

A Deep Learning Approach to Hedging Financial Derivatives with Price Impact

John Huynh

Matrikelnummer: 401283

Ausarbeitung zur Erlangung des akademischen Grades

**Bachelor of Science
Mathematik**

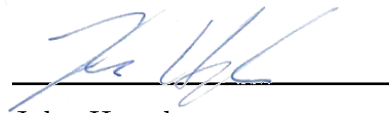
Erstgutachter: Prof. Dr. Christoph Belak

Zweitgutachter: Prof. Dr. Peter Bank

Eidesstattliche Erklärung zur Bachelorarbeit

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 09.05.2022



John Huynh

Zusammenfassung

Wir stellen ein Deep Learning Framework zum Hedging von Finanzderivaten in Märkten mit Friktionen vor, das erstmals in [Büh+19] präsentiert wurde. Die grundlegende Idee besteht darin, Hedgingstrategien durch neuronale Netze zu parametrisieren und die für neuronale Netzwerke bekannten effizienten Optimierungsalgorithmen zu nutzen, um eine optimale Hedgingstrategie zu erhalten. Dieser Ansatz ist nicht von spezifischen Marktdynamiken abhängig und ist daher vollständig datengetrieben. Insbesondere ermöglicht er die Modellierung von Marktfriktionen wie Transaktionskosten, Liquiditätsbeschränkungen und Market Impact.

Die Leistungsfähigkeit des Deep Hedging Algorithmus wird zunächst an einem synthetischen Markt mit Black-Scholes Dynamik ohne Friktionen demonstriert. Wir stellen fest, dass die durch neuronale Netze parametrisierten Hedgingstrategien in diesem Fall in der Lage sind, die optimale Hedgingstrategie gut zu approximieren. Darüber hinaus untersuchen wir die Leistung des Deep Hedging Ansatzes in einem Szenario mit Friktionen, genauer gesagt bei Market Impact. Zu diesem Zweck betrachten wir ein Finanzmodell mit permanentem Preis Impact, wie in [BLZ16] untersucht, und leiten eine perfekte Hedgingstrategie in einem speziellen Fall her. Unsere numerischen Experimente zeigen vielversprechende Ergebnisse für die untersuchte Impact Struktur.

Abstract

We present a deep learning framework for hedging financial derivatives in markets with frictions, which was first proposed in [Büh+19]. The basic idea is to represent hedging strategies by deep neural networks and to utilize the efficient optimization algorithms known for neural networks to obtain an optimal hedging strategy. This approach is not dependent on specific market dynamics and is thus entirely data-driven. In particular, it allows modelling market frictions such as transaction costs, liquidity constraints and market impact.

The feasibility of the deep hedging algorithm is first demonstrated in a synthetic market driven by the Black-Scholes model without frictions. We find that in this setting, the neural network hedging strategy is able to approximate the optimal hedging strategy well. Further, we investigate the performance of the deep hedging approach in the presence of frictions, namely market impact. For this purpose, we consider a financial model with permanent price impact, as studied in [BLZ16], and derive a perfect hedging strategy in a special case. Our numerical experiments show promising results for the studied impact structure.

Contents

1	Introduction	1
2	Pricing and Hedging in Financial Markets with Frictions	3
2.1	Discrete-time Markets with Frictions	3
2.2	Convex Risk Measures	7
2.3	Pricing and Hedging Using Convex Risk Measures	12
3	Deep Learning Approximation	15
3.1	Neural Networks and Universal Approximation	15
3.2	Approximating Hedging Strategies with Deep Neural Networks	17
3.3	Learning the Optimal Hedging Strategy	22
4	Numerical Experiments	26
4.1	Black-Scholes Model	26
4.2	Setting and Implementation	27
4.3	Numerical Results	28
5	Markets with Permanent Price Impact	33
5.1	Impact Rule	33
5.2	Theoretical Benchmark	35
5.3	Numerical Results	37
6	Conclusion	40
	References	42

1 Introduction

In recent years, deep learning has seen tremendous growth in its popularity and usefulness. This came through a series of overwhelming successes in widely different application areas such as image classification, speech recognition and natural language processing. Needless to say, deep learning techniques have also had a major impact on the space of quantitative finance. In this thesis, we take a closer look at the application of deep learning in the context of pricing and risk management. To be more precise, we consider a deep learning approach to the pricing and hedging of financial derivatives.

To motivate this, suppose we want to sell a derivative. What price should we charge and how do we hedge our position to reduce our risk exposure? A tractable solution to this problem can be obtained in idealized frictionless markets. Typically, one assumes that the future evolution of the price of the underlyings is given by some stochastic model. The model is then calibrated using the market prices of liquidly traded options. Subsequently, it is possible to compute a fair price for the derivative using concepts of risk-neutral pricing. To hedge the position after the sale, one usually calculates the so-called *Greeks*, i.e. the sensitivity of the model price w.r.t. changes of the underlyings and other state variables, and trades accordingly. Note that in general, we may already have an existing portfolio of assets, or we may want to sell multiple different derivatives. Under the idealized market assumption the hedging of each derivative can be handled separately, or differently stated, pricing and risk of a portfolio of derivatives is linear. However, in real markets this is not really true any more, since any kind of trading is subject to market frictions such as transaction costs, market impact and liquidity constraints. As a result, it is necessary that one adds their own “intuitive” adjustments to the trading strategy given by the computed Greeks. For instance, selling a particular derivative may be beneficial to reduce the risk of the already existing portfolio, and thus it can be priced more favourably. In essence, the use of these idealized models is somewhat unsatisfactory, but it is prevalent in practice due to a lack of efficient alternatives.

This deficiency is addressed in [Büh+19] where the authors propose a deep learning approach to hedge derivatives. The basic idea is to represent hedging strategies by deep neural networks and to utilize the efficient optimization algorithms known for neural networks to obtain an optimal hedging strategy. Indeed, the experiments in [Büh+19] illustrate the high feasibility of this *deep hedging* algorithm and showed that the neural network hedging performed very well compared to the benchmark in a Heston model. The advantage of this approach is that it does not require to assume anything about the underlying market dynamics, and it thus entirely data-driven. In particular, it allows including market frictions such as transaction costs, liquidity constraints and market impact. Consequently, [Büh+19] included experiments with proportional transaction costs, which

showed very promising results. In addition, there have also been tests which indicate the feasibility of the approach even in a high-dimensional setting.

The aim of this thesis is to investigate the performance of the deep hedging approach in the presence of frictions, namely market impact. For this purpose, we consider a financial model with permanent price impact as studied in [BLZ16]. We find that the neural network hedging is able to approximate the optimal hedging strategy well even in the presence of market impact. We have implemented the algorithm in Python using PyTorch.

The thesis is organized as follows: First, we recall the setup of the original deep hedging framework (as proposed in [Büh+19]), namely we provide the theoretical framework for pricing and hedging using convex risk measures in discrete-time markets with frictions. Section 3 introduces the approach of using neural networks to approximate optimal hedging strategies and presents the theoretical arguments derived in [Büh+19]. In particular, we also point out practical considerations regarding the optimization of neural networks. In Section 4, several numerical experiments are performed similar to the ones presented in [Büh+19], but assuming that the dynamics follow the Black-Scholes model instead of the Heston model. These experiments demonstrate the feasibility and accuracy of the proposed method. In Section 5, we consider a market with frictions, namely, permanent price impact. We briefly outline the impact rule (as studied in [BLZ16]) and derive a perfect hedging strategy in a special case. The hedging strategy is then used as benchmark in our numerical experiments, which indicates the usefulness of our approach even in the presence of frictions.

2 Pricing and Hedging in Financial Markets with Frictions

We start by introducing the setting we work in, that is, discrete-time financial markets with frictions. This is followed by a brief introduction of convex risk measures with some interesting examples. We also draw a connection between convex risk measures and the so-called optimized certainty equivalents. Finally, we discuss the pricing and hedging of derivatives using convex risk measures.

2.1 Discrete-time Markets with Frictions

We consider a discrete-time financial market model with a fixed, finite time horizon $T > 0$. The market consists of $d \in \mathbb{N}$ assets which can be traded at times $0 = t_0 < t_1 < \dots < t_n = T$, $n \in \mathbb{N}$. Throughout, we work on a discrete probability space $\Omega = \{\omega_1, \omega_2, \dots, \omega_N\}$, $N \in \mathbb{N}$, and let \mathbb{P} be a probability measure on Ω such that $\mathbb{P}(\{\omega\}) > 0$ for every $\omega \in \Omega$. We denote by $\mathcal{X} := \{X: \Omega \rightarrow \mathbb{R}\}$ the set of all real-valued random variables over Ω .

Moreover, we model the flow of information in our market using a stochastic process $(I_k)_{k=0, \dots, n}$ where $I_k \in \mathbb{R}^r$, $r \in \mathbb{N}$, contains all the new market information which becomes available at time t_k . This information can include any kind of trading signal such as ask and bid prices of the assets, market costs, risk limits, news, etc. Essentially, this setup allows us to include any kind of quantitative information that a trader might find useful to derive a trading decision. Let $\mathbb{F} = (\mathcal{F}_k)_{k=0, \dots, n}$ be the natural filtration generated by $I = (I_k)_{k=0, \dots, n}$, i.e. $\mathcal{F}_k := \sigma(I_0, \dots, I_k)$. Thus, \mathcal{F}_k represents all information available up to t_k . We can now define our financial market.

Definition 2.1. A *financial market* is an \mathbb{R}^d -valued \mathbb{F} -adapted stochastic process $S = (S^1, \dots, S^d)$, i.e. S_k is \mathcal{F}_k -measurable for all $k = 0, \dots, n$. The value S_k^i denotes the (mid-)price of the i -th asset at time t_k .

Remark 2.2. It is important to mention that the assets in our market do not necessarily need to be primary assets like equities but can also be secondary assets such as liquid options, see for instance Example 2.13.

In what follows, we assume that there exists a 0th asset S^0 which plays the role of a (locally) *riskless bond*, see [FS16, Example 5.5]. We write $\bar{S} := (S^0, S)$. To simplify the notation, we further assume that the *spot rates* are zero, i.e. $S^0 \equiv 1$.

Definition 2.3. A *trading strategy* is an \mathbb{F} -adapted \mathbb{R}^{1+d} -valued process

$$\bar{\delta} = (\delta^0, \delta) = (\delta_k^0, \delta_k^1, \delta_k^2, \dots, \delta_k^d)_{k=0, 1, \dots, n-1}$$

where δ_k^i denotes the number of assets S^i held at time t_k .

Notice that δ reflects the amount of assets we hold in the market S , while δ^0 corresponds to the amount of money we hold in S^0 . Since we assume $S_0 \equiv 1$, we can interpret δ^0 as the amount of money we hold in cash. For simplicity, we assume in the following that we have $\delta_{-1} = \delta_n = 0$ for every trading strategy $\bar{\delta}$. In particular, this implies that we always fully liquidate our entire portfolio invested in the market at terminal time T .

Definition 2.4. The *wealth process* $(X_m^{\bar{\delta}})_{m=0,\dots,n}$ associated with a trading strategy $\bar{\delta}$ is given by

$$X_m^{\bar{\delta}} := \bar{\delta}_m \cdot \bar{S}_m := \sum_{i=0}^d \delta_m^i S_m^i \quad \text{for } m = 0, \dots, n.$$

The *gains process* associated with a trading strategy $\bar{\delta}$ is defined as

$$(\bar{\delta} \bullet \bar{S})_m := \sum_{k=0}^{m-1} \bar{\delta}_k \cdot (\bar{S}_{k+1} - \bar{S}_k) = \sum_{k=0}^{m-1} \sum_{i=0}^d \delta_k^i (S_{k+1}^i - S_k^i)$$

for $m = 0, \dots, n$.

The gains process represents the accumulated gains and losses resulting from our trading strategy $\bar{\delta}$ and the asset price fluctuations up to time t_m . In the following, we also sometimes write $(\bar{\delta} \bullet \bar{S})_T$ for the total accumulated gains and losses up until time T , i.e. $(\bar{\delta} \bullet \bar{S})_T := (\bar{\delta} \bullet \bar{S})_n$. In the same manner, we may occasionally write S_T for the terminal price instead of S_n , i.e. $S_T := S_n$.

Remark 2.5. Note that we assumed S_0 to be constant, and hence we have

$$\begin{aligned} (\bar{\delta} \bullet \bar{S})_m &= \sum_{k=0}^{m-1} \sum_{i=0}^d \delta_k^i (S_{k+1}^i - S_k^i) \\ &= \sum_{k=0}^{m-1} \sum_{i=1}^d \delta_k^i (S_{k+1}^i - S_k^i) = \sum_{k=0}^{m-1} \delta_k \cdot (S_{k+1} - S_k) =: (\delta \bullet S)_m \end{aligned}$$

for all $m = 0, \dots, n$. This means that the gains process does not depend on δ^0 (and S^0). And in what follows we thus write $(\delta \bullet S)_m$ instead of $(\bar{\delta} \bullet \bar{S})_m$ (and also $(\delta \bullet S)_T$ instead of $(\bar{\delta} \bullet \bar{S})_T$).

As is typical, we also assume that all trading strategies $\bar{\delta}$ are *self-financed*, i.e. $\bar{\delta}_{k-1} \cdot \bar{S}_k = \bar{\delta}_k \cdot \bar{S}_k$ for all $k = 0, \dots, n$. To wit, at each trading time, our position is always rearranged in such a way that its present value stays the same. In particular, this ensures that any trading strategy can be financed without any external cash in- or outflows. And

this implies that the value of our position is completely determined by the gains process and the initial investment.

Proposition 2.6. *Let $\bar{\delta} = (\delta^0, \delta)$ be a trading strategy. Then, $\bar{\delta}$ is self-financed if and only if*

$$X_m^{\bar{\delta}} = \bar{\delta}_0 \cdot \bar{S}_0 + (\delta \bullet S)_m$$

holds for all $m = 0, \dots, n$.

Proof. See [FS16, Proposition 5.7]. □

Remark 2.7. In fact, one can show that for a trading strategy $\bar{\delta} = (\delta^0, \delta)$, the process δ^0 is uniquely determined by the d -dimensional process δ and the initial investment $\bar{\delta}_0 \cdot \bar{S}_0$. Consequently, if we are given an initial wealth $p_0 \in \mathbb{R}$ and an arbitrary \mathbb{F} -adapted d -dimensional process δ , then there is a unique process δ^0 such that $\bar{\delta} := (\delta^0, \delta)$ is a self-financing trading strategy and $p_0 = \bar{\delta}_0 \cdot \bar{S}_0$. See also [FS16, Remark 5.8].

In light of the above remark, we consider the following alternative definition of (self-financing) trading strategies.

Definition 2.8. Let $\delta = (\delta_k^1, \delta_k^2, \dots, \delta_k^d)_{k=0,1,\dots,n-1}$ be an \mathbb{F} -adapted \mathbb{R}^d -valued process and $p_0 \in \mathbb{R}$. Then we say that δ is a *trading strategy with initial wealth p_0* . The set of all trading strategies is denoted by \mathcal{H} .

Moreover, we denote by $X^\delta := X^{\bar{\delta}}$ the wealth process of a trading strategy δ with initial wealth p_0 , where $\bar{\delta}$ is the corresponding (uniquely determined) self-financing strategy as constructed in Remark 2.7.

