A Deep Learning Approach to Probability of Shortfall in Defined Contribution Plan Optimization

by

Zeyu Zhang

A research paper
presented to the University of Waterloo
in fulfillment of the
research paper requirement for the degree of
Master of Mathematics
in
Data Science

Waterloo, Ontario, Canada, 2025

Author's Declaration

I hereby declare that I am the sole author of this research paper. This is a true copy of the research paper, including any required final revisions, as accepted by my examiners.

I understand that my research paper may be made electronically available to the public.

Abstract

This study presents a neural network (NN)-based approach to compute the optimal decumulation strategy for retirees with defined contribution (DC) pension plans. Unlike traditional approaches to expected shortfall problems, we introduce a probability of shortfall (PS) term to quantify the risk of fund depletion, balancing withdrawal amounts with the likelihood that terminal wealth falls below a threshold. We demonstrate that one of the main challenges in solving this problem is the non-differentiability of the indicator function, which introduces difficulties in calculating the derivative using Monte Carlo simulation, limiting the effectiveness of gradient-based optimization. To overcome this, we propose the sigmoid function as a differentiable approximation to the indicator function. Through extensive numerical experiments, we identify good choices of NN hyper-parameters and validate the approach against partial differential equation (PDE) results as a ground truth. Our findings show that the NN solution consistently produces results closely matching those of the PDE method. Furthermore, robustness testing on synthetic and historical datasets highlights the NN's reliability and practical applicability. Our results highlight the NN's ability to offer computationally efficient and highly accurate solutions for optimal decumulation strategies. While the probability of shortfall is more interpretable than the expected shortfall, as it quantifies the probability of shortfall occurrence, it does not capture the magnitude of shortfalls. This trade-off underscores the importance of selecting an appropriate risk measure based on investor preferences.

Acknowledgements

I would like to thank my mother for everything.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traditional defined benefit (DB) pension plans have been gradually disappearing from the employment landscape, largely due to their prohibitive cost and the longevity risk they pose to employers. By 2024, only 15% of private-sector workers in the United States retained access to a DB plan, in contrast to 70% who were enrolled in defined contribution (DC) plans (U.S. Bureau of Labor Statistics, 2024). Unlike DB plans, which guarantee retirees fixed monthly payments based on factors like tenure and salary, DC plans shift the burden of managing retirement savings onto individual investors.

This shift introduces a formidable challenge: crafting optimal withdrawal and allocation strategies tailored to retirees' diverse needs and risk preferences. Nobel Laureate William Sharpe aptly described this problem as "the nastiest, hardest problem in finance" (Ritholz, 2017). Despite well-established rules of thumb, such as the Bengen Rule (commonly known as the 4% Rule) (Bengen, 1994), existing strategies often fall short of ensuring optimal outcomes. Indeed, reviews of decumulation strategies (Bernhardt and Donnelly, 2018; MacDonald et al., 2013) highlight the inherent difficulty of addressing retirees' conflicting objectives—such as balancing current withdrawals with long-term fund sustainability—with a one-size-fits-all solution.

To better manage the trade-offs between withdrawals and the risk of depleting retirement savings, Forsyth (2022) introduced a dynamic decumulation strategy framed as a constrained optimal stochastic control problem. This strategy spans a standard 30-year investment horizon and employs dynamic programming techniques to numerically solve the associated Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE). The resulting framework identifies Pareto-optimal solutions by scalarizing conflicting objectives—maximizing expected withdrawals while minimizing the risk of shortfall, measured

by the lower tail of the terminal wealth distribution. Constraints, including a minimum withdrawal and investment limits to prevent shorting or leveraging, further refine this strategy.

While PDE-based methods provide mathematically rigorous solutions, they can be computationally intensive and challenging to scale. There are several neural network and deep learning methods for optimal stochastic control problems which have been proposed before, taking a stacked neural network approach as in Buehler et al. (2019); Han and E (2016); Tsang and Wong (2020) or a hybrid dynamic programming and reinforcement learning approach (Huré et al., 2021), as well as using time as an input to the NN was also suggested in Laurière et al. (2021). We especially want to mention a neural network-based framework proposed by Chen et al. (2023) bypasses dynamic programming to approximate optimal solutions efficiently. The results obtained from the proposed NN framework closely align with those derived from PDE methods. The NN framework has the following advantages:

- The framework is entirely data-driven and does not rely on a parametric model, making it highly adaptable in selecting training data and reducing susceptibility to model misspecification.
- The control is learned directly, leveraging control's low dimensionality. This approach eliminates the need for dynamic programming and avoids error propagation. Additionally, the NN method extends naturally to high-dimensional problems, such as those involving multiple assets.
- When the control varies continuously over time, the NN framework preserves this continuity. If the control is discontinuous, the NN provides a smooth yet highly accurate approximation.

In this paper, we apply the above NN framework to measure risk using the probability of shortfall. In contrast to the expected shortfall problem, there is little previous work about probability of shortfall as a risk measure. Gridin and Golubin (2023) demonstrated the generalized convexity of shortfall probability under certain normality assumptions, contrasting mean-shortfall-probability with mean-value-at-risk as objective functions. Calafiore (2013) presented a novel data-driven approach for portfolio optimization that relies on historical data rather than parametric distributional assumptions.

