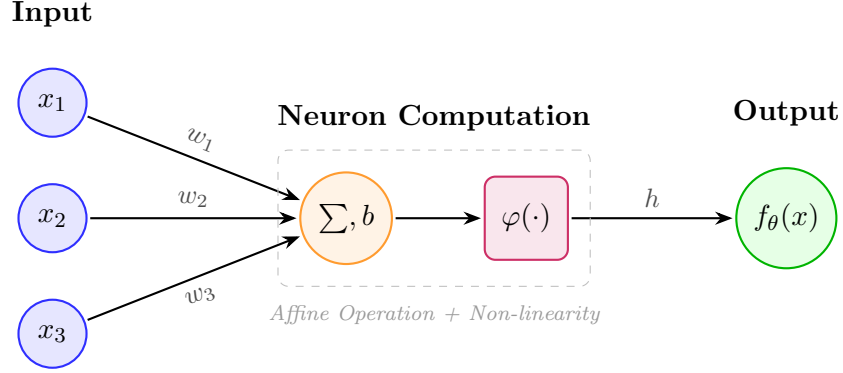


Figure 3.1: Schematic representation of a single computational layer in a Feedforward Neural Network with input $d_0 = 3$, affine combination $z = Wx + b$, activation function φ , and output $f_\theta(x)$.

Now we highlight some key structural features of Definition 3.1.1, to provide a better understanding of it.

Remark 3.1.1. The integer d_ℓ represents the width of layer ℓ , while L represents the depth of the neural network. We say that a network with $L \geq 2$ is a deep neural network.

Remark 3.1.2. In this remark, we briefly explain the indexing of the layers (see Figure 3.2). The index $\ell = 0$ corresponds to the input layer, which receives as input $x \in \mathbb{R}^{d_0}$. For $\ell = 1, \dots, L - 1$, we indicate ℓ -th hidden layer. This layer produces the vector $h^{(\ell)} \in \mathbb{R}^{d_\ell}$, obtained by composing the affine map $x \mapsto W_\ell x + b_\ell$ with the nonlinearity φ . Finally, the indexing $\ell = L$ indicates the output layer. Here, generally, there is no activation function to leave the image of f_θ unconstrained.

Remark 3.1.3. The total number of trainable parameters of a neural network can be easily calculated by the following formula:

$$\sum_{\ell=1}^L (d_\ell \cdot d_{\ell-1} + d_\ell). \quad (3.3)$$

Now, we want to linger on the essential role of the activation function φ . Without nonlinearity, the composition $A_L \circ \dots \circ A_1$ of the affine maps $A_\ell(\cdot) := W_\ell(\cdot) + b_\ell$ would itself reduce to a single affine map, and the network could represent only affine functions of the input. By interleaving affine transformations with φ , the network acquires the capacity to approximate complex nonlinear mappings. We now make this precise.

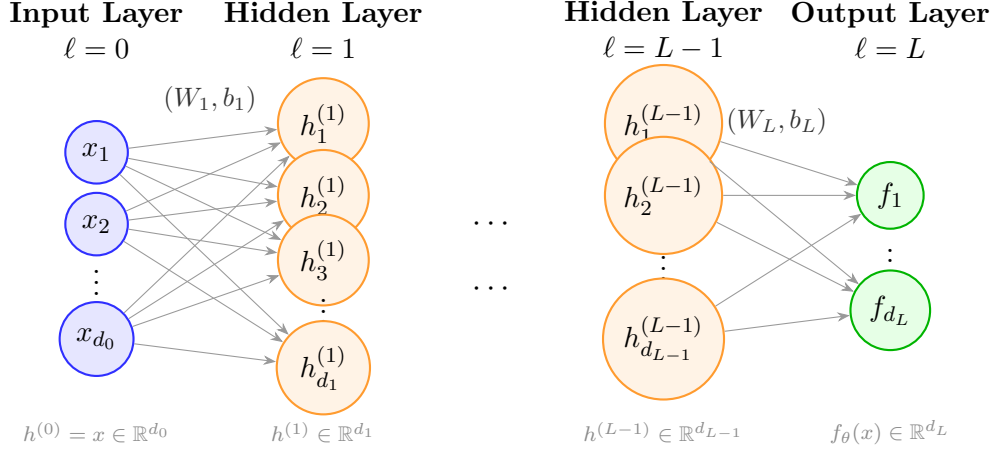


Figure 3.2: Architecture of a deep feedforward neural network of depth L and architecture $(d_0, d_1, \dots, d_{L-1}, d_L)$. Each hidden layer ℓ maps vector $h^{(\ell-1)}$ to $h^{(\ell)}$ via affine parameterization (W_ℓ, b_ℓ) and activation function φ , with output $f_\theta(x)$.

Definition 3.1.2 (Activation Function). A measurable function $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ is called an activation function. For $n \in \mathbb{N}$ and a vector $z = (z_1, \dots, z_n)^\top \in \mathbb{R}^n$, its componentwise extension is defined by

$$\varphi(z) := (\varphi(z_1), \dots, \varphi(z_n))^\top \in \mathbb{R}^n. \quad (3.4)$$

The choice of activation function directly impacts the functioning of the neural network and the efficiency of training. We present as examples two widely used activation functions. We will recall them later in the implementation section.

Example 3.1.1 (Rectified Linear Unit). *The rectified linear unit (ReLU) activation function is defined by*

$$\varphi(x) := \max(x, 0) = x^+, \quad x \in \mathbb{R}. \quad (3.5)$$

The ReLU is a quite popular activation function because of its properties. It is piecewise linear, computationally efficient and helps alleviate the vanishing gradient problem since its derivative is 1 for $x > 0$. The ReLU is continuous but not differentiable at $x = 0$ therefore, in gradient-based training, the subgradient $\partial\varphi(0) \in [0, 1]$ is used at that point.

Example 3.1.2 (Sigmoid). *The sigmoid activation function is defined by*

$$\varphi(x) := \frac{1}{1 + e^{-x}}, \quad x \in \mathbb{R}. \quad (3.6)$$

This function is smooth and strictly increasing. With $\varphi: \mathbb{R} \rightarrow (0, 1)$, the limits are $\lim_{x \rightarrow -\infty} \varphi(x) = 0$, and $\lim_{x \rightarrow +\infty} \varphi(x) = 1$.

3.2 Universal Approximation Theorems

Having defined the basic structure of a neural network and its building blocks, we now present the central theoretical property of this class of functions: the Universal Approximation Theorem (UAT). We present two versions. The first, due to Hornik, 1991, is the classical result on compact subsets of $\mathbb{R}^{d_0}$ and constitutes the theoretical foundation for the deep hedging framework developed in this thesis. The second, a global extension by Cuchiero et al., 2026 to weighted spaces, provides the theoretical basis for a natural generalisation to path-dependent payoffs, which we leave for future work.

We start with the result proposed by Hornik, 1991, following a setup similar to Buehler et al., 2019.

Theorem 3.2.1 (Universal Approximation). *Suppose $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ is bounded and non-constant. Then the following hold:*

- (i) *For any finite Borel measure μ on $(\mathbb{R}^{d_0}, \mathcal{B}(\mathbb{R}^{d_0}))$ and any $1 \leq p < \infty$, the set $\bigcup_{N \in \mathbb{N}} \mathcal{NN}(d_0, N, 1; \varphi)$ is dense in $L^p(\mathbb{R}^{d_0}, \mu)$.*
- (ii) *If in addition $\varphi \in C(\mathbb{R})$, then $\bigcup_{N \in \mathbb{N}} \mathcal{NN}(d_0, N, 1; \varphi)$ is dense in $C(\mathbb{R}^{d_0})$ with respect to the topology of uniform convergence on compact sets.*

Proof. See Hornik, 1991. □

Remark 3.2.1. Theorem 3.2.1 states that depth $L = 2$ with arbitrary width already suffices for universal approximation, so the result extends immediately to the deeper networks of Definition 3.1.1. Moreover, we can generalize the result also for $d_1 > 1$. The conditions on σ are mild: any bounded non-constant activation qualifies, and in particular the sigmoid of Example 3.1.2 satisfies them directly, however it should be noted that also ReLU activation function of Example 3.1.1 is valid, despite being not bounded.

In the deep hedging framework of this thesis, each network receives a finite-dimensional input, so Theorem 3.2.1 already provides the required theoretical justification. However, a natural and important extension of deep hedging considers path-dependent payoffs, where the strategy depends on the full price history. This history lies in a non-compact path space, and Theorem 3.2.1(ii) no longer applies.

The following global result, due to Cuchiero et al., 2026, provides the appropriate theoretical foundation for this extension. We briefly recall their framework in a simplified way.

Let (X, τ_X) be a completely regular Hausdorff topological space. A lower semi-continuous function $\psi: X \rightarrow (0, \infty)$ is called an *admissible weight function* if every pre-image $K_R := \{x \in X : \psi(x) \leq R\}$ is compact with respect to τ_X , for all $R > 0$. The pair (X, ψ) is then called a *weighted space*. Given such a space and a Banach space $(Y, \|\cdot\|_Y)$, we define

$$B_\psi(X; Y) := \overline{C_b(X; Y)}^{\|\cdot\|_{B_\psi}}, \quad \|f\|_{B_\psi(X; Y)} := \sup_{x \in X} \frac{\|f(x)\|_Y}{\psi(x)}, \quad (3.7)$$

as the closure of bounded continuous functions under the weighted norm, writing $B_\psi(X)$ when $Y = \mathbb{R}$.

The Universal Approximation Theorem in this setting can be understood as a constructive application of the Stone–Weierstrass theorem: one builds, from the neural network class, a dense subalgebra or submodule of $B_\psi(X; Y)$, and invokes approximation by that structure. To make this precise, we introduce two conditions on the network architecture.

Definition 3.2.1 (Additive Point Separating Family). Let (X, ψ) be a weighted space. A set $L \subseteq B_\psi(X)$ of bounded continuous functions is called an *additive point separating family* if it is closed under addition, contains all constant functions (i.e. the map $x \mapsto c$ belongs to L for every $c \in \mathbb{R}$), and is point separating, i.e. for every $x_1 \neq x_2$ in X there exists $l \in L$ with $l(x_1) \neq l(x_2)$.

Definition 3.2.2 (Activating Family). A family Φ of continuous functions $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ is called *activating* if, for any Banach space Y and any $a < b$, the set

$$\mathcal{A} := \left\{ \sum_{i=1}^n \gamma_i \varphi_i(w_i \cdot + b_i) + \gamma_0 \mid \gamma_0, \gamma_i \in Y, w_i \in \mathbb{N}_0, b_i \in \mathbb{R}, \varphi_i \in \Phi, n \in \mathbb{N} \right\} \quad (3.8)$$

is dense in $C([a, b]; Y)$. It suffices to verify this for $Y = \mathbb{R}$, since the result then extends to all finite-dimensional subspaces of Y , and by a vector-valued Stone–Weierstrass argument to all Banach spaces Y .

Remark 3.2.2. Any bounded non-constant function $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ is discriminatory in the sense of Hornik, 1991, and every discriminatory function generates an activating family $\Phi = \{\varphi\}$. The ReLU $\varphi(x) = \max(x, 0)$ also generates an activating family, as already stated in Remark 3.2.1.

Theorem 3.2.2 (Global Universal Approximation, Cuchiero et al., 2026). *Let (X, ψ) be a weighted space, Y a Banach space, Φ an activating family, and $L \subseteq B_\psi(X)$ an additive point separating family. Define the set of neural networks*

$$\mathcal{NN}_\Phi := \left\{ X \ni x \mapsto \sum_{i=1}^n \gamma_i \varphi_i(l_i(x)) + \gamma_0 \mid \gamma_0, \gamma_i \in Y, l_i \in L, \varphi_i \in \Phi, n \in \mathbb{N} \right\}. \quad (3.9)$$

Then $\mathcal{NN}_\Phi$ is dense in $B_\psi(X; Y)$.

Proof. See Cuchiero et al., 2026. □

Remark 3.2.3. When X is an infinite-dimensional path space equipped with an admissible weight ψ , the networks in $\mathcal{NN}_\Phi$ are called *functional input neural networks*; the theorem guarantees their density in the corresponding weighted function space.

In conclusion, Theorem 3.2.2 is a genuine strengthening of Theorem 3.2.1. The approximation error is now measured by the weighted norm $\|\cdot\|_{B_\psi(X; Y)}$, which controls the behavior of f and its approximation uniformly over all of X , not merely on a fixed compact subset.

3.3 Training a Neural Network

Having established that neural networks can approximate a given target function, we now turn to the question of how the parameters are chosen in practice. The objective is to find θ such that f_θ approximates a given target mapping on a dataset. This is achieved by minimising a *loss function* via an iterative gradient-based algorithm, using the *backpropagation* procedure to compute the required gradients efficiently.

3.3.1 Loss Function

We start by defining the loss function by using a slightly different formalism compared to the classical ML literature to better adapt to the deep hedging frameworks we will use in the next chapters.

Definition 3.3.1 (Loss Function and Risk Functional). Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $X : \Omega \rightarrow \mathbb{R}^{d_0}$ be a random input variable.

A (pointwise) loss function is a measurable function $\ell : \mathbb{R}^{d_L} \rightarrow \mathbb{R}$. The associated *risk functional* is defined by

$$\mathcal{L}(\theta) := \mathbb{E}[\ell(f_\theta(X))]. \quad (3.10)$$

Given a dataset $\{x^{(j)}\}_{j=1}^M$ of independent samples of X , the risk functional is approximated by the empirical average

$$\mathcal{L}_M(\theta) := \frac{1}{M} \sum_{j=1}^M \ell(f_\theta(x^{(j)})). \quad (3.11)$$

The training problem consists in solving

$$\theta^* \in \arg \min_{\theta} \mathcal{L}(\theta), \quad (3.12)$$

or, in practice, its empirical counterpart with $\mathcal{L}_M$.

Remark 3.3.1. In classical supervised learning, one is given input-output pairs $\{(x^{(j)}, y^{(j)})\}_{j=1}^M \subset \mathbb{R}^{d_0} \times \mathbb{R}^{d_L}$ and the loss depends explicitly on the target, typically through a function of the form $\ell(f_\theta(x), y)$. A common choice is the squared loss $\ell(f_\theta(x), y) = \|f_\theta(x) - y\|^2$, in which case the empirical risk reduces to the mean squared error.

In deep hedging context, no explicit target variable is available, since the loss function is defined directly in terms of the output of the network, for instance through a performance or risk criterion. In particular, in the deep hedging framework, the loss is typically given by a convex risk measure applied to the terminal profit and loss generated by a trading strategy parameterized by f_θ . Despite this difference in interpretation, the mathematical structure remains the same as one seeks to optimize over a class of functions by minimizing an expected loss.

The optimisation problem is typically solved by gradient descent. Starting from an initial parameter vector θ_0 , the parameters are updated iteratively according to the discretized formula

$$\theta_{t+1} := \theta_t - \eta_t \nabla_{\theta} \mathcal{L}_M(\theta_t), \quad (3.13)$$

where $\eta_t > 0$ denotes the learning rate at iteration $t \in \mathbb{N}$.

3.3.2 Mini-Batch Stochastic Gradient Descent

Since evaluating $\mathcal{L}_M(\theta)$ on the full sample at each step can be computationally expensive, a common solution is to work with mini-batches. Let $B_t \subset \{1, \dots, M\}$ be a randomly sampled index set of size $|B_t| = M' \ll M$. Similarly to equation 3.11, the mini-batch approximation of the empirical risk is

$$\hat{\mathcal{L}}_{B_t}(\theta) := \frac{1}{M'} \sum_{j \in B_t} \ell(f_\theta(x^{(j)})). \quad (3.14)$$

Consequently, equation 3.13 becomes

$$\theta_{t+1} := \theta_t - \eta_t \nabla_{\theta} \widehat{\mathcal{L}}_{B_t}(\theta_t). \quad (3.15)$$

The mini-batch gradient is an unbiased or approximately unbiased estimator of the full gradient expressed in equation 3.13 and its use serves a dual purpose: it reduces the per-step computational cost from $\mathcal{O}(M)$ to $\mathcal{O}(M')$, and it introduces stochastic noise into the gradient estimate, which is known empirically to help escape sharp local minima and improve generalisation.

One complete pass through the full sample is called an *epoch*. The dataset is partitioned into $\lceil M/M' \rceil$ non-overlapping batches per epoch, and batches are reshuffled at the beginning of each epoch.

Remark 3.3.2. In all numerical experiments of this thesis, we use the *Adam* optimiser (Kingma & Ba, 2014), which adapts the learning rate for each parameter coordinate using estimates of the first and second moments of the gradient. Adam can be seen as a preconditioned variant of (3.15) and consistently outperforms vanilla gradient descent in deep learning practice.

3.3.3 Backpropagation

The key computational task in (3.15) is the efficient evaluation of $\nabla_{\theta} \widehat{\mathcal{L}}_{B_t}(\theta)$. Backpropagation (Rumelhart et al., 1986) computes these derivatives by repeated application of the chain rule, exploiting the recursive structure of Definition 3.1.1.

For the sake of clarity, we first consider a single sample x and write the forward pass as

$$h^{(0)} = x, \quad z^{(\ell)} = W_{\ell} h^{(\ell-1)} + b_{\ell}, \quad h^{(\ell)} = \varphi(z^{(\ell)}), \quad (3.16)$$

for $\ell = 1, \dots, L-1$ and output

$$f_{\theta}(x) = h^{(L)} := W_L h^{(L-1)} + b_L. \quad (3.17)$$

We denote the corresponding sample loss¹ by

$$\mathcal{L}(\theta; x) := \ell(f_{\theta}(x)). \quad (3.18)$$

The backpropagated error signal at layer ℓ is defined by

$$\mathbf{e}^{(\ell)} := \frac{\partial \mathcal{L}(\theta; x)}{\partial z^{(\ell)}}, \quad (3.19)$$

¹In the supervised learning setting, the sample loss depends explicitly on a target variable y and takes the form $\mathcal{L}(\theta; x, y) = \ell(f_{\theta}(x), y)$. All subsequent derivations remain unchanged.

while at the output layer we set

$$\mathbf{e}^{(L)} := \nabla_{h^{(L)}} \mathcal{L}(\theta; x). \quad (3.20)$$

With this notation, the backward recursion becomes

$$\mathbf{e}^{(\ell)} = (W_{\ell+1}^\top \mathbf{e}^{(\ell+1)}) \odot \varphi'(z^{(\ell)}), \quad (3.21)$$

where $\odot$ denotes the Hadamard product (componentwise multiplication). The gradients with respect to the parameters are given by

$$\frac{\partial \mathcal{L}}{\partial W_\ell} = \mathbf{e}^{(\ell)} (h^{(\ell-1)})^\top, \quad \frac{\partial \mathcal{L}}{\partial b_\ell} = \mathbf{e}^{(\ell)}. \quad (3.22)$$

For a mini-batch B_t , the empirical loss is

$$\widehat{\mathcal{L}}_{B_t}(\theta) := \frac{1}{|B_t|} \sum_{j \in B_t} \mathcal{L}(\theta; x^{(j)}), \quad (3.23)$$

and, by linearity of differentiation, its gradient satisfies

$$\nabla_\theta \widehat{\mathcal{L}}_{B_t}(\theta) = \frac{1}{|B_t|} \sum_{j \in B_t} \nabla_\theta \mathcal{L}(\theta; x^{(j)}). \quad (3.24)$$

In practice, this means that backpropagation is applied to each sample in the mini-batch and the resulting gradients are averaged.

The algorithm therefore consists of two passes through the network: a forward pass, which computes the activations and pre-activations, and a backward pass, which propagates the error signal from the output layer to the input layer and assembles the parameter gradients.

Remark 3.3.3. If φ is not differentiable at isolated points, as in the case of ReLU, the above formulas are interpreted using a subgradient at those points.

Chapter 4

Standard Deep Hedging

4.1 Market Setup

In this section, we introduce the setup we used for the Standard Deep Hedger (STD), which will be also used later for the Ambiguity Averse Deep Hedging (AADH) extension. We follow the notation and structure of the original setup proposed by Buehler et al., 2019, namely a discrete-time financial market with frictions. This discrete-time formulation will also serve us as a numerical approximation of continuous-time models via time discretisation and Monte Carlo simulation.

We consider a finite time horizon $T > 0$ and trading dates $0 = t_0 < t_1 < \dots < t_n = T$, with $n \in \mathbb{N}$. The market is defined on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$, where $\Omega = \{\omega_1, \dots, \omega_N\}$, with $N \in \mathbb{N}$, and the probability measure $\mathbb{P}$ satisfies $\mathbb{P}(\{\omega_i\}) > 0$ for all $i = 1, \dots, N$. It is important to note that we assume Ω is finite for the numerical solution of the hedging problem. Information in the market is described by the filtration $\mathbb{F} = (\mathcal{F}_k)_{k=0, \dots, n}$, generated by an $\mathbb{R}^r$ -valued stochastic process $I = (I_k)_{k=0, \dots, n}$, $r \in \mathbb{N}$, where I_k represents all newly available market information at time t_k . In simpler terms, $\mathcal{F}_k := \sigma(I_0, \dots, I_k)$, so that $\mathcal{F}_k$ contains all information available up to time t_k . Moreover, we denote by $\mathcal{X} := \{X : \Omega \rightarrow \mathbb{R}\}$ the set of all real-valued random variables on Ω .

In the setup we consider, the market consists of $d \in \mathbb{N}$ tradable financial instruments which can be equities as well as options or swaps. In our mathematical notation, we will always consider d instruments: this helps us to generalize the problem also for very complex portfolios of derivatives. However, for the sake of simplicity, in the implementation we will consider $d = 1$. In other words, we will consider only one derivative instrument with the associated underlying or hedging instrument.