Remark 2.9. It is important to point out that in the following we only consider unconstrained trading strategies \mathcal{H} . However, it would be possible to also include additional trading constraints and thus consider a restricted set of trading strategies $\mathcal{H}^{\text{constrained}} \subset \mathcal{H}$. This would be more reasonable since it would allow us to model liquidity, asset availability or trading restrictions. For a thorough treatment, we refer to [Büh+19, p. 5].

In real financial markets, any trading activity is usually subject to transaction costs. These transaction costs include, for example, brokers' commissions and bid-ask spreads. In our setup, we model trading costs as follows. We consider *cost functions* $c_k: \Omega \times \mathbb{R}^d \rightarrow \mathbb{R}_+$, $k = 0, \dots, n$, which satisfy the following properties:

- (i) *normalized:* $c_k(\omega, 0) = 0$ for all $\omega \in \Omega$,
- (ii) *adapted:* $\omega \mapsto c_k(\omega, y)$ is \mathcal{F}_k -measurable for all $y \in \mathbb{R}^d$,

(iii) *upper semi-continuous*: $\limsup_{y' \rightarrow y} c_k(\omega, y') \leq c_k(\omega, y)$ for all $\omega \in \Omega$ and $y \in \mathbb{R}^d$.

Now, if we want to acquire a position $y \in \mathbb{R}^d$ in the market S at time t_k , then this transaction will incur the (random) additional cost $c_k(y)$. In total, the transaction cost for a trading strategy $\delta \in \mathcal{H}$ up until time T is thus

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

Example 2.10. This setup allows for a variety of different types of transaction costs. For example, we may consider proportional transaction cost by defining

$$c_k(y) := \sum_{i=1}^d c_k^i S_k^i |y^i|, \quad y \in \mathbb{R}^d,$$

where $c_k^i > 0$ and y^i denotes the number of shares we want to acquire in the i -th asset. Similarly, we can consider fixed transaction costs. Let $\varepsilon > 0$ and $c_k^i > 0$, then we can define

$$c_k(y) := \sum_{i=1}^d c_k^i \mathbb{1}_{|y^i| \geq \varepsilon}, \quad y \in \mathbb{R}^d.$$

Remark 2.11. In our setting it would also be possible to model market impact in the sense that the asset distribution is affected by our trading. We will take a closer look at this in Section 5.

A major subject of interest in mathematical finance is the study of *financial derivatives*, i.e. securities whose value is dependent on other reference securities. These reference securities are often referred to as *underlyings*.

Definition 2.12. A real-valued random variable $Z \in \mathcal{X}$ is referred to as (*European*) *contingent claim*. A European contingent claim $Z \in \mathcal{X}$ is called a *derivative (of the underlying assets S^0, S^1, \dots, S^d)* if Z is measurable with respect to the σ -algebra generated by the price process $(S_k)_{k=0, \dots, n}$.

A contingent claim has the following financial interpretation: it is an asset which yields the amount $Z(\omega)$ at time T depending on the market scenario $\omega \in \Omega$. The time T is often referred to as the *maturity* of Z . Needless to say, maturities before the end of the time horizon T are also possible, however in what follows we only consider contingent claims that expire at time T .

While the payoff of contingent claims can depend on any arbitrary event, the payoff of derivatives is *derived* from the price process $(S_k)_{k=0,\dots,n}$. For instance, *options* are financial contracts which give the buyer the opportunity to buy or sell the underlying asset(s) for certain conditions which typically depend on the price process of the underlyings. The two most important examples of such options are call and put options.

Example 2.13. A *call option* is a derivative defined by

$$Z := (S_T^i - K)^+ := \max\{0, S_T^i - K\},$$

where K is a fixed price, called the *strike price*. This can be interpreted as follows, the buyer of a call option has the right, but not the obligation, to buy the asset S^i at time T for the strike price K .

In contrast, a *put option* gives the right, but not the obligation, to sell the asset S^i at time T for the strike price K . This yields the payoff

$$Z := (K - S_T^i)^+ := \max\{0, K - S_T^i\}.$$

Call and put options are a very common type of derivatives traded in a real-world financial market. Note that we only consider European-style options here, i.e. options which can only be exercised at maturity T . Of course, there also exist options which are more *exotic*, and we refer to [FS16, Chapter 5.3 and 5.4] for examples.

2.2 Convex Risk Measures

Complete markets are idealized financial markets where every contingent claim can be perfectly replicated by using dynamic trading strategies. In such a complete market, the price of a contingent claim is unique and coincides with the wealth process of a perfect replication strategy.

However, in our setting we want to take a more realistic view of the financial world by taking into account market frictions such as transactions costs and market impact. In these *incomplete* financial market models, a contingent claim typically will not admit a perfect hedge and the question of pricing and hedging becomes much more involved. While the concept of *superhedging* provides a method to eliminate all risks, the required cost is usually too high. For example, it was shown in [SSC95] that in the Black-Scholes model with proportional transaction cost the superhedging strategy for a call option is the trivial strategy of buying the underlying stock and holding until maturity and therefore the price of the call is equal to the initial price of the stock which is unreasonably expensive.

The underlying problem is that the principle of no-arbitrage in markets with frictions

is often insufficient to determine meaningful prices for financial derivatives. In such cases, it is therefore necessary to look for an alternative approach. From the perspective of the seller of a financial derivative, it makes sense to keep the risk generated by the sale of the derivative within an acceptable range. A popular concept for measuring risk are the so-called *convex risk measures*.

Definition 2.14. A mapping $\rho: \mathcal{X} \rightarrow \mathbb{R}$ is called a *convex risk measure* if it satisfies the following conditions for every $X, Y \in \mathcal{X}$.

- (i) *Monotonicity*: If $X \geq Y$, then $\rho(X) \leq \rho(Y)$.
- (ii) *Convexity*: If $\lambda \in [0, 1]$, then $\rho(\lambda X + (1 - \lambda)Y) \leq \lambda\rho(X) + (1 - \lambda)\rho(Y)$.
- (iii) *Cash Invariance*: If $c \in \mathbb{R}$, then $\rho(X + c) = \rho(X) - c$.

For $X \in \mathcal{X}$, we interpret $\rho(X)$ as the *risk associated with the financial position X* and we call X *acceptable* if $\rho(X) \leq 0$. We say that ρ is *normalized* if $\rho(0) = 0$.

The financial interpretation of the monotonicity condition can be stated as follows: the risk of a position decreases if the payoff profile becomes more favourable. The idea behind the convexity condition is that diversification should not increase the risk. To wit, assume we are presented with two different investment strategies leading to the possible future outcome X and Y given the resources available to us. If we choose to diversify by spending only the fraction λ of our resources on the first strategy and the remaining to the second, we obtain the payoff $\lambda X + (1 - \lambda)Y$. Hence, convexity ensures that diversification is not penalized but encouraged. Finally, the cash invariance condition is motivated by interpretation of $\rho(X)$ as the amount of capital required to be added to the position X in order to make it acceptable. Hence, if cash in the amount of $c \in \mathbb{R}$ is added to the position in the risk-free asset, the capital requirement is reduced by the same amount. In particular, this means that $\rho(X + \rho(X)) = 0$ for any $X \in \mathcal{X}$. For a detailed introduction to the theory of risk measures, we refer to [FS16, Chapter 4].

To get a better understanding, let us take a look at two straightforward examples.

Example 2.15 ([FS16, Example 4.8]). The most pessimistic investor will consider the potential loss which occurs in the worst-case scenario. This idea leads to the following risk measure defined by

$$\rho_{\max}(X) := - \inf_{\omega \in \Omega} X(\omega), \quad X \in \mathcal{X},$$

which is known as the *worst-case risk measure*. It is easy to check that ρ_{\max} is indeed a

convex risk measure. Note that for any other convex risk measure ρ , we obtain

$$\rho(X) \leq \rho\left(-\inf_{\omega \in \Omega} X(\omega)\right) = \rho_{\max}(X) \quad \text{for all } X \in \mathcal{X}.$$

In this sense, ρ_{\max} is the most conservative measure of risk. In contrast, consider

$$\rho_{\text{neutral}}(X) = \mathbb{E}[-X], \quad X \in \mathcal{X},$$

which represents the risk preference of a risk-neutral investor. It is clear that ρ_{neutral} is also a convex risk measure.

Next, we consider a special class of risk measures which can be defined in terms of loss functions.

Lemma 2.16 ([Büh+19, Lemma 3.7]). *Let $\ell: \mathbb{R} \rightarrow \mathbb{R}$ be a loss function, i.e. continuous, non-decreasing and convex. Then*

$$\rho(X) := \inf_{w \in \mathbb{R}} \{w + \mathbb{E}[\ell(-X - w)]\}, \quad X \in \mathcal{X} \quad (2.1)$$

defines a convex risk measure.

Proof. To see this, we simply need to verify that the desired properties are satisfied. Indeed, let $X, Y \in \mathcal{X}$ and observe

- (i) Monotonicity: Suppose $X \geq Y$, then $-X - w \leq -Y - w$ holds for any $w \in \mathbb{R}$ and because ℓ is non-decreasing, we obtain

$$\rho(X) = \inf_{w \in \mathbb{R}} \{w + \mathbb{E}[\ell(-X - w)]\} \leq \inf_{w \in \mathbb{R}} \{w + \mathbb{E}[\ell(-Y - w)]\} = \rho(Y).$$

- (ii) Cash invariance: Let $c \in \mathbb{R}$, then

$$\begin{aligned} \rho(X + c) &= \inf_{w \in \mathbb{R}} \{w + \mathbb{E}[\ell(-(X + c) - w)]\} \\ &= -c + \inf_{w \in \mathbb{R}} \{(w + c) + \mathbb{E}[\ell(-X - (w + c))]\} \\ &= -c + \rho(X). \end{aligned}$$

- (iii) Convexity: Let $\lambda \in [0, 1]$. Then, for each $w \in \mathbb{R}$ and $w_1, w_2 \in \mathbb{R}$ such that $w = \lambda w_1 + (1 - \lambda)w_2$, we obtain

$$\begin{aligned} \ell(-(\lambda X + (1 - \lambda)Y) - w) &= \ell(\lambda(-X - w_1) + (1 - \lambda)(-Y - w_2)) \\ &\leq \lambda \ell(-X - w_1) + (1 - \lambda) \ell(-Y - w_2), \end{aligned}$$

using the convexity of ℓ . It follows that

$$\begin{aligned}
& \rho(\lambda X + (1 - \lambda)Y) \\
&= \inf_{w \in \mathbb{R}} \{w + \mathbb{E}[\ell(-(\lambda X + (1 - \lambda)Y) - w)]\} \\
&\leq \inf_{w_1, w_2 \in \mathbb{R}} \{\lambda w_1 + (1 - \lambda)w_2 + \mathbb{E}[\lambda \ell(-X - w_1) + (1 - \lambda)\ell(-Y - w_2)]\} \\
&= \inf_{w_1 \in \mathbb{R}} \{\lambda w_1 + \mathbb{E}[\lambda \ell(-X - w_1)]\} + \inf_{w_2 \in \mathbb{R}} \{(1 - \lambda)w_2 + \mathbb{E}[(1 - \lambda)\ell(-Y - w_2)]\} \\
&= \lambda \rho(X) + (1 - \lambda)\rho(Y).
\end{aligned}$$

□

Remark 2.17. If $u: \mathbb{R} \rightarrow \mathbb{R}$ is a *utility function*, i.e. strictly concave, strictly increasing and continuous, then we can construct a loss function $\ell: \mathbb{R} \rightarrow \mathbb{R}$ by defining $\ell(x) := -u(-x)$ for all $x \in \mathbb{R}$. In particular, choosing the loss function this way, (2.1) coincides with the so-called *optimized certainty equivalents*. The optimized certainty equivalent (OCE) is a decision theoretic criterion based on utility functions, and a detailed study about its connection with convex risk measures is given in [BT07]. In light of this, we will also refer to convex risk measures which take the form (2.1) for some loss function ℓ as optimized certainty equivalents.

Definition 2.18. Let $\lambda > 0$. We call

$$\rho_{\text{entropic}}^\lambda(X) := \frac{1}{\lambda} \log(\mathbb{E}[\exp(-\lambda X)]) \quad (2.2)$$

the *entropic risk measure*.

Remark 2.19 ([Büh+19, Example 3.8]). It turns out that the entropic risk measure is an optimized certainty equivalent. Indeed, if we set $\ell(x) := \exp(\lambda x) - \frac{1+\log(\lambda)}{\lambda}$, $x \in \mathbb{R}$, then one can show that (2.1) can be solved explicitly and coincides with 2.2, see also [BT07, Example 2.1]. In particular, Lemma 2.16 implies that the entropic risk measure is indeed a convex risk measure.

One intuitive approach to manage the risk of a position is to control the probability that a loss occurs.

Definition 2.20. The *Value at Risk at level $\gamma \in (0, 1)$* of a financial position $X \in \mathcal{X}$ is given by

$$\begin{aligned}
\text{VaR}_\gamma(X) &:= \inf\{m \in \mathbb{R}: \mathbb{P}(X < -m) \leq \gamma\} \\
&= \inf\{m \in \mathbb{R}: \mathbb{P}(X + m < 0) \leq \gamma\}.
\end{aligned}$$

This can be interpreted as follows, $\text{VaR}_\gamma(X)$ is the smallest amount of money such that it keeps the probability of a negative outcome below the level γ if we add it to the position X . It is easy to see that VaR_γ satisfies monotonicity and cash-invariance. However, it is not convex and may discourage diversification by incentivizing to concentrate risk on an event with very small probability. For a specific example of this behaviour, we refer to [FS16, Example 4.46]. Another drawback of using the Value-at-Risk criterion is that it only takes the probability of a loss into account, but not the size of the loss. In cases where a loss is to be avoided at all cost this may be reasonable, but usually other measures of risk will be more appropriate. A popular risk measure which fixes these problems is *Average Value at Risk*.

Definition 2.21. For a financial position $X \in \mathcal{X}$, we define its *Average Value at Risk at level α* as

$$\text{AVaR}_\alpha(X) := \frac{1}{\alpha} \int_0^\alpha \text{VaR}_\gamma(X) d\gamma \quad (2.3)$$

for some $\alpha \in (0, 1]$.

While Value at Risk only takes the probability of a loss into consideration, Average Value at Risk considers the expected shortfall in case a loss occurs and is thus more sensitive to the shape of the tail loss distribution.