Building on the neural network (NN) framework established by Chen et al. (2023), our work explores a new dimension: measuring the probability of shortfall—the likelihood that terminal wealth falls below a specified target—rather than expected shortfall. This shift in focus moves from quantifying the shortfall's magnitude to assessing its probability. We

benchmark NN results against PDE-based solutions, which serve as the ground truth. In addition, we experiment with hyperparameter tuning and other refinements to enhance NN performance. Finally, we examine whether the probability of shortfall is an effective risk measure.

Compared to earlier studies, our approach expresses the probability of shortfall differently, introduces alternative assumptions for stock and bond dynamics, and employs neural networks to solve the associated optimization problem, a departure from prior reliance on traditional numerical methods.

Here we also want to emphasize some key differences between the expected shortfall problem (referred to as the ES problem for the remainder of this paper) which has been solved by the same NN framework by Chen et al. (2023) and our new probability of shortfall problem (referred to as the PS problem for the remainder of this paper):

1. Expected Shortfall and Probability of Shortfall: The expected shortfall measures the expected magnitude of wealth falling short of the target, while the probability of shortfall quantifies the likelihood of such an event. Mathematically, probability of shortfall is expressed as the expected value of an indicator function that flags whether terminal wealth is below the target.

2. Optimal Control and Targeted Wealth: For the ES problem, the NN identifies both the optimal control strategy and the optimal value at risk (VAR). In contrast, the PS problem assumes a fixed target wealth, requiring the NN to focus solely on identifying optimal controls.

Our findings contribute to the growing body of research on dynamic retirement strategies, providing a fresh perspective on the probability of shortfall, demonstrating how neural networks can be effectively leveraged in portfolio optimization, and discussing whether the probability of shortfall is a good risk measure.

Chapter 2

Problem Formulation

2.1 Overview

The investment scenario described in Forsyth (2022) considers an investor who, upon retirement, begins with a portfolio of specified wealth. The investment horizon is divided into a fixed number of equally spaced re-balancing intervals, typically annually. At each re-balancing time, the investor decides first how much to withdraw within a prescribed range and then allocates the remaining wealth (after withdrawal) across available assets. The allocation is constrained to avoid shorting or leverage and can include a broad stock index fund and a constant maturity bond index fund.

Between re-balancing periods, the portfolio's wealth evolves based on the dynamics of the underlying assets. If the portfolio's wealth falls below zero due to withdrawals, it is liquidated, trading ceases, debt accrues at the borrowing rate, and withdrawals are limited to the minimum allowable amount. Importantly, in this framework, the final withdrawal and allocation occur in the 29th year of the investment horizon. The portfolio is then liquidated to determine the terminal wealth at the end of the 30th year. This structure assumes that the investor holds other assets, such as real estate, which are not fungible with the investment portfolio. This real estate can be regarded as a hedge of last resort, to fund shortfall.

The investor's objective is twofold: to maximize a weighted sum of total withdrawals and minimize the probability that the terminal wealth falls below a target value. Mathematically, this probability is framed as the expected value of an indicator function representing whether the terminal wealth is less than the target. The following sections detail

the mathematical formulation of this optimization problem, applicable to both PDE and NN methods.

2.2 Stochastic Process Model

Let S_t and B_t represent the real (i.e. inflation-adjusted) amounts invested in the stock index and a constant maturity bond index, respectively. These assets are modeled with correlated jump diffusion models. These parametric stochastic differential equations (SDEs) allow us to model non-normal asset returns. The SDEs are used in solving the PDE, and generating training data with Monte Carlo simulations in the proposed NN framework.

If a jump is triggered, $S_t = \xi^s S_{t^-}$ where ξ^s is a jump multiplier and $S_{t^-} = S(t-\epsilon)$, $\epsilon \to 0^+$. $\log(\xi^s)$ is assumed to follow a double exponential distribution. The jump is either upward or downward, with probabilities μ^s and $1-\mu^s$ respectively. The density function for $y = \log(\xi^s)$ is

$$f^s(y) = u^s \eta_1^s e^{-\eta_1^s y} \mathbf{1}_{y \geq 0} + (1-u^s)\eta_2^s e^{\eta_2^s y} \mathbf{1}_{y<0} \ , \tag{2.2.1}$$

where η_1^s and η_2^s are rate parameters for the exponential distributions that describe the magnitude of the upward and downward jumps, respectively; $\mathbf{1}_{y\geq 0}$ and $\mathbf{1}_{y<0}$ are indicator functions that restrict the distribution to positive or negative values.

We also define the expected change in $\xi^s - 1$:

$$\gamma_\xi^s \quad = \quad E[\xi^s - 1] = \frac{u^s \eta_1^s}{\eta_1^s - 1} + \frac{(1-u^s)\eta_2^s}{\eta_2^s + 1} - 1 \ . \tag{2.2.2}$$

In the absence of control, S_t evolves according to

$$\frac{dS_t}{S_{t^-}} \quad = \quad \left(\mu^s - \lambda_\xi^s \gamma_\xi^s\right) \, dt + \sigma^s \, dZ^s + d\left(\sum_{i=1}^{\pi_t^s} (\xi_i^s - 1)\right) , \tag{2.2.3}$$

where μ^s is the (uncompensated) drift rate, σ^s is the volatility, dZ^s is the increment of a Wiener process, π_t^s is a Poisson process with positive intensity parameter λ_ξ^s, and ξ_i^s are i.i.d. positive random variables having distribution (2.2.1). Moreover, ξ_i^s, π_t^s, and Z^s are assumed to all be mutually independent.