The prices are described by an $\mathbb{R}^d$ -valued, $\mathbb{F}$ -adapted stochastic process $S =$

$(S_k)_{k=0,\dots,n}$. This can be also represented by the vector $S_k = (S_k^1, \dots, S_k^d)$ which, for all k , denotes the prices of the d instruments at time t_k . To be more precise, the prices we consider are the so called "mid-prices", so the average price between bid and ask in the financial markets.

Regarding other assumptions, we do not impose completeness of the market, nor do we assume that every contingent claim can be replicated. In particular, we allow for market frictions and incomplete hedging.

Finally, the objective of deep hedging will be to hedge a contingent claim Z , that is, an $\mathcal{F}_T$ -measurable random variable representing the payoff of a liability at maturity T .

Example 4.1.1 (European Call Option). *Let $S = (S_k)_{k=0,\dots,n}$ be the price process of the underlying asset, and let $K > 0$ be a fixed strike. A European call option with maturity T has the payoff*

$$Z = (S_n - K)^+. \quad (4.1)$$

4.2 Hedging Strategies

4.2.1 Classical Delta Hedging

Before introducing the deep hedging strategy, we switch to continuous time and briefly recall the classical delta hedging proposed by Black and Scholes, 1973. Differently from our discrete-time framework with frictions, the classical delta hedging serves as a benchmark in the idealized case of continuous time under idealized assumptions. We consider a call option with payoff $Z = (S_T - K)^+$ in the continuous-time setting.

In the Black-Scholes model, the underlying asset price follows a geometric Brownian motion

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \quad (4.2)$$

where $\mu \in \mathbb{R}$ is the drift, $\sigma > 0$ is the volatility and W is the Brownian motion. The market is assumed to be frictionless, allowing continuous trading and no transaction costs. Moreover, borrowing and lending are possible at a constant risk-free rate r . In our implementation, for simplicity, we consider $r = 0$ for all $t \in [0, T]$.

Let $V(t, S_t)$ denote the price at time t of the option written on the underlying asset with maturity T and payoff Z . Applying Itô's-Doeblin formula to $V(t, S_t)$ gives

$$dV = \left(\frac{\partial V}{\partial t} + \mu S_t \frac{\partial V}{\partial S} + \frac{1}{2} \sigma^2 S_t^2 \frac{\partial^2 V}{\partial S^2} \right) dt + \sigma S_t \frac{\partial V}{\partial S} dW_t. \quad (4.3)$$

However, the Wiener process underlying Equations 4.2 and 4.3 is the same. Therefore, let's consider a self-financing portfolio consisting of one short position in the derivative and a long position Δ_t in the underlying asset, whose value is

$$\Pi_t = -V(t, S_t) + \Delta_t S_t. \quad (4.4)$$

Choosing

$$\Delta_t = \frac{\partial V}{\partial S}(t, S_t), \quad (4.5)$$

eliminates the stochastic component and the resulting portfolio becomes locally riskless and must therefore earn the risk-free rate. This no-arbitrage argument implies that V satisfies the Black-Scholes partial differential equation

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0, \quad (4.6)$$

with terminal condition specified in Equation 4.1 in the case of an European call option.

Now, considering a strike price K , the famous solution for the Black-Scholes PDE (Equation 4.6) is given by

$$V(t, S) = S\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2). \quad (4.7)$$

The symbol Φ indicates the cumulative standard normal distribution

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{z^2}{2}} dz. \quad (4.8)$$

and

$$d_1 = \frac{\log(S/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \quad (4.9)$$

$$d_2 = d_1 - \sigma\sqrt{T-t}. \quad (4.10)$$

To conclude, delta is given by

$$\Delta_t = \frac{\partial V}{\partial S}(t, S_t) = \Phi(d_1). \quad (4.11)$$

Hence, in the Black-Scholes framework, continuous rebalancing of the portfolio according to Δ_t perfectly replicates the option payoff, and this perfect replication is exact due to continuous trading, absence of transaction costs and market completeness.

However, these assumptions are not satisfied in the discrete-time market with frictions considered in our setup and described in Section 4.1. In fact, in real-world market, the exact replication is no longer possible, and this motivates the risk-minimization approach used in deep hedging.

4.2.2 Strategies for Deep Hedging

In deep hedging, under the conditions specified in Section 4.1, to hedge a contingent claim $Z \in \mathcal{X}$, we trade in the d available financial instruments whose price process is $S = (S_k)_{k=0,\dots,n}$. In the case of deep hedging the trading strategy is defined as an $\mathbb{R}^d$ -valued, $\mathcal{F}$ -adapted stochastic process $\delta = (\delta_k)_{k=0,\dots,n-1}$, with $\delta_k = (\delta_k^1, \dots, \delta_k^d) \in \mathbb{R}^d$, where δ_k^i denotes the number of units of asset i held on the considered time interval.

Intuitively, the positions of the trading strategy are represented by the letter δ since what we want to do practically is to substitute the classical δ -hedging strategy with the deep hedging strategy. In other words, the objective is the same so we held the same notation.

Moreover, it should be noted that in our setup we set $\delta_{-1} := 0$ and $\delta_n := 0$. These two are the initial and final condition and indicate that no position is held before trading begins and all positions are liquidated at maturity respectively.

In practice, however, trading strategies may be subject to more constraints, for instance due to market frictions, liquidity, or position limits. We denote by $\mathcal{H}^u$ the set of all unconstrained trading strategies and to address these constraints we consider a subset $\mathcal{H} \subset \mathcal{H}^u$ of admissible trading strategies. At each time t_k , the admissible positions are restricted to a set $\mathcal{H}_k \subset \mathbb{R}^d$. We call a strategy δ admissible if $\delta_k \in \mathcal{H}_k$ for all $k = 0, \dots, n-1$.

All trading is assumed to be self-financing. This means that changes in the portfolio value are solely generated by gains and losses from trading and by transaction costs, without additional cash injections after the initial time.

Let $P_0 \in \mathbb{R}$ denote the initial capital received at time t_0 , for instance, the premium of the sold derivatives. For a trading strategy $\delta \in \mathcal{H}$, the cumulative gains from trading up to time T are given by

$$(\delta \cdot S)_T := \sum_{k=0}^{n-1} \delta_k \cdot (S_{k+1} - S_k). \quad (4.12)$$

More precisely, for each rebalancing at time t_k we incur costs related to the traded amount. In the implementation, we consider proportional transaction costs, so we define the total transaction costs as

$$C_T(\delta) := \sum_{k=0}^n c_k S_k |\delta_k - \delta_{k-1}|, \quad (4.13)$$

where $c_k \geq 0$ denotes a proportional transaction cost coefficient. In our numerical experiments, we assume $c_k \equiv c$ constant over time and no transaction costs at maturity.

The terminal profit and loss (P&L) associated with the strategy δ is therefore given by

$$\text{P\&L}_T(P_0, \delta, Z) := -Z + P_0 + (\delta \cdot S)_T - C_T(\delta). \quad (4.14)$$

This quantity will be evaluated and optimised with a neural network according to a suitable risk criterion.

4.3 Deep Hedging

4.3.1 From the Ideal Hedging Problem to the Objective Functional

We now connect the market setup of the previous section with the training problem of deep hedging following the notions outlined in Section 3.3. The central object is not a target label, as in classical supervised learning, but the terminal profit and loss generated by a trading strategy.

By recalling how we computed the loss in equation 3.11, we first define the risk measure that is minimized.

Definition 4.3.1 (Convex Risk Measure). Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $\mathcal{X} := \{X : \Omega \rightarrow \mathbb{R}\}$ be the space of all real-valued random financial positions. A functional $\rho : \mathcal{X} \rightarrow \mathbb{R}$ is called a convex risk measure if for all $X, X_1, X_2 \in \mathcal{X}$ the following properties hold:

1. Decreasing monotonicity: if $X_1 \leq X_2$, then $\rho(X_1) \geq \rho(X_2)$.
2. Convexity: $\rho(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda \rho(X_1) + (1 - \lambda)\rho(X_2)$ for $\lambda \in [0, 1]$.
3. Cash invariance: $\rho(X + c) = \rho(X) - c$ for $c \in \mathbb{R}$.

In the original Deep Hedging formulation and in this thesis we use a convex risk measure for its clear economic meaning. In Definition 4.3.1, the random variable X represents the terminal wealth, while $-X$ represents a liability. Moreover, the definition tells us that monotonicity indicates that a more favorable position requires less capital injection. Convexity formalizes the principle that diversification does not increase risk. Cash invariance ensures that adding deterministic capital reduces required capital one-for-one.

Example 4.3.1 (Entropic Risk Measure). *For a fixed risk aversion parameter $\gamma > 0$, the entropic risk measure is defined by*

$$\rho_\gamma(X) := \frac{1}{\gamma} \log \mathbb{E} [e^{-\gamma X}] . \quad (4.15)$$

Definition 4.3.2 (Objective Functional). Let $\ell : \mathbb{R} \rightarrow \mathbb{R}$ be a measurable loss function. We define the objective associated with a strategy δ by

$$\mathcal{J}(P_0, Z, \delta) := \mathbb{E}[\ell(P \& L_T(P_0, \delta, Z))] . \quad (4.16)$$

The corresponding hedging problem is

$$\pi(Z) := \inf_{\delta \in \mathcal{H}} \mathcal{J}(P_0, Z, \delta) . \quad (4.17)$$

4.3.2 Parametrization of the Trading Strategy

The functional $\pi(Z)$ in Equation 4.17 is the optimal value of the hedging problem over the whole admissible strategy set $\mathcal{H}$, as presented in Subsection 4.3.1. In the terminology of the deep hedging literature, this is the idealized problem. In practice, one replaces $\mathcal{H}$ by a parametrized family of strategies generated by neural networks, following the framework introduced in Chapter 3.

More precisely, let δ^θ denote the trading strategy produced by a neural network with parameter vector $\theta \in \Theta$. The deep hedging problem then becomes the finite-dimensional optimization problem

$$\pi_{\mathcal{NN}}(Z) := \inf_{\theta \in \Theta} \mathcal{J}(P_0, Z, \delta^\theta) . \quad (4.18)$$

Thus, $\pi(Z)$ is the abstract optimization value over all admissible strategies, whereas $\pi_{\mathcal{NN}}(Z)$ is its neural-network approximation, obtained by restricting the optimization to a fixed architecture $f_\theta \in \mathcal{NN}(d_0, \dots, d_L; \varphi)$. Consistency of this approximation as the network capacity grows (i.e. as Θ ranges over an increasing family of architectures) follows from the universal approximation argument of Buehler et al., 2019, Prop. 4.9, which we do not reprove here.

Remark 4.3.1. In classical supervised learning, the loss is usually written directly as a function of the parameters, e.g. $\mathcal{L}(\theta)$. In deep hedging, by contrast, the parameter dependence enters through the strategy δ^θ , while the objective is evaluated on the resulting terminal profit and loss. This is why the notation $\pi(Z)$ is natural for the ideal problem, and why the training stage is best viewed as the minimization of a parametrized approximation of that value functional.

Following Buehler et al., 2019, the trading strategy at each rebalancing date t_k is Markovian, depending only on the current market information I_k and the previously held position:

$$\delta_k^\theta = F_{\theta_k}(I_k, \delta_{k-1}^\theta), \quad (4.19)$$

with $F_{\theta_k} \in \mathcal{NN}(d_0, \dots, d_L; \varphi)$. In the original semi-recurrent construction, each time step t_k is assigned its own network F_{θ_k} with an independent parameter vector θ_k , so that $\theta = (\theta_0, \dots, \theta_{n-1})$.

In this thesis we instead adopt the weight-sharing variant discussed in Buehler et al., 2019, Remark 4.5, and more similar to the approach in Jones et al., 2025: a single network f_θ with a common parameter vector θ is used across all rebalancing dates, and the time index is passed explicitly as an additional input feature. The trading strategy is thus given by

$$\delta_k^\theta = f_\theta(T - t_k, S_k, \delta_{k-1}^\theta), \quad f_\theta \in \mathcal{NN}(d_0, \dots, d_L; \varphi), \quad (4.20)$$

for $k = 0, \dots, n-1$, with $\delta_{-1}^\theta := 0$ (see Figure 4.1). This reduces the total number of trainable parameters from that of n independent networks down to a single network, thereby decreasing computational complexity and training time.

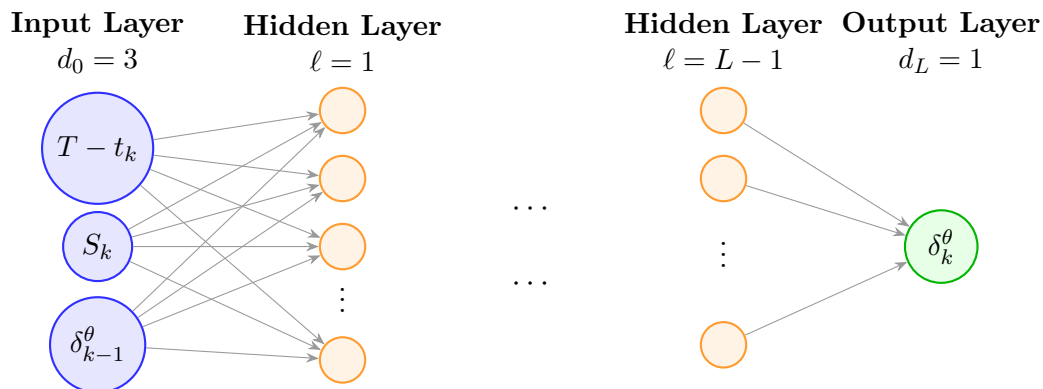


Figure 4.1: Neural network parametrization of the trading strategy $\delta_k^\theta = f_\theta(T - t_k, S_k, \delta_{k-1}^\theta)$ at rebalancing date t_k . The single parameter vector $\theta \in \Theta$ is shared across all time steps $k = 0, \dots, n-1$.

4.4 Deep Hedging Algorithm

To summarize and provide a more practical view of the framework presented in this chapter, Deep Hedging can be seen as a parametrization of the admissible trading

strategy by means of a neural network. Instead of optimizing directly over the whole strategy space $\mathcal{H}$, we optimize the objective in Equation 4.18 with respect to the neural-network parameters θ . In our implementation, the same network is used at every rebalancing date, with the time to maturity included as an input, as described in Equation 4.20. The training procedure is summarized in Algorithm 1.

Algorithm 1 Deep Hedging Training Algorithm

Require: N_{train} , N_{epochs} , minibatch size B , trading dates $0 = t_0 < \dots < t_n = T$, P_0 , payoff Z , market model, neural network f_θ , objective/risk measure ρ

Ensure: Trained network f_{θ^*}

```

1: Initialize  $\theta$  randomly
2: Generate  $N_{\text{train}}$  market paths
    $\left\{ (S_k^{(j)})_{k=0}^n \right\}_{j=1}^{N_{\text{train}}} \quad \triangleright \text{Training set}$ 
3: Compute the corresponding payoffs  $Z^{(j)}$ 
4: for  $e = 1, \dots, N_{\text{epochs}}$  do
5:   Shuffle the training paths and form minibatches
6:   for each minibatch  $\mathcal{B}$  do
7:     for each  $j \in \mathcal{B}$  do
8:        $\delta_{-1}^{(j)} \leftarrow 0, \quad G^{(j)} \leftarrow 0, \quad C^{(j)} \leftarrow 0$ 
9:     end for
10:    for  $k = 0, \dots, n-1$  do
11:      for each  $j \in \mathcal{B}$  do
12:         $\delta_k^{(j)} \leftarrow f_\theta(T - t_k, S_k^{(j)}, \delta_{k-1}^{(j)}) \quad \triangleright \text{Hedging position}$ 
13:         $G^{(j)} \leftarrow G^{(j)} + \delta_k^{(j)}(S_{k+1}^{(j)} - S_k^{(j)}) \quad \triangleright \text{Trading gain}$ 
14:         $C^{(j)} \leftarrow C^{(j)} + c_k S_k^{(j)} |\delta_k^{(j)} - \delta_{k-1}^{(j)}| \quad \triangleright \text{Transaction costs}$ 
15:      end for
16:    end for
17:    for each  $j \in \mathcal{B}$  do
18:       $\delta_n^{(j)} \leftarrow 0 \quad \triangleright \text{Liquidation at maturity}$ 
19:       $\text{P\&L}_T^{(j)} \leftarrow -Z^{(j)} + P_0 + G^{(j)} - C^{(j)}$ 
20:    end for
21:     $\hat{\rho}_{\mathcal{B}}(\theta) \leftarrow \rho(\{\text{P\&L}_T^{(j)}\}_{j \in \mathcal{B}}) \quad \triangleright \text{Empirical minibatch objective}$ 
22:     $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta \hat{\rho}_{\mathcal{B}}(\theta))$ 
23:  end for
24: end for
25: return  $f_{\theta^*}$ 

```

Remark 4.4.1. In Algorithm 1, we set $P_0 = 0$, consistently with the numerical implementation considered in the experimental part. More generally, P_0 may be treated as exogenously given, optimized jointly with the neural-network parameters, or obtained through an alternative pricing procedure. The minibatch training procedure follows the approach discussed in Subsection 3.3.2.

Chapter 5

Ambiguity-Averse Deep Hedging

5.1 Ambiguity Aversion

Classical expected utility theory, and by extension the risk-measure framework of Section 4.3.1, assumes that the agent evaluates outcomes with respect to a single, known probability measure $\mathbb{P}$. However, Ellsberg, 1961 showed that this is violated by decision makers whenever they face *model uncertainty* or, in simpler words, *ambiguity*. In our case, uncertainty is not only about a random outcome (such as the P&L of the hedging strategy), but also about which probability measure governs it in the first place. In estimating the parameters used to train the deep hedger, we run into exactly this issue (Cetingoz & Lehalle, 2026; Duembgen & Rogers, 2014). This distinction between quantifiable *risk* (uncertainty within a known model) and unquantifiable *ambiguity* (uncertainty about the model itself) motivates a separate treatment of the two.

Definition 5.1.1 (Maxmin Expected Utility). Let $\mathcal{Q}$ be a (non-empty, convex, weak*-compact) set of probability measures on $(\Omega, \mathcal{F})$, representing the decision maker's set of plausible models, and let U be a utility function. A maxmin expected utility (MEU) agent evaluates a position $X \in \mathcal{X}$ by

$$\inf_{Q \in \mathcal{Q}} \mathbb{E}_Q[U(X)]. \quad (5.1)$$

Remark 5.1.1. The topological properties of $\mathcal{Q}$ ensure mathematical and economic soundness. Weak*-compactness guarantees that the continuous mapping $Q \mapsto \mathbb{E}_Q[U(X)]$ attains its infimum as a minimum via the extreme value theorem, ruling out infinite or unattainable worst-case scenarios. Convexity reflects economic indifference to model randomization; because expectation is linear in Q , the infimum over $\mathcal{Q}$ equals

that over its closed convex hull, providing the necessary structure to apply minimax theorems in dual optimization.

The MEU framework of Gilboa and Schmeidler, 1989 was proposed to rationalize the Ellsberg paradox (Ellsberg, 1961). However, it is often criticised as excessively conservative, since the entire decision is based on the single worst model in $\mathcal{Q}$, regardless of how implausible it is relative to the others. As we discuss in Section 5.2, robustification approaches for deep hedging that instantiate this worst-case criterion (Limmer & Horvath, 2024; Lütkebohmert et al., 2022) might encounter this issue, motivating the smooth ambiguity approach of Jones et al., 2025 that we adopt.

The framework presented in the following Definition 5.1.2 comes from the work of Klibanoff et al., 2005 and is used as a key step in Jones et al., 2025.

Definition 5.1.2 (Smooth Ambiguity Aversion). Let $\mathcal{Q}$ be a set of candidate models as above, equipped with a probability measure μ over $\mathcal{Q}$ representing the agent’s second-order belief about which model is correct. Let U_1 be an “inner” utility function, evaluated under each candidate model, and let U_2 be an “outer” utility function, evaluated over the induced distribution of certainty equivalents. A smooth-ambiguity-averse agent evaluates $X \in \mathcal{X}$ by

$$\mathbb{E}_{Q \sim \mu} \left[U_2 \left(\mathbb{E}_Q [U_1(X)] \right) \right]. \quad (5.2)$$

The smooth ambiguity aversion in Equation 5.2 nests both extremes as special cases. First, taking U_2 linear collapses the two expectations into one and recovers Bayesian (ambiguity-neutral) expected utility with respect to the averaged model $\int_{\mathcal{Q}} Q d\mu(Q)$. Second, sending the concavity of U_2 to infinity recovers the MEU criterion of Definition 5.1.1 as a limiting case. The concavity of U_2 thus plays the role of an *ambiguity aversion* parameter. It should be noted that this is separate from the risk aversion given by U_1 , allowing the two functions to be calibrated independently.

Finally, requiring U_2 to be a convex risk measure guarantees that ambiguity is measured on the same scale as risk (in this case, in units of capital) and inherits the same properties established in Definition 4.3.1.

5.2 Ambiguity Aversion in Deep Hedging

The result of deep hedging is stated with respect to a *fixed* market measure $\mathbb{P}$: the deep hedging strategy δ^θ is trained to be optimal under the particular market generator used to produce the training paths. In practice, this generator is a proxy for the true, unknown market dynamics, recalibrated periodically to the most recent data.

Whenever the distribution realised during live hedging deviates from the distribution used for training (a *distributional shift*), the standard deep hedger (SDH) is not able to remain robust.