Proposition 2.22. For $\alpha \in (0, 1)$ and $X \in \mathcal{X}$, we have

$$\text{AVaR}_\alpha(X) = \frac{1}{\alpha} \mathbb{E} [(\text{VaR}_\alpha(X) + X)^-] + \text{VaR}_\alpha(X). \quad (2.4)$$

Moreover, if X has a continuous distribution we even have

$$\text{AVaR}_\alpha(X) = \frac{1}{\alpha} \mathbb{E} [-X \mathbb{1}_{\{-X \geq \text{VaR}_\alpha(X)\}}] = \mathbb{E} [-X \mid -X \geq \text{VaR}_\alpha(X)]. \quad (2.5)$$

Proof. This is an immediate consequence of [FS16, Proposition 4.51 and Corollary 4.54]. \square

The formulas (2.4) and (2.5) are also the reason why Average Value at Risk (AVaR) is sometimes called *expected shortfall (ES)* or *conditional value at risk (CVaR)*, respectively. In addition, it turns out that AVaR_α can be also represented as optimized certainty equivalent by setting the loss function as $\ell(x) := \frac{1}{1-\alpha} \max\{x, 0\}$ for all $x \in \mathbb{R}$ in (2.1), i.e.

$$\text{AVaR}_\alpha(X) = \inf_{w \in \mathbb{R}} \left\{ w + \frac{1}{\alpha} \mathbb{E} [(-X - w)^+] \right\}.$$

A proof of this fact can be found in [FS16, Proposition 4.51]. In particular, Lemma 2.16 implies that Average Value at Risk is indeed a convex risk measure.

Remark 2.23. An advantage of using Average Value at Risk is that the parameter $\alpha \in (0, 1]$ can be used to control the level of risk-aversion. For $\alpha = 1$, one can prove that $\int_0^1 \text{VaR}_\gamma(X) d\gamma = \mathbb{E}[-X]$ which is an immediate consequence of [FS16, Lemma A.23]. In particular, this implies

$$\text{AVaR}_1(X) = \int_0^1 \text{VaR}_\gamma(X) d\gamma = \mathbb{E}[-X] = \rho_{\text{neutral}}(X),$$

for all $X \in \mathcal{X}$, see also [FS16, p. 234]. On the other hand, for α close to 0, notice that for $X \in \mathcal{X}$ we have

$$\lim_{\gamma \searrow 0} \text{VaR}_\gamma(X) = \inf\{m \in \mathbb{R} : \mathbb{P}(X + m < 0) \leq 0\} = -\inf_{\omega \in \Omega} X(\omega) = \rho_{\max}(X),$$

which is the worst-case risk measure ρ_{\max} , see Example 2.15. Hence, it is reasonable to define

$$\text{AVaR}_0(X) := \text{VaR}_0(X) := \rho_{\max}(X), \quad X \in \mathcal{X},$$

see also [FS16, Remark 4.50]. To sum up, Average Value at Risk ranges from risk-neutral to extremely risk-averse for different levels of α , i.e. risk-neutral for α close to 1 and very risk-averse for α close to 0.

Of course, there exist plenty of other convex risk measures which are also used in practice beyond the ones presented here. For other interesting examples and a more thorough treatment of the theory of risk measures, we refer to [FS16, Chapter 4].

2.3 Pricing and Hedging Using Convex Risk Measures

In this section, we will focus on how convex risk measures can be used to price financial derivatives in markets with frictions.

Recall that the cash invariance condition of convex risk measures ensures that the value $\rho(X)$ can be interpreted as a cash requirement to make the position $X \in \mathcal{X}$ acceptable. Hence, when selling a financial derivative $Z \in \mathcal{X}$ we could charge $\rho(-Z)$ as a price since this guarantees that our position stays acceptable. However, this price would usually be too high since it excludes the possibility of hedging our position, i.e. trading in the market to decrease our risk exposure. For example, if we follow the (self-financing) trading strategy $\delta \in \mathcal{H}$, the final position at time T becomes $-Z + (\delta \bullet S)_T$ and the associated risk is $\rho(-Z + (\delta \bullet S)_T)$. Note that following the trading strategy δ can be subject to transaction cost such that the risk actually happens to be $\rho(-Z + (\delta \bullet S)_T - C_T(\delta))$. Nevertheless, this risk may very well be much less than without trading, supposing that

δ hedges reasonably well. From the seller's point of view, it now makes sense to find a trading strategy which minimizes this risk. This idea gives rise to the following definition.

Definition 2.24. Let $X \in \mathcal{X}$ and ρ be a convex risk measure. Then, we call

$$\pi(X) := \inf_{\delta \in \mathcal{H}} \rho(X + (\delta \bullet S)_T - C_T(\delta)) \quad (2.6)$$

the *minimal cost of risk for X under the risk measure ρ* . Moreover, any trading strategy $\delta^* \in \mathcal{H}$ that is a minimizer of the optimization problem (2.6) will be referred to as *optimal hedging strategy for X under the risk measure ρ* .

For the remainder of this section, let us fix a convex risk measure ρ and consider the corresponding π as defined in (2.6). The value $\pi(-Z)$ can be interpreted as the risk we take by selling a financial derivative Z and following the optimal strategy δ^* minimizing the risk (if it exists). Furthermore, we observe that π itself is a convex risk measure under suitable conditions on the cost function.

Proposition 2.25 ([Büh+19, Proposition 3.2]). *Assume that the cost function C_T is convex, i.e. $y \mapsto c_k(\omega, y)$ is convex for each $\omega \in \Omega$ and $k = 0, \dots, n$. Then, the functional π is a convex risk measure.*

Proof. See [Büh+19, Proposition 3.2]. □

In particular, since π is a convex risk measure itself, $\pi(-Z)$ can also be interpreted as a cash requirement. This simply means that $\pi(-Z)$ is the minimal amount of capital which has to be added to the position $-Z$ to make the terminal position acceptable under the risk measure ρ , if the seller is able to hedge optimally. Therefore, it would be reasonable to take $\pi(-Z)$ as the minimal price for selling a financial derivative Z . However, this would ignore the fact that if we decide not to sell Z (our position would then be 0) there might be other trading opportunities which can be exploited. After all, we did not assume that $\pi(0) = 0$. In general, it is possible that $\pi(0) < 0$, which essentially means that if we trade optimally, we can withdraw $-\pi(0) > 0$ amount of cash while our position still stays acceptable. For instance, consider ρ_{neutral} from Example 2.15 and a security S which has positive expected return. A strategy δ which simply buys and holds the stock until time T will have negative risk, and in particular we have $\pi(0) < 0$.

Thus, a more valid approach would be to charge the amount of cash such that the seller becomes indifferent between selling the financial derivative Z and not doing so. This motivates the following definition.

Definition 2.26. For any financial derivative $Z \in \mathcal{X}$, we call

$$p(Z) := \pi(-Z) - \pi(0)$$

the indifference price of Z .

In particular, by cash-invariance, the indifference price $p(Z)$ satisfies the relation $\pi(-Z + p(Z)) = \pi(0)$. In addition, charging the indifference price is further justified by the following observation.

Lemma 2.27 ([Büh+19, Lemma 3.3]). *Suppose there are no transaction costs $C_T \equiv 0$. Let $Z \in \mathcal{X}$ be a financial derivative which is attainable, i.e. there exists a trading strategy $\delta^* \in \mathcal{H}$ and initial value $p_0 \in \mathbb{R}$ such that $Z = p_0 + (\delta^* \bullet S)_T$. Then the indifference price coincides with the initial value, i.e. $p(Z) = p_0$.*

Proof. First note that $\delta \mapsto (\delta \bullet S)_T$ is, in fact, linear (for each ω). Using this and the assumption that Z is attainable, we obtain

$$-Z + (\delta \bullet S)_T = -(p_0 + (\delta^* \bullet S)_T) + (\delta \bullet S)_T = ([\delta - \delta^*] \bullet S)_T - p_0.$$

for all $\delta \in \mathcal{H}$. In particular, cash-invariance of ρ implies

$$\rho(-Z + (\delta \bullet S)_T) = p_0 + \rho([\delta - \delta^*] \bullet S)_T.$$

By taking the infimum over $\delta \in \mathcal{H}$ on both sides it follows that

$$\pi(-Z) = \inf_{\delta \in \mathcal{H}} \rho(-Z + (\delta \bullet S)_T) = p_0 + \inf_{\delta \in \mathcal{H}} \rho([\delta - \delta^*] \bullet S)_T.$$

Notice that $\mathcal{H} - \delta^* = \mathcal{H}$, i.e. for every $\delta \in \mathcal{H}$ there exists some $\delta' \in \mathcal{H}$ such that $\delta = \delta' - \delta^*$. Consequently, we have

$$\inf_{\delta \in \mathcal{H}} \rho([\delta - \delta^*] \bullet S)_T = \inf_{\delta \in \mathcal{H}} \rho((\delta \bullet S)_T) = \pi(0)$$

and inserting this into the equation above yields $\pi(-Z) = p_0 + \pi(0)$ and thus $p(Z) = \pi(-Z) - \pi(0) = p_0$. \square

3 Deep Learning Approximation

As mentioned in the introduction, an efficient method for the numerical solution of the optimization problem (2.6) is presented in [Büh+19]. The basic idea is to parameterize strategies by neural networks and to exploit the efficient optimization algorithms known for neural networks. We start by briefly recalling the definition of (feed forward) neural networks and their universal approximation property. The following presentation is largely based on [Büh+19, Section 4].

3.1 Neural Networks and Universal Approximation

Artificial neural network were inspired by the structure of neural networks in the brain. Roughly speaking, a large number of neurons are connected to each other in a complex communication network which enables the brain to perform complex computations. Artificial neural networks are commonly described as graphs, where the nodes correspond to neurons while edges represent the links between them. Feed forward neural networks are networks in which the underlying graph does not contain cycles. A mathematical definition of feed forward neural networks is given below.

Definition 3.1. Let $L, N_0, N_1, \dots, N_L \in \mathbb{N}$ and $\sigma: \mathbb{R} \rightarrow \mathbb{R}$. We refer to σ as the *activation function*, to L as the *number of layers* and to N_0, N_L and N_ℓ with $\ell = 1, \dots, L-1$ as the *dimension of the input, output, and ℓ -th hidden layer*, respectively. Moreover, for any $\ell = 1, \dots, L$, let $A^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $b^\ell \in \mathbb{R}^{N_\ell}$ and define the corresponding affine function $W^\ell: \mathbb{R}^{N_{\ell-1}} \rightarrow \mathbb{R}^{N_\ell}$ by $W(x) := A^\ell x + b^\ell$. Then, the function

$$F: \mathbb{R}^{N_0} \rightarrow \mathbb{R}^{N_L}, \quad x \mapsto (W^L \circ \sigma \circ W^{L-1} \circ \sigma \circ \dots \circ W^2 \circ \sigma \circ W^1)(x),$$

where σ is applied componentwise, is called a *(feed forward) neural network*. We refer to $A^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ as the *weight matrices* and to $b^\ell \in \mathbb{R}^{N_\ell}$ as the *bias vectors*. The value of A_{ij}^ℓ is interpreted as the *weight of the edge connecting the node i of layer $\ell-1$ to the node j of layer ℓ* , and b_j^ℓ can be interpreted as the *bias of the node j in layer ℓ* .

Thus, by definition, a (feed forward) neural network is simply a finite concatenation of affine functions and the activation function σ . Figure 1 shows a possible architecture of a (feed forward) neural network.

Remark 3.2. In principle, the activation function σ can be chosen arbitrarily. However, some activation functions work better than others, depending on the task. The question of how to choose the activation is still a very active area of research, and there are not many definitive guidelines. Nevertheless, there are a variety of widely used activation functions

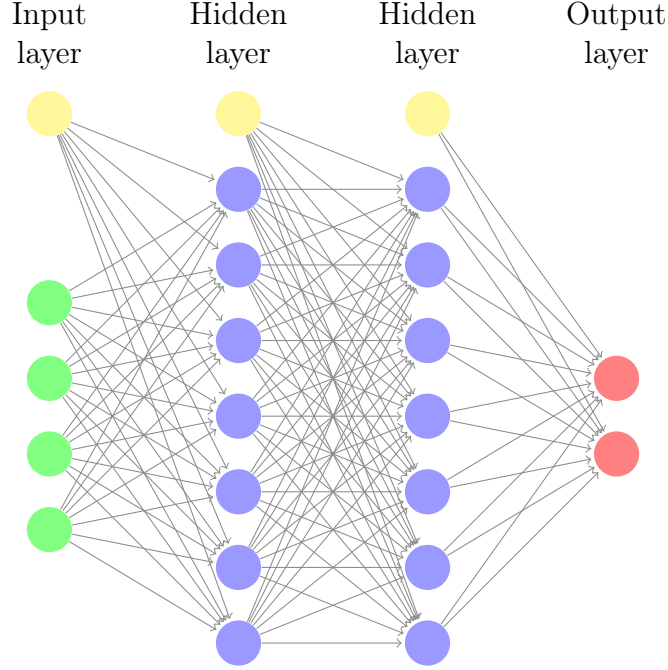


Figure 1: Visualization of a neural network with $L = 3$ layers, input dimension $N_0 = 4$ and output dimension $N_3 = 2$.

which work well in practice. A good default choice is the rectified linear unit, which is defined as $\text{ReLU}(x) := x^+ := \max\{0, x\}$. Others include the logistic sigmoid function $\sigma(x) := 1/(1 + e^{-x})$ or the hyperbolic tangent function $\tanh(x) := (e^x - e^{-x})/(e^x + e^{-x})$. For a more detailed description of other types of activation functions, we refer to [GBC16, Chapter 6.3].

Definition 3.3. For $d_0, d_1 \in \mathbb{N}$ and $\sigma: \mathbb{R} \rightarrow \mathbb{R}$, we denote by $\mathcal{N}_{\infty, d_0, d_1}^\sigma$ the set of neural networks with input dimension d_0 , output dimension d_1 and activation function σ .

It is well known that neural networks are universal approximators in the sense that they are capable of approximating any multivariate function arbitrarily well. To illustrate this, let us state the following two classical results.

Theorem 3.4 (Universal Approximation). *Let σ be continuous. Then $\mathcal{N}_{\infty, d_0, 1}^\sigma$ is dense in the space of continuous function $C(\mathbb{R}^{d_0})$ mapping from \mathbb{R}^{d_0} to \mathbb{R} for the topology of uniform convergence on compact sets if and only if σ is not a polynomial.*

Moreover, for any nonnegative finite measure μ on $(\mathbb{R}^{d_0}, \mathcal{B}(\mathbb{R}^{d_0}))$ with compact support and $1 \leq p < \infty$, we have that the set $\mathcal{N}_{\infty, d_0, 1}^\sigma$ is dense in $L^p(\mathbb{R}^{d_0}, \mathcal{B}(\mathbb{R}^{d_0}), \mu)$ if and only if σ is not a polynomial.

Proof. This was first proven in [Les+93, Theorem 1 and Proposition 2]. We also refer to the survey paper by [Pin99]. \square

In fact, Theorem 3.4 was proven for neural networks with a single hidden layer, which are often referred to as *shallow networks*. This implies however that the above result does not address the question of depth and efficiency. The (single) hidden layer may require infeasibly many nodes to approximate a function reasonably well. In practice, usually deeper neural networks with many hidden layers are used, as they seem to have more expressivity than shallow networks, see [GBC16, Chapter 6.4] and the references given there. The benefits of deep neural networks over shallow ones are also discussed in the review paper [GRK20]. It also provides a comprehensive overview of the large variety of approximation results for neural networks, such as approximation rates for classical function spaces and approximation results for special neural network architectures.

Note that Theorem 3.4 easily generalizes to $\mathcal{N}_{\infty, d_0, d_1}^\sigma$ with $d_1 > 1$ because an \mathbb{R}^{d_1} -valued neural network can be simply constructed from \mathbb{R} -valued neural networks. For notational convenience, let us now fix an activation function σ (see Remark 3.2 for examples) and in what follows we write $\mathcal{N}_{\infty, d_0, d_1} := \mathcal{N}_{\infty, d_0, d_1}^\sigma$.