From the equation above, it is clear that the stock price has a combined distribution. Usually, the stock price follows a continuous path, with small fluctuations due to market

activity, captured by the $\sigma^s\, dZ^s$ term. This part of the model is similar to the classical Geometric Brownian Motion used in the Black-Scholes model, but here it is adjusted by the drift term $\mu^s - \lambda_\xi^s \gamma_\xi^s$ to account for jumps. But occasionally the stock price experiences sudden jumps, which are modeled by the Poisson process. The jumps are not continuous; they occur at random times with a rate λ_ξ^s and their magnitude is determined by the distribution of ξ_i^s.

Similarly, let the amount in the bond index be $B_{t^-} = B(t-\epsilon), \epsilon \to 0^+$. In the absence of control, B_t evolves as

$$\frac{dB_t}{B_{t^-}} = \left(\mu^b - \lambda_\xi^b \gamma_\xi^b + \mu_c^b \mathbf{1}_{\{B_{t^-}<0\}}\right) dt + \sigma^b\, dZ^b + d\left(\sum_{i=1}^{\pi_t^b} (\xi_i^b - 1)\right), \tag{2.2.4}$$

where the terms in Equation (2.2.4) are defined analogously to Equation (2.2.3). In particular, π_t^b is a Poisson process with positive intensity parameter λ_ξ^b, and ξ_i^b has distribution

$$f^b(y = \log \xi^b) = u^b \eta_1^b e^{-\eta_1^b y} \mathbf{1}_{y \geq 0} + (1-u^b)\eta_2^b e^{\eta_2^b y} \mathbf{1}_{y<0}\ , \tag{2.2.5}$$

and $\gamma_\xi^b = E[\xi^b - 1]$. ξ_i^b, π_t^b, and Z^b are assumed to all be mutually independent. The term $\mu_c^b \mathbf{1}_{\{B_{t^-}<0\}}$ in Equation (2.2.4) represents the borrowing spread. The diffusion processes are correlated, i.e. $dZ^s \cdot dZ^b = \rho_{sb}\, dt$. The stock and bond jump processes are assumed to be mutually independent.

We define the investor's total wealth at time t as

$$\text{Total wealth } \equiv W_t = S_t + B_t\ . \tag{2.2.6}$$

Barring insolvency, shorting stock and using leverage (i.e. borrowing) are not permitted, a realistic constraint in the context of DC retirement plans. Furthermore, if wealth ever goes below zero, the portfolio is liquidated, trading ceases, and debt accumulates at the borrowing rate. We emphasize that we are assuming that the retiree has other assets (i.e., residential real estate) that can be used to fund any accumulated debt.

2.3 Notational Conventions

We define the finite set of discrete withdrawal and re-balancing times $\mathcal{T}$:

$$\mathcal{T} = \{t_0 = 0, t_1, t_2, \ldots, t_M = T\}\ , \tag{2.3.1}$$

where $t_0 = 0$ marks the beginning of the investment period, and $t_M = T$ represents the end. The re-balancing times are assumed to be evenly spaced, so $t_i - t_{i-1} = \Delta t = T/M$ is constant for all i. To simplify notation, we occasionally use $S_t \equiv S(t)$, $B_t \equiv B(t)$, and $W_t \equiv W(t)$ to denote the state variables for stock, bond, and total wealth, respectively.

At each re-balancing time $t_i \in \mathcal{T}$, the investor first withdraws an amount q_i of cash from the portfolio and then re-balances the remaining wealth. Taxes are disregarded in this framework, reflecting the tax-advantaged nature of most retirement accounts.

For an arbitrary time-dependent function $f(t)$, the shorthand notation:

$$f(t_i^+) = \lim_{\epsilon \to 0^+} f(t_i + \epsilon) \ , \quad f(t_i^-) = \lim_{\epsilon \to 0^+} f(t_i - \epsilon) \ , \tag{2.3.2}$$

is used to denote values just after and just before t_i, respectively.

The controlled underlying multidimensional process is denoted as:

$$X(t) = (S(t), B(t)) \ , \quad t \in [0, T] \ , \tag{2.3.3}$$

where $x = (s, b)$ represents the realized state of the system.

At re-balancing times $t_i \in \mathcal{T}$, the investor applies the withdrawal control q_i, which is determined by the state of the portfolio just before withdrawal:

$$q_i(\cdot) = q_i(X(t_i^-)) = q(X(t_i^-), t_i) \ . \tag{2.3.4}$$

This control modifies the wealth of the portfolio as follows:

$$W(t_i^+) = W(t_i^-) - q_i \ , \quad t_i \in \mathcal{T} \ , \tag{2.3.5}$$

where $W(t_i^-)$ is the wealth before withdrawal, and $W(t_i^+)$ is the wealth after withdrawal.