One natural route to robustness is distributionally robust optimization (DRO). Here, the hedging objective is optimized against the worst case over an ambiguity set of measures, for instance a Wasserstein ball around the training measure (Bartl et al., 2021; Rahimian & Mehrotra, 2022), or by explicitly randomising the parameters of the market generator during training (Lütkebohmert et al., 2022). However, such approaches directly instantiate the maxmin criterion of Definition 5.1.1 and inherit its drawbacks. One problem is that they can require charging a substantially higher option premium to compensate for the resulting overly conservative hedging (Lütkebohmert et al., 2022). Furthermore, alternative robustification approaches, such as training an adversarial market generator alongside the hedger (Limmer & Horvath, 2024), might raise suspicion for adding another layer of complexity to an already black-box deep hedging strategy.

The approach we build on instead, following the Ambiguity Averse Deep Hedger (AADH) of Jones et al., 2025, uses Definition 5.1.2 directly. It partitions the *existing* training data into clusters according to features believed relevant to out-of-sample robustness, treats each cluster as a candidate model, and applies an inner risk measure per cluster and an outer ambiguity measure across clusters. No additional or adversarially generated data is required, and, no additional premium need be charged for the added robustness.

To summarize, the Ambiguity Averse Deep Hedging (AADH) approach sits between the traditional (and maybe overly conservative) robustification methods and the ambiguity neutrality of the standard deep hedging (SDH) framework. By carefully choosing the smooth ambiguity aversion parameter in the function aggregator (U_2 in Equation 5.2) one can decide how much to "robustify" the model, while avoiding drawbacks such as having to charge a higher option premium, or using different generators and sets of parameters for each generated market scenario.

5.2.1 Feature Clustering

Let $\mathcal{D} := \{S^j\}_{j=1}^{N_{train}} \subset (\mathbb{R}^d)^{n+1}$ denote the set of N_{train} sample paths of the hedging instruments generated by the market generator, as used to train the standard deep hedger of Section 4.4. Let

$$\Psi : (\mathbb{R}^d)^{n+1} \rightarrow \mathbb{R}^{n_f} \quad (5.3)$$

be a feature map whose n_f components are pathwise functionals of interest computed from a single sample path. In our case we chose realised volatility, autocorrelation

of returns, and maximum drawdown. We consider a feature valid only if it can be computed pathwise from information available prior to training (i.e. it must not depend on the trading strategy).

Given a fixed number of clusters $k \in \mathbb{N}$, a clustering algorithm applied to the feature set $\mathcal{U} := \Psi(\mathcal{D}) \subset \mathbb{R}^{n_f}$ produces a labelling function

$$\chi : \mathbb{R}^{n_f} \rightarrow \{1, \dots, k\}, \quad (5.4)$$

which in turn partitions the training paths into k disjoint clusters $\mathcal{C}_1, \dots, \mathcal{C}_k \subset \mathcal{D}$, where $\mathcal{C}_i := \Psi^{-1}(\chi^{-1}(i)) \cap \mathcal{D}$, together with their empirical probabilities

$$\mu_i := \frac{|\mathcal{C}_i|}{|\mathcal{D}|}. \quad (5.5)$$

Since the components of Ψ are in general not measured on comparable scales, $\mathcal{U}$ is normalised (in our implementation linearly onto the hypercube $[0, 1]^{n_f}$ with $n_f = 3$) prior to clustering.

5.2.2 K-Means Clustering

We use k -means clustering to obtain the labelling function χ from the (normalised) feature set $\mathcal{U}$. Given $\mathcal{U} = \{f^j\}_{j=1}^{N_{train}} \subset \mathbb{R}^{n_f}$ and a target number of clusters k , k -means seeks centroids $\{c_1, \dots, c_k\} \subset \mathbb{R}^{n_f}$ minimising the within-cluster sum of squares

$$\sum_{i=1}^k \sum_{f^j: \chi(f^j)=i} \|f^j - c_i\|_2^2, \quad (5.6)$$

with $\chi(f^j) := \arg \min_i \|f^j - c_i\|_2$. The procedure used to build the feature clusters is summarised in Algorithm 2.

To provide visual understanding of Algorithm 2, Figure 5.1 illustrates the clustering of the training paths in the three-dimensional feature space defined by maximum drawdown, autocorrelation, and realised volatility. Figure 5.2 shows the corresponding price paths grouped according to their assigned clusters. From a financial perspective, these clusters can be interpreted as distinct market regimes, characterised by different combinations of market features and therefore potentially requiring either regime-specific hedging strategies or a single ambiguity-averse strategy designed to remain robust across regimes.

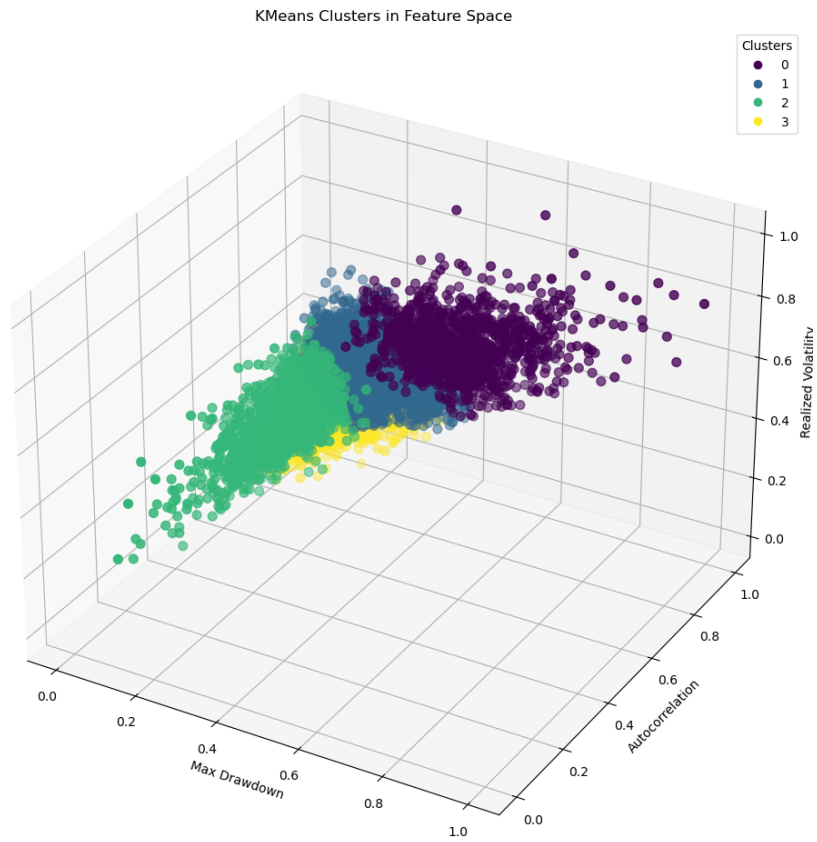


Figure 5.1: Paths defined as points clustered using K-Means algorithm. Features used: Max Drawdown, Autocorrelation and Realized Volatility.

Algorithm 2 Feature Clustering

Require: Training paths $\mathcal{D} = \{S^j\}_{j=1}^{N_{train}}$; feature map $\Psi : (\mathbb{R}^d)^{n+1} \rightarrow \mathbb{R}^{n_f}$; number of clusters k

- 1: **for** $j = 1, \dots, N_{train}$ **do**
 - 2: $f^j \leftarrow \Psi(S^j)$
 - 3: **end for**
 - 4: $\mathcal{U} \leftarrow \{f^j\}_{j=1}^{N_{train}}$, normalised componentwise
 - 5: $\chi \leftarrow \text{KMEANS}(\mathcal{U}, k)$
 - 6: **for** $i = 1, \dots, k$ **do**
 - 7: $\mathcal{C}_i \leftarrow \Psi^{-1}(\chi^{-1}(i)) \cap \mathcal{D}$
 - 8: $\mu_i \leftarrow |\mathcal{C}_i|/|\mathcal{D}|$
 - 9: **end for**
 - 10: **return** partition $\{\mathcal{C}_i\}_{i=1}^k$ and empirical measure $\{\mu_i\}_{i=1}^k$
-

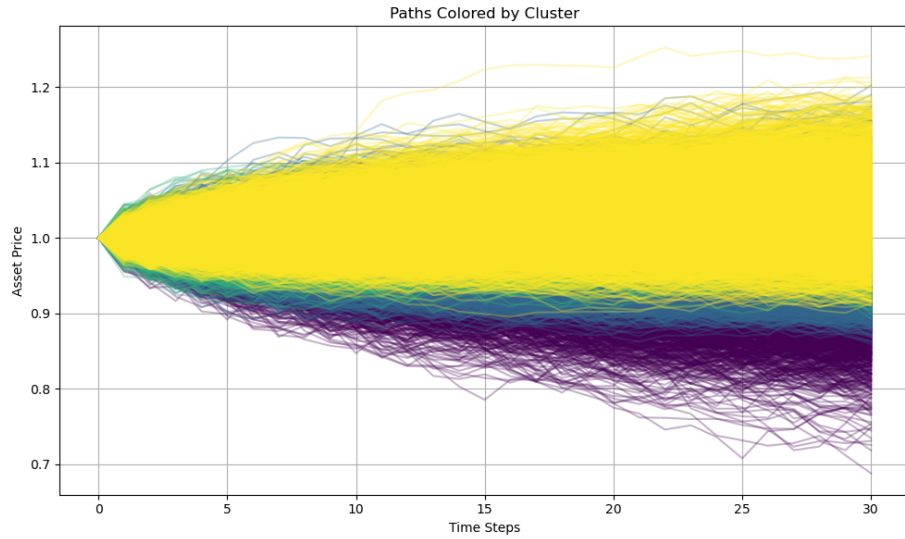


Figure 5.2: Price paths divided in regimes based on K-Means clustering.

5.3 The Ambiguity-Averse Objective

In this section, we consider $X_T^\delta := \text{P\&L}_T(P_0, \delta, Z)$ for simplicity of notation. We recall that $\text{P\&L}_T(P_0, \delta, Z)$ denotes the terminal wealth of a hedging strategy δ , as in Equation 4.14. Given a feature partition $\{\mathcal{C}_i, \mu_i\}_{i=1}^k$ as constructed in Section 5.2.2, each cluster $\mathcal{C}_i$ induces a conditional market measure $\mathbb{P}_i(\cdot) := \mathbb{P}(\cdot \mid \chi(\Psi(S)) = i)$. We treat these k conditional measures as candidate models for a random measure $\tilde{\mathbb{P}}$, taking value $\mathbb{P}_i$ with probability μ_i , i.e. with law

$$\mu = \sum_{i=1}^k \mu_i \Delta_{\mathbb{P}_i}, \quad (5.7)$$

where $\Delta_{\mathbb{P}_i}$ denotes the Dirac measure centered at the conditional market measure $\mathbb{P}_i$.

Following the smooth-ambiguity-averse construction of Section 5.1, we equip $\mathcal{X}$ with an inner convex risk measure ρ_1 (Definition 4.3.1), evaluated separately under each $\mathbb{P}_i$, and an outer convex risk measure ρ_2 , evaluated over the distribution of these k per-cluster risks. The ambiguity-averse deep hedging problem is then to find

$$\inf_{\theta \in \Theta} \rho_{2, \tilde{\mathbb{P}} \sim \mu} \left(-\rho_{1, \tilde{\mathbb{P}}} \left(X_T^{\delta^\theta} \right) \right), \quad (5.8)$$

where $\rho_{1, \tilde{\mathbb{P}}}(X_T^{\delta^\theta})$ takes value $\rho_{1, \mathbb{P}_i}(X_T^{\delta^\theta})$ with probability μ_i .

In this thesis, as in Jones et al., 2025, we take ρ_2 to be the entropic risk measure of Example 4.15, applied over the measure μ :

$$\rho_2(Y) := \frac{1}{\gamma} \log \mathbb{E}_\mu [e^{-\gamma Y}] = \frac{1}{\gamma} \log \left(\sum_{i=1}^k \mu_i e^{-\gamma (-\rho_{1, \mathbb{P}_i}(X_T^{\delta^\theta}))} \right), \quad (5.9)$$

for an ambiguity aversion parameter $\gamma > 0$.

The number of clusters k interpolates between two limiting cases we investigate below, by first defining the optimized certainty equivalent risk measures (Ben-Tal & Teboulle, 2007), needed for the proofs.

Definition 5.3.1 (Optimized Certainty Equivalent). Let $\ell : \mathbb{R} \rightarrow \mathbb{R}$ be a measurable, non-decreasing, convex loss function. The optimized certainty equivalent (OCE) risk measure associated to ℓ is

$$\rho(X) := \inf_{w \in \mathbb{R}} \{w + \mathbb{E}[\ell(-X - w)]\}, \quad X \in \mathcal{X}. \quad (5.10)$$

Proposition 5.3.1 (Trivial clustering: $k = 1$). *Suppose the feature partition consists of a single cluster ($k = 1$, $\mathcal{C}_1 = \mathcal{D}$, $\mu_1 = 1$). Then, for any outer ambiguity measure ρ_2 normalised so that $\rho_2(0) = 0$, the objective (5.8) coincides with the standard deep hedging objective under the risk measure ρ_1 , i.e. $\pi_{\mathcal{NN}}(Z)$ of Equation 4.18.*

Proof. With $k = 1$, $\tilde{\mathbb{P}}$ is degenerate, taking the value $\mathbb{P}$ with probability 1; hence $\rho_{1,\tilde{\mathbb{P}}}(X_T^{\delta^\theta}) = \rho_{1,\mathbb{P}}(X_T^{\delta^\theta})$ almost surely, a deterministic constant which we denote by c . By cash-invariance of ρ_2 applied to $-c$, $\rho_2(-c) = \rho_2(0 + (-c)) = \rho_2(0) - (-c) = \rho_2(0) + c$. Since $\rho_2(0) = 0$ by assumption, this gives $\rho_{2,\tilde{\mathbb{P}} \sim \mu}(-\rho_{1,\tilde{\mathbb{P}}}(X_T^{\delta^\theta})) = \rho_2(-c) = c = \rho_{1,\mathbb{P}}(X_T^{\delta^\theta})$, recovering the standard deep hedging objective. $\square$

Proposition 5.3.2 (Maximal clustering: $k = N_{train}$). *Suppose ρ_2 is an OCE risk measure in the sense of Definition 5.3.1, associated to loss function ℓ , and $\rho_1(0) = \rho_0$. If the feature partition consists of $k = N_{train}$ singleton clusters, one per training path, then the objective (5.8) coincides (up to the constant ρ_0) with a standard deep hedging objective under the risk measure ρ_2 , evaluated with respect to the empirical measure $\hat{\mathbb{P}}$ of the training set.*

Proof. With $k = N_{train}$, each $\mathbb{P}_i$ is a Dirac mass on the i -th training path ω_i . Writing $x_i := X_T^{\delta^\theta}(\omega_i)$, cash-invariance of ρ_1 applied to the (deterministic, under $\mathbb{P}_i$) constant x_i gives $\rho_{1,\mathbb{P}_i}(X_T^{\delta^\theta}) = \rho_1(0 + x_i) = \rho_1(0) - x_i = \rho_0 - x_i$. Writing $\eta \in \mathbb{R}$ for the OCE optimisation variable,

$$\begin{aligned} \rho_{2,\tilde{\mathbb{P}} \sim \mu}(-\rho_{1,\tilde{\mathbb{P}}}(X_T^{\delta^\theta})) &= \inf_{\eta \in \mathbb{R}} \left\{ \eta + \mathbb{E}_\mu \left[\ell \left(\rho_{1,\tilde{\mathbb{P}}}(X_T^{\delta^\theta}) - \eta \right) \right] \right\} \\ &= \inf_{\eta \in \mathbb{R}} \left\{ \eta + \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \ell(\rho_0 - x_i - \eta) \right\} \\ &= \inf_{\eta' \in \mathbb{R}} \left\{ \eta' - \rho_0 + \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \ell(-x_i - \eta') \right\} \quad (\eta' := \eta + \rho_0) \\ &= \inf_{\eta' \in \mathbb{R}} \left\{ \eta' + \mathbb{E}_{\hat{\mathbb{P}}} \left[\ell \left(-X_T^{\delta^\theta} - \eta' \right) \right] \right\} - \rho_0 = \rho_{2,\hat{\mathbb{P}}}(X_T^{\delta^\theta}) - \rho_0, \end{aligned}$$

using $\mu_i = 1/N_{train}$ for all i . $\square$

Propositions 5.3.1 and 5.3.2 justify viewing k as a hyperparameter interpolating between two standard deep hedgers, one minimising the inner risk measure ρ_1 (ambiguity-neutral) and one minimising the outer ambiguity measure ρ_2 (maximal ambiguity aversion, given the available features): a finite $k > 1$ balances the two. In other words, if $k = 1$ then we have a standard deep hedger minimizing ρ_1 over the full

training measure, while if $k = N_{train}$ we obtain a standard deep hedger minimizing ρ_2 over the empirical measure.

Finally, we want to show that Equation 5.9 is a particular case of Equation 5.1 (Klibanoff et al., 2005, Proposition 3).

Proposition 5.3.3 (Ambiguity aversion in the limit $\gamma \rightarrow \infty$). *Assume $\mu_i > 0$ for all $i = 1, \dots, k$. Then*

$$\lim_{\gamma \rightarrow \infty} \rho_{2, \tilde{\mathbb{P}} \sim \mu} \left(-\rho_{1, \tilde{\mathbb{P}}} \left(X_T^{\delta^\theta} \right) \right) = \max_{1 \leq i \leq k} \rho_{1, \mathbb{P}_i} \left(X_T^{\delta^\theta} \right). \quad (5.11)$$

That is, in the limit of infinite ambiguity aversion, the AADH objective reduces to a discrete maxmin criterion over the k candidate cluster measures $\{\mathbb{P}_1, \dots, \mathbb{P}_k\}$, coinciding with the maxmin expected utility criterion of Definition 5.1.1 restricted to $\mathcal{Q} := \{\mathbb{P}_1, \dots, \mathbb{P}_k\}$ and $U := -\rho_1$.

Proof. Write $r_i := \rho_{1, \mathbb{P}_i}(X_T^{\delta^\theta})$ and $r_{\max} := \max_{1 \leq i \leq k} r_i$, and let $i^* \in \arg \max_i r_i$. By Equation 5.9,

$$\rho_{2, \tilde{\mathbb{P}} \sim \mu} \left(-\rho_{1, \tilde{\mathbb{P}}} \left(X_T^{\delta^\theta} \right) \right) = \frac{1}{\gamma} \log \left(\sum_{i=1}^k \mu_i e^{\gamma r_i} \right). \quad (5.12)$$

Since $\mu_i \leq 1$ for all i and $\mu_{i^*} > 0$, we have the two-sided bound

$$\mu_{i^*} e^{\gamma r_{\max}} \leq \sum_{i=1}^k \mu_i e^{\gamma r_i} \leq \left(\sum_{i=1}^k \mu_i \right) e^{\gamma r_{\max}} = e^{\gamma r_{\max}}. \quad (5.13)$$

Taking logarithms and dividing by $\gamma > 0$,

$$r_{\max} + \frac{1}{\gamma} \log \mu_{i^*} \leq \frac{1}{\gamma} \log \left(\sum_{i=1}^k \mu_i e^{\gamma r_i} \right) \leq r_{\max}. \quad (5.14)$$

Since μ_{i^*} is a fixed positive constant independent of γ , we have $\frac{1}{\gamma} \log \mu_{i^*} \rightarrow 0$ as $\gamma \rightarrow \infty$, so both bounds converge to $r_{\max}$. By the squeeze theorem,

$$\lim_{\gamma \rightarrow \infty} \rho_{2, \tilde{\mathbb{P}} \sim \mu} \left(-\rho_{1, \tilde{\mathbb{P}}} \left(X_T^{\delta^\theta} \right) \right) = r_{\max} = \max_{1 \leq i \leq k} \rho_{1, \mathbb{P}_i} \left(X_T^{\delta^\theta} \right). \quad (5.15)$$

To connect this to Definition 5.1.1, recall that convex risk measures and utility functions use opposite monotonicity conventions: ρ_1 is decreasing in the quality of the position, while a utility function U is increasing. Setting $U := -\rho_1$ translates one convention into the other, giving

$$\max_{1 \leq i \leq k} \rho_{1, \mathbb{P}_i} \left(X_T^{\delta^\theta} \right) = - \min_{1 \leq i \leq k} \left(-\rho_{1, \mathbb{P}_i} \left(X_T^{\delta^\theta} \right) \right) = - \inf_{Q \in \{\mathbb{P}_1, \dots, \mathbb{P}_k\}} \mathbb{E}_Q \left[U \left(X_T^{\delta^\theta} \right) \right], \quad (5.16)$$

where the middle equality uses that the minimum over the finite set $\{\mathbb{P}_1, \dots, \mathbb{P}_k\}$ coincides with the infimum, and $\mathbb{E}_{\mathbb{P}_i}[U(X)] = -\rho_{1, \mathbb{P}_i}(X)$ since $\mathbb{P}_i$ is the (degenerate, cluster-conditional) measure under which ρ_1 is evaluated. Up to the overall sign convention, the right-hand side is exactly the maxmin criterion of Definition 5.1.1 with $\mathcal{Q} = \{\mathbb{P}_1, \dots, \mathbb{P}_k\}$. $\square$

In conclusion, Proposition 5.3.3 shows that γ and k act on different AADH characteristics of ambiguity aversion. The number of clusters k determines which *candidate models* $\{\mathbb{P}_1, \dots, \mathbb{P}_k\}$ are available to be robustified against, while γ determines *how aggressively* the outer measure penalises different risks across them. While $\gamma \rightarrow \infty$ recovers the classical, and overly conservative, worst-case robustification of Definition 5.1.1 (restricted to the k clusters), $\gamma \rightarrow 0$ recovers ambiguity neutrality with respect to μ (proof omitted).