3.2 Approximating Hedging Strategies with Deep Neural Networks

By Theorem 3.4, neural networks can approximate any continuous function with arbitrary accuracy. This heavily suggests to use neural networks to approximate the optimal hedging strategy for (2.6). Hence, the basic idea here is to consider trading strategies $\delta = (\delta_k)_{k=0, \dots, n-1} \in \mathcal{H}$ where δ_k is given by the output of a neural network $F_k \in \mathcal{N}_{\infty, d_0, d_1}$ at each time step t_k . There are several things that need to be discussed regarding this approach.

Let us start with the following question: What is the input and output dimension d_0 and d_1 ? Since δ_k is d -dimensional, it is clear that we should have $d_1 = d$ for the output dimension. But what about the input dimension? Or more generally, what should the input of F_k even be? To answer this, first recall that trading strategies are \mathbb{F} -adapted. This simply means that δ_k is \mathcal{F}_k -measurable for all $k = 0, \dots, n-1$ and hence, there must exist some measurable function $f_k: \mathbb{R}^{r(k+1)} \rightarrow \mathbb{R}^d$ such that $\delta_k = f_k(I_0, \dots, I_k)$ for each $k = 0, \dots, n-1$. In that sense δ_k can be written as a function of the information available in our market up until time t_k . With this in mind, it seems reasonable to take I_0, \dots, I_k as the input for the neural network F_k which represents the trading strategy at time t_k , i.e. $\delta_k = F_k(I_0, \dots, I_k)$. In the presence of market frictions, however, it is also useful to include the previous position δ_{k-1} as an additional input for the neural network F_k . The reason for this is that the transaction costs to acquire a new position δ_k is given by $c_k(\delta_k - \delta_{k-1})$ and thus depends on the previous position δ_{k-1} . Similar effects also hold for other market frictions, such as market impact.

To summarize, the idea is to find an optimal trading strategy which can be represented by a neural network with input I_0, \dots, I_k and δ_{k-1} at each time step t_k , i.e. $\delta_k = F_k(I_0, \dots, I_k, \delta_{k-1})$. Note that the input δ_{k-1} for the neural network F_k at time t_k is given by the output of the neural network F_{k-1} at time t_{k-1} . In particular, the entire trading strategy $\delta = (\delta_k)_{k=0, \dots, n-1}$ can be represented by one deep neural network with a semi-recurrent structure.

However, there is a problem that occurs in practice. The set $\mathcal{N}_{\infty, d_0, d_1}$ contains all neural networks with arbitrarily many hidden layers and nodes. This is of course impossible to represent in a computer. Besides, to efficiently find the optimal weights, we require a parameterization for the set of neural networks. Therefore, let us consider a sequence of subsets approximating the set $\mathcal{N}_{\infty, d_0, d_1}$.

Definition 3.5. We call a sequence of subsets $\{\mathcal{N}_{M, d_0, d_1}\}_{M \in \mathbb{N}}$ of $\mathcal{N}_{\infty, d_0, d_1}$ an *approximating sequence* of $\mathcal{N}_{\infty, d_0, d_1}$ if it satisfies the following properties:

- (i) $\mathcal{N}_{M, d_0, d_1} \subset \mathcal{N}_{M+1, d_0, d_1}$ for all $M \in \mathbb{N}$,
- (ii) $\bigcup_{M \in \mathbb{N}} \mathcal{N}_{M, d_0, d_1} = \mathcal{N}_{\infty, d_0, d_1}$,
- (iii) for any $M \in \mathbb{N}$, the set $\mathcal{N}_{M, d_0, d_1}$ can be parameterized with $\Theta_{M, d_0, d_1} \subset \mathbb{R}^{q_M}$ for some $q_M \in \mathbb{N}$, i.e. $\mathcal{N}_{M, d_0, d_1} = \{F^\theta : \theta \in \Theta_{M, d_0, d_1}\}$.

Example 3.6 ([Büh+19, Remark 4.3]). The natural choice for an approximating sequence of $\mathcal{N}_{\infty, d_0, d_1}$ is to take all neural networks with a *fixed architecture*. For this purpose, consider non-decreasing sequences $\{L^{(M)}\}_{M \in \mathbb{N}}$ and $\{N_1^{(M)}\}_{M \in \mathbb{N}}, \dots, \{N_{L^{(M)}-1}^{(M)}\}_{M \in \mathbb{N}}$. Then, for $\mathcal{N}_{M, d_0, d_1}$, we can take the set of neural networks with $L^{(M)}$ hidden layers and dimension $N_1^{(M)}, \dots, N_{L^{(M)}-1}^{(M)}$ for each layer, respectively. It is easily seen that by adding an additional layer to a neural network, it is able to represent more functions. The same holds when increasing the amount of nodes per layer. In particular, this implies that the first condition (i) holds, i.e. $\mathcal{N}_{M, d_0, d_1} \subset \mathcal{N}_{M+1, d_0, d_1}$. If we further suppose that the number of hidden layers and their respective number of nodes is strictly increasing in M , then the second condition (ii) is also satisfied. Moreover, $\mathcal{N}_{M, d_0, d_1}$ can be naturally parameterized by the weight matrices A^ℓ and bias vectors b^ℓ because of its fixed architecture. The parameter is given by

$$\theta = ((A^\ell, b^\ell))_{\ell=1}^{L^{(M)}} \in \bigtimes_{\ell=1}^{L^{(M)}} \left(\mathbb{R}^{N_\ell^{(M)} \times N_{\ell-1}^{(M)}} \times \mathbb{R}^{N_\ell^{(M)}} \right) =: \Theta_{M, d_0, d_1},$$

where Θ_{M, d_0, d_1} can be simply identified with \mathbb{R}^{q_M} for $q_M = \sum_{\ell=1}^{L^{(M)}} N_\ell^{(M)} N_{\ell-1}^{(M)} + N_\ell^{(M)}$. Thus, this choice for $\{\mathcal{N}_{M, d_0, d_1}\}_{M \in \mathbb{N}}$ indeed yields an approximating sequence of $\mathcal{N}_{\infty, d_0, d_1}$.

In the following, let us fix the approximating sequence $\{\mathcal{N}_{M,d_0,d_1}\}_{M \in \mathbb{N}}$ of $\mathcal{N}_{\infty,d_0,d_1}$ and its parameterization Θ_{M,d_0,d_1} by taking the one which is described in Example 3.6 above.

Definition 3.7. We denote by

$$\begin{aligned}\mathcal{H}_M &:= \left\{ (\delta_k)_{k=0,\dots,n-1} \in \mathcal{H} : \delta_k = F_k(I_0, \dots, I_k, \delta_{k-1}), F_k \in \mathcal{N}_{M,r(k+1)+d,d} \right\} \\ &= \left\{ (\delta_k)_{k=0,\dots,n-1} \in \mathcal{H} : \delta_k = F^{\theta_k}(I_0, \dots, I_k, \delta_{k-1}), \theta_k \in \Theta_{M,r(k+1)+d,d} \right\}\end{aligned}$$

the set of all trading strategies which can be represented by neural networks. Note that each trading strategy $\delta \in \mathcal{H}_M$ can be parameterized by a parameter

$$\theta = \bigotimes_{k=0}^{n-1} \theta_k \in \bigotimes_{k=0}^{n-1} \Theta_{M,r(k+1)+d,d} =: \Theta_M$$

where θ_k is the corresponding parameter for the neural network F_k at time step t_k and we denote by $\delta^\theta = (\delta_k^\theta)_{k=0,\dots,n-1}$ the trading strategy with $\delta_k^\theta = F^{\theta_k}(I_0, \dots, I_k, \delta_{k-1}^\theta)$ for $k = 0, \dots, n-1$.

Remark 3.8 ([Büh+19, Remark 4.6]). Suppose that S is an (\mathbb{F}, \mathbb{P}) -Markov process and $Z = g(S_T)$ for some measurable function $g: \mathbb{R}^d \rightarrow \mathbb{R}$. In this setting, it turns out that the optimal hedging strategy in (2.6) can be written as $\delta_k = f_k(I_k, \delta_{k-1})$ for some $f_k: \mathbb{R}^{r+d} \rightarrow \mathbb{R}^d$, i.e. the optimal strategy only depends on the current information. In that sense, including past information does not improve the optimal hedging strategy and thus, in this setting, it makes more sense to consider neural network trading strategies δ with $\delta_k = F_k(I_k, \delta_{k-1})$. For a proof of such a result, we refer to [Hin70, Theorem 18.4].

Remark 3.9 ([Büh+19, Remark 4.5]). Note that it would also be possible to consider a *truly* recurrent structure by taking a single neural network which approximates the optimal hedge at all time steps t_k . Of course, this single neural network would then have to take an additional input to be capable of differentiating between the time steps, e.g. take the time to maturity $T - t_k$ as added input. Thus, our goal would be to find an optimal parameter θ for the hedging strategy δ given by $\delta_k = F^\theta(I_k, \delta_{k-1}, T - t_k)$ for $k = 0, \dots, n-1$. See [GBC16, Chapter 10] for more details on recurrent neural networks.

Definition 3.10. For any $X \in \mathcal{X}$, we define

$$\begin{aligned}\pi_M(X) &:= \inf_{\delta \in \mathcal{H}_M} \rho(X + (\delta \bullet S)_T - C_T(\delta)) \\ &= \inf_{\theta \in \Theta_M} \rho(X + (\delta^\theta \bullet S)_T - C_T(\delta^\theta))\end{aligned}\tag{3.1}$$

as the minimal cost of risk for X under the risk measure ρ which can be attained by neural network trading strategies in \mathcal{H}_M .

Thus, by computing $\pi_M(X)$ instead of $\pi(X)$, the infinite-dimensional problem of finding an optimal hedging strategy in (2.6) is essentially reduced to a finite-dimensional problem of finding an optimal parameter $\theta \in \Theta_M$ for our neural network trading strategy in (3.1). The following proposition shows that the neural network price $\pi_M(-Z) - \pi_M(0)$ is indeed capable of approximating the exact indifference price $p(Z)$.

Proposition 3.11 ([Büh+19, Proposition 4.9]). *For any $X \in \mathcal{X}$, we have*

$$\lim_{M \rightarrow \infty} \pi_M(X) = \pi(X).$$

Proof. Step 1. First, observe that $(\mathcal{H}_M)_{M \in \mathbb{N}}$ is monotone increasing in the sense that $\mathcal{H}_M \subset \mathcal{H}_{M+1} \subset \mathcal{H}$ for all $M \in \mathbb{N}$. And hence by definition of π_M , see (3.1), it follows that $(\pi_M(X))_{M \in \mathbb{N}}$ is a monotone decreasing sequence for all $X \in \mathcal{X}$. Thus, we only need to prove that for any $\varepsilon > 0$, there exists some $M \in \mathbb{N}$ such that $\pi_M(X) \leq \pi(X) + \varepsilon$.

For this purpose, let us fix an $X \in \mathcal{X}$ and some $\varepsilon > 0$. By definition of $\pi(X)$, see (2.6), there exists a trading strategy $\delta \in \mathcal{H}$ such that

$$\rho(X + (\delta \bullet S)_T - C_T(\delta)) \leq \pi(X) + \frac{\varepsilon}{2}. \quad (3.2)$$

It is not clear however that there exists a large enough $M \in \mathbb{N}$ such that the strategy δ is contained in \mathcal{H}_M . Be that as it may, if we can find an $M \in \mathbb{N}$ such that there exists another trading strategy $\delta' \in \mathcal{H}_M$ which approximates δ in the sense that we have

$$\rho(X + (\delta' \bullet S)_T - C_T(\delta')) \leq \rho(X + (\delta \bullet S)_T - C_T(\delta)) + \frac{\varepsilon}{2}, \quad (3.3)$$

then we can combine this with (3.2) to obtain

$$\rho(X + (\delta' \bullet S)_T - C_T(\delta')) \leq \pi(X) + \varepsilon.$$

And since $\delta' \in \mathcal{H}_M$ for some $M \in \mathbb{N}$, using (3.1), we conclude that

$$\pi_M(X) \leq \rho(X + (\delta' \bullet S)_T - C_T(\delta')) \leq \pi(X) + \varepsilon$$

which is what we wanted to show.

Step 2. It remains to prove the existence of a strategy $\delta' \in \mathcal{H}_M$ satisfying (3.3) for some $M \in \mathbb{N}$.

Remember that $\delta \in \mathcal{H}$ is an \mathbb{F} -adapted stochastic process which means that δ_k is \mathcal{F}_k -measurable for all $k = 0, \dots, n-1$. Hence, there exists a measurable function $f_k: \mathbb{R}^{r(k+1)} \rightarrow \mathbb{R}^d$ such that $\delta_k = f_k(I_0, \dots, I_k)$ for all $k = 0, \dots, n-1$. We denote by $f_k^i: \mathbb{R}^{r(k+1)} \rightarrow \mathbb{R}$ the i -th component of f_k . By Theorem 3.4 (for $p = 1$), $\mathcal{N}_{\infty, r(k+1), 1}$ is

dense in $L^1(\mathbb{R}^{r(k+1)}, \mu)$ for any finite measure μ on $(\mathbb{R}^{r(k+1)}, \mathcal{B}(\mathbb{R}^{r(k+1)}))$ with compact support. With this in mind, our goal is now to approximate f_k^i with neural networks using Theorem 3.4. For this purpose, consider the image measure $\mu := \mathbb{P} \circ (I_0, \dots, I_k)^{-1}$. Then,

$$\begin{aligned} \|f_k^i\|_{L^1(\mathbb{R}^{r(k+1)}, \mu)} &= \int_{\mathbb{R}^{r(k+1)}} |f_k^i| d\mu = \int_{\Omega} |f_k^i(I_0, \dots, I_k)| d\mathbb{P} \\ &= \sum_{m=1}^N |f_k^i(I_0(\omega_m), \dots, I_k(\omega_m))| \cdot \mathbb{P}(\{\omega_m\}) < \infty \end{aligned}$$

where we used that Ω is finite and hence $f_k^i \in L^1(\mathbb{R}^{r(k+1)}, \mu)$ for any $i = 1, \dots, d$. In particular, the finiteness of Ω implies that μ must have compact support. Consequently, by Theorem 3.4, there exists a sequence of neural networks $(F_{k,n}^i)_{n \in \mathbb{N}} \subset \mathcal{N}_{\infty, r(k+1), 1}$, which converges to f_k^i in $L^1(\mathbb{R}^{r(k+1)}, \mu)$. Therefore, we have

$$\begin{aligned} \sum_{m=1}^N |F_{k,n}^i(I_0(\omega_m), \dots, I_k(\omega_m)) - f_k^i(I_0(\omega_m), \dots, I_k(\omega_m))| \cdot \mathbb{P}(\{\omega_m\}) \\ = \int_{\mathbb{R}^{r(k+1)}} |F_{k,n}^i - f_k^i| d\mu \longrightarrow 0 \quad \text{for } n \rightarrow \infty \end{aligned} \quad (3.4)$$

for all $i = 1, \dots, d$ and $k = 0, \dots, n-1$. Thus, we can consider the sequence of trading strategies $(\delta^n)_{n \in \mathbb{N}}$ defined by $\delta_k^n := F_{k,n}^i(I_0, \dots, I_k)$ for $k = 0, \dots, n-1$. Then, our assumption that $\mathbb{P}(\{\omega\}) > 0$ for all $\omega \in \Omega$ and (3.4) implies

$$\lim_{n \rightarrow \infty} \delta_k^n(\omega) = \delta_k(\omega) \quad \text{for all } \omega \in \Omega \text{ and } k = 0, \dots, n-1. \quad (3.5)$$

In particular, we have that $(\delta^n \bullet S)_T$ converges to $(\delta \bullet S)_T$ (for all ω) as $n \rightarrow \infty$. Note that while δ^n is a trading strategy constructed from neural networks, it is not contained in \mathcal{H}_M for any $M \in \mathbb{N}$. This is because at each time step, the input of the neural network $F_{k,n}$ does *not* include the additional argument δ_{k-1}^n , see Definition 3.7. Nevertheless, it is easy to see that for each δ^n there exists a corresponding trading strategy $\hat{\delta}^n$ such that its output at each time step is given by a neural network which takes $I_0, \dots, I_k, \hat{\delta}_{k-1}^n$ as an input *and* has exactly the same output as $F_{k,n}(I_0, \dots, I_k)$. In particular, $\hat{\delta}^n$ satisfies (3.5) as well, and we have $\hat{\delta}^n \in \mathcal{H}_M$ for some $M \in \mathbb{N}$. Thus, without loss of generality, we consider $\hat{\delta}^n$ instead of δ^n and we write $\delta^n := \hat{\delta}^n$.