Following withdrawal, the investor determines the portfolio allocation control p_i, which depends on the state of the portfolio after withdrawal:

$$p_i(\cdot) = p_i(X(t_i^+)) = p(X(t_i^+), t_i) \ . \tag{2.3.6}$$

The allocation proportion for stocks is then given by:

$$p_i(X(t_i^+)) = p(X(t_i^+), t_i) = \frac{S(t_i^+)}{S(t_i^+) + B(t_i^+)} \ . \tag{2.3.7}$$

To simplify the notation, we note that the withdrawal control $q_i(\cdot)$ depends only on wealth before withdrawal, W_i^-, and the allocation control $p_i(\cdot)$ depends only on wealth after withdrawal, W_i^+.

Instantaneous re-balancing is assumed, ensuring that no changes in asset prices occur in the interval (t_i^-, t_i^+). A control at t_i is therefore represented by the pair:

$$(q_i(\cdot), p_i(\cdot)) \in \mathcal{Z}(W_i^-, W_i^+, t_i) \ , \tag{2.3.8}$$

where $\mathcal{Z}(W_i^-, W_i^+, t_i)$ defines the set of admissible controls at t_i.

Constraints on controls:

- Allocation Constraints: No shorting or leverage is allowed, provided the portfolio remains solvent.
- Withdrawal Bounds: Withdrawals are limited to a specified minimum and maximum.
- Insolvency: If $W_i^+ < 0$ (portfolio wealth becomes negative after withdrawal), the portfolio is liquidated. Trading ceases, debt accumulates at the borrowing rate, and withdrawals are restricted to the minimum allowable amount.

At the end of the investment period, the asset is sold and the terminal wealth is realized. Mathematically, the constraints ensure that the controls remain within admissible bounds while respecting the described withdrawal and allocation rules.

$$\mathcal{Z}_q(W_i^-,t_i) = \begin{cases} [q_{\min}, q_{\max}] \ ; \ t_i \in \mathcal{T} \ ; \ W_i^- > q_{\max} \\ [q_{\min}, W_i^-] \ ; \ t_i \in \mathcal{T} \ ; \ q_{\min} < W_i^- < q_{\max} \\ \{q_{\min}\} \ ; \ t_i \in \mathcal{T} \ ; \ W_i^- < q_{\min} \end{cases} , \tag{2.3.9}$$

$$\mathcal{Z}_p(W_i^+,t_i) = \begin{cases} [0,1] & W_i^+ > 0 \ ; \ t_i \in \mathcal{T} \ ; \ t_i \neq t_M \\ \{0\} & W_i^+ \leq 0 \ ; \ t_i \in \mathcal{T} \ ; \ t_i \neq t_M \\ \{0\} & t_i = T \end{cases} , \tag{2.3.10}$$

$$\mathcal{Z}(W_i^-, W_i^+,t_i) = \mathcal{Z}_q(W_i^-,t_i) \times \mathcal{Z}_p(W_i^+,t_i) \ . \tag{2.3.11}$$

At each re-balancing time, we seek the optimal control for all possible combinations of $(S(t), B(t))$ having the same total wealth. Hence, the controls for both withdrawal and allocation are formally a function of wealth and time before withdrawal (W_i^-,t_i), but for

implementation purposes, it will be helpful to write the allocation as a function of wealth and time after withdrawal (W_i^+,t_i). The admissible control set $\mathcal{A}$ can be written as

$$\mathcal{A} = \left\{ (q_i, p_i)_{0 \le i \le M} : (q_i, p_i) \in \mathcal{Z}(W_i^-, W_i^+, t_i) \right\} . \tag{2.3.12}$$

An admissible control $\mathcal{P} \in \mathcal{A}$, can be written as

$$\mathcal{P} = \{ (q_i(\cdot), p_i(\cdot)) \ : \ i = 0, \ldots, M \} . \tag{2.3.13}$$

2.4 Risk: Probability of Shortfall

Assume that $g(W_T)$ is the probability density of the terminal wealth W_T at $t = T$. Then suppose

$$\int_{-\infty}^{W^*} g(W_T) \, dW_T = \text{Prob}(W_T < W^*) , \tag{2.4.1}$$

where W^* is a defined threshold wealth value. We then define the probability of shortfall (PS) as the probability of terminal wealth is in the worst section, which is $W_T < W^*$. Mathematically,

$$\text{PS} = \text{Prob}(W_T < W^*) = E[\mathbf{1}_{W_T < W^*}] , \tag{2.4.2}$$

where $\mathbf{1}_{W_T < W^*}$ is an indicator function:

$$\mathbf{1}_{W_T < W^*} = \begin{cases} 1 & \text{if } W_T < W^*, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4.3}$$

Under a control $\mathcal{P}$ and initial state X_0, PS can be written as:

$$\text{PS}(X_0^-, t_0^-) \ = \ E_{\mathcal{P}}^{X_0^-, t_0^-}[\mathbf{1}_{W_T < W^*}] . \tag{2.4.4}$$

In our problem, we want to minimize PS which is equivalent to maximizing -PS:

$$\max_{\mathcal{P} \in \mathcal{A}} \ - E^{X_0^-, t_0^-}[\mathbf{1}_{W_T < W^*}] . \tag{2.4.5}$$