5.4 Ambiguity-Averse Deep Hedgers in Practice

Before turning to the implementation, it is worth noting explicitly where the theoretical construction of Section 5.3 differs from the numerical experiments. Proposition 5.3.2, the result characterising the AADH objective in the limit $k = N_{train}$, relies on cash-invariance of the inner risk measure ρ_1 , a convex risk measure hypothesis (Definition 4.3.1). Neither this thesis nor Jones et al., 2025 implements the AADH with such a ρ_1 in practice. We therefore retain Proposition 5.3.2 as a result about the idealised construction, justifying the interpretation of k as interpolating between two well-defined limiting objectives. Proposition 5.3.1 and Proposition 5.3.3 are unaffected by this substitution, as neither uses cash-invariance of ρ_1 in their proof.

Therefore, we use the following inner loss function

$$\rho_1(X_T^{\delta^\theta}) := \text{Var}(X_T^{\delta^\theta, \text{before costs}}) + \lambda \mathbb{E}[\text{TC}_T(\delta^\theta)], \quad (5.17)$$

where $X_T^{\delta^\theta, \text{before costs}}$ denotes the terminal P&L before transaction costs, $\text{TC}_T(\delta^\theta)$ the total transaction costs of Equation 4.13, and $\lambda > 0$ a risk-aversion parameter.

Lemma 5.4.1. The functional ρ_1 of Equation 5.17 is not a convex risk measure in the sense of Definition 4.3.1.

Proof. It suffices to exhibit a violation of cash-invariance and of monotonicity.

Cash-invariance. For any $c \in \mathbb{R}$, since $\text{Var}(X_T^{\delta^\theta, \text{before costs}} + c) = \text{Var}(X_T^{\delta^\theta, \text{before costs}})$ and $\mathbb{E}[\text{TC}_T(\delta^\theta)]$ does not depend on the constant c , $\rho_1(X_T^{\delta^\theta} + c) = \rho_1(X_T^{\delta^\theta}) \neq \rho_1(X_T^{\delta^\theta}) - c$ unless $c = 0$.

Monotonicity. Take $\lambda = 0$ for simplicity and let $X_1 \geq 0$ be a non-degenerate (i.e. non-almost-surely-constant) random variable with $\text{Var}(X_1) > 0$, and set $X_2 := 2X_1$. Then $X_1 \leq X_2$ pointwise, yet $\rho_1(X_2) = \text{Var}(2X_1) = 4\text{Var}(X_1) > \text{Var}(X_1) = \rho_1(X_1)$, contradicting the required decreasing monotonicity $X_1 \leq X_2 \Rightarrow \rho_1(X_1) \geq \rho_1(X_2)$. $\square$

Remark 5.4.1. Beyond violating cash-invariance and monotonicity, ρ_1 as defined in Equation 5.17 is, strictly speaking, not even a map $\mathcal{X} \rightarrow \mathbb{R}$ in the sense required by Definition 4.3.1: it depends on the strategy δ^θ through the separate decomposition of the terminal wealth into P&L before costs and transaction costs, rather than through the terminal wealth $X_T^{\delta^\theta}$ alone.

Finally, this departure from Definition 4.3.1 is a deliberate modelling choice. As shown by Horikawa and Nakagawa, 2024, training a deep hedger to minimise a convex risk measure can incentivise the network to learn a statistical arbitrage strategy alongside the hedge itself, exploiting any residual drift in the training measure $\mathbb{P}$. This problem is amplified in the ambiguity-averse setting: individual feature clusters $\mathcal{C}_i$ (or market regimes, in financial interpretation), being much smaller and more homogeneous subsets of the training data than the full set $\mathcal{D}$, are considerably more susceptible to such arbitrage than $\mathcal{D}$ itself. Equation 5.17 avoids this failure mode.

5.5 The Ambiguity-Averse Deep Hedging Algorithm

We now summarise the training procedure for the ambiguity-averse deep hedger (AADH), adapting Algorithm 1 to the nested objective of Equation 5.8. Feature clustering (Algorithm 2) is run once, prior to training, to obtain the partition $\{\mathcal{C}_i\}_{i=1}^k$ and empirical probabilities $\{\mu_i\}_{i=1}^k$ of Equation 5.5. During training, each minibatch is stratified across clusters, so that a batch of size B contains approximately $B\mu_i$ paths from cluster $\mathcal{C}_i$, for every $i = 1, \dots, k$. Within each cluster, the network computes hedging positions and terminal P&L exactly as in Algorithm 1; the inner loss ρ_1 of Equation 5.17 is then evaluated per cluster, and the k resulting values are aggregated by the outer entropic risk measure ρ_2 of Equation 5.9. The full procedure is summarised in Algorithm 3.

Algorithm 3 Ambiguity-Averse Deep Hedging Training Algorithm

Require: N_{train} , N_{epochs} , minibatch size B , trading dates $0 = t_0 < \dots < t_n = T$, P_0 , payoff Z , market model, network f_θ , clusters k , ambiguity parameter $\gamma > 0$, risk-aversion $\lambda > 0$

Ensure: Trained network f_{θ^*}

```
1: Initialize  $\theta$  randomly
2: Generate paths  $\{(S_k^{(j)})_{k=0}^n\}_{j=1}^{N_{\text{train}}}$  and payoffs  $Z^{(j)}$ 
3:  $\{\mathcal{C}_i, \mu_i\}_{i=1}^k \leftarrow \text{FEATURECLUSTERING (Alg. 2)}$ 
4: for  $e = 1, \dots, N_{\text{epochs}}$  do
5:   Form stratified minibatches:  $\mathcal{B}$  has  $B_i \approx B\mu_i$  paths from  $\mathcal{C}_i$ 
6:   for each minibatch  $\mathcal{B}$  do
7:     for  $i = 1, \dots, k$  do
8:       for each  $j \in \mathcal{B} \cap \mathcal{C}_i$  do
9:          $\delta_{-1}^{(j)}, G^{(j)}, C^{(j)} \leftarrow 0$ 
10:      end for
11:      for  $k' = 0, \dots, n-1$  do
12:        for each  $j \in \mathcal{B} \cap \mathcal{C}_i$  do
13:           $\delta_{k'}^{(j)} \leftarrow f_\theta(T - t_{k'}, S_{k'}^{(j)}, \delta_{k'-1}^{(j)})$ 
14:           $G^{(j)} \leftarrow G^{(j)} + \delta_{k'}^{(j)} (S_{k'+1}^{(j)} - S_{k'}^{(j)})$ 
15:           $C^{(j)} \leftarrow C^{(j)} + c_{k'} S_{k'}^{(j)} |\delta_{k'}^{(j)} - \delta_{k'-1}^{(j)}|$ 
16:        end for
17:      end for
18:      for each  $j \in \mathcal{B} \cap \mathcal{C}_i$  do
19:         $\delta_n^{(j)} \leftarrow 0; \quad X^{(j)} \leftarrow -Z^{(j)} + P_0 + G^{(j)}$ 
20:      end for
21:       $r_i(\theta) \leftarrow \text{Var}_{j \in \mathcal{B} \cap \mathcal{C}_i}(X^{(j)}) + \lambda \text{Mean}_{j \in \mathcal{B} \cap \mathcal{C}_i}(C^{(j)})$ 
22:    end for
23:     $r_{\max} \leftarrow \max_i r_i(\theta)$ 
24:     $\hat{\rho}_{2,\mathcal{B}}(\theta) \leftarrow r_{\max} + \frac{1}{\gamma} \log \left( \sum_{i=1}^k \mu_i e^{\gamma(r_i(\theta) - r_{\max})} \right)$ 
25:     $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta \hat{\rho}_{2,\mathcal{B}}(\theta))$ 
26:  end for
27: end for
28: return  $f_{\theta^*}$ 
```

Remark 5.5.1. As in Algorithm 1, we set $P_0 = 0$. Two features distinguish Algorithm 3 from the standard deep hedging procedure. First, the time-step loop is repeated once per cluster within each minibatch, so that the network sees k disjoint sub-batches per gradient step rather than one. Second, minibatches must be strat-

ified across clusters: since ρ_1 requires an empirical variance and mean *within* each cluster, a cluster absent from a minibatch (or represented by too few paths) yields a degenerate or overly noisy estimate of $r_i(\theta)$, and, more subtly, an unstratified batch that omits cluster i altogether would require renormalising $\{\mu_i\}$ over the clusters actually present for the aggregation step to remain a valid probability-weighted risk measure. Neither of these issues arises for the standard deep hedger of Algorithm 1, since its objective is evaluated once over the whole minibatch. Beyond this training-time overhead and the one-off cost of Algorithm 2, the AADH requires no additional training data, no auxiliary market generator, and no higher option premium than the SDH, in contrast to the DRO-based robustification approaches discussed in Section 5.2.

Chapter 6

Data Generation and Distributional Shifts

6.1 Synthetic Distributional Shifts

The ambiguity-averse construction of Chapter 5 addresses model uncertainty within a single training measure $\mathbb{P}$: feature clustering partitions a given training set into candidate regimes $\{\mathbb{P}_i\}_{i=1}^k$, all generated by the same underlying market model. It does not, however, address what happens when the measure realised at test time differs structurally from the measure used for training.

The central motivation for studying this question comes from a pattern already visible in the numerical results of Jones et al. (2025): when the AADH is trained and evaluated on paths from the same generator, it consistently underperforms the standard deep hedger, whereas on a range of out-of-sample test sets (real equity indices, the COVID period, an alternative volatility model) it consistently outperforms it. This pattern is documented only through a handful of fixed, externally given test sets, each at an unknown and uncontrolled distance from the training distribution. It leaves open the more basic question this chapter is designed to answer: as the test distribution is moved continuously away from the training distribution, at what point, and how quickly, does the AADH's advantage over the SDH appear?

Rather than comparing a synthetic training measure against a handful of fixed, externally given test sets, we construct a single family of probability measures $\{\mathbb{P}_\xi\}_{\xi \in [0,1]}$ on path space, parametrised by a scalar ξ that controls the distance between any two members of the family. Training and testing under different values of ξ then lets us trace the gap between the SDH and the AADH as an explicit function of the distributional distance separating the training and test regimes, rather than as

an isolated, uncontrolled comparison on a fixed pair of datasets. We formalise this comparison in Section 6.3, after constructing the mixture family itself.

6.1.1 The Two Generators

We use two structurally different stochastic models (Cont, 2001) for the underlying asset prices we use to train the deep hedgers. We use a Heston model since it is the baseline of many deep hedging applications and one of the most widely used and flexible models in mathematical finance. The second one is a GARCH model, chosen for its capacity to capture volatility clustering and autocorrelation.

Heston. The Heston model (Heston, 1993) is a continuous-time stochastic volatility model in which the asset price S_t and its instantaneous variance v_t follow

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S, \quad (6.1)$$

$$dv_t = \kappa(\theta - v_t) dt + \sigma_v \sqrt{v_t} dW_t^v, \quad (6.2)$$

where v_t follows a mean-reverting CIR-type process with long-run level θ^1 , mean-reversion speed κ , and volatility-of-volatility σ_v , and the driving Brownian motions are correlated

$$dW_t^S dW_t^v = \rho dt. \quad (6.3)$$

In our implementation, paths are simulated by an Euler-Maruyama discretisation of Equations 6.1, 6.2 and 6.3. Moreover, since the discretised CIR process can become negative even though the continuous-time process almost surely does not (under the Feller condition), we reflect v_t if it falls below zero at each time step (Gatheral, 2011; Lord et al., 2010).

GARCH(1,1). The GARCH(1,1) model (Bollerslev, 1986), extending the ARCH model of Engle (1982), is instead a discrete-time model for volatility clustering. Following Section 4.1, we fix the discretisation $0 = t_0 < \dots < t_n = T$, with $\Delta t := T/n$. For $j = 1, \dots, n$, let h_j denote the conditional variance of the log-return over $(t_{j-1}, t_j]$, i.e. an $\mathcal{F}_{t_{j-1}}$ -measurable quantity, and let

$$\varepsilon_j := \sqrt{h_j} z_j, \quad z_j \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1), \quad (6.4)$$

¹Note that θ , in this setting, does not represent the vector of parameters of the neural network.

denote the shock realised over that same interval. The GARCH(1,1) recursion is

$$h_j = \omega_{\Delta t} + \alpha \varepsilon_{j-1}^2 + \beta h_{j-1}, \quad j = 2, \dots, n, \quad (6.5)$$

with $\alpha, \beta \geq 0$ and $\alpha + \beta < 1$ imposed for covariance-stationarity, and log-prices evolve as

$$\log S_{t_j} = \log S_{t_{j-1}} + \mu \Delta t - \frac{1}{2} h_j + \varepsilon_j, \quad j = 1, \dots, n. \quad (6.6)$$

The intercept follows from the long-run variance σ_{LR}^2 : taking expectations in Equation 6.5 under stationarity gives

$$\omega_{\Delta t} = (1 - \alpha - \beta) \sigma_{\text{LR}}^2. \quad (6.7)$$

In Figure 6.1, the prices generated by the GARCH(1,1) model described above.

Both generators return simulated paths as (time-step $\times$ path) matrices, with the same orientation.

6.1.2 The Two Extremal Measures

Before constructing the mixture family itself, we make precise what it means to generate paths using only one of the two market models. These two measures serve as the endpoints of the continuum we construct in Section 6.1.3, and every mixture $\mathbb{P}_\xi$ used for training or testing in this thesis is built by interpolating between them.

Fix a time horizon T , trading dates $0 = t_0 < \dots < t_n = T$ as in Section 4.1, and consider the path space $\mathcal{S} := \mathbb{R}^{n+1}$ (again we take $d = 1$). Each of the two generators of Section 6.1.1 induces a probability measure on the measurable space $(\mathcal{S}, \mathcal{B}(\mathcal{S}))$. We denote these two *extremal* measures by

$$\mathbb{P}_\text{H} := \text{law of } (S_{t_0}, \dots, S_{t_n}) \text{ under the discretised Heston model}, \quad (6.8)$$

$$\mathbb{P}_\text{G} := \text{law of } (S_{t_0}, \dots, S_{t_n}) \text{ under the GARCH(1,1)}. \quad (6.9)$$

We want to remark a crucial difference of origin: the $\mathbb{P}_i$ of Chapter 5 are created by clustering a single, already-generated training set. $\mathbb{P}_\text{H}$ and $\mathbb{P}_\text{G}$ are *exogenous*, specified in advance by choosing two structurally different market models.

6.1.3 The Mixture Family

Definition 6.1.1 (Distributional-Shift Mixture). For $\xi \in [0, 1]$, the *mixture measure* $\mathbb{P}_\xi$ on $(\mathcal{S}, \mathcal{B}(\mathcal{S}))$ is defined by

$$\mathbb{P}_\xi := \xi \mathbb{P}_\text{H} + (1 - \xi) \mathbb{P}_\text{G}. \quad (6.10)$$

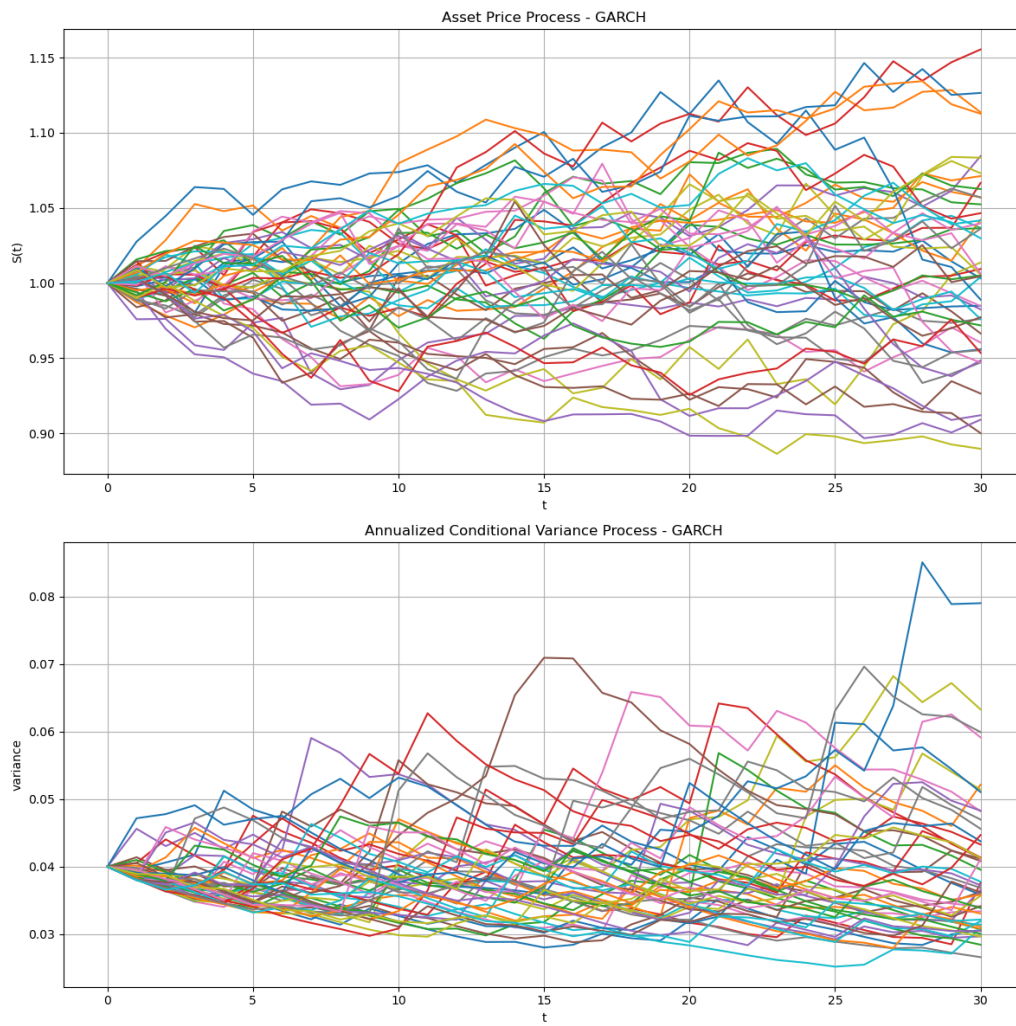


Figure 6.1: Simulated asset price paths generated using the GARCH(1,1) model.

We refer to ξ as the *Heston weight*. Equivalently, $\mathbb{P}_\xi$ is the law of $Y := B Y_H + (1 - B) Y_G$, where $B \sim \text{Bernoulli}(\xi)$, $Y_H \sim \mathbb{P}_H$, $Y_G \sim \mathbb{P}_G$ are mutually independent.

In other words, $\{\mathbb{P}_\xi\}_{\xi \in [0,1]}$ traces a continuum of probability measures on path space connecting them and the endpoints recover the two pure regimes, $\mathbb{P}_0 = \mathbb{P}_G$ and $\mathbb{P}_1 = \mathbb{P}_H$.

In the following two results we want to make precise in what sense this continuum is well-behaved, and how it interacts with the risk functionals of Chapter 5. We take inspiration from Huber, 1992 to build the following Lemma.

Lemma 6.1.1 (Total-Variation Lipschitz Continuity). For $\xi, \xi' \in [0, 1]$,

$$d_{\text{TV}}(\mathbb{P}_\xi, \mathbb{P}_{\xi'}) = |\xi - \xi'| d_{\text{TV}}(\mathbb{P}_H, \mathbb{P}_G), \quad (6.11)$$

where $d_{\text{TV}}(P, Q) := \sup_{A \in \mathcal{B}(\mathcal{S})} |P(A) - Q(A)|$.

Proof. For any $A \in \mathcal{B}(\mathcal{S})$,

$$\mathbb{P}_\xi(A) - \mathbb{P}_{\xi'}(A) = (\xi - \xi') \mathbb{P}_H(A) + (\xi' - \xi) \mathbb{P}_G(A) = (\xi - \xi') (\mathbb{P}_H(A) - \mathbb{P}_G(A)). \quad (6.12)$$

Taking the supremum over A gives $d_{\text{TV}}(\mathbb{P}_\xi, \mathbb{P}_{\xi'}) = |\xi - \xi'| \sup_A |\mathbb{P}_H(A) - \mathbb{P}_G(A)| = |\xi - \xi'| d_{\text{TV}}(\mathbb{P}_H, \mathbb{P}_G)$. $\square$

Lemma 6.1.1 shows that the total-variation distance between two members of the family scales exactly linearly with the distance between their parameters: doubling $|\xi - \xi'|$ doubles $d_{\text{TV}}(\mathbb{P}_\xi, \mathbb{P}_{\xi'})$. In other words, the map $\xi \mapsto \mathbb{P}_\xi$ is Lipschitz continuous: equal steps in ξ always produce equal steps in total-variation distance, wherever they are taken in $[0, 1]$. This is what makes $|\xi_{\text{train}} - \xi_{\text{test}}|$ a rigorous measure of the magnitude of the distributional shift between training and test. This is what motivates the section: rather than some arbitrary or uncontrolled notion of "how different two datasets look," we have constructed a rigorous way of comparing the two.