Step 3. The proof is completed by showing that there exists some $n \in \mathbb{N}$ such that $\delta' := \delta^n$ satisfies (3.3).

Recall that we required the cost function C_T to be upper semi-continuous, i.e.

$$\limsup_{y' \rightarrow y} C_T(\omega, y') \leq C_T(\omega, y) \quad \text{for all } \omega \in \Omega \text{ and } y \in \mathbb{R}^d.$$

In particular, this combined with (3.5) implies

$$\begin{aligned} \liminf_{n \rightarrow \infty} (X + (\delta^n \bullet S)_T - C_T(\delta^n)) &= X + (\delta \bullet S)_T - \limsup_{n \rightarrow \infty} C_T(\delta^n) \\ &\geq X + (\delta \bullet S)_T - C_T(\delta). \end{aligned} \tag{3.6}$$

Furthermore, in our finite setting, it turns out that ρ is continuous. Indeed, since Ω is finite, we can identify \mathcal{X} with \mathbb{R}^N and thus ρ can be viewed as a convex function $\rho: \mathbb{R}^N \rightarrow \mathbb{R}$. However, it is a well known fact that any convex function $\rho: \mathbb{R}^N \rightarrow \mathbb{R}$ must be continuous. For a proof, we refer to [Roc70, Theorem 10.1]. It now follows from the continuity of ρ that

$$\begin{aligned} \liminf_{n \rightarrow \infty} \rho(X + (\delta^n \bullet S)_T - C_T(\delta^n)) &= \rho\left(\liminf_{n \rightarrow \infty} (X + (\delta^n \bullet S)_T - C_T(\delta^n))\right) \\ &\leq \rho(X + (\delta \bullet S)_T - C_T(\delta)) \end{aligned}$$

where we used (3.6) and the monotonicity of ρ in the second step. As a result, there must exist some (sufficiently large) $n \in \mathbb{N}$ such that

$$\rho(X + (\delta^n \bullet S)_T - C_T(\delta^n)) \leq \rho(X + (\delta \bullet S)_T - C_T(\delta)) + \frac{\varepsilon}{2},$$

i.e. δ^n satisfies (3.3). This completes the proof. \square

The above result is due to [Büh+19, Proposition 4.9] and the proof above is basically just a slightly more structured (and extended) version of the proof given there. However, [Büh+19, Proposition 4.9] only holds for activation functions σ which are bounded and nonconstant. Since we used another version of the Universal Approximation Theorem 3.4, our result is true for any continuous σ which is not a polynomial. Note that the finiteness of Ω is used several times in the proof. Nevertheless, Proposition 3.11 can be generalized to also hold for general Ω (under suitable conditions). In fact, all the results prior to this remain valid for general Ω if we assume the usual appropriate integrability conditions and additionally conditions on the chosen convex risk measure ρ (only for Proposition 3.11).

3.3 Learning the Optimal Hedging Strategy

Proposition 3.11 provides a strong theoretical foundation for using neural networks to represent hedging strategies. For any $X \in \mathcal{X}$, the minimal cost of risk $\pi(X)$ can be

approximated with $\pi_M(X)$ arbitrarily well by choosing M large enough. From a practical point of view, computing $\pi(X)$ is an infinite-dimensional problem of finding an optimal hedging strategy while computing $\pi_M(X)$ only requires to find an optimal (finite-dimensional) parameter $\theta \in \Theta_M$. Recall that $\theta \in \Theta_M$ simply represents the values of the weights of the neural network. In fact, another big advantage of using neural networks is that a specialized set of optimization techniques have been developed to find a (close-to) optimal set of weights θ . We will briefly introduce these optimization techniques and how they can be applied in our setting. For a deeper discussion on optimization techniques for neural network training, we refer to [GBC16, Chapter 8].

To demonstrate the basic concept, we will concentrate on the setting where ρ is given by the entropic risk measure $\rho_{\text{entropic}}^\lambda$ for some $\lambda > 0$, see Example 2.18. In that case, the optimization problem (3.1) becomes

$$\begin{aligned}\pi_M(X) &= \inf_{\theta \in \Theta_M} \frac{1}{\lambda} \log \mathbb{E} [\exp (-\lambda X + (\delta^\theta \bullet S)_T - C_T(\delta^\theta))] \\ &= \frac{1}{\lambda} \log \underbrace{\inf_{\theta \in \Theta_M} \mathbb{E} [\exp (-\lambda X + (\delta^\theta \bullet S)_T - C_T(\delta^\theta))]}_{=: J^*(\theta)}\end{aligned}\quad (3.7)$$

where we used that the natural logarithm is a strictly increasing function. The goal is thus to minimize $J^*(\theta)$. However, the true underlying distribution of the real market S (and the position X) is in general unknown. In that sense, minimizing $J^*(\theta)$ is not a classical optimization problem, but a *machine learning problem*. In these kinds of problems we only have a training set of samples given, and it is assumed that the training set is sampled i.i.d. from the real market distribution. In practice, this usually means having historical data of the real market. Note that this is also the reason why we do not require a classical model of the market and why the approach is completely data-driven.

Thus, in what follows, we will assume we have a training dataset with $N_{\text{samples}} < N$ samples given by a subset $\{\omega_1^{\text{data}}, \dots, \omega_{N_{\text{samples}}}^{\text{data}}\} =: \Omega^{\text{data}} \subset \Omega$, in the sense that we observed the entire information process $\bar{I}(\omega)$ where $\bar{I} := (I_0, \dots, I_n)$ in the sampled scenarios $\omega \in \Omega^{\text{data}}$. Instead of minimizing $J^*(\theta)$, the idea is now to replace the expectation in (3.7) by an expectation over the samples weighted by their respective probabilities

$$J(\theta) := \sum_{m=1}^{N_{\text{samples}}} \exp (-\lambda X(\omega_m^{\text{data}}) + (\delta^\theta \bullet S)_T(\omega_m^{\text{data}}) - C_T(\delta^\theta)(\omega_m^{\text{data}})) \frac{N}{N_{\text{samples}}} \mathbb{P}(\{\omega_m^{\text{data}}\}). \quad (3.8)$$

The hope is that minimizing the *empirical risk* $J(\theta)$ leads to a significant reduction of the *risk* $J^*(\theta)$ (our true objective) as well. This process of minimizing $J(\theta)$ instead of

$J^*(\theta)$ is known as *empirical risk minimization*. The most effective modern approaches to minimizing $J(\theta)$ are based on gradient descent. The idea is to start with some initial guess $\theta^{(0)}$ and then iteratively take a small step in the opposite direction of the gradient

$$\theta^{(j+1)} := \theta^{(j)} - \eta_j \nabla J(\theta^{(j)}), \quad j \in \mathbb{N}, \quad (3.9)$$

for some step size η_j , $j \in \mathbb{N}$. Under suitable assumptions on J and the step size sequence $(\eta_j)_{j \in \mathbb{N}}$, it is guaranteed that $\theta^{(j)}$ converges to a local minimum of J as $j \rightarrow \infty$. However, to apply this algorithm, we need an efficient way to compute the gradient $\nabla J(\theta^{(j)})$. There are two key ingredients that enable this calculation effectively.

- ▷ The first is the observation that the objective $J(\theta)$ in (3.8) decomposes as a sum over the training samples. Computing this sum accurately is very expensive because it requires evaluating the model on each sample in the entire training dataset. Alternatively, in each iteration j of the gradient descent algorithm (3.9), we can estimate the sum $J(\theta^{(j)})$ by randomly (uniformly) sampling a small number $N_{\text{batch}} \ll N_{\text{samples}}$ of examples $\{\omega_1^{(j)}, \dots, \omega_{N_{\text{batch}}}^{(j)}\} =: \Omega^{(j)} \subset \Omega^{\text{data}}$ from the training dataset and then take the average over these examples weighted by their respective probabilities

$$\widehat{J(\theta)} := \sum_{m=1}^{N_{\text{batch}}} \exp(-\lambda X(\omega_m^{(j)}) + (\delta^\theta \bullet S)_T(\omega_m^{(j)}) - C_T(\delta^\theta)(\omega_m^{(j)})) \frac{N}{N_{\text{batch}}} \mathbb{P}(\{\omega_m^{(j)}\}). \quad (3.10)$$

By using this estimate instead, we obtain the most basic form of the *stochastic gradient descent* algorithm which is, along with its variants, the most used optimization algorithm for deep learning. For an explicit algorithm, we refer to [GBC16, Chapter 8.3]. The subset $\Omega^{(j)} \subset \Omega^{\text{data}}$ is often referred to as a *(mini-)batch*, while the number N_{batch} is referred to as the *batch size*. The step size sequence $(\eta_j)_{j \in \mathbb{N}}$ is also called the *learning rate* and, in practice, it is important to slowly decrease the learning rate to guarantee convergence, see [GBC16, Chapter 8.3.1]. Moreover, to implement stochastic gradient descent, the dataset is usually shuffled once and decomposed into batches. Then, the dataset (which is divided into batches) is passed through multiple times, i.e. an update step (3.9) is performed for each batch using the estimate (3.10), see [GBC16, Algorithm 8.1]. These pass-throughs are often referred to as *epochs*. During the first epoch the stochastic estimate (3.10) is an unbiased estimate of the risk (3.7), but in the following epochs the estimate becomes biased because the samples have already been used (see also [GBC16, Chapter 8.1.3]). However, the introduced bias is usually negligibly small compared to the decreased training error when using additional epochs.

- ▷ The second ingredient is that the gradient w.r.t. θ of the hedging strategy, which is given by the output of a neural network F^θ , can be computed efficiently using the so-called *back-propagation* algorithm. The basic idea is that, because of the special structure of neural networks, one can recursively apply the chain rule to efficiently calculate the gradients, see [Büh+19, p. 19]. We refer to [GBC16, Chapter 6.5] for more details.

Thus, by combining the stochastic gradient descent algorithm to optimize θ and back-propagation to efficiently compute the desired gradient, we obtain a powerful method to optimize our neural network. In fact, under suitable conditions, stochastic gradient descent even guarantees convergence to a global minimum, see for instance [Ber+21, Theorem 1.14] and the references given there. However, these conditions usually include convexity, which is not satisfied in our setting and hence the algorithm may only converge to a local minimum. Nevertheless, it is common belief that finding the global minimum (of the training set) is in practice irrelevant, since it often leads to *overfitting*. Moreover, research indicates that most local minima are equivalent and yield similar performance, at least given a sufficiently large neural network. In particular, the probability of finding a “bad” local minimum (in the sense that it has high value) decreases quickly with the size of the neural network. For more details, we refer to [Cho+15] (see also [GBC16, Chapter 8.2.2]).

Note that in this brief introduction of optimization techniques for neural network training, we assumed that the risk measure ρ is given by the entropic risk measure $\rho_{\text{entropic}}^\lambda$. However, the optimization techniques presented here can be easily generalized to other risk measures. In particular, they can be applied to OCE risk measures (2.1) which requires to additionally optimize over the parameter $w \in \mathbb{R}$. For a fuller treatment, we refer to [Büh+19, Chapter 4.3]. In fact, it is even possible to extend our setup to the case of general convex risk measures: By using the robust representation of convex risk measures (see [FS16, Chapter 4.2]), one can approximate (2.6) by a minimax problem over neural networks. This construction is shown in [Büh+19, Chapter 4.5], where it is argued that the objective functional of this approximate problem is also open to the deep learning optimization techniques presented above, namely stochastic gradient descent and back-propagation. We will not discuss this extension in more detail here. But the crucial point which we want to make here is that our setup is not just limited to the entropic risk measure, but can be applied to any convex risk measure.

4 Numerical Experiments

In the previous two sections, we talked about how to price and hedge financial derivatives in markets with frictions and how to numerically approximate the optimal hedging strategy using neural networks. In this section, we will now demonstrate the feasibility of this approach in numerical experiments.

We start by briefly recalling the Black-Scholes model and introduce the setting for our experiments. Then, we take a look at how the neural network hedging strategy compares to the benchmark delta hedging strategy and how different risk preferences affect the learned neural network hedging strategy.

4.1 Black-Scholes Model

We first consider a one-dimensional Black-Scholes model which is given by the following dynamics

$$S_0 = s_0 \quad \text{and} \quad S_t = S_0 \exp \left(\left(\mu - \frac{1}{2} \sigma^2 \right) t + \sigma W_t \right), \quad t \in [0, T],$$

where $\mu \in \mathbb{R}$, $\sigma, s_0 > 0$ and W_t is a one-dimensional Brownian motion. The constant s_0 is simply the initial value, while μ is referred to as the *drift* and σ as the *volatility*. The Black-Scholes model is the standard reference model in the context of option pricing. The usefulness of this model stems from the fact that the risk-neutral price of a call option can be computed explicitly using the famous *Black-Scholes formula*. Indeed, recall that the payoff of a call option with maturity T is given by $Z = (S_T - K)^+$ for some strike price $K > 0$. Then, the *unique* risk-neutral price of the call option at time t is given by

$$u(t, S_t) := S_t \Phi(d_+) - K e^{-r(T-t)} \Phi(d_-), \quad t \in [0, T], \quad (4.1)$$

where $\Phi: \mathbb{R} \rightarrow [0, 1]$ denotes the cumulative distribution function of a standard normal random variable and

$$d_{\pm} = \frac{\log(S_t/K) + (r \pm \sigma^2/2)(T-t)}{\sigma \sqrt{T-t}}.$$

Note that r refers to the interest rate which we assumed to be zero, i.e. $r = 0$. Moreover, assuming a frictionless market and continuous rebalancing, the option payoff can be replicated perfectly when charging the risk-neutral price (4.1) at time 0. In fact, the replication strategy is given by

$$\Delta_t := \partial_x u(t, S_t) = \Phi(d_+), \quad t \in [0, T], \quad (4.2)$$

where u is defined as in (4.1) and $\partial_x u$ denotes the partial derivative of u with the respect to the second component. The replication strategy can be interpreted as follows: by holding Δ_t units of the underlying asset, it is possible to completely hedge-out any risk associated with selling the option when charging the risk-neutral price at time 0. This strategy is also referred to as *delta hedging*. For a complete derivation of the above formulas and a deeper discussion of the Black-Scholes model, we refer to [FS16, Ch. 5.7]. Note however that the delta hedging strategy requires the ability to trade continuously.