2.5 Reward: Expected Total Withdrawals

We use expected total withdrawals as a reward measure. Mathematically, we define expected withdrawals (EW) as

$$\mathrm{EW}(X_0^-, t_0^-) = E_{\mathcal{P}}^{X_0^-, t_0^-}\left[\sum_{i=0}^{M} q_i\right] . \tag{2.5.1}$$

Remark 2.5.1 (No discounting, no mortality weighting)**.** *It is important to note that we do not discount future cash flows in Equation (2.5.1). Since all quantities are assumed to be real (i.e., inflation-adjusted), this effectively implies a real discount rate of zero, a conservative assumption. This approach aligns with the classical work of Bengen (1994). Furthermore, we do not apply mortality weighting to cash flows, which is also consistent with Bengen (1994). For a discussion of this perspective, often summarized as a plan to live, not a plan to die, see Pfau (2018)."*

2.6 Defining a Common Objective Function

Since expected withdrawals (EW) and probability of shortfall (PS) are conflicting measures, we use a scalarization method to determine Pareto optimal points for this multi-objective problem. For a given constant κ, we seek the optimal control $\mathcal{P}_0$ such that the following is maximized,

$$\mathrm{EW}(X_0^-, t_0^-) + \kappa(-\mathrm{PS}(X_0^-, t_0^-)) . \tag{2.6.1}$$

We define (2.6.1) as the EW-PS problem ($EP_{t_0}(\kappa)$) and write the problem formally as

$(EP_{t_0}(\kappa))$:

$$J\left(s,b,t_0^-\right) = \sup_{\mathcal{P}_0 \in \mathcal{A}} \left\{ E_{\mathcal{P}_0}^{X_0^-, t_0^-} \left[\sum_{i=0}^{M} q_i \; + \; \kappa \left(-\mathbf{1}_{W_T < W^*} \right) \overbrace{+\epsilon W_T}^{stabilization} \right. \right.$$
$$\left. \left. \Big| X(t_0^-) = (s,b) \right] \right\}$$

$$\text{subject to} \begin{cases} (S_t, B_t) \text{ follow processes (2.2.3) and (2.2.4)}; \quad t \notin \mathcal{T} \\ W_i^+ = S_i^- + B_i^- - q_i \,; \; X_i^+ = (S_i^+, B_i^+) \\ S_i^+ = p_i(\cdot) W_i^+ \,; \; B_i^+ = (1 - p_i(\cdot)) W_i^+ \\ (q_i(\cdot), p_i(\cdot)) \in \mathcal{Z}(W_i^-, W_i^+, t_i) \\ i = 0, \ldots, M \; ; \; t_i \in \mathcal{T} \end{cases} \tag{2.6.2}$$

The stabilization term ϵW_T serves to avoid ill-posedness. When $W_t \gg W^*$ and $t \to T$, the control has minimal impact on the objective function. This occurs because, in this regime, the probability of shortfall term satisfies $E[\mathbf{1}_{W_T < W^*}] \simeq 0$, meaning the allocation control contributes little to the PS term in the objective. Additionally, for sufficiently high values of W_t, withdrawals are capped at $q_{\max}$, making the withdrawal control independent of W_t in this scenario. The stabilization term ϵW_T is introduced to mitigate potential ill-posedness in this region, ensuring numerical stability and well-defined control behavior.

The objective function in (2.6.2) serves as the basis for the value function in the PDE framework and the loss function for the NN method.

Chapter 3

Neural Network Formulation

We now formally describe the proposed NN framework and demonstrate aforementioned properties. We approximate the control $\mathcal{P}$ directly by using feed-forward, fully-connected neural networks. Given parameters $\boldsymbol{\theta_p}$ and $\boldsymbol{\theta}_q$, i.e. NN weights and biases, $\hat{p}(W_i^+, t_i, \boldsymbol{\theta}_p)$ and $\hat{q}(W_i^-, t_i, \boldsymbol{\theta}_q)$ approximate the controls p_i and q_i respectively,

$$\begin{aligned}
\hat{q}(W_i^-, t_i^-, \boldsymbol{\theta}_q) &\simeq q_i(W_i^-) \ ; \ i = 0, \dots, M \\
\hat{p}(W_i^+, t_i^+, \boldsymbol{\theta}_p) &\simeq p_i(W_i^+) \ ; \ i = 0, \dots, M-1 \\
\hat{\mathcal{P}} = \{(\hat{q}(\cdot), \hat{p}(\cdot))\} &\simeq \mathcal{P} \ .
\end{aligned}$$

The functions $\hat{p}$ and $\hat{q}$ take time as one of the inputs, and therefore we can use two NN functions to approximate control $\mathcal{P}$ across time instead of defining a NN at each rebalancing time. In this section, we discuss how we solve the problem (2.6.2) using this approximation and then provide a description of the NN architecture that is used. We discuss the precise formulation used by the NN, including the activation functions that encode the stochastic constraints.

3.1 Neural Network Optimization for $EP_{t_0}(\kappa)$

We begin by formulating the neural network (NN) optimization problem based on the stochastic optimal control problem (2.6.2). In particular, we recall that in formulations (2.3.4) and (2.3.6), the controls q_i and p_i are functions of wealth and time.