Proposition 6.1.1 (Mixture Moment Decomposition). *Let $f : \mathcal{S} \rightarrow \mathbb{R}$ be any functional with finite second moment under both $\mathbb{P}_H$ and $\mathbb{P}_G$, and write $m_H := \mathbb{E}_{\mathbb{P}_H}[f]$, $m_G := \mathbb{E}_{\mathbb{P}_G}[f]$, $v_H := \text{Var}_{\mathbb{P}_H}(f)$, $v_G := \text{Var}_{\mathbb{P}_G}(f)$. Then, under $\mathbb{P}_\xi$,*

$$\mathbb{E}_{\mathbb{P}_\xi}[f] = \xi m_H + (1 - \xi) m_G, \quad (6.13)$$

$$\text{Var}_{\mathbb{P}_\xi}(f) = \xi v_H + (1 - \xi) v_G + \xi(1 - \xi) (m_H - m_G)^2. \quad (6.14)$$

Proof. Using the two-stage representation of Definition 6.1.1, condition on the regime indicator B . Equation 6.13 is immediate from the law of total expectation: $\mathbb{E}[f] = \mathbb{E}[\mathbb{E}[f \mid B]] = \xi m_H + (1 - \xi) m_G$. For the variance, the law of total variance gives $\text{Var}(f) = \mathbb{E}[\text{Var}(f \mid B)] + \text{Var}(\mathbb{E}[f \mid B])$. The first term is $\xi v_H + (1 - \xi) v_G$. For the second, $\mathbb{E}[f \mid B]$ takes value m_H with probability ξ and m_G with probability $1 - \xi$, so it is itself a two-point random variable with variance $\xi(1 - \xi)(m_H - m_G)^2$, by the standard formula for the variance of a Bernoulli-mixed two-point distribution. $\square$

It is interesting to notice that the variance under the mixture is not simply the convex combination $\xi v_H + (1 - \xi) v_G$ of the two within-regime variances: there is an additional, strictly positive term $\xi(1 - \xi)(m_H - m_G)^2$ whenever the two regimes disagree on the mean of f . This extra term vanishes at the endpoints $\xi \in \{0, 1\}$ (pure regimes) and is maximised at $\xi = 1/2$ where regime ambiguity is greatest. Second, since the inner loss ρ_1 of Equation 5.17 is itself of the form $\text{Var}(\cdot) + \lambda \mathbb{E}[\cdot]$, Proposition 6.1.1 applies directly to it, and gives, for a fixed trained strategy δ^{θ^*} , a closed-form prediction for how its risk should vary as a function of the test-time mixture weight.

We can now formalize the question of this thesis. For a hedging model $M \in \{\text{SDH}, \text{AADH}\}$ trained under ξ_{train} and evaluated under ξ_{test} , denote its out-of-distribution risk by $R^M(\xi_{\text{train}}, \xi_{\text{test}})$ (formalised in Section 6.3), and define the *ambiguity-aversion gain*

$$\Delta(\xi_{\text{train}}, \xi_{\text{test}}) := R^{\text{SDH}}(\xi_{\text{train}}, \xi_{\text{test}}) - R^{\text{AADH}}(\xi_{\text{train}}, \xi_{\text{test}}). \quad (6.15)$$

What Jones et al. (2025) reports corresponds to our $\Delta(\xi, \xi) \lesssim 0$ along the diagonal, and $\Delta(\xi_{\text{train}}, \xi_{\text{test}}) > 0$ for at least some pairs with $\xi_{\text{train}} \neq \xi_{\text{test}}$. In Chapter 7 we discuss their claim. To summarize what we did in this Subsection, Lemma 6.1.1 makes $|\xi_{\text{train}} - \xi_{\text{test}}|$ a rigorous proxy for distributional distance, and Proposition 6.1.1 isolates the part of any risk change that is attributable purely to the mixture structure itself, rather than to the network’s own generalisation behaviour. Together, they let us ask, in a controlled and quantitative way, exactly how Δ depends on the distance between training and test regimes. We investigate this in Chapters 6 and 7.

6.2 Data Generation

Definition 6.1.1 specifies $\mathbb{P}_\xi$ theoretically. However, in practice, we have finite samples. Given a target number of paths N and a mixture weight $\xi \in [0, 1]$, we construct an empirical dataset approximating $\mathbb{P}_\xi$ as follows.

1. Set $n_H := \text{round}(\xi N)$ and $n_G := N - n_H$.
2. Simulate n_H independent paths from the Heston generator and n_G independent paths from the GARCH(1,1) generator, as described in Section 6.1.1.
3. Concatenate the two batches of paths and shuffle the paths to get a set of N paths.

We denote the resulting empirical measure by

$$\widehat{\mathbb{P}}_\xi^{(N)} := \frac{n_H}{N} \widehat{\mathbb{P}}_H^{(n_H)} + \frac{n_G}{N} \widehat{\mathbb{P}}_G^{(n_G)}, \quad (6.16)$$

where $\widehat{\mathbb{P}}_H^{(n_H)}$ and $\widehat{\mathbb{P}}_G^{(n_G)}$ are the empirical measures of the two sub-samples. Using the strong law of large numbers for each generator we have that $\widehat{\mathbb{P}}_\xi^{(N)} \rightarrow \mathbb{P}_\xi$ weakly as $N \rightarrow \infty$ (with $n_H/N \rightarrow \xi$), so that $\widehat{\mathbb{P}}_\xi^{(N)}$ is a consistent estimator (i.e. the estimator converges weakly to the true distribution almost surely) of the theoretical mixture of Definition 6.1.1.

Remark 6.2.1. The random shuffle of paths in step 3 does not affect the empirical measure itself. Its purpose is purely practical since we ensure that mini-batches drawn sequentially from the dataset are themselves approximately balanced across the two generators. This is particularly important for training the algorithms. Secondly, both the training dataset and every test dataset used in this thesis are generated independently by the procedure above, allowing the training mixture weight ξ_{train} and any test mixture weight ξ_{test} to be chosen independently of one another.

6.3 All Together: The Distributional-Shift Experiment

We now put together Sections 6.1 and 6.2 into the experiment we use in Chapter 7.

Definition 6.3.1 (Training and Testing under a Mixture). Fix a hedging model (SDH or AADH), a training mixture weight $\xi_{\text{train}} \in [0, 1]$, and a training sample size N_{train} . Let $\theta^*(\xi_{\text{train}})$ denote the (approximately) optimal network parameters obtained by training the model, via Algorithm 1 or Algorithm 3 as appropriate, on data generated from $\widehat{\mathbb{P}}_{\xi_{\text{train}}}^{(N_{\text{train}})}$. For a test mixture weight $\xi_{\text{test}} \in [0, 1]$ and an evaluation risk functional ρ^{eval} , we define the *out-of-distribution risk*

$$R(\xi_{\text{train}}, \xi_{\text{test}}) := \rho^{\text{eval}}\left(X_T^{\delta^{\theta^*(\xi_{\text{train}})}}; \widehat{\mathbb{P}}_{\xi_{\text{test}}}^{(N_{\text{test}})}\right), \quad (6.17)$$

i.e. the risk, measured by ρ^{eval} , of the terminal P&L generated by the strategy trained under ξ_{train} , when evaluated on paths sampled from $\widehat{\mathbb{P}}_{\xi_{\text{test}}}^{(N_{\text{test}})}$.

Evaluating $R(\xi_{\text{train}}, \xi_{\text{test}})$ over a grid of pairs $(\xi_{\text{train}}, \xi_{\text{test}}) \in [0, 1]^2$ will produce a risk surface we will explain in more detail in Chapter 7. In practice, its diagonal $R(\xi, \xi)$ measures in-distribution performance, while the off-diagonal entries measure the difference induced by the mismatch between training and test conditions (with $|\xi_{\text{train}} - \xi_{\text{test}}|$ controlling the distance of Lemma 6.1.1).

Proposition 6.1.1 yields an explicit prediction for one slice of this surface. Fix ξ_{train} and the resulting trained strategy $\delta^{\theta^*}(\xi_{\text{train}})$, and take $\rho^{\text{eval}} = \text{Var}$, applied to the P&L before costs generated by this fixed strategy. Writing m_H, m_G, v_H, v_G for the mean and variance of this (fixed) strategy's before-cost P&L under the two pure regimes, Equation 6.14 gives

$$R(\xi_{\text{train}}, \xi_{\text{test}}) = \xi_{\text{test}} v_H + (1 - \xi_{\text{test}}) v_G + \xi_{\text{test}}(1 - \xi_{\text{test}}) (m_H - m_G)^2, \quad (6.18)$$

a quadratic function of ξ_{test} alone, for each fixed ξ_{train} . This gives us a model-free baseline that isolate possible model generalization failure.

Chapter 7

Implementation of the Experiment and Results

7.1 Experiment Setup

In this section, we explain the full pipeline in practical terms that we used to construct the deep hedgers and implement the full synthetic distributional-shift experiment.

7.1.1 Data Configuration

We start from the data generation. Following Subsection 6.1.1, we used for the Heston model the parameters reported in Jones et al. (2025), in order to remain as close as possible to the training environment used in that study and to faithfully reproduce its baseline Heston setting. For the GARCH(1,1) model, we instead chose parameters designed to produce realistic stock-index-like return dynamics, in particular its autocorrelation dynamics. Note that the assumption of GARCH model representing a stock-index is particularly relevant.

For both models, we set the initial asset price and strike price to $S_0 = K = 1$, and considered a maturity of $T = \frac{30}{252}$, corresponding to 30 trading days under the convention of 252 trading days per year. We discretised the paths using $N = 30$ time steps, so that the discretisation corresponds to one trading day per step. The risk-free rate and the drift parameter were both set to zero. The resulting Heston

parameters were

$$\begin{aligned} v_0 &= 0.04, & \kappa &= 2.0, & \theta &= 0.04, \\ \sigma_v &= 3.5, & \rho &= -0.7, & r &= 0, \\ \mu &= 0. \end{aligned} \tag{7.1}$$

In Figure 7.1 we show the Heston price and volatility processes, using the parameters outlined above.

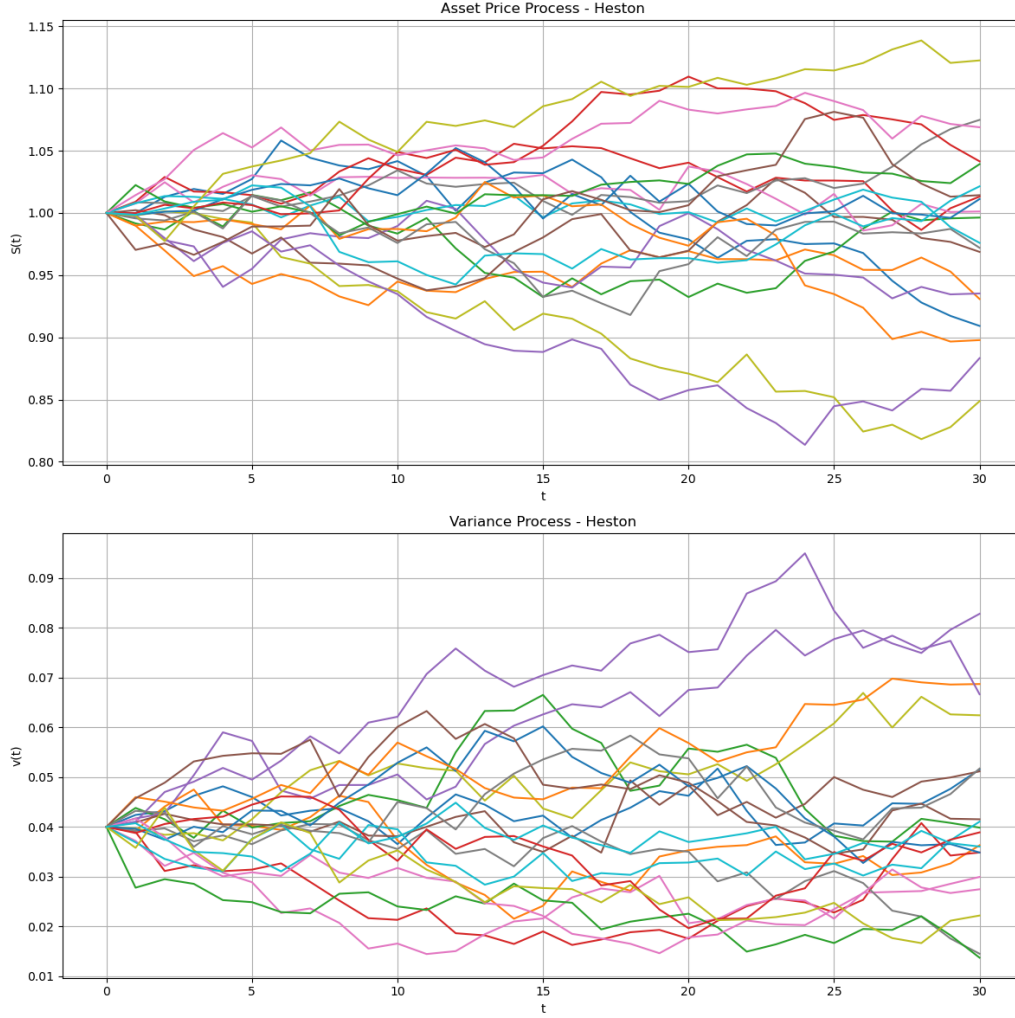


Figure 7.1: Simulated asset price paths and volatility paths generated using the Heston model.

For the GARCH(1,1) model, we used

$$\begin{aligned}\alpha &= 0.08, & \beta &= 0.90, & h_0 &= 0.04, \\ \sigma_{\text{LR}}^2 &= 0.04, & \mu &= 0.\end{aligned}\tag{7.2}$$

Here, σ_{LR}^2 denotes the unconditional long-run variance and serves to calculate ω , while h_0 the initial conditional variance. The GARCH parameters were chosen with the objective of generating paths that are close to those of a broad stock index such as the S&P 500. For example, the choice $\alpha + \beta = 0.98$ produces highly persistent conditional volatility, which is consistent with the volatility clustering typically observed in equity-index returns.

The drift was set to zero in both generators. This choice does not affect the interpretation of the distributional-shift experiment, since the purpose of the two generators is to create structurally different volatility and return dynamics.

For each experiment and parameter configuration, we generated $N_{\text{sim}} = 100,000$ independent paths, with a Train, Validation, Test split of respectively 60%, 20% and 20% of the paths. The same random seed, $\text{seed} = 1$, was used throughout the data-generation procedure to ensure reproducibility of the simulations.

Before training the hedging networks, we preprocess the generated training paths according to the feature-clustering procedure described in Section 5.2.1. We used $k = 4$ clusters and autocorrelation, maximum drawdown and volatility as features (Figure 7.2). Importantly, the clustering is performed on the realised characteristics of the training paths and is therefore endogenous to the training dataset: the four regimes are not assigned according to whether a path was generated by Heston or GARCH. This distinction is particularly important for the mixture experiment.

7.1.2 Training Configuration

Both the standard deep hedger (SDH) and the ambiguity-averse deep hedger (AADH) are trained using the same basic optimisation and network configuration. This ensures that differences in performance are attributable to the objectives of the two methods rather than to different network architectures or optimisation techniques.

The networks are trained using mini-batches of size $\text{batch size} = 1000$ for 200 epochs with Adam optimizer (Kingma & Ba, 2014). The learning rate is initially set to $\text{lr}_0 = 10^{-4}$, and is reduced

$$\eta = \begin{cases} 10^{-4}, & \text{for epochs } 0, \dots, 99, \\ 10^{-5}, & \text{for epochs } 100, \dots, 149, \\ 10^{-6}, & \text{for epochs } 150, \dots, 200. \end{cases}\tag{7.3}$$

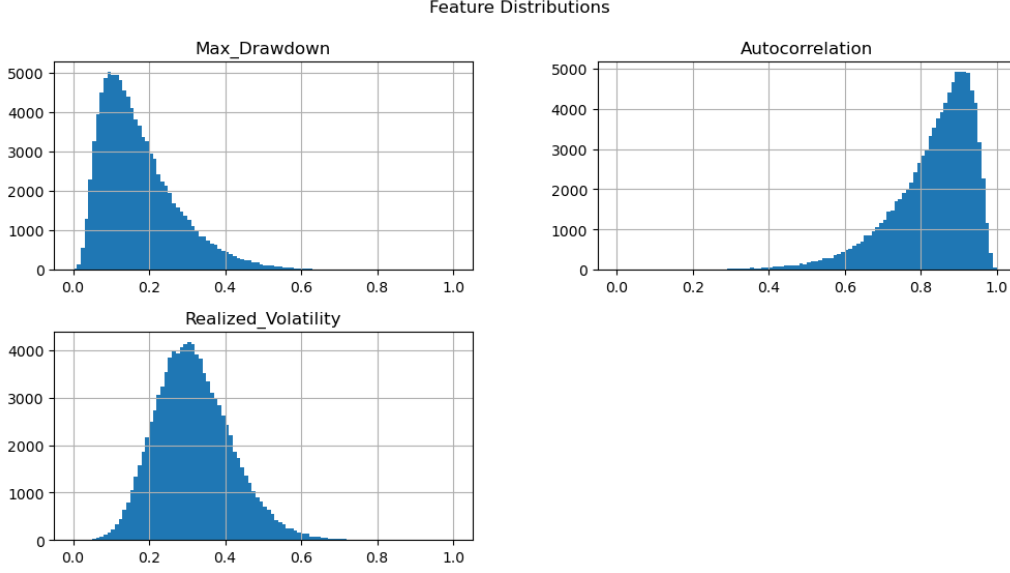


Figure 7.2: Equation 5.3 in practice. The paths are normalized following three different target features, which are then used in K-Means clustering (Algorithm 2).

The proportional transaction-cost rate (Equation 4.13) is set to $c = 0.002$. Thus, the transaction costs entering the hedging P&L (Equation 4.14) are proportional to the traded amount, with c representing a cost of 0.2% per unit of traded underlying. This is in line with real-world transaction costs. Furthermore, as payoff, we use an European Call Option (Equation 4.1) hedging a single asset ($d = 1$). However, the framework we presented in the previous chapters has been made deliberately as general as possible, so in future implementation, different payoffs or more assets can be used without the need of changing the underlying theory or algorithmic structure.

The loss parameters are fixed across the SDH and AADH experiments. In the notation of Chapter 5, the parameter governing the inner loss is $\lambda = \frac{1}{100}$, while the parameter governing the outer ambiguity-averse objective is $\gamma = 250$. The same values of λ and γ are used for all mixture weights in the synthetic distributional shifts and for both hedging models.

7.1.3 Deep-Hedging Network Architecture

Following the notation of Section 4.1, the hedging strategy is represented by a feed-forward neural network whose input at each trading date consists of the current asset price S_k , the previous hedge position δ_{k-1} , and the time to maturity $T - t_k$.

(Equation 4.20). Thus, with one underlying asset and three features per time step, the input vector lies in $\mathbb{R}^3$. The network contains three hidden layers, each with 100 neurons and a ReLU activation function (Equation 3.1.1). The final linear layer has one output, corresponding to the hedge position δ_t in the single underlying asset.

To enforce the economically admissible range of the hedge ratio for the European call considered in this thesis, the output layer is followed by a customized sharp sigmoid activation

$$f(x) = \frac{1}{1 + e^{-(x-x_0) \cdot s}} \quad (7.4)$$

centered at $x_0 = 0.5$ and scaled by $s = 10$. The activation maps the network output into the interval $[0, 1]$, so that the resulting hedge position is consistent with the usual delta range of a call option.

The resulting architecture is detailed in Table 7.1. The architecture and all hyperparameters are kept fixed for both the SDH and AADH.

Table 7.1: Deep-Hedging Feed-Forward Neural Network Architecture.

Layer	Description	Dimension	Activation
Input	$S_k, \delta_{k-1}, T - t_k$	3	—
Hidden 1	Dense / Fully Connected	100	ReLU
Hidden 2	Dense / Fully Connected	100	ReLU
Hidden 3	Dense / Fully Connected	100	ReLU
Output	Hedge Position (δ_k)	1	Sigmoid (Eq. 7.4)

In conclusion, the only difference between the two training procedures is therefore their objective: the SDH minimises the standard hedging loss, whereas the AADH incorporates the feature-clustered regimes through the ambiguity-averse outer objective described in Chapter 5. In this way, the experiment isolates the effect of ambiguity aversion making the results as comparable as possible.

7.2 Results

7.2.1 In Sample Results

We now present the main results obtained in our experiments. This first section uses the original setting proposed in Jones et al., 2025: both the SDH and the AADH are trained on Heston-generated paths and tested either on the same model or on

out-of-sample empirical paths (i.e. market indices). We also report some results on the AADH that are not shown in the original paper.

One of the most important metrics in deep hedging is the P&L diagram, which shows the distribution of the profits and losses generated by the trading strategy through Equation 4.14, evaluated on the Test set. Our results confirm the findings of Buehler et al., 2019, where the presence of transaction costs produces left-skewed distributions. In Figure 7.3 we can see the different shapes of the two distributions. The AADH distribution (in orange) presents a fatter left tail than the SDH distribution (in blue); its peak, however, is slightly shifted towards smaller losses. This is consistent with the theory, since the AADH, being ambiguity-averse, hedges less aggressively.