4.2 Setting and Implementation

We now introduce the setting for our numerical experiments. We consider the discretized one-dimensional Black-Scholes model as defined above and thus set $d = 1$. For the model parameters, we take $\mu = 0$, $\sigma = 0.2$ and $s_0 = 100$. Moreover, we choose a time horizon of 30 trading days with daily rebalancing, i.e. $T = 30/365$ and $n = 30$ where the trading dates are given by $t_k = k/365$ for $k = 0, \dots, n$. Further, we assume no transaction costs, i.e. $C_T \equiv 0$. Note that sample trajectories of the discretized Black-Scholes model can be efficiently generated using various numerical methods, see [Gla03, Chapter 3.1 and 3.2] for more details. In particular, generating independent samples of S can be viewed as sampling from a uniform distribution on a very large but finite probability space Ω . Finally, we choose $\rho = \text{AVaR}_\alpha$ as our risk measure, as this allows to easily model different risk preferences by adjusting the parameter $\alpha \in (0, 1)$ appropriately, see Remark 2.23.

Our goal is now to hedge a call option with maturity T and strike price $K = s_0$, i.e. $Z = (S_T - K)^+$. Recall that a call option can be replicated perfectly by following the delta hedging strategy (4.2) continuously. However, this is impossible in real market situations, since we can only rebalance our portfolio at discrete time points. This coincides with our current setting where we only allow trading once a day, which will thus inevitably lead to a (small) hedging error. Nonetheless, we may take a discretized version of the delta hedging strategy, i.e. $\delta_k^{\text{BS}} := \Delta_{t_k}$ for $k = 0, \dots, n-1$ where Δ is defined as in (4.2), and additionally charging the risk-neutral price $q := u(0, s_0)$ where u is given by (4.1). This serves as our benchmark, and in the following we will refer to it as the *model hedge*.

Let us now describe the modelling choices for our deep learning approach to hedging, as explained in Section 3. Remember that the hedging strategy is given as the output of a semi-recurrent neural network, i.e. the number of units held at time t_k is represented by $\delta_k^\theta = F^{\theta_k}(I_k, \delta_{k-1}^\theta)$ where F^{θ_k} is a feed forward neural network. Note that θ_k parameterizes the neural network F^{θ_k} and simply represents the values of the weight matrices and biases (which are different for each k). In the following, we will choose $I_k = \log(S_k)$ as the network input. Furthermore, for each neural network F^{θ_k} we choose the following

dimension parameters $L = 3$, $N_0 = 2d$, $N_1 = N_2 = d + 15$ and $N_3 = d$, see Definition 3.1, and we fix $\text{ReLU}(x) = x^+$ as the activation function, see Remark 3.2.

After specifying our modelling choices, we can now use the algorithm as described in Section 3.3 to learn a (close-to) optimal hedging strategy. We first generate $N_{\text{samples}} = 4 \cdot 10^6$ sample trajectories of our (simulated) market, which will serve as our training dataset Ω^{data} . As mentioned above, each sample $S(\omega_m)$ with $\omega_m \in \Omega^{\text{data}}$ can be viewed as sampling from a uniform distribution on a very large but finite probability space $\Omega = \{\omega_1, \dots, \omega_N\}$. In particular, we have $\mathbb{P}(\{\omega_m\}) = \frac{1}{N}$ for all $m = 1, \dots, N$. Note that for each observed scenario $\omega \in \Omega^{\text{data}}$, the corresponding payoff is simply given by $Z(\omega) := (S_T(\omega) - K)^+$. To hedge the call option, we can now use this training dataset and apply the methodology illustrated in Section 3.3 to learn a (close-to) optimal parameter θ for (3.1) with $X = -Z$. To compare the performance of the neural network hedging strategy with the model hedge, we will further simulate another set of 10^6 sample paths and use them as an out-of-sample test dataset. Recall that we choose Average Value at Risk to optimize our neural network hedging strategy, while the methodology in Section 3.3 is described for the entropic risk measure. However, it is straightforward to adapt (3.7) accordingly using Proposition 2.5.

We have implemented the algorithm in Python using PyTorch to build and train the neural networks. In particular, PyTorch includes an automatic differentiation system which makes computing the gradients with the back-propagation algorithm extremely convenient, see [Pas+19]. Note that we use the so-called *Adam algorithm* to optimize our neural network. Adam is a slightly more sophisticated version of the stochastic gradient descent algorithm with an adaptive learning rate, which was introduced in [KB14]. For the algorithm, we choose a learning rate of $\eta = 0.005$ and a batch size of $N_{\text{batch}} = 256$. In addition, we include the technique of *batch normalization* in each layer of each network before applying the activation function. Batch normalization is a method that normalizes the input of each layer by recentering and rescaling. This can greatly accelerate the training of neural networks. For more details, we refer to the original paper [IS15].

4.3 Numerical Results

We now present the results of our numerical experiments. Recall that our objective is to sell a call option $Z = (S_T - K)^+$. For our benchmark, i.e. the model hedge, we first charge the risk-neutral price q and follow the delta hedging strategy δ^{BS} which leads to the terminal payoff $q - Z + (\delta^{\text{BS}} \bullet S)_T$. Ideally this terminal payoff would always be 0, but because of the discretization there will be a (small) hedging error. Note that the risk-neutral price can be easily computed by applying the Black-Scholes formula (4.1) which yields $q \approx 2.2871$.

To obtain our deep hedging strategy, we train the semi-recurrent neural network with our implemented algorithm using our training data set of (simulated) sample paths for 40 epochs. We denote by θ the parameter of the trained neural network, i.e. the values of the weights and biases. Moreover, we denote by δ^θ the corresponding neural network hedging strategy, and we set

$$p_0^\theta := \rho(-Z + (\delta^\theta \bullet S)_T),$$

which is an approximation of (3.1). Recall that the indifference price is defined as $p(Z) = \pi(-Z) - \pi(0)$. One can show that if the market S is a martingale, then we have $\pi(0) = \rho(0)$. For a proof, we refer to [Büh+19, Proposition 3.10]. Remember that our Black-Scholes market is indeed a martingale because we chose $\mu = 0$. Since our risk measure of choice is Average Value at Risk, i.e. $\rho = \text{AVaR}_\alpha$, an easy computation shows that we also have $\rho(0) = 0$. Thus, in our setting the indifference price is simply $p(Z) = \pi(-Z)$ and in particular, Proposition 3.11 implies that p_0^θ is an approximation to the indifference price $p(Z)$. Hence, if we want to sell Z , we should charge p_0^θ as a price and then follow the trading strategy δ^θ . This yields the terminal payoff $p_0^\theta - Z + (\delta^\theta \bullet S)_T$. We can now use the out-of-sample test dataset to evaluate the terminal hedging error by computing $p_0^\theta - Z + (\delta^\theta \bullet S)_T$ for each sample in the test set. However, to obtain a better comparison with the benchmark, we will actually consider the payoff $q - Z + (\delta^\theta \bullet S)_T$, i.e. instead of charging p_0^θ we ask for the risk-neutral price q and then follow δ^θ . We can now plot the hedging error for each test sample in a histogram and compare the neural network hedging strategy with the model hedge. This is shown in Figure 2 where we used $\rho = \text{AVaR}_\alpha$ with risk preference $\alpha = 0.5$. We observe that the hedging error for δ^{BS} and δ^θ is very similar. In particular, the algorithm computed $p_0^\theta = 2.5598$ as a price. We can also compare the strategies δ_k^θ and δ_k^{BS} at a fixed time-point $t_k = 15/365$ conditional on S_k . If we want to make such a comparison, however, we face the following problem: The model hedge strategy $\delta_k^{\text{BS}} = \Delta_{t_k} = \partial_x u(t_k, S_{t_k})$ only depends on the current price (and some fixed model parameters). On the other hand, the neural network strategy $\delta_k^\theta = F^{\theta_k}(I_k, \delta_{k-1}^\theta)$ takes an additional input δ_{k-1}^θ . Hence, to compare the strategies, let us use a much simpler network structure which only takes the current information as an input, namely $\delta_k^\theta = F^{\theta_k}(I_k)$. With this, we can now plot δ_k^θ and δ_k^{BS} conditional on S_k on a grid for a fixed-time point $t_k = 15/365$. This is illustrated in Figure 3. It is interesting to note that the hedging performance for this simpler structure is very similar compared to the semi-recurrent network structure, see Figure 4. By all means, this is also expected since we assumed no market frictions such as transaction costs or market impact, as explained in the beginning of Section 3.2.

Note that the choice of risk preference, i.e. $\alpha = 0.5$, is crucial for the behaviour of the

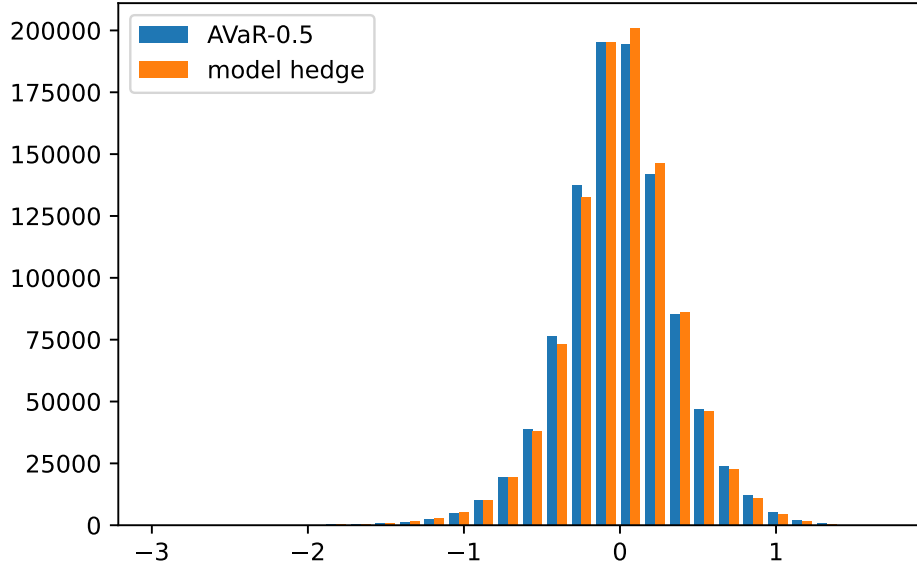


Figure 2: PnL comparison of model hedge and deep hedge associated to $\rho = \text{AVaR}_\alpha$ with risk preference $\alpha = 0.5$.

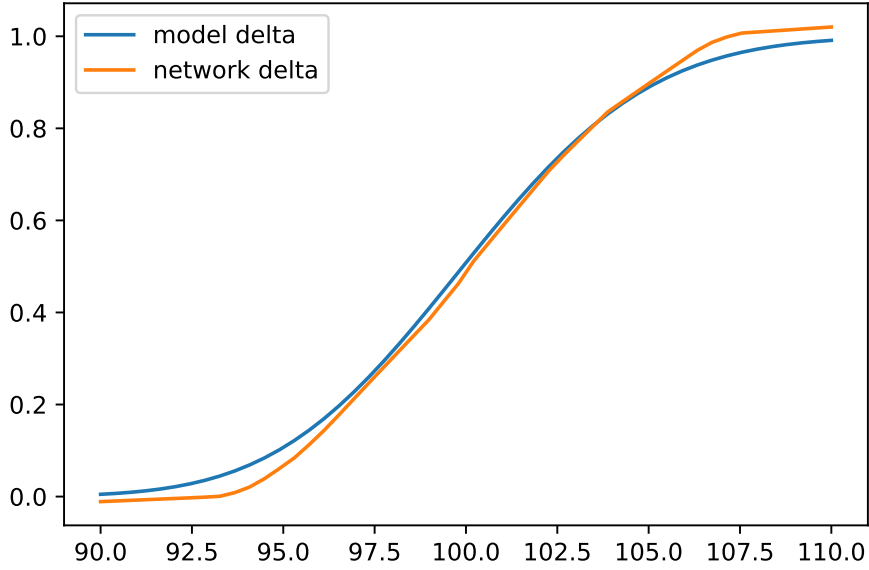


Figure 3: Comparison of δ_k^{BS} and neural network approximation δ_k^θ as a function of S_k for $t_k = 15$ days.

learned trading strategy. If we instead take $\alpha = 0.01$ and train our neural network with $\text{AVaR}_{0.01}$, we obtain a significantly higher risk-adjusted price $p_0^\theta = 3.3505$. Of course this is anticipated, since extreme losses have to be avoided with a much higher priority. And as a result, we need to charge a higher price. This can be seen in Figure 5 which depicts

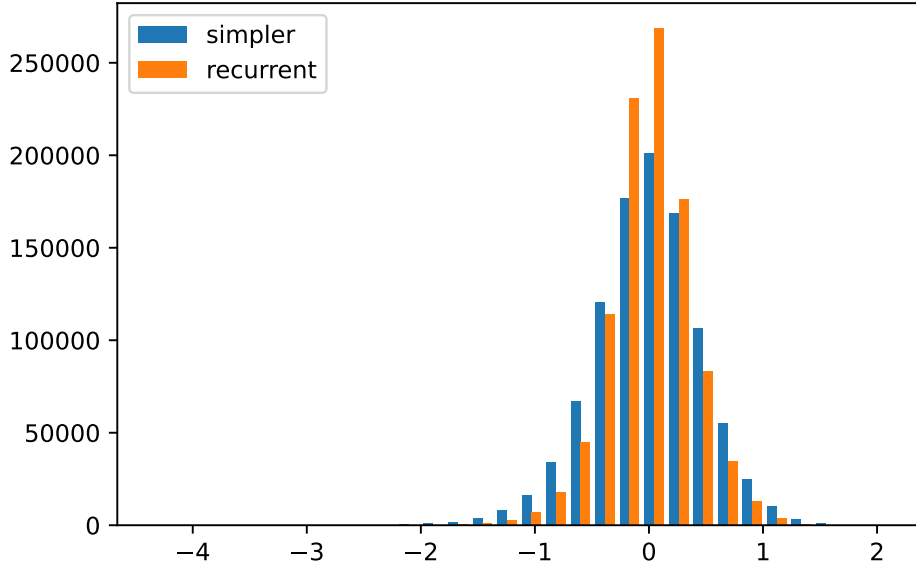


Figure 4: Comparison of semi-recurrent and simpler network structure.

a histogram comparing the hedging error for the neural network hedging strategy trained with $\text{AVaR}_{0.5}$ and $\text{AVaR}_{0.01}$, respectively, while each charging the corresponding price p_0^θ . To yield a better comparison, we can again take the risk-neutral price q for both strategies