Our objective is to determine the NN weights $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}_q$ by solving (2.6.2), where the NN parameterized functions $\hat{q}(W_i^-, t_i^-, \boldsymbol{\theta}_q)$ and $\hat{p}(W_i^+, t_i^+, \boldsymbol{\theta}_p)$ approximate feasible controls $(q_i, p_i) \in \mathcal{Z}(W_i^-, W_i^+, t_i)$ for $t_i \in \mathcal{T}$.

For a given set of controls $\hat{\mathcal{P}}$, we define the NN performance criterion V_{NN} as:

$$V_{NN}(\hat{\mathcal{P}}, s,b, t_0^-) \;=\; E_{\hat{\mathcal{P}}_0}^{X_0^-, t_0^-}\left[\sum_{i=0}^{M} \hat{q}_i \;+\; \kappa\Big(-\mathbf{1}_{W_T<W^*}\Big) + \epsilon W_T \Big| X(t_0^-) = (s,b) \right] .$$

$$\text{subject to } \begin{cases} (S_t, B_t) \text{ follow processes (2.2.3) and (2.2.4)}; \quad t \notin \mathcal{T} \\ W_i^+ = S_i^- + B_i^- - q_i \,;\; X_i^+ = (S_i^+, B_i^+) \\ S_i^+ = \hat{p}_i(\cdot) W_i^+ \,;\; B_i^+ = (1 - \hat{p}_i(\cdot)) W_i^+ \\ (\hat{q}_i(\cdot), \hat{p}_i(\cdot)) \in \mathcal{Z}(W_i^-, W_i^+, t_i) \\ i = 0, \ldots, M \;;\; t_i \in \mathcal{T} \end{cases} \tag{3.1.1}$$

The optimal value function J_{NN} (at t_0^-) is then given below,

$$J_{NN}(s, b, t_0^-) = \sup_{\hat{\mathcal{P}} \in \mathcal{A}} \; V_{NN}(\hat{\mathcal{P}}, s,b, t_0^-) \; . \tag{3.1.2}$$

Next we describe the structure of the neural networks and feasibility encoding.

3.2 Neural Network Framework

Consider two fully connected feed-forward neural networks (NNs), where $\hat{p}$ and $\hat{q}$ are parameterized by vectors $\boldsymbol{\theta}_p \in \mathbb{R}^{\nu_p}$ and $\boldsymbol{\theta}_q \in \mathbb{R}^{\nu_q}$, respectively, representing the NN weights and biases. The two networks may differ in their activation functions, as well as in the number of hidden layers and nodes per layer.

Each NN takes the same type of input, $(W(t_i), t_i)$, but their specific roles differ. The withdrawal NN, $\hat{q}$, receives the state variable before withdrawal, $(W(t_i^-), t_i)$, while the allocation NN, $\hat{p}$, takes in the state variable after withdrawal, $(W(t_i^+), t_i)$.

To ensure that NN generates feasible controls as specified in (3.2.2), we apply a modified sigmoid activation function to scale the output of the withdrawal NN, $\hat{q}$, in accordance with the constraints of the $EP_{t_0}(\kappa)$ problem (2.6.2) on the withdrawal amount q_i, as defined in Equation (2.3.9). This transformation enables unconstrained optimization of the NN training parameters.

Specifically, for $x \in [0,1]$, we use the function $f(x) := a+(b-a)x$ to scale the output to the range $[a,b]$. We restrict the withdrawal $\hat{q}$ to lie within $[q_{\min}, q_{\max}]$, where the withdrawal range $q_{\max} - q_{\min}$ depends on wealth W^-, as defined in (2.3.9). The range of permitted withdrawals is given by:

$$\text{range} = \begin{cases} q_{\max} - q_{\min} \ , \text{ if } W_i^- > q_{\max} \ ; \\ W^- - q_{\min} \ , \ \text{ if } q_{\min} < W_i^- < q_{\max} \ ; \\ 0 \ , \qquad\qquad \text{if } W_i^- < q_{\min} \ . \end{cases}$$

More concisely, we have the following mathematical expression:

$$\text{range} = \max\left((\min(q_{\max}, W^-) - q_{\min}), 0\right) .$$

Let $z \in \mathbb{R}$ be the NN output before the final output layer of $\hat{q}$. Note that z depends on input features, state and time, before being transformed by the activation function. We then have the following expression for the final withdrawal,

$$\begin{aligned} \hat{q}(W^-, t, \boldsymbol{\theta_q}) &= q_{\min} + \text{range} \cdot \left(\frac{1}{1+e^{-z}}\right) \\ &= q_{\min} + \max\left((\min(q_{\max}, W^-) - q_{\min}), 0\right) \left(\frac{1}{1+e^{-z}}\right) . \end{aligned}$$

Note that the sigmoid function $\frac{1}{1+e^{-z}}$ is a mapping from $\mathbb{R} \to [0,1]$.

Similarly, we use a softmax activation function on the NN output of the $\hat{p}$, in order to impose no-shorting and no-leverage constraints.