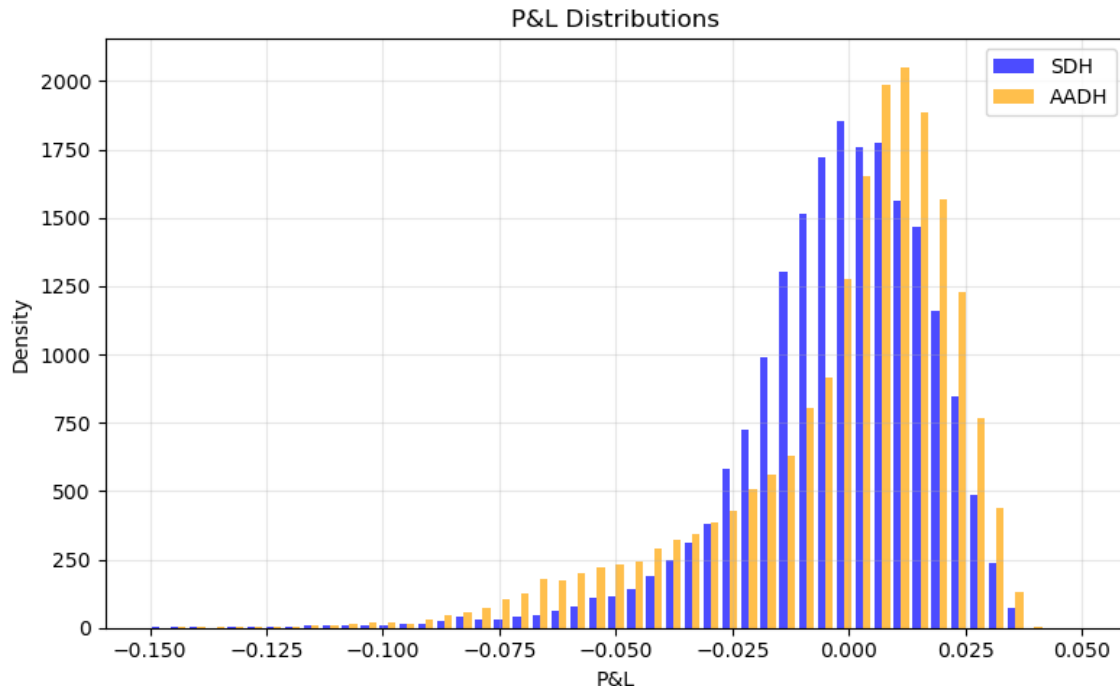


Figure 7.3: P&L Distribution.

Table 7.2 summarises the main statistics of the P&L distribution. The AADH exhibits a higher variance than the SDH, but a lower mean loss. We remark that in our setup $P_0 = 0$, so the option premium charged upfront should be added back to these figures before they are interpreted as economic losses.

	Mean	Variance
SDH	-0.00294	0.02108
AADH	-0.00241	0.02646

Table 7.2: P&L Summary Statistics using the Test set.

In Figure 7.4 we show the delta trading strategy together with the associated price paths. Visually, the strategy is well behaved: the hedge starts at approximately $\delta \approx 0.55$ at time $t = 0$, consistent with an at-the-money position far from maturity, and, as maturity approaches, $\delta \rightarrow 1$ for in-the-money calls and $\delta \rightarrow 0$ for out-of-the-money calls.

Figure 7.5 shows again that the AADH trades less aggressively, accounting for uncertainty. The left panel shows the two learned strategies at the last time step before maturity, while the right panel shows their difference. The AADH holds more of the underlying than the SDH when the call is out-of-the-money and less when it is in-the-money, so that its delta profile is flatter around the strike: the AADH therefore displays a lower gamma. Equivalently, the AADH behaves approximately as a Black-Scholes delta hedger with a higher implied volatility parameter, which is the same qualitative effect reported in Jones et al., 2025.

Finally, we test the trained models both on the synthetic Test set and on real-world out-of-sample data. Following Jones et al., 2025, we use ten years of daily open prices of a series of stock indices, from the 3rd of March 2014 to the 31st of October 2024. Test paths are obtained with a sliding window of the appropriate length, shifted one trading day at a time, and each path is normalised by its initial value so that $S_0 = K = 1$, consistently with the training setup.

Table 7.3 reports the results. Starting from the Test set (Heston), our numbers are very close to those of Jones et al., 2025: the SDH beats the AADH on every metric except transaction costs, where the AADH is consistently cheaper. Comparing the variances directly, we obtain 4.437×10^{-4} against their 4.824×10^{-4} for the SDH, and 6.925×10^{-4} against their 6.900×10^{-4} for the AADH. We focus on variance rather than on risk because, following Buehler et al., 2019, we do not charge transaction costs on the terminal liquidation, so our transaction-cost figures, and hence our risk figures, are lower than theirs by construction. However, since the transaction costs are scaled by $\lambda = 1/100$ risks are slightly lower but completely in line with the original ones.

According to Jones et al., 2025, out-of-sample data, or periods characterised by distributional shifts, should be precisely where the AADH outperforms the SDH:

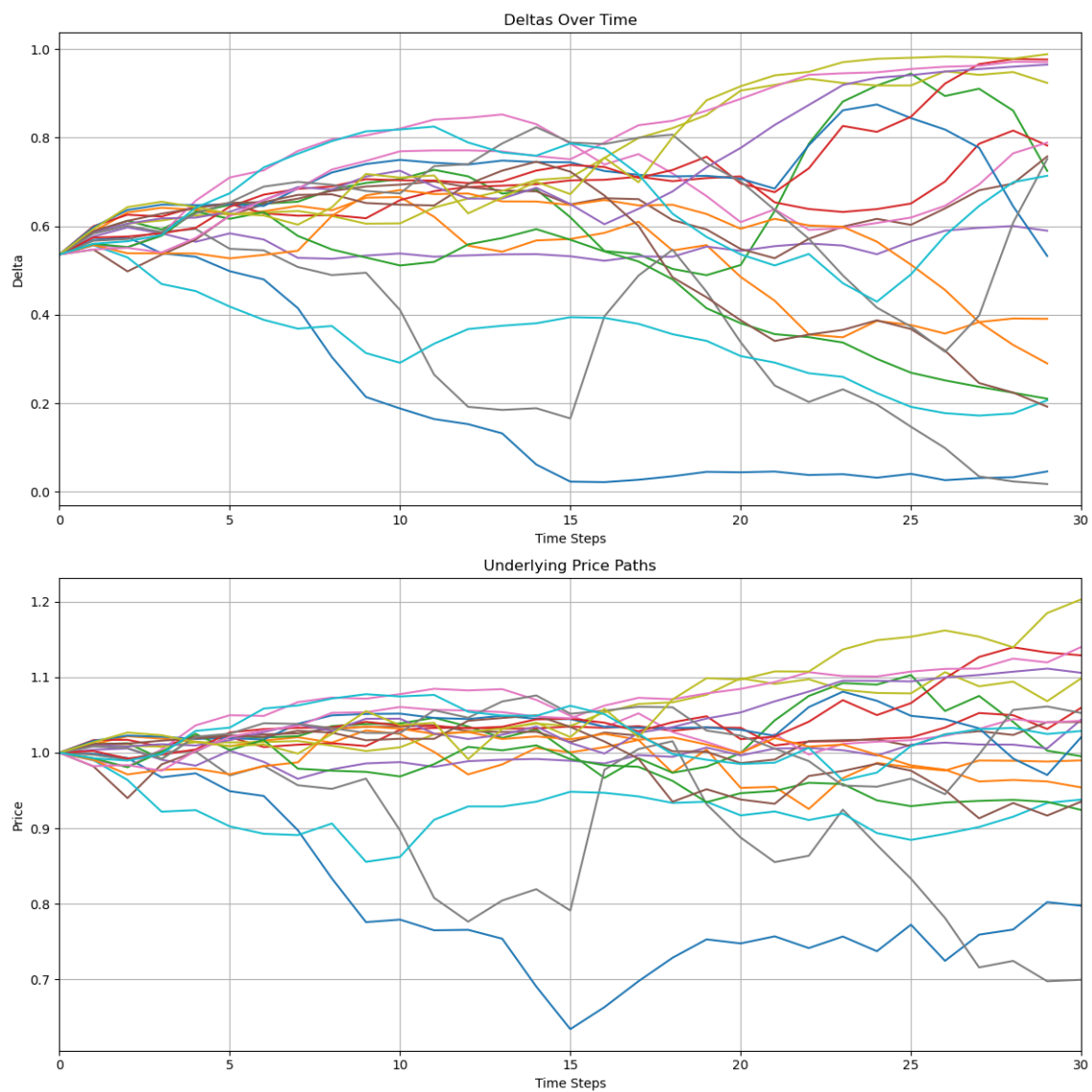


Figure 7.4: AADH Delta Trading Strategy with the associated price path.

Evaluation Metric	Evaluation data	SDH	AADH
Risk	Heston: test set	4.497×10^{-4}	7.028×10^{-4}
	10y SX5E	1.609×10^{-4}	2.146×10^{-4}
	10y S&P 500	1.409×10^{-4}	1.820×10^{-4}
	10y FTSE 100	1.401×10^{-4}	1.692×10^{-4}
	10y N225	1.639×10^{-4}	1.804×10^{-4}
	10y Nasdaq 100	1.870×10^{-4}	2.112×10^{-4}
Variance	Heston: test set	4.437×10^{-4}	6.925×10^{-4}
	10y SX5E	1.425×10^{-4}	1.977×10^{-4}
	10y S&P 500	1.245×10^{-4}	1.667×10^{-4}
	10y FTSE 100	1.249×10^{-4}	1.536×10^{-4}
	10y N225	1.430×10^{-4}	1.617×10^{-4}
	10y Nasdaq 100	1.685×10^{-4}	1.935×10^{-4}
Transaction costs	Heston: test set	3.136×10^{-3}	2.582×10^{-3}
	10y SX5E	2.362×10^{-3}	2.101×10^{-3}
	10y S&P 500	2.236×10^{-3}	2.019×10^{-3}
	10y FTSE 100	2.164×10^{-3}	1.965×10^{-3}
	10y N225	2.494×10^{-3}	2.190×10^{-3}
	10y Nasdaq 100	2.723×10^{-3}	2.293×10^{-3}

Table 7.3: Performance and transaction costs of the Heston-trained deep hedgers (SDH and AADH) on the Heston Test set and on ten years of daily open prices of five equity indices. *Risk* is the inner loss of Equation 5.17, *Variance* is the variance of the terminal P&L, and *Transaction costs* is the expected total cost incurred over a path. The lower value in each row is highlighted in bold.

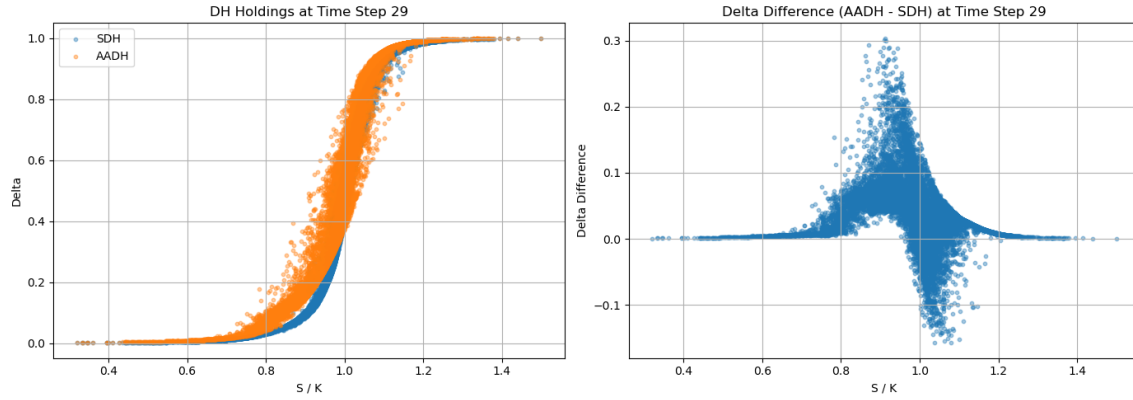


Figure 7.5: Learned delta trading strategy and difference at the last time step before maturity.

in their results the variance and the risk of the AADH are consistently lower than those of the SDH on real indices. Our results indicate the opposite. The SDH attains a lower variance and a lower risk than the AADH on all five indices, while our data do confirm the original finding that the AADH consistently spends less on transaction costs. We stress that Table 7.3 is obtained with the sliding window applied in chronological order; the ordering convention used to build the empirical test paths turns out to be decisive, and we return to this point in Section 7.3.

This discrepancy between our results and those of Jones et al., 2025 motivates the synthetic distributional-shift experiment of Chapter 6.1, in which the GARCH generator is taken to represent the dynamics of an equity index, so that the AADH’s behaviour relative to the SDH can be studied as a controlled function of the distance between the training and test measures.

Before moving on, we want to remark that we repeated the empirical comparison using closing instead of opening prices, changing architecture, extending the sample to more recent time periods, charging transaction costs also on the terminal liquidation, and varying the remaining setup choices of Section 7.1. In every one of these variants the SDH attained a lower risk and a lower variance than the AADH on real index data, while the AADH remained consistently cheaper in transaction costs. The discrepancy with Jones et al., 2025 is therefore stable across our implementation choices, which is what motivates the controlled study of the next section.

7.2.2 Controlled Distributional Shifts Results

Since we obtained results that differ from those of Jones et al., 2025, we now study the behaviour of the AADH under controlled synthetic distributional shifts, putting into practice the framework introduced in Section 6.3.

We are interested in determining whether there exist regions of the risk surface in which the AADH outperforms the SDH. We first let ξ vary on a grid with step 0.25, from 0 to 1. We therefore train five different models, with a smaller number of epochs than in the original setting because of the computational cost, and we test each of them on every mixture weight, following Equation 6.17. Figures 7.6 and 7.7 report the risk and variance surfaces, expressed as the difference between the figures of the SDH and those of the AADH. A negative entry therefore means that the SDH attains a lower risk, or a lower variance, than the AADH. The surfaces are negative at every point of the grid: the AADH never outperforms the SDH, neither in distribution nor under any of the shifts we consider. On the other hand, we confirm that the AADH always beats the SDH on transaction costs, which are consistently lower.

This finding is worth emphasising. We started from the assumption that the GARCH generator represents the dynamics of a stock index, so that for $\xi = 0$ (no Heston paths) we would have expected the AADH to outperform the SDH, in line with Jones et al., 2025. The synthetic experiment instead reproduces the pattern already observed on empirical data in Table 7.3, which supports our earlier results and raises questions about the out-of-sample conclusions of the original paper.

We now return to the full setup described in Section 7.1, rather than the reduced version adopted for Figures 7.6 and 7.7, and we examine the effect of the distributional shift on a model trained on Heston data only ($\xi_{train} = 1$). Since the surfaces display no region in which the AADH outperforms, our aim here is to inspect more closely the relation between test shift and performance, in the configuration that is closest to the one for which Jones et al., 2025 report an advantage for the AADH.

Table 7.4 reports this experiment: a model trained exclusively on Heston-generated paths ($\xi_{train} = 1$), tested on the synthetic mixtures $\mathbb{P}_{\xi_{test}}$. Figure 7.8 displays the same results as differences between the two hedgers. We again observe a strong similarity with the last row of Figures 7.6 and 7.7, so the shorter training used there does not alter the conclusions. The AADH fails to beat the SDH on risk and variance at every mixture weight, and beats it on transaction costs at every mixture weight. We remark that here we adopt a finer partition of ξ close to the GARCH end of the family, since those are the mixtures that should resemble equity-index data most closely.

In Figure 7.9 we test the prediction of Equation 6.18 of Section 6.3. The gap between the realised variance and the closed-form quadratic in ξ_{test} is essentially

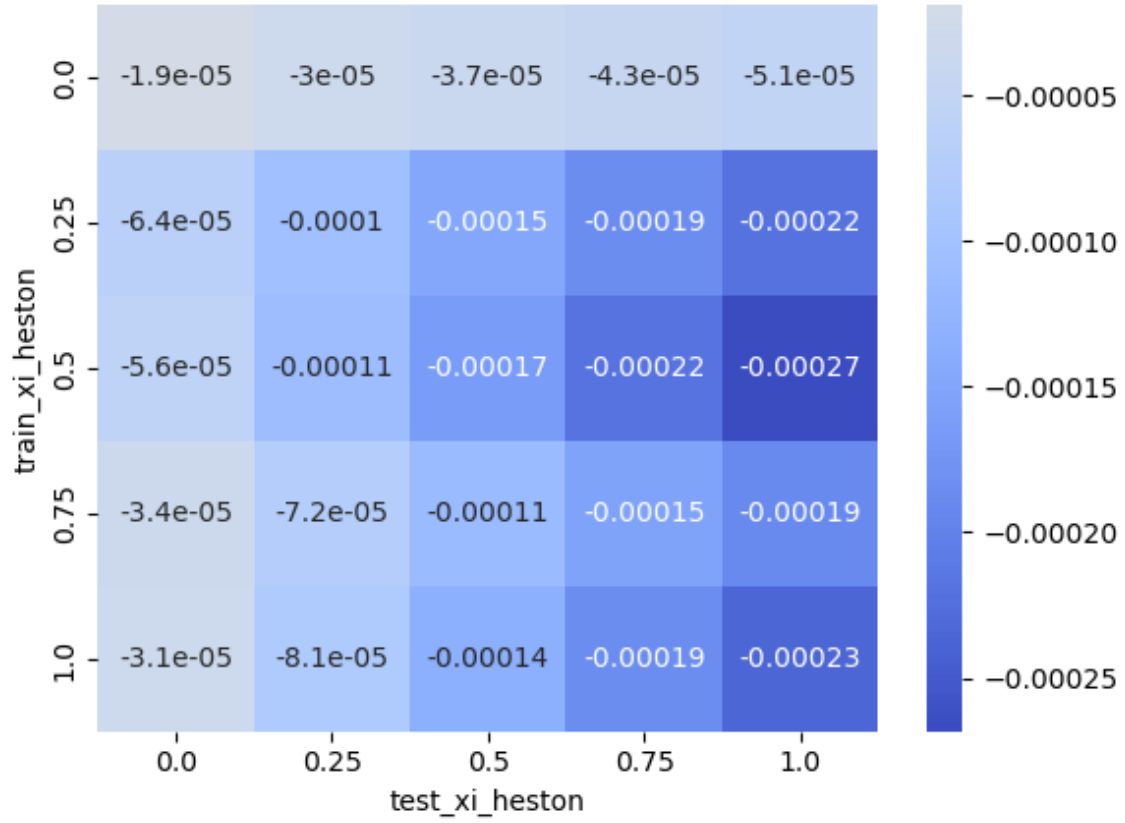


Figure 7.6: Risk surface with controlled distributional shifts. Each entry is the risk of the SDH minus the risk of the AADH: a positive entry means that the AADH has the lower risk, a negative entry that the SDH has the lower risk.

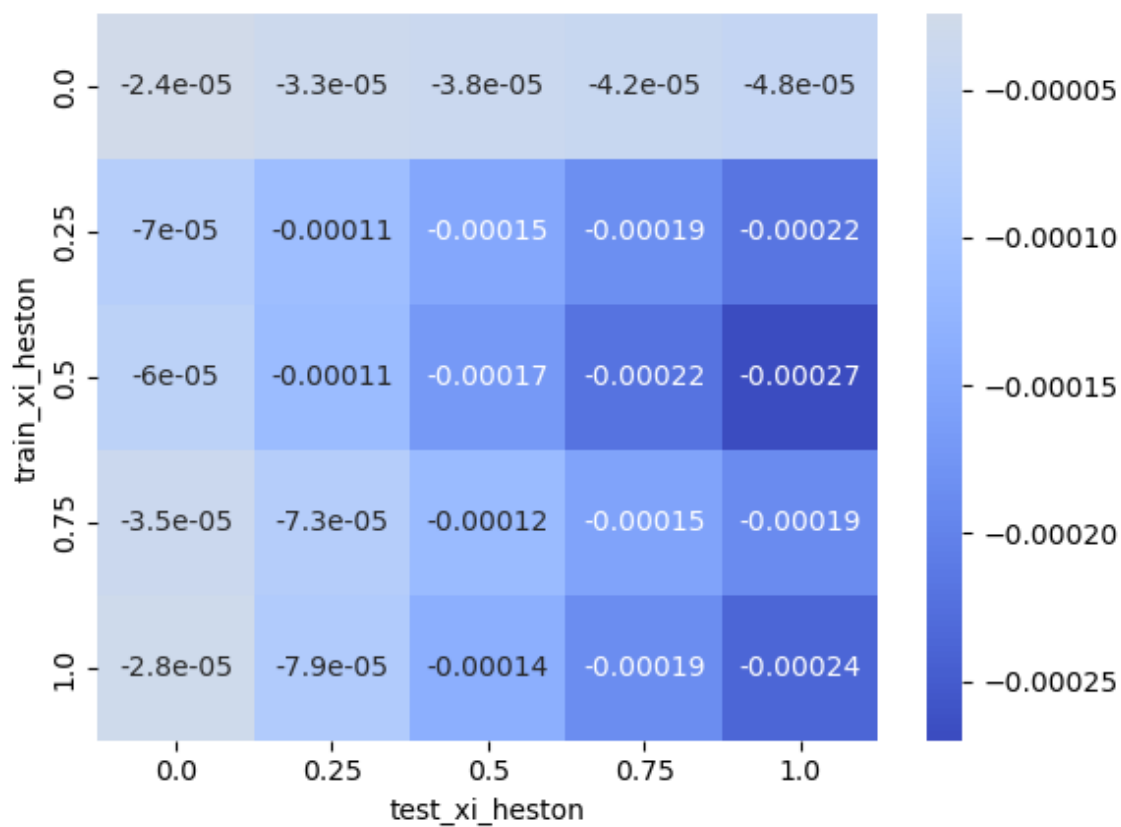


Figure 7.7: Variance surface with controlled distributional shifts. Each entry is the variance of the SDH minus the variance of the AADH: a positive entry means that the AADH has the lower variance, a negative entry that the SDH has the lower variance.