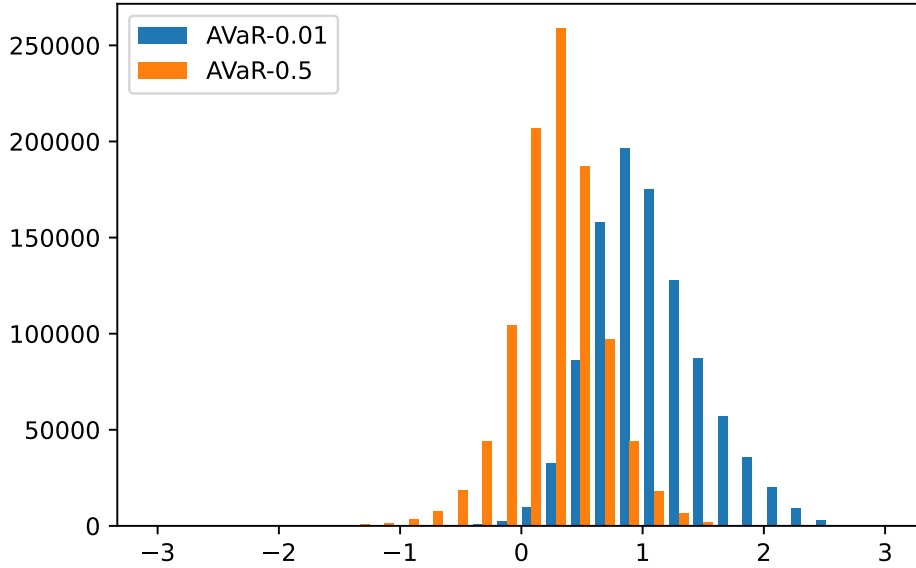


Figure 5: Comparison of deep hedge trained with $\text{AVaR}_{0.01}$ and $\text{AVaR}_{0.5}$.

instead. This is illustrated in Figure 6. We can clearly observe that the strategy trained with $\alpha = 0.5$ is more centred at 0 and has a smaller mean hedging error. Meanwhile, the strategy trained with $\alpha = 0.01$ yields smaller extreme losses which can be seen in the table below Figure 6. The above example demonstrates that by the choice of our risk measure we are able to calculate optimal hedging strategies which are adjusted according

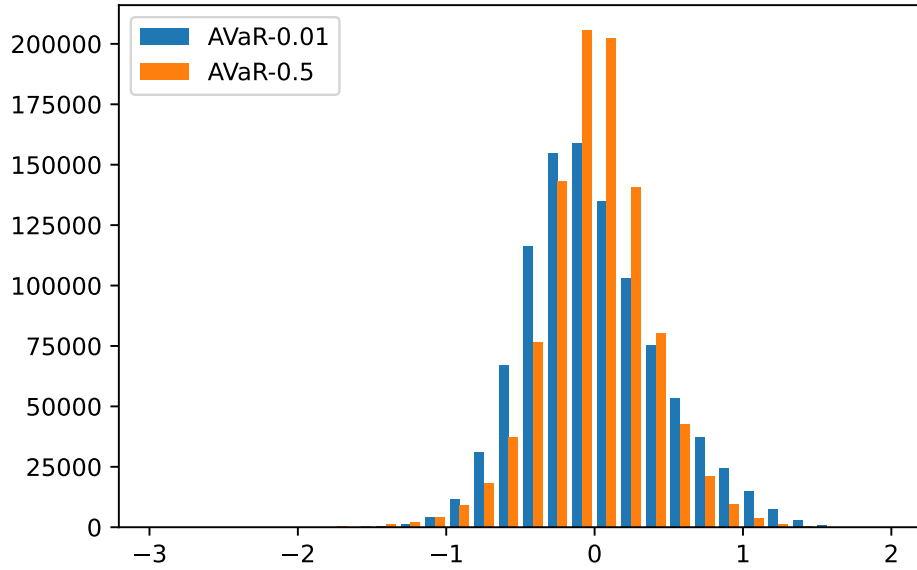


Figure 6: Comparison of deep hedge trained with $\text{AVaR}_{0.01}$ and $\text{AVaR}_{0.5}$, normalized to risk-neutral price.

to our risk preference.

It is worth pointing out again that the deep hedging approach is model independent. This means that no information about the Black-Scholes model is used in the algorithm. We merely need a model to generate sample paths of the price process to use for the training process. As a matter of fact, we do not even require a model: It would also be possible to take (historical) sample paths of a real financial market. Nonetheless, we have decided to assume that the underlying dynamics follow a Black-Scholes model. This is more convenient and most importantly it provides a benchmark such that we can compare the performance of our approach. Having said that, we could have also used any other (more realistic) model, e.g. a stochastic volatility model, to simulate sample paths. In fact, the feasibility of this approach was first illustrated in [Büh+19, Section 5] where the market process S followed (a discretization of) a Heston model. In particular, the numerical experiments given there showed that the neural network hedging strategy was also able to recover the corresponding model hedge for the Heston model well.

5 Markets with Permanent Price Impact

In this section, we further illustrate the usefulness of the deep learning approach by numerically calculating the indifference price and the optimal hedging strategy in a market with frictions. To be more precise, we consider a financial model with permanent price impact.

5.1 Impact Rule

For simplicity, we will assume that our financial market consists of a single stock, i.e. $d = 1$. The following presentation is based on [BLZ16].

The impact of a trading strategy on the price process is modelled through an impact function $f \in C_b^2(\mathbb{R})$, i.e. $f: \mathbb{R} \rightarrow \mathbb{R}$ is twice continuously differentiable and bounded. We further assume f to be (strictly) positive. Using this impact function, the market impact of our trading is modelled as follows: If we want to buy $y \in \mathbb{R}$ number of shares of the asset and the asset has a price of x before the trade, then the price after buying y shares will become $x + f(x)y$. In that sense, the impact of our trading on the price is linear. Note that the price of the asset will increase if we buy shares, and the price will decrease if we sell shares. Since the price impact is applied continuously, the cost of buying y shares is not simply xy but is given by

$$\int_0^y (x + f(x)\iota) d\iota = xy + \frac{1}{2}f(x)y^2.$$

This means that due to price impact the price of each additional share we want to buy becomes more expensive. In particular, the average cost of each share when buying y shares is simply

$$\frac{1}{y} \int_0^y (x + f(x)\iota) d\iota = x + \frac{1}{2}f(x)y.$$

Let us now take a closer look at how the dynamics of the stock price S is influenced by our trading in the market. For this purpose, let $\delta = (\delta_k)_{k=0,\dots,n-1} \in \mathcal{H}$ be a trading strategy and let $s_0 > 0$ be the initial value of the stock S . We write $S_{0-} := s_0$ for the value of S at time 0 *before trading*. At time 0 we decide to buy $\delta_0 \in \mathbb{R}$ shares of S which has an impact on the price according to the above impact rule, and we write

$$S_0 := S_{0-} + f(S_{0-})\delta_0 = s_0 + f(s_0)\delta_0$$

which is the price of S at time 0 *after trading*. More generally, for any $k = 0, \dots, n-1$,

given the price S_{k-} at time t_k *before trading*, we have

$$S_k := S_{k-} + f(S_{k-})(\delta_k - \delta_{k-1}) \quad (5.1)$$

as the price at time t_k *after trading*. The reason we trade $\delta_k - \delta_{k-1}$ shares is because this is exactly the amount of shares we need to buy to obtain the position δ_k given our previous position δ_{k-1} . Note that the cost for this trade is

$$S_{k-}(\delta_k - \delta_{k-1}) + \frac{1}{2}f(S_{k-})(\delta_k - \delta_{k-1})^2.$$

On the other hand we obtain $\delta_k - \delta_{k-1}$ more stock. In addition, the value of the stock changed through our trade from S_{k-} to S_k , which also changes the value of our previously held stock δ_{k-1} . In total, the total effect of this trade on our wealth is given by

$$\begin{aligned} & - \left(S_{k-}(\delta_k - \delta_{k-1}) + \frac{1}{2}f(S_{k-})(\delta_k - \delta_{k-1})^2 \right) + S_k(\delta_k - \delta_{k-1}) + (S_k - S_{k-})\delta_{k-1} \\ &= (S_k - S_{k-})(\delta_k - \delta_{k-1}) - \frac{1}{2}f(S_{k-})(\delta_k - \delta_{k-1})^2 + (S_k - S_{k-})\delta_{k-1} \\ &= f(S_{k-})(\delta_k - \delta_{k-1})(\delta_k - \delta_{k-1}) - \frac{1}{2}f(S_{k-})(\delta_k - \delta_{k-1})^2 + (S_k - S_{k-})\delta_{k-1} \\ &= \frac{1}{2}f(S_{k-})(\delta_k - \delta_{k-1})^2 + (S_k - S_{k-})\delta_{k-1} \end{aligned}$$

where we used the relation $S_k - S_{k-} = f(S_{k-})(\delta_k - \delta_{k-1})$ in the second equation, see (5.1). All in all, the wealth process associated with a trading strategy $\delta \in \mathcal{H}$ and initial wealth $p_0 \in \mathbb{R}$ in this setting with market impact takes the following form

$$\begin{aligned} X_m^\delta &= p_0 + \sum_{k=1}^m (S_{k-} - S_{k-1})\delta_{k-1} + \sum_{k=0}^m \left(\frac{1}{2}f(S_{k-})(\delta_k - \delta_{k-1})^2 + (S_k - S_{k-})\delta_{k-1} \right) \\ &= p_0 + \sum_{k=1}^m (S_k - S_{k-1})\delta_{k-1} + \frac{1}{2} \sum_{k=0}^m f(S_{k-})(\delta_k - \delta_{k-1})^2 \\ &= p_0 + (\delta \bullet S)_m + \frac{1}{2} \sum_{k=0}^m f(S_{k-})(\delta_k - \delta_{k-1})^2 \end{aligned}$$

for $m = 0, \dots, n$. Note however that X_m^δ is not the actual *liquidation value* at time t_m unless $\delta_m = 0$. This is because the liquidation of a position δ_m will have an impact on the market and does not generate cash equal to $S_m\delta_m$. Nonetheless, the value $X_T^\delta := X_n^\delta$ at terminal time T will actually represent our actual wealth (in cash) since we assumed $\delta_n = 0$, i.e. full liquidation.

5.2 Theoretical Benchmark

As noted in Section 4.3, the availability of a benchmark is very useful because it allows us to compare the performance of our trained neural network strategy. Hence, the goal of this section is to derive a benchmark in a market with permanent price impact as described in the previous section.

For this purpose, we assume that between any two trading times t_{k-1} and t_k the price of the stock develops according to the dynamics of a strong solution of the stochastic differential equation

$$dS_t = \mu(S_t) dt + \sigma(S_t) dW_t \quad \text{and} \quad S_0 = s_0$$

where W is a Brownian motion, $s_0 > 0$ and $\mu, \sigma, \sigma^{-1}: \mathbb{R} \rightarrow \mathbb{R}$ are Lipschitz and bounded. To be more precise, for any $k = 1, \dots, n$, we have

$$S_{k-} = S_{k-1} + \int_{t_{k-1}}^{t_k} \mu(S_s) ds + \int_{t_{k-1}}^{t_k} \sigma(S_s) dW_s. \quad (5.2)$$

The above equation (5.2) together with (5.1) thus determine the dynamics of stock price process S . In this setting, one can derive a quasi-linear pricing equation which holds in the sense of viscosity solutions. And when it admits a *smooth solution*, it also provides a perfect hedging strategy. In fact, the solution holds in continuous time and is derived by considering the corresponding discrete time trading dynamics and choosing a finer and finer time grid, i.e. passing to the limit $n \rightarrow \infty$. This was proven in [BLZ16].

However, the treatment of this topic would be out of scope for this thesis, which is why we will skip the details and only consider a simple case. Namely, we consider the case where the impact function f is constant, i.e. $f(x) = \lambda > 0$ for all $x \in \mathbb{R}$. This is a very simple market impact model, but it allows us to obtain a tractable solution as follows. Consider the partial differential equation

$$\begin{aligned} -\partial_t u(t, x) - \frac{1}{2} \sigma(x + \partial_x u(t, x) \lambda)^2 \partial_{xx}^2 u(t, x) &= 0, & (t, x) \in [0, T] \times (0, \infty) \\ u(T, x) &= g(x), & x \in (0, \infty). \end{aligned} \quad (5.3)$$

where $g: (0, \infty) \rightarrow \mathbb{R}$ is a measurable function. Suppose that $u: [0, T] \times (0, \infty) \rightarrow \mathbb{R}$ is a smooth solution of (5.3), i.e. u solves (5.3) such that u is continuous on $[0, T] \times (0, \infty)$ and the restriction of u to $[0, T] \times (0, \infty)$ is once continuously differentiable in t and twice continuously differentiable in x . Then, a perfect hedging strategy for $Z := g(S_T) \in \mathcal{X}$ is given by

$$\Delta_t := \partial_x u(t, S_t - \lambda \Delta_t), \quad t \in [0, T]. \quad (5.4)$$

This is a special case of a much more general result which was proven in [BLZ16]. We refer to [BLZ16, Section 3.4] in particular. Note that this result holds in continuous time only, and not under discrete time dynamics as described in the previous section.

Observe that (5.3) is considerably simplified if we assume that σ is constant. In fact, we obtain the usual heat equation. Notice that if we further assume that μ is constant, the underlying dynamics of the market (5.2) are described by the so-called *Bachelier model*. Hence, let us first take a closer look at the Bachelier model, but *without market impact*.

Remark 5.1. Recall that the dynamics of the Bachelier model are given by

$$S_t = s_0 + \mu t + \sigma W_t, \quad t \in [0, T].$$

Set $g(x) = (x - K)^+$, i.e. our objective is hedging a call option again. Then, one can show that the risk-neutral price of a call option on S with strike $K > 0$ and maturity $T > 0$ is given by

$$u(t, S_t) := (S_t - K) \Phi \left(\frac{S_t - K}{s_0 \sigma \sqrt{T - t}} \right) + s_0 \sigma \sqrt{T - t} \varphi \left(\frac{S_t - K}{s_0 \sigma \sqrt{T - t}} \right), \quad (5.5)$$

where $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ and $\Phi: \mathbb{R} \rightarrow [0, 1]$ denote the probability density function and cumulative distribution function of a standard normal random variable, respectively. In particular, one can verify that u defined by (5.5) solves (5.3). Similarly to the Black-Scholes model, it is also possible to perfectly hedge a call option in the Bachelier model using the *delta hedging strategy* which is given by

$$\partial_x u(t, S_t) = \Phi \left(\frac{S_t - K}{s_0 \sigma \sqrt{T - t}} \right). \quad (5.6)$$

For a quick review of the Bachelier model and in particular the results stated in this remark, we refer to [DS06, Chapter 4.3].