With these output activation functions, it can be easily verified that $(\hat{q}_i(\cdot), \hat{p}_i(\cdot)) \in \mathcal{Z}\left(W_i^-, W_i^+, t_i\right)$ always. Using defined NN, this transforms problem (3.1.2) of finding an optimal $\hat{\mathcal{P}}$ to solving the optimization problem below:

$$\begin{aligned} \hat{J}_{NN}(s,b,t_0^-) &= \sup_{\boldsymbol{\theta_q} \in \mathbb{R}^{\nu_q}} \sup_{\boldsymbol{\theta_p} \in \mathbb{R}^{\nu_p}} \hat{V}_{NN}(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p, s, b, t_0^-) \\ &= \sup_{(\boldsymbol{\theta_q}, \boldsymbol{\theta_p}) \in \mathbb{R}^{\nu_q + \nu_p}} \hat{V}_{NN}(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p, s, b, t_0^-) \end{aligned} \tag{3.2.1}$$

where $\hat{V}_{NN}$ is defined as:

$$\hat{V}_{NN}(\boldsymbol{\theta}_q,\boldsymbol{\theta}_p, s,b, t_0^-) = \frac{1}{N}\sum_{j=1}^{N}\left[\sum_{i=0}^{M}\hat{q}((W_i)^j, t_i; \boldsymbol{\theta}_q) \;+\; \kappa\left(-\mathbf{1}_{(W_i)^j<W^*}\right) + \epsilon(W_T)^j \middle| X(t_0^-) = (s,b)\right]$$

$$\text{subject to } \begin{cases} ((S_t)^j, (B_t)^j) \text{ drawn from the } j^{th} \text{ sample of returns;} \quad t \notin \mathcal{T} \\ (W_i^+)^j = (S_i^-)^j + (B_i^-)^j - \hat{q}\left((W_{t_i}^-)^j, t_i, \boldsymbol{\theta_q}\right) \;;\; (X_i^+)^j = (S_i^+, B_i^+)^j \\ (S_i^+)^j = \hat{p}\left((W_i^+)^j, t_i, \boldsymbol{\theta_p}\right)\; (W_i^+)^j \;;\; (B_i^+)^j = (1-\hat{p}\left((W_i^+)^j, t_i, \boldsymbol{\theta_p}\right))\; (W_i^+)^j \\ (\hat{q}_i(\cdot), \hat{p}_i(\cdot)) \in \mathcal{Z}\left((W_i^-)^j, (W_i^+)^j, t_i\right) \\ i = 0, \ldots, M \;;\; t_i \in \mathcal{T} \end{cases}, \tag{3.2.2}$$

where the superscript j represents the j^{th} path of joint asset returns and N is the total number of sampled paths. For subsequent benchmark comparison, we generate price paths using processes (2.2.3) and (2.2.4). However, we are agnostic as to the method used to generate these paths. We assume that random sample paths are independent. However, correlation between returns of different assets within a sample path can be modeled.

It is important to note that while the original control $\mathcal{P}$ is constrained by (2.3.12), the reformulated problem in (3.2.1) allows for unconstrained optimization over $\boldsymbol{\theta}_q$ and $\boldsymbol{\theta}_p$. This enables us to solve it directly using standard gradient descent techniques. Specifically, we employ the Adam stochastic gradient descent algorithm to determine the optimal parameters $\boldsymbol{\theta}_{\boldsymbol{q}}^*$ and $\boldsymbol{\theta}_{\boldsymbol{p}}^*$.

The output of NN $\hat{q}$ represents the withdrawal amount, while the output of NN $\hat{p}$ determines the asset allocation weights.

Figure 3.2.1 illustrates the proposed NN framework, highlighting the following key aspects:

- Time is an input to both NNs, which means that the parameter vectors $\boldsymbol{\theta}_q$ and $\boldsymbol{\theta}_p$ remain constant and do not vary over time.
- At each re-balancing time, the observed wealth before withdrawal is used to construct the feature vector for $\hat{q}$. The computed withdrawal then determines the wealth after withdrawal, which serves as an input feature for $\hat{p}$.
- Standard sigmoid activation functions are applied at each hidden layer output.

- The activation function for the withdrawal control $\hat{q}$ differs from that of the allocation control $\hat{p}$. $\hat{q}$ uses a modified sigmoid function to transform its output according to (2.3.9). $\hat{p}$ employs a softmax activation function, ensuring that its output consists of strictly positive portfolio weights summing to one, as specified in (2.3.10). By designing NN outputs this way, we can enforce feasibility constraints through activation functions while maintaining an unconstrained optimization framework for training.