Evaluation Metric	ξ_{test}	SDH	AADH
Risk	0.00	1.573×10^{-4}	1.855×10^{-4}
	0.03	1.710×10^{-4}	2.073×10^{-4}
	0.04	1.768×10^{-4}	2.175×10^{-4}
	0.05	1.796×10^{-4}	2.186×10^{-4}
	0.10	2.043×10^{-4}	2.560×10^{-4}
	0.15	2.302×10^{-4}	2.961×10^{-4}
	0.25	2.701×10^{-4}	3.592×10^{-4}
	0.50	3.591×10^{-4}	5.038×10^{-4}
	1.00	4.497×10^{-4}	7.028×10^{-4}
Variance	0.00	1.342×10^{-4}	1.638×10^{-4}
	0.03	1.487×10^{-4}	1.861×10^{-4}
	0.04	1.547×10^{-4}	1.966×10^{-4}
	0.05	1.577×10^{-4}	1.978×10^{-4}
	0.10	1.838×10^{-4}	2.361×10^{-4}
	0.15	2.112×10^{-4}	2.773×10^{-4}
	0.25	2.533×10^{-4}	3.419×10^{-4}
	0.50	3.476×10^{-4}	4.899×10^{-4}
	1.00	4.437×10^{-4}	6.925×10^{-4}
Transaction costs	0.00	2.740×10^{-3}	2.317×10^{-3}
	0.03	2.753×10^{-3}	2.325×10^{-3}
	0.04	2.756×10^{-3}	2.327×10^{-3}
	0.05	2.762×10^{-3}	2.331×10^{-3}
	0.10	2.782×10^{-3}	2.345×10^{-3}
	0.15	2.799×10^{-3}	2.356×10^{-3}
	0.25	2.835×10^{-3}	2.381×10^{-3}
	0.50	2.938×10^{-3}	2.450×10^{-3}
	1.00	3.136×10^{-3}	2.582×10^{-3}

Table 7.4: Performance of the two deep hedgers trained on pure Heston data ($\xi_{train} = 1$) and evaluated on the mixture measures $\hat{\mathbb{P}}_{\xi_{test}}$ of Definition 6.1.1, where ξ denotes the Heston weight, so that $\xi = 0$ corresponds to pure GARCH paths and $\xi = 1$ to pure Heston paths. The lower value in each row is highlighted in bold.

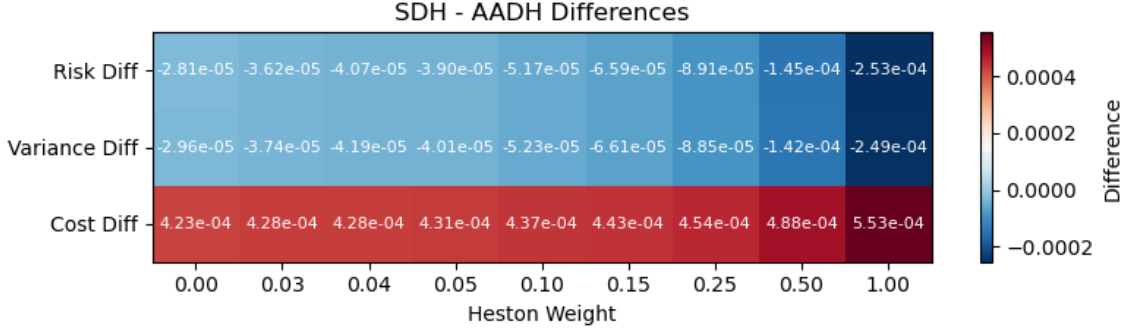


Figure 7.8: Difference in performance between the SDH and the AADH across the main metrics.

negligible, which indicates that the outperformance of the SDH over the AADH in terms of risk cannot be attributed to model misspecification or to a failure of the training procedure. We also note that the gap between the two hedgers is far from constant across ξ_{test} : at $\xi_{test} = 1$, i.e. in distribution, the SDH outperforms the AADH by -2.49×10^{-4} in variance, whereas at $\xi_{test} = 0$, the point of maximal shift away from the training measure, the gap shrinks to -2.96×10^{-5} , roughly a ninefold reduction. This is the classical robustness trade-off one would expect from an ambiguity-averse objective: the AADH gives up some in-sample optimality and recovers part of it as the test distribution moves away from the training one (result consistent with DRO). In our setting, however, this recovery is not sufficient to close the gap entirely, let alone reverse its sign: the SDH remains the better hedger, in terms of both risk and variance, at every mixture weight we tested.

Finally, as a rule of thumb, we check whether the controlled distributional shifts are consistent with the results obtained on empirical data, by locating each index at the mixture weight that reproduces its observed level of risk. We start from the S&P 500. For $\xi = 0$, that is, for pure GARCH paths, the SDH variance is 1.245×10^{-4} on the real index against 1.342×10^{-4} on the synthetic mixture, while for the AADH it is 1.667×10^{-4} against 1.638×10^{-4} . The two pairs of figures are very close, which is consistent with our assumption that an equity index should reproduce almost exactly the behaviour, and hence the risk level, of pure GARCH paths. More interesting is the expected difference between the two hedgers: under the theoretical mixture (see also Figure 7.8) the difference is -0.296×10^{-4} , while on the real index it is -0.422×10^{-4} . The two values are not identical, but they are of the same order and the same sign, and the same holds for the overall risk. In other

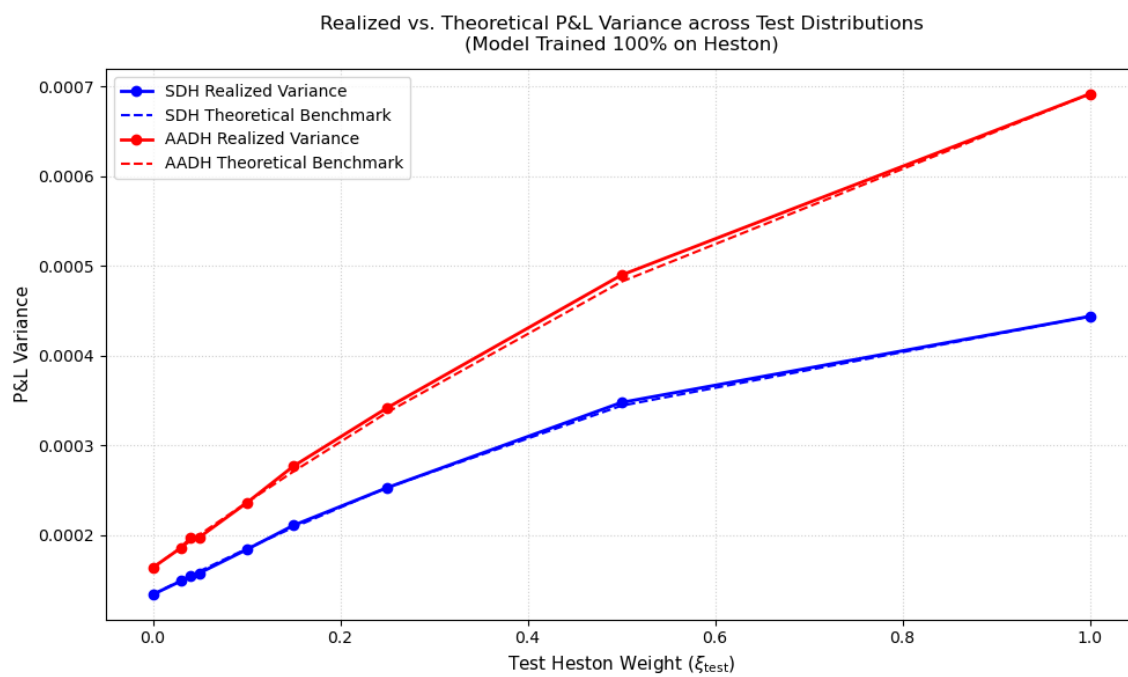


Figure 7.9: Theoretical variance under the controlled distributional shifts compared with the realised results.

words, the synthetic distributional shifts explain, with a good degree of accuracy, both the risk levels and the performance gap observed between the two models on real-world data.

We repeat the exercise on a different index, the Nasdaq 100, to show that the same principle holds. The realised variances are 1.685×10^{-4} for the SDH and 1.935×10^{-4} for the AADH, which correspond to a mixture weight of $\xi = 0.05$ (95% GARCH, 5% Heston). This is a reasonable value, given that the Nasdaq 100 is more volatile than the other indices considered and that our Heston parameters produce high-volatility paths. The associated theoretical variances are 1.577×10^{-4} and 1.979×10^{-4} for the SDH and the AADH respectively, and the difference between the two hedgers is -0.250×10^{-4} under the theoretical mixture against -0.401×10^{-4} on the real index. Again, up to a small approximation, the controlled shifts account for the difference in performance observed on the empirical data.

We close this section by stressing two points. First, the degree of approximation in the risk levels described above is entirely expected, given the difference between real market dynamics and the two generators we employ. We assume that GARCH represents the dynamics of an equity index, while Heston contributes high-volatility paths: neither generator can capture the full complexity of real data, and once this limitation is acknowledged the explanatory power of the mixture family is satisfactory for our purposes. Second, even with this degree of approximation, our results appear to contradict the out-of-sample results of Jones et al., 2025. For comparison, our performance differences, which the theoretical shifts confirm, are -0.296×10^{-4} (theoretical) against -0.422×10^{-4} (real) for the S&P 500, and -0.250×10^{-4} against -0.401×10^{-4} for the Nasdaq 100. The corresponding empirical figures reported in the original paper are $+0.790 \times 10^{-4}$ for the S&P 500 and $+0.754 \times 10^{-4}$ for the Nasdaq 100, with the values for the Nikkei 225 (-0.098×10^{-4}), the EURO STOXX 50 ($+0.109 \times 10^{-4}$) and the FTSE 100 ($+0.130 \times 10^{-4}$) also lying outside the range predicted by our mixture family. Our experiment is therefore unable to provide a theoretical explanation for the results obtained empirically in Jones et al., 2025. In Section 7.3 we discuss why.

7.3 Some Considerations on the Results

In this section we explain the reasons behind the difference between our results and those of the seminal paper of Jones et al., 2025, highlighted in Subsection 7.2.2. We describe our thought process and the rich and constructive exchange we subsequently had with the authors of the paper on the matter.

We started from the results on the test set reported in Subsection 7.2.1. There

we confirmed all the findings of the paper: when testing on Heston paths generated with the same parameters used for training, the risk metrics we obtain are essentially identical to those reported in the original work. The discrepancy arises only on out-of-sample data, which led us to suspect an issue in the real data used for testing.

We then tried a number of different configurations, as described in Subsection 7.2.1, also aligning the few modelling choices in which we had departed from the original paper (600 epochs, terminal transaction costs, and the computation of δ_{k+1} rather than δ_k , as in Buehler et al., 2019). This again produced nearly identical numbers on the test set, and therefore pointed towards a difference in the collection or preprocessing convention of the real data.

At that stage we formulated the conjecture that the original paper might have adopted a different time convention from ours when constructing the sliding windows, running backwards in time. To give a concrete example, if yesterday’s price is \$100 and today’s price is \$110, the forward convention records an increase of +10%, whereas the backward convention (from today to yesterday) records a decrease of −9.1%. Under this backward convention, the results we obtained on real index data were remarkably close to the published ones. For the S&P 500, for instance, we obtained a variance of 2.903×10^{-4} against 2.547×10^{-4} in Jones et al., 2025 for the SDH, and 1.855×10^{-4} against 1.757×10^{-4} for the AADH. We also recovered the fact that on the Nikkei 225 the variance of the SDH is slightly lower than that of the AADH, which is the only case in the paper where this behaviour is documented.

A second observation pointed in the same direction. We noticed that the AADH (and also the SDH) is particularly exposed to sharp price *increases* rather than to sharp decreases. Testing on COVID-19 data, we found that most of the risk was carried by a handful of paths corresponding to the rally that started at the beginning of April 2020, when the indices gained +20% to +25% in a couple of weeks. This is consistent with the previous point: under the backward convention, the sharp declines of the COVID-19 crisis are turned into even sharper increases (a −30% move becomes a +43% move once the window is reversed). Taken together, these two findings seemed to indicate a discrepancy in the data preprocessing.

We then contacted the authors of Jones et al., 2025, with whom we had a detailed and very constructive exchange on the matter. On inspecting their code, they found an error in the generation of the Heston paths which transformed the intended $\rho = -0.7$ into $\rho \approx +0.71$. This is clearly not what the authors intended, especially since two positively correlated Brownian motions in the Heston model are at best unlikely and at worst dangerous when modelling the behaviour of equity paths. Figure 7.10 illustrates the paths generated under the two values of ρ . While the variance processes are identical in distribution, the price processes differ substantially: un-

der a positive correlation, periods of high volatility are accompanied by sharp price increases rather than by sharp price decreases.

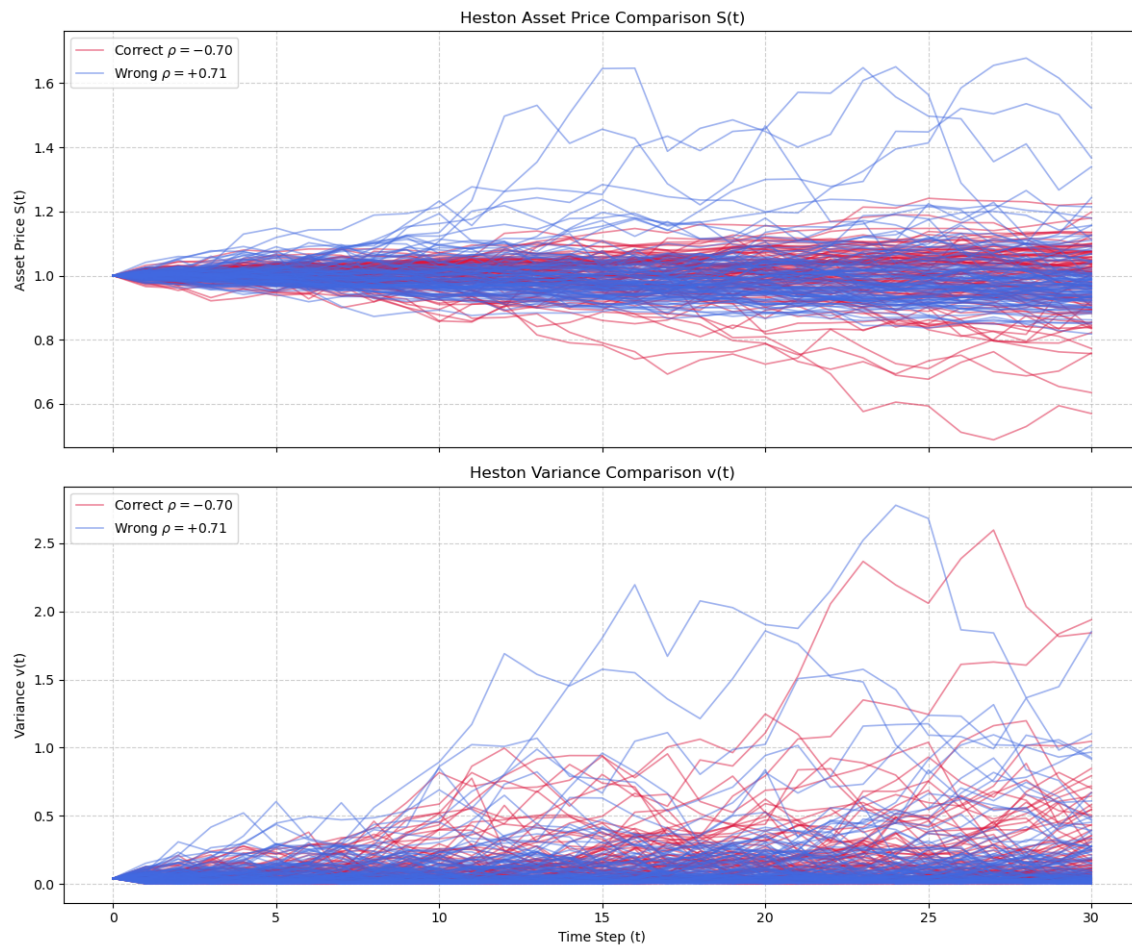


Figure 7.10: Comparison of the Heston price and variance processes under the correct and the incorrect ρ parameter.

This discovery leads to two further considerations. First, as the authors of Jones et al., 2025 themselves suggested, the published experiments can be reread as a study of the robustness of deep hedging to the misspecification of the correlation parameter. Seen in this light, the result is relevant in its own right: it indicates that an ambiguity-averse objective may recover part of the performance lost to a structurally misspecified generator, and it would constitute a very interesting contribution.

Second, and more speculatively, the sign of ρ offers a natural explanation for the upside sensitivity described above. In a preliminary analysis of the COVID-19 evaluation data, we examined which paths drive the risk of the two hedgers. We found that a very small fraction of paths, roughly 0.5% of the evaluation paths and 0.2% of the whole dataset, accounts for a disproportionate share of the total risk, and that these paths are concentrated on a handful of dates during the COVID-19 rebound. Removing them lowers the risk of both hedgers substantially, with the AADH proving relatively more resilient. As already noted, these high-risk paths correspond to sharp upward, not downward, moves.

This connects directly to the correlation error. A Heston model with $\rho \approx +0.71$ produces a training distribution in which volatility spikes coincide with upward price moves, and therefore one whose extreme paths are sharp rallies; the standard parametrisation with $\rho = -0.7$ concentrates its extreme paths on the downside. Since, for a short call position, it is precisely the upside moves that appear to drive the hedging risk, the incorrect ρ may have inadvertently produced training data better suited to the empirical evaluation, not because it is more realistic, but because it contains the relevant type of tail event. In other words, the out-of-sample advantage of the AADH documented in Jones et al., 2025 may be inseparable from the direction of the extreme moves present in its training measure, which would also explain why our controlled distributional shifts, built on a correctly specified $\rho = -0.7$ Heston generator, never reproduce it.

We stress that this remains an empirical conjecture, drawn from a separate and preliminary analysis, and that we do not attempt to prove it here. A natural next step, which we leave to future work, would be to track which feature clusters carry the largest share of risk across training runs, and to verify whether this is systematically linked to the sign of ρ and, more generally, to the direction of the extreme moves contained in the training measure.

Chapter 8

Conclusion

In this thesis we studied the behaviour of ambiguity-averse deep hedging under controlled distributional shifts. After introducing the mathematical objects involved, feedforward neural networks and their approximation properties in Chapter 3, and the standard deep hedging framework of Buehler et al., 2019 in Chapter 4, we presented in Chapter 5 the ambiguity-averse extension of Jones et al., 2025, based on feature clustering and on the smooth ambiguity aversion model of Klibanoff et al., 2005. We made precise the role of the two hyperparameters introduced by that construction, showing that the number of clusters k interpolates between a standard deep hedger minimising the inner risk measure and one minimising the outer ambiguity measure, and that the ambiguity aversion parameter γ recovers the maxmin criterion of Gilboa and Schmeidler, 1989 in the limit $\gamma \rightarrow \infty$.

The main methodological contribution of the thesis is the framework of Chapter 6. Rather than evaluating an ambiguity-averse hedger against a handful of fixed and externally given test sets, each lying at an unknown distance from the training measure, we constructed a one-parameter family of mixture measures $\mathbb{P}_\xi = \xi \mathbb{P}_H + (1 - \xi) \mathbb{P}_G$ interpolating between a Heston and a GARCH(1,1) generator. We showed the mathematical properties of this object and we derived a closed-form decomposition of the variance of any functional under the mixture. This provides a model-free prediction against which the realised risk of a trained strategy can be compared.

The empirical results of Chapter 7 can be summarised in three points. First, on in-distribution data our implementation reproduces the numbers of Jones et al., 2025 closely, and confirms the qualitative behaviour of the AADH: a flatter delta profile, a lower gamma, and systematically lower transaction costs, together with nearly identical risk metrics on the test set. Second, on real equity indices, and on every point of the synthetic risk surface, the standard deep hedger attains a lower risk and a

lower variance than the ambiguity-averse one, while the AADH remains consistently cheaper: this is the opposite of the out-of-sample conclusion of the original paper, and it proved stable across every configuration we tried. Third, the theoretical mixture family accounts well for the risk levels observed on real indices, and, more importantly, for the size and sign of the gap between the two hedgers, which places the empirical result and the synthetic experiment on a common footing.

As discussed in Section 7.3, the origin of this divergence was identified through a direct and very helpful exchange with the authors of Jones et al., 2025, who found that their Heston generator had effectively been run with $\rho \approx +0.71$ rather than the intended $\rho = -0.7$. This does not invalidate the ambiguity-averse construction, which remains theoretically well founded: it relocates the question. What the published experiments demonstrate is arguably the resilience of an ambiguity-averse objective to a structurally misspecified generator. This is a very interesting result in itself.

Several directions remain open. The most immediate is the conjecture formulated in Section 7.3: attributing risk to individual feature clusters across training runs, and testing whether the clusters that dominate the loss are systematically tied to the sign of ρ and to the direction of the extreme moves in the training measure. A positive answer may turn the present negative result into a constructive prescription: the features used for clustering, and the tails of the generator, should be chosen jointly with the payoff being hedged.