We can now turn to the case *with market impact*. In this situation, we obtain that the perfect hedging strategy Δ is given by

$$\Delta_t = \partial_x u(t, S_t - \Delta_t \lambda) = \Phi \left(\frac{S_t - \Delta_t \lambda - K}{s_0 \sigma \sqrt{T - t}} \right). \quad (5.7)$$

This is an immediate consequence of (5.4) and (5.6). Notice that the perfect hedging strategy (5.7) consists in following the usual delta hedging strategy (5.6) but for $\partial_x u$ computed at the value of the stock which would be obtained if the position in stocks was liquidated, i.e. $S_t - \Delta_t \lambda$. One may also observe that in (5.7) the hedging strategy Δ_t

depends on itself and there is no obvious way to compute Δ_t directly. However, (5.7) can be solved numerically, e.g. using a fixed-point iteration. Note that an easy calculation shows that (5.7) is indeed a contraction for $t \in [0, T)$ if

$$\frac{\lambda}{s_0 \sigma \sqrt{2\pi(T-t)}} < 1 \quad (5.8)$$

is satisfied.

5.3 Numerical Results

Finally, we test our deep learning approach to a market situation with permanent price impact, as described in Section 5.1. For this purpose we consider a discretized one-dimensional Bachelier model with model parameters $\mu = 0$, $\sigma = 0.2$ and $s_0 = 100$. In addition, we assume the presence of permanent price impact as described in Section 5.1 with constant impact function $f \equiv \lambda$ and we choose $\lambda = 1$. This can be interpreted as follows: if we buy one share of the asset, the price of the asset increases by one. In particular, for this choice of parameters, (5.8) is satisfied for all t_k , $k = 0, \dots, n-1$. As in the Black-Scholes case, our objective will be to hedge a call option with maturity $T = 30/365$ and strike price $K = s_0$. Note again that (5.7) is only a perfect hedging strategy defined in continuous time and thus requires the ability of continuous trading. Since we only allow daily rebalancing, there will inevitably be a (small) hedging error again. Nevertheless, we use a discretized version $\delta_k^B := \Delta_{t_k}$, $k = 0, \dots, n-1$, where Δ is defined by (5.7), as our model hedge which serves as a benchmark. Moreover, we proceed precisely as in the Black-Scholes case regarding the modelling choices for the deep learning algorithm, see Section 4.2. Note that we have also chosen $\rho = \text{AVaR}_\alpha$ with $\alpha = 0.5$ as our risk preference.

We can now compare the performance of the trained neural network strategy δ^θ with our model hedge δ_k^B . The hedging error is plotted as a histogram in Figure 7 below. We observe that the hedging error for δ^θ and δ_k^B is very similar. Moreover, the neural network computed the price $p_0^\theta = 2.5753$ as an approximation of the indifference price. Note that the risk neutral price can be easily computed using (5.5) which yields $q \approx 2.2875$. Thus, we conclude that the neural network strategy δ^θ is able to approximate well the optimal strategy in (2.6), even in the presence of permanent market impact.

In Figure 4 we showed that the hedging performance for a neural network with a simple structure, i.e. $\delta_k^\theta = F^{\theta_k}(I_k)$, is similar compared to the semi-recurrent structure $\delta_k^\theta = F^{\theta_k}(I_k, \delta_{k-1}^\theta)$. This, however, is not true any more in our setting, where we assume the presence of market frictions. Indeed, this was illustrated in [Büh+19, Figure 5].

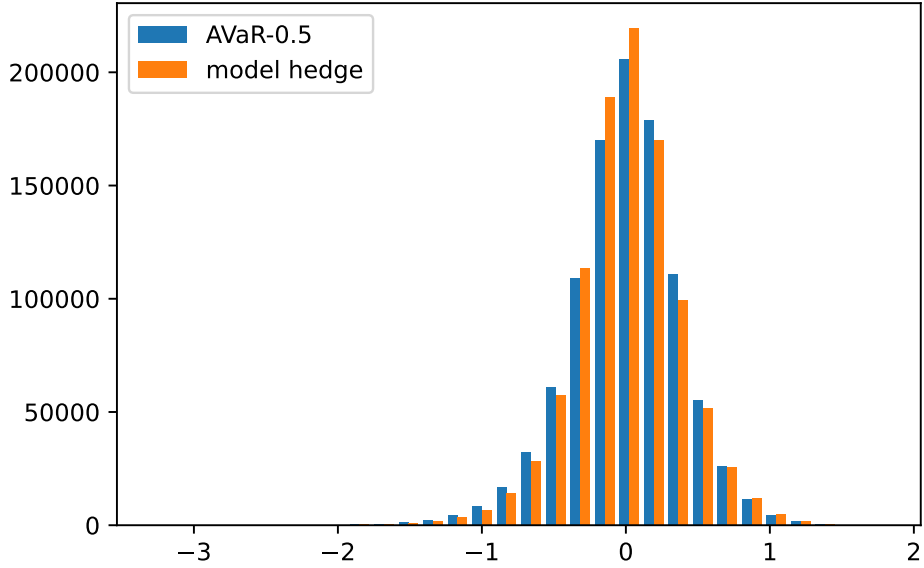
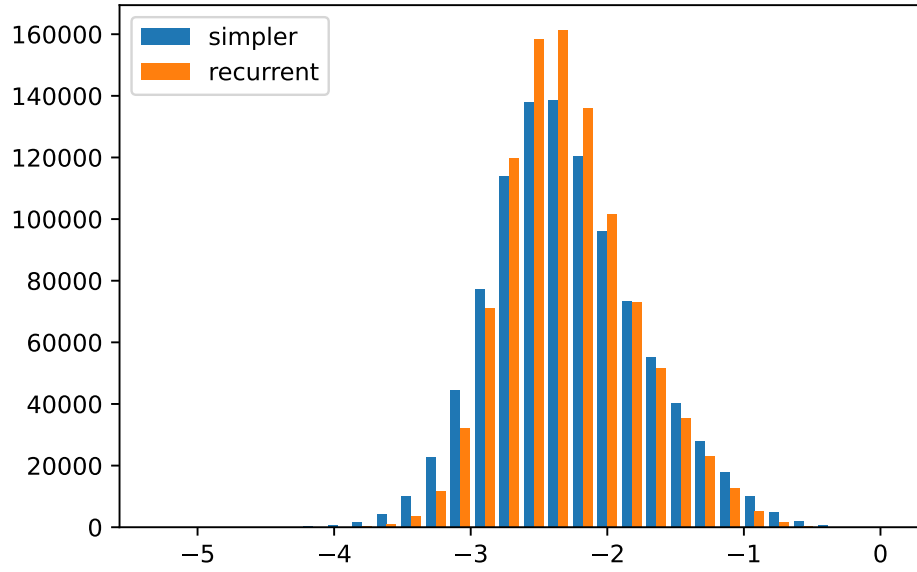


Figure 7: Comparison of model hedge and deep hedge associated to $\text{AVaR}_{0.5}$ in the presence of permanent price impact.

There, the authors considered a setting with proportional transaction costs, see Example 2.10. Here, we want to do a similar experiment but assuming different market frictions, namely the presence of permanent price impact instead of transactions costs. For this purpose, we consider the same setting as above, except that we choose the risk preference $\alpha = 0.01$. Figure 8 compares the hedging performance for both neural network structures. We observe that the hedging performance of the neural network with the simple structure is only slightly worse than the neural network with the semi-recurrent structure, despite the presence of market impact. This can be explained as follows: At time t_k we trade $\delta_k - \delta_{k-1}$ shares of the asset. Hence, to make a trading decision, δ_{k-1} is relevant to measure the market impact of our trading. However, the optimal hedging strategy which is given by (5.7) does not depend on the previous position but only the current price. With this in mind, it makes sense that the hedging performance of both neural network structures should be similar. The next remark further highlights the limitations of the impact model we considered for this experiment, that is, permanent price impact as described in Section 5.1 with constant impact function $f \equiv \lambda$.

Remark 5.2 ([BLZ16, Section 3.4]). Recall that each time we trade, the price is shifted by $\lambda(\delta_k - \delta_{k-1})$. Since we required $\delta_{-1} = \delta_n = 0$, we have that the total impact of our trading on the price is null: $\sum_{k=0}^n \lambda(\delta_k - \delta_{k-1}) = \lambda(\delta_n - \delta_0) = 0$. Moreover, we assumed that σ is constant and in particular, the underlying dynamics of the price process is not changed by the impact of our trading, i.e. $S_T = s_0 + \sigma W_T$. Note that price impact can be interpreted as liquidation costs: when buying, we pay an additional cost, but it moves the price up, when selling back, we pay a cost again. However, because we pushed up



	Mean Loss	Price
simpler	-2.288	3.6403
recurrent	-2.288	3.3916

Figure 8: Comparison of semi-recurrent and simpler network structure (with permanent price impact and $\text{AVaR}_{0.01}$).

the price, we are able to sell at a higher price. In fact, one can show that this perfectly cancels in our setting where our trading has no effect on the underlying dynamics of S and the impact function f is constant, see [BLZ16, Section 3.4].

In light of this, the permanent price impact model we considered for our experiments is a rather simple one. Nonetheless, it allowed us to obtain a useful benchmark, see (5.7).

6 Conclusion

In this thesis, we have presented the deep hedging framework as proposed in [Büh+19]. First, we provided a brief exposition on discrete time markets with frictions. In particular, we dealt with pricing and hedging in such markets using convex risk measures. Moreover, we observed that there is a strong theoretical justification to use neural networks to approximate optimal hedging strategies. Our experiments demonstrated a convincing hedging performance for this approach in a Black-Scholes market. In addition, the deep hedging algorithm is capable of learning risk-adjusted hedging strategies by choosing the risk measure appropriately. We have also investigated the performance of the deep hedging approach in markets with frictions, namely, in the presence of permanent price impact. Our experiments showed that the neural network hedging strategy was able to approximate the optimal hedging strategy well for the impact structure we considered.

Note however that this impact structure is very simple, as seen in Remark 5.2. A more realistic model for market impact is given by the square-root formula, i.e. the price change is proportional to \sqrt{y} where y is the number of shares traded (instead of linear in y). The square-root formula is widely used in practice to provide an estimate of market impact before a trade. In fact, the square-root formula has been empirically verified, see for instance [Tót+11]. Thus, our thesis leaves room for further research on the performance of the deep hedging algorithm with more realistic market impact rules. In particular, another interesting challenge is to model the impact of trading in one asset on another asset, for instance when trading options (as hedging instruments), as noted in [Büh+19, Remark 2.2].

Let us make a few comments on further research. First, it should be noted that the objective (3.1) depends on the derivative Z we want to hedge. This means that every time we want to sell a different derivative, we need to completely retrain the model. This is of course a severe drawback. Hence, it would be very useful to come up with a different problem formulation that would include the derivative itself as a state variable of the model. This model might take a much longer time to train, but the advantage is that we only need to train the model once and can then use it flexibly for any derivative which we want to sell.

Moreover, we emphasize that although we have used a Black-Scholes model and a Bachelier model to generate market scenarios for our experiments, the deep hedging approach is in fact inherently model agnostic. The more similar our training data is to real market data, the better is the hedging performance of the neural network strategy in the real market. In particular, this implies that we should rather use real market data for our training samples. However, there is often not sufficient data available, for instance when the focus is on daily return data. This problem is amplified by the in-

herent non-stationarity of market data, in the sense that data from even ten years back exhibit very different statistical characteristics than the data today. This is the reason we chose to use synthetic data generated from a classic stochastic model, as this is much more convenient. Having said that, there is an increasing interest towards *market scenario generators*. A statistically driven market simulator based on generative modelling is presented in [Büh+20].

It is also worth pointing out that we have chosen exactly the same neural network structure as proposed in [Büh+19], with the same amount of layers and nodes. We find that increasing the model complexity, by adding more nodes per layer, did not seem to result in an improved hedging performance. However, changing the network structure might lead to faster training times and better hedging performance. This is indicated by [Ima+21] which suggests using a so-called *no-transaction band network*. Indeed, their experiments demonstrate that for European and lookback options, this architecture quickly attains a better hedging strategy in comparison to a standard feed forward network. Further, [HTŽ21] propose to use a slightly changed “fully recurrent” architecture, which performs significantly better for the rough Bergomi model. Rough volatility models are known to reflect the reality of financial markets better than classical Markovian models do. Hence, their findings also give an indication how a naive application of model architectures to real data could lead to substantial errors.

In conclusion, the deep hedging approach presented in this thesis is an exciting recent development in the application of deep learning in finance and has opened a lot of directions for research both for academics and for practitioners.

References

- [Ber+21] Julius Berner et al. *The Modern Mathematics of Deep Learning*. 2021. arXiv: 2105.04026 [cs.LG].
- [BLZ16] Bruno Bouchard, Grégoire Loeper, and Yiyi Zou. “Almost-sure hedging with permanent price impact”. In: *Finance and Stochastics* 20.3 (2016), pp. 741–771.
- [BT07] Aharon Ben-Tal and Marc Teboulle. “An Old-New Concept of Convex Risk Measures: The Optimized Certainty Equivalent”. In: *Mathematical Finance* 17.3 (2007), pp. 449–476.
- [Büh+19] Hans Bühler et al. “Deep Hedging”. In: *Quantitative Finance* 19.8 (2019), pp. 1271–1291.
- [Büh+20] Hans Bühler et al. *A Data-driven Market Simulator for Small Data Environments*. 2020. arXiv: 2006.14498 [q-fin.ST].
- [Cho+15] Anna Choromanska et al. “The Loss Surfaces of Multilayer Networks”. In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. 2015, pp. 192–204.
- [DS06] Freddy Delbaen and Walter Schachermayer. *The Mathematics of Arbitrage*. Springer Finance. Springer Berlin Heidelberg, 2006.
- [FS16] Hans Föllmer and Alexander Schied. *Stochastic Finance: An Introduction in Discrete Time*. De Gruyter, 2016.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [Gla03] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Stochastic Modelling and Applied Probability. Springer New York, 2003.
- [GRK20] Ingo Gühring, Mones Raslan, and Gitta Kutyniok. *Expressivity of Deep Neural Networks*. 2020. arXiv: 2007.04759 [cs.LG].
- [Hin70] Karl Hinderer. *Foundations of Non-stationary Dynamic Programming with Discrete Time Parameter*. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, 1970.
- [HTŽ21] Blanka Horvath, Josef Teichmann, and Žan Žurič. “Deep Hedging under Rough Volatility”. In: *Risks* 9.7 (2021).
- [Ima+21] Shota Imaki et al. *No-Transaction Band Network: A Neural Network Architecture for Efficient Deep Hedging*. 2021. arXiv: 2103.01775 [q-fin.CP].

- [IS15] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 448–456.
- [KB14] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (2014).
- [Les+93] Moshe Leshno et al. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. In: *Neural Networks* 6.6 (1993), pp. 861–867.
- [Pas+19] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* 32. 2019, pp. 8024–8035.
- [Pin99] Allan Pinkus. “Approximation theory of the MLP model in neural networks”. In: *Acta Numerica* 8 (1999), pp. 143–195.
- [Roc70] Ralph Tyrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
- [SSC95] H. Mete Soner, Steven E. Shreve, and Jaksa Cvitanic. “There is no Nontrivial Hedging Portfolio for Option Pricing with Transaction Costs”. In: *The Annals of Applied Probability* 5.2 (1995), pp. 327–355.
- [Tót+11] Bence Tóth et al. “Anomalous Price Impact and the Critical Nature of Liquidity in Financial Markets”. In: *Physical Review X* (2011).