A second direction concerns the relative scale of the two terms in the inner loss. In a separate experiment we replaced $\lambda = 1/100$ with $\lambda = 1/10$, while retaining the sigmoid constraint that forces $\delta \in (0, 1)$ and therefore removes the statistical arbitrage concern of Horikawa and Nakagawa, 2024. This places variance and transaction costs on a comparable scale, and under this setup the distributional-shift pattern on the synthetic grid becomes considerably more pronounced. In our preliminary tests (not included in the paper) two of the five indices show a marginal improvement of the AADH over the SDH that is not present under the original λ . Since the AADH systematically trades less, any objective that assigns more weight to trading costs mechanically favours it, and characterising this trade-off appears to be a more promising route to an advantage than robustness to shifts alone. A related direction would be to replace the inner loss of Equation 5.9 with a convex risk measure in the sense of Definition 4.3.1, rather than the non-convex variance-based criterion we use, at the cost of reintroducing the statistical arbitrage issue of Horikawa and Nakagawa, 2024 that motivated the original choice.

Finally, the framework of Chapter 6 is deliberately generic and can be extended in several ways: to richer mixture families including jump diffusions or rough volatility

generators (Fukasawa et al., 2021; Horvath et al., 2021), to deep market generators in place of parametric ones (Cetingoz & Lehalle, 2026), to other payoffs, and in particular to the forward start option, for which Jones et al., 2025 report their strongest results and whose path dependence would make the ambiguity-averse objective a more natural fit. On the theoretical side, the global approximation result of Cuchiero et al., 2026 provides the foundation needed to extend the whole construction to genuinely path-dependent strategies on non-compact path spaces, which we have left for future work.

Bibliography

- Bartl, D., Drapeau, S., Obłój, J., & Wiesel, J. (2021). Sensitivity analysis of wasserstein distributionally robust optimization problems. *Proceedings. Mathematical, Physical, and Engineering Sciences*, 477(2256), 20210176.
- Ben-Tal, A., & Teboulle, M. (2007). An old-new concept of convex risk measures: The optimized certainty equivalent. *Mathematical Finance*, 17(3), 449–476.
- Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, 81(3), 637–654.
- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3), 307–327.
- Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8), 1271–1291.
- Cannelli, L., Nuti, G., Sala, M., & Szehr, O. (2023). Hedging using reinforcement learning: Contextual k-armed bandit versus q-learning. *The Journal of Finance and Data Science*, 9, 100101.
- Cao, J., Chen, J., Farghadani, S., Hull, J., Poulos, Z., Wang, Z., & Yuan, J. (2023). Gamma and vega hedging using deep distributional reinforcement learning. *Frontiers in Artificial Intelligence*, 6, 1129370.
- Cao, J., Chen, J., Hull, J., & Poulos, Z. (2021). Deep hedging of derivatives using reinforcement learning. *arXiv preprint arXiv:2103.16409*.
- Cetingoz, A. R., & Lehalle, C.-A. (2026). Synthetic data for portfolios: A throw of the dice will never abolish chance. *Quantitative Finance*, 1–32.
- Charpentier, A., Elie, R., & Remlinger, C. (2023). Reinforcement learning in economics and finance: A. charpentier et al. *Computational Economics*, 62(1), 425–462.
- Cont, R. (2001). Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative finance*, 1(2), 223.
- Cuchiero, C., Schmock, P., & Teichmann, J. (2026). Global universal approximation of functional input maps on weighted spaces. *Constructive Approximation*, 1–76.

- Duembgen, M., & Rogers, L. C. G. (2014). Estimate nothing. *Quantitative Finance*, 14(12), 2065–2072.
- Ellsberg, D. (1961). Risk, ambiguity, and the savage axioms. *The quarterly journal of economics*, 75(4), 643–669.
- Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the econometric society*, 987–1007.
- Fecamp, S., Mikael, J., & Warin, X. (2020). Deep learning for discrete-time hedging in incomplete markets. *Journal of computational Finance*, 25(2).
- Fukasawa, M., Horvath, B., & Tankov, P. (2021). Hedging under rough volatility. *arXiv preprint arXiv:2105.04073*.
- Gatheral, J. (2011). *The volatility surface: A practitioner’s guide*. John Wiley & Sons.
- Gilboa, I., & Schmeidler, D. (1989). Maxmin expected utility with non-unique prior. *Journal of mathematical economics*, 18(2), 141–153.
- Giurca, A., & Borovkova, S. (2021). Delta hedging of derivatives using deep reinforcement learning. *Available at SSRN 3847272*.
- Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT press Cambridge.
- Halperin, I. (2017). Qlbs: Q-learner in the black-scholes (-merton) worlds. *arXiv preprint arXiv:1712.04609*.
- Halperin, I. (2019). The qlbs q-learner goes nuclear: Fitted q iteration, inverse rl, and option portfolios. *Quantitative Finance*, 19(9), 1543–1553.
- Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, 6(2), 327–343.
- Horikawa, H., & Nakagawa, K. (2024). Relationship between deep hedging and delta hedging: Leveraging a statistical arbitrage strategy. *Finance Research Letters*, 62, 105101.
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2), 251–257.
- Horvath, B., Teichmann, J., & Žurič, Ž. (2021). Deep hedging under rough volatility. *Risks*, 9(7), 138.
- Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution* (pp. 492–518). Springer.
- Jones, A., Horvath, B., Reisinger, C., Wood, B., Bai, L., & Akkari, A. (2025). Ambiguity-averse deep hedging with feature clustering. *Available at SSRN 5390563*.

- Karoui, N. E., Jeanblanc-Picqu , M., & Shreve, S. E. (1998). Robustness of the black and scholes formula. *Mathematical finance*, 8(2), 93–126.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Klibanoff, P., Marinacci, M., & Mukerji, S. (2005). A smooth model of decision making under ambiguity. *Econometrica*, 73(6), 1849–1892.
- Kolm, P. N., & Ritter, G. (2019). Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, 1(1), 159–171.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444.
- Limmer, Y., & Horvath, B. (2024). Robust hedging gans: Towards automated robustification of hedging strategies. *Applied Mathematical Finance*, 31(3), 164–201.
- Livieri, G., Mouti, S., Pallavicini, A., & Rosenbaum, M. (2018). Rough volatility: Evidence from option prices. *IIE transactions*, 50(9), 767–776.
- Lord, R., Koekkoek, R., & Dijk, D. V. (2010). A comparison of biased simulation schemes for stochastic volatility models. *Quantitative Finance*, 10(2), 177–194.
- L tkebohmert, E., Schmidt, T., & Sester, J. (2022). Robust deep hedging. *Quantitative Finance*, 22(8), 1465–1480.
- Mikkil , O., & Kannianen, J. (2023). Empirical deep hedging. *Quantitative Finance*, 23(1), 111–122.
- Rahimian, H., & Mehrotra, S. (2022). Frameworks and results in distributionally robust optimization. *Open Journal of Mathematical Optimization*, 3, 1–85.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088), 533–536.
- Sutton, R. S., Barto, A. G., & Barto, A. (1998). *Reinforcement learning: An introduction* (Vol. 1). MIT press Cambridge.
- Wu, D., & Jaimungal, S. (2023). Robust risk-aware option hedging. *Applied Mathematical Finance*, 30(3), 153–174.

Appendix A

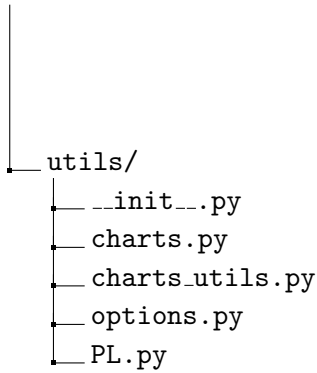
Repository Structure and Code

The full implementation comprises approximately 9,000 lines of Python code, built on `PyTorch`. The repository is organized into several Python folders supporting data generation, dataset construction, ambiguity modelling, neural-network models, training, risk measurement, and visualization utilities. The root directory also contains experiment notebooks, testing scripts, project configuration files, and documentation. The large data collections are represented by placeholders in the diagram rather than being listed individually. In particular, `dataset/real_data/` contains the source data files used by the experiments, including CSV datasets and scripts for constructing rolling and inverted data windows.

The code is publicly available at <https://github.com/DavideDevetak24/Ambiguity-Averse-Deep-Hedging-of-Options-Under-Controlled-Distributional-Shifts/tree/main>. For clarity, this public repository contains a reduced version of the project, limited to the main experiments. The full directory tree, reproduced below, includes additional exploratory notebooks and test scripts that were part of the development process but were left out of the public release to avoid unnecessary clutter.

```
AADH_Scripts/  
├── __init__.py  
├── README.md  
├── .env  
├── .vscode/  
│   └── settings.json  
├── costs_test.py  
├── Idea.1.ipynb  
├── Images.ipynb  
└── Main.ipynb
```

- main_mix_1.ipynb
- main_mix_2.ipynb
- main_mix_3.ipynb
- Main_MLP_test_1.ipynb
- Main_MLP_test_2.ipynb
- Main_test_1_architectures.ipynb
- Main_test_2_one_cluster.ipynb
- Main_test_4_covid.ipynb
- Main_test_5_covid.ipynb
- Main_test_6_covid.ipynb
- ambiguity_measures/
 - __init__.py
 - ambiguity_loss_fn.py
 - KMeans_algo.py
- data_generators/
 - __init__.py
 - cev.py
 - garch.py
 - heston.py
- dataset/
 - __init__.py
 - dataset.py
 - real_data/
 - *Data source files: CSV datasets*
 - real_data_window.py
 - real_data_inverted_window.py
- models/
 - __init__.py
 - models_utils.py
 - neural_network.py
- risk_measures/
 - __init__.py
 - measures.py
 - test_metrics.py
- training/
 - __init__.py
 - train_loop_nn.py
 - train_utils.py



Relevant Code Excerpts

The sections below reproduce a selection of the core parts of code we discussed in the main text. Again, we remark that this is only a tiny fraction of what we included in the repository, however it is useful to understand in a more practical sense the logic we described theoretically in the previous chapters. We choose to order the code chunks following the flow of the main pipeline.

Listing A.1: Heston Model for Data Generation

```

1  import numpy as np
2  import pandas as pd
3
4  S0 = 100          # initial stock price
5  v0 = 0.04         # initial variance
6  T = 1            # time horizon
7  N = 252          # number of yearly steps
8  r = 0.03         # risk-free rate
9  mu = r           # drift under risk neutral measure
10 kappa = 1         # rate of mean reversion
11 theta = 0.04      # long-term variance
12 sigma_v = 2       # volatility of volatility
13 rho = -0.7        # correlation between Brownian motions
14 K = 100           # strike price
15 n_simulations = 100 # number of Monte Carlo simulations
16
17 def heston_data_generator(S0, v0, T, N, r, mu, kappa, theta, sigma_v,
18                           rho, n_simulations=1000, risk_neutrality=True, random=True
19                           , seed=42):
20
21     if risk_neutrality == True:
22         mu = r
23     else:
24         mu = mu
25
26     dt = T / N
27
28     data_S = np.empty((N+1, n_simulations))
29     data_v = np.empty((N+1, n_simulations))
30     data_S[0, :] = S0
31     data_v[0, :] = v0
32     rng = np.random.default_rng(seed if random else 42)
33
34     Z = rng.multivariate_normal(

```

```

33     mean=[0.0, 0.0],
34     cov=[[1.0, rho], [rho, 1.0]],
35     size=(N, n_simulations)
36 )
37 dW_s = np.sqrt(dt) * Z[:, :, 0]
38 dW_v = np.sqrt(dt) * Z[:, :, 1]
39
40 for j in range(1, N+1):
41     vt = np.abs(data_v[j-1, :])
42     sqrt_vt = np.sqrt(vt)
43     data_S[j, :] = data_S[j-1, :] * np.exp((mu - 0.5 * vt) * dt + sqrt_vt * dW_s
44         [j-1, :])
45     v_new = data_v[j-1, :] + kappa * (theta - vt) * dt + sigma_v * sqrt_vt *
46         dW_v[j-1, :]
47     data_v[j, :] = np.abs(v_new)
48
49 return pd.DataFrame(data_S), pd.DataFrame(data_v)

```

Listing A.2: Neural Network Model

```

1 import torch
2 import torch.nn as nn
3 import torch.nn.functional as F
4 from models.models_utils import CustomSharpSigmoid
5
6 class HedgingNN(nn.Module):
7     def __init__(self,
8         hidden_layers_dim=100,
9         num_hidden_layers=3,
10        n_features=3,
11        n_assets=1,
12        center=0.5,
13        scale=10.0):
14        super(HedgingNN, self).__init__()
15        self.hidden_layers_dim = hidden_layers_dim
16        self.num_hidden_layers = num_hidden_layers
17        self.n_assets = n_assets
18        self.n_features = n_features
19        input_dim = n_assets * n_features
20        layers = []
21        layers.append(nn.Linear(input_dim, hidden_layers_dim))
22        layers.append(nn.ReLU())
23        for _ in range(self.num_hidden_layers):
24            layers.append(nn.Linear(hidden_layers_dim, hidden_layers_dim))
25            layers.append(nn.ReLU())
26        layers.append(nn.Linear(hidden_layers_dim, n_assets))
27        layers.append(CustomSharpSigmoid(center=center, scale=scale))
28        self.network = nn.Sequential(*layers)

```

Listing A.3: Inner Loss Function

```

1 import torch
2
3 class AADHInnerLoss(nn.Module):
4     def __init__(self, lam: float = 1/100):
5         super().__init__()
6         self.lam = float(lam)

```

```

7     def forward(self, pl_no_tc: torch.Tensor, tc: torch.Tensor) -> torch.Tensor:
8         if pl_no_tc.numel() > 1:
9             V_pl_no_tc = torch.var(pl_no_tc, unbiased=True)
10        else:
11            V_pl_no_tc = torch.tensor(1e-5, device=pl_no_tc.device) # added variance
12            floor
13        E_tc = torch.mean(tc)
14        return V_pl_no_tc + self.lam * E_tc

```

Listing A.4: Outer Loss Function

```

1  import torch
2  import torch.nn as nn
3
4  class AADHOuterLoss(nn.Module):
5      def __init__(self, alpha: float = 250):
6          super().__init__()
7          self.alpha = float(alpha)
8      def forward(self, inner_losses: torch.Tensor, cluster_probs: torch.Tensor) ->
9          torch.Tensor:
10         r_max = inner_losses.max()
11         exponential = torch.exp(self.alpha * (inner_losses - r_max))
12         expectation = torch.sum(cluster_probs * exponential)
13         outer_loss = r_max + (1.0 / self.alpha) * torch.log(expectation)
14         return outer_loss

```

Listing A.5: Train Loop

```

1  def train_one_epoch_AADH(model, loader, optimizer, cluster_probs, inner_loss=
2      AADHInnerLoss(), outer_loss=AADHOuterLoss(), option_type_fn=eur_call_payoff,
3      strike=100.0, cost_rate=0.0, device='cpu'):
4      model.train()
5      device = torch.device(device)
6      cluster_probs_tensor = _cluster_probs_to_tensor(cluster_probs, device)
7      use_full_batch_forward = _can_use_full_batch_aadh_forward(model)
8      running_loss = torch.zeros(), device=device)
9      n = 0
10     for paths, labels in loader:
11         paths = _to_device(paths, device)
12         labels = _to_device(labels, device)
13         outer_loss_value = _compute_aadh_batch_loss(model, paths, labels,
14             cluster_probs_tensor,
15             inner_loss, outer_loss,
16             option_type_fn, strike,
17             cost_rate,
18             use_full_batch_forward)
19         optimizer.zero_grad(set_to_none=True)
20         outer_loss_value.backward()
21         clip_grad_norm_(model.parameters(), 1.0)
22         optimizer.step()
23         running_loss += outer_loss_value.detach() * batch_size
24         n += batch_size
25     epoch_loss = (running_loss / n).item()
26     return epoch_loss

```

Listing A.6: Distributional Shifts Experiment

```

1 def run_mixed_generator_experiment(heston_params,
2                                   garch_params,
3                                   n_train_paths=2000,
4                                   n_test_paths=1000,
5                                   xi_step=0.1,
6                                   xi_values=None,
7                                   n_clusters=5,
8                                   batch_size=100,
9                                   train_val_proportions=(0.8, 0.2),
10                                  epochs=50,
11                                  cost_rate=0.0,
12                                  learning_rate=1e-3,
13                                  lam=1/100,
14                                  alpha=250,
15                                  option_type_fn=None,
16                                  strike=1.0,
17                                  adaptive_learning=False,
18                                  schedule=None,
19                                  model_class=None,
20                                  model_kwargs=None,
21                                  device="cpu",
22                                  seed=42,
23                                  shuffle=False,
24                                  num_workers=0,
25                                  pin_memory=None,
26                                  persistent_workers=False,
27                                  quiet=True):
28
29     if n_train_paths <= 0 or n_test_paths <= 0:
30         raise ValueError("n_train_paths and n_test_paths must be positive.")
31
32     if len(train_val_proportions) != 2:
33         raise ValueError("train_val_proportions must contain train and validation
34                             proportions.")
35
36     train_prop, val_prop = train_val_proportions
37     if abs((train_prop + val_prop) - 1.0) > 1e-10:
38         raise ValueError("train_val_proportions must sum to 1.")
39
40     if model_class is None:
41         model_class = HedgingNN
42     if model_kwargs is None:
43         model_kwargs = {}
44     if option_type_fn is None:
45         option_type_fn = eur_call_payoff
46
47     device = torch.device(device)
48     xi_grid = _mixture_grid(xi_step=xi_step, xi_values=xi_values)
49     test_path_cache = {}
50
51     for test_idx, test_xi in enumerate(xi_grid):
52         paths, n_heston, n_garch = _generate_mixed_paths(
53             heston_params,
54             garch_params,
55             n_test_paths,
56             test_xi,

```

```

56         seed + 100_000 + test_idx * 1_000
57     )
58     test_path_cache[test_xi] = {
59         "paths": paths,
60         "n_heston": n_heston,
61         "n_garch": n_garch,
62     }
63
64     loader_kwargs = {
65         "batch_size": batch_size,
66         "shuffle": False,
67         "num_workers": num_workers,
68         "pin_memory": torch.cuda.is_available() if pin_memory is None else
69             pin_memory,
70         "persistent_workers": persistent_workers and num_workers > 0,
71     }
72
73     rows = []
74
75     for train_idx, train_xi in enumerate(xi_grid):
76         train_paths, train_heston_count, train_garch_count = _generate_mixed_paths(
77             heston_params,
78             garch_params,
79             n_train_paths,
80             train_xi,
81             seed + 10_000 + train_idx * 1_000
82         )
83
84         train_features = compute_feature_matrix(train_paths)
85         train_labels, kmeans_model = kmeans_clustering(train_features, n_clusters=
86             n_clusters)
87         train_dataset = AADeepHedgingDataset(train_paths, train_labels)
88         train_loader, val_loader, _ = build_loaders(
89             train_dataset,
90             batch_size=batch_size,
91             set_proportions=[train_prop, val_prop, 0.0],
92             shuffle=shuffle,
93             num_workers=num_workers,
94             pin_memory=pin_memory,
95             persistent_workers=persistent_workers
96         )
97
98         model_seed = seed + train_idx
99         aadh_model = _train_mix_model(
100             train_main, model_class, model_kwargs, train_loader, val_loader,
101             option_type_fn, strike, cost_rate, lam, alpha, epochs,
102             learning_rate, True, adaptive_learning, schedule, device,
103             model_seed, quiet
104         )
105
106         sdh_model = _train_mix_model(
107             train_main, model_class, model_kwargs, train_loader, val_loader,
108             option_type_fn, strike, cost_rate, lam, alpha, epochs,
109             learning_rate, False, adaptive_learning, schedule, device,
110             model_seed, quiet
111         )
112
113     for test_xi in xi_grid:
114         test_paths = test_path_cache[test_xi]["paths"]

```

```

112     test_features = compute_feature_matrix(test_paths)
113     test_labels = kmeans_model.predict(test_features)
114     test_dataset = AADeepHedgingDataset(test_paths, test_labels)
115     test_loader = DataLoader(test_dataset, **loader_kwargs)
116
117     aadh_risk, aadh_variance, aadh_costs = test_result_metrics(
118         aadh_model,
119         test_loader,
120         option_type_fn=option_type_fn,
121         strike=strike,
122         cost_rate=cost_rate,
123         lam=lam,
124         device=device
125     )
126     sdh_risk, sdh_variance, sdh_costs = test_result_metrics(
127         sdh_model,
128         test_loader,
129         option_type_fn=option_type_fn,
130         strike=strike,
131         cost_rate=cost_rate,
132         lam=lam,
133         device=device
134     )
135
136     rows.append({
137         "train_xi_heston": train_xi,
138         "test_xi_heston": test_xi,
139         "train_heston_paths": train_heston_count,
140         "train_garch_paths": train_garch_count,
141         "test_heston_paths": test_path_cache[test_xi]["n_heston"],
142         "test_garch_paths": test_path_cache[test_xi]["n_garch"],
143         "AADH_overall_risk": aadh_risk,
144         "AADH_variance": aadh_variance,
145         "AADH_transaction_costs": aadh_costs,
146         "SDH_overall_risk": sdh_risk,
147         "SDH_variance": sdh_variance,
148         "SDH_transaction_costs": sdh_costs,
149         "risk_diff_SDH_minus_AADH": sdh_risk - aadh_risk,
150         "variance_diff_SDH_minus_AADH": sdh_variance - aadh_variance,
151         "cost_diff_SDH_minus_AADH": sdh_costs - aadh_costs,
152     })
153
154     return pd.DataFrame(rows)

```

AI Disclaimer

The AI tool Claude Sonnet 5 (Anthropic) was minimally used solely for linguistic refinement and the validation of selected proofs. All core ideas, research, and final derivations remain entirely the original work of the author.