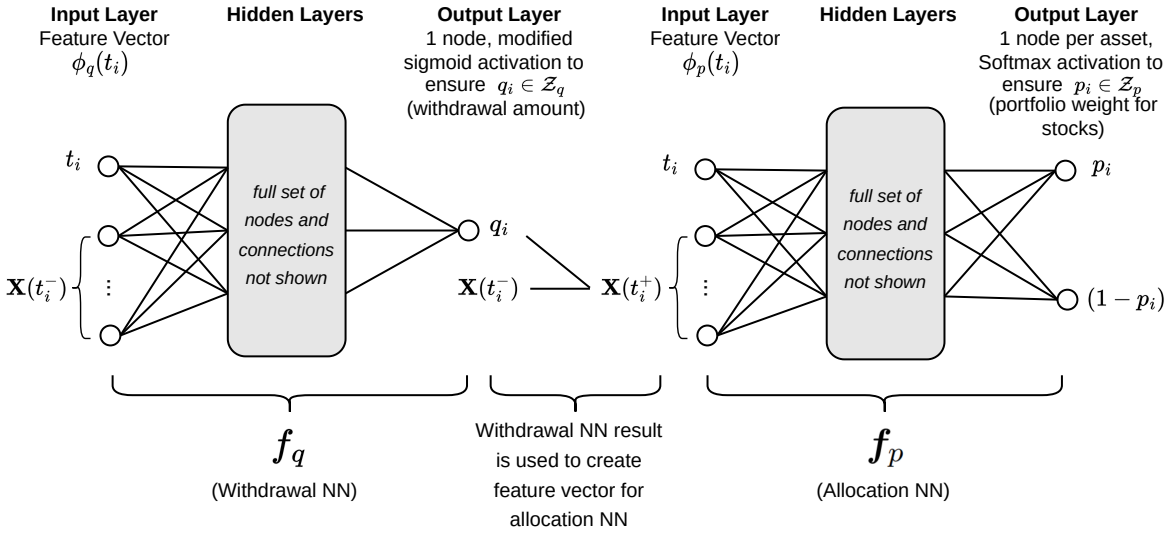


FIGURE 3.2.1: *Illustration of the NN framework as per Section 3.2, adapted from Chen et al. (2023).*

Chapter 4

Data

For our computational study, we use monthly data from the Center for Research in Security Prices (CRSP) spanning the period from January 1926 to December 2022. Specifically, we use the T-bill index for bond assets and the CRSP value-weighted total return index for stock assets. Since these indices are reported in nominal terms, we adjust them for inflation using the U.S. Consumer Price Index (CPI), also provided by CRSP. This ensures that all returns are expressed in real terms, aligning with the goal of modeling retirement wealth in inflation-adjusted terms.

4.1 Stochastic Model Calibration

Throughout this study, we refer to *synthetic data* as data generated by parametric stochastic models (SDEs), described in Section 2.2. The parameters of these models are estimated using the threshold technique (Cont and Mancini, 2011; Dang and Forsyth, 2016; Mancini, 2009), ensuring consistency with historical CRSP data. To maintain a real-return perspective, the data is inflation-adjusted before calibration. Table 4.1.1 presents the parameter values obtained through this process, where μ represents drift rate, σ represents volatility, λ represents intensity of the Poisson process, u represents probability of upward jump. The parameters η_1 and η_2 denote the jump multipliers for upward and downward jumps, respectively. Finally, ρ_{sb} represents the correlation, which is computed by excluding returns that coincide with jumps in either series, ensuring a more accurate representation of market dynamics (see Dang and Forsyth (2016) for details on jump filtering techniques).

Stock	μ^s	σ^s	λ^s	u^s	η_1^s	η_2^s	ρ_{sb}
	0.086211	0.1477986	0.319587	0.2258064	4.35814	5.540746	0.0875
T-bill	μ^b	σ^b	λ^b	u^b	η_1^b	η_2^b	ρ_{sb}
	0.0031	0.0133	0.4845	0.4043	65.3426	56.9756	0.0875

TABLE 4.1.1: *Sample period 1926:1 to 2022:12.*

4.2 Bootstrap Resampling

We refer to *historical data* as data generated using the stationary block bootstrap method (Dichtl et al., 2016a; Patton et al., 2009; Politis and Romano, 1994; Politis and White, 2004). This method involves resampling historical CRSP data by repeatedly drawing random blocks of variable length, with replacement. The block size follows a geometric distribution with a specified expected length. To preserve asset return correlations, we employ a paired sampling strategy, simultaneously selecting stock and bond returns from corresponding time periods. This technique effectively reshuffles historical data while retaining some aspects of its serial correlation. For details on block bootstrap resampling, see Forsyth and Vetzal (2019).

In this study, we apply the historical data for robustness checks, as detailed in Section 8.3. Note that The probability of duplicating a resampled path remains negligible for any practical sample size (Ni et al., 2022). We will also test the sensitivity of the results in a range of block sizes in numerical experiments.

4.3 Conclusion

To train our neural networks effectively, we require a sufficiently large number of sampled paths (N) to adequately capture market dynamics. First, we generate training data using Monte Carlo simulations of the parametric models defined in (2.2.3) and (2.2.4). However, in our proposed data-driven neural network framework, we can also rely solely on non-parametric historical return trajectories rather than the explicit parametric model structures.

In later sections, we present results comparing neural networks trained on parametrically simulated data and historically resampled data. Additionally, we evaluate the robust-

ness of the neural network framework using out-of-distribution test scenarios and analyze control sensitivity using historical data.

Chapter 5

Challenges of the PS problem

In this section, we explore the challenges of solving the PS problem using Monte Carlo simulation coupled with stochastic gradient descent. Specifically, we simplify the problem to a one-period portfolio allocation setting and compare the derivative valued of the PS term obtained from the exact density function and Monte Carlo simulation. Our goal is to illustrate how the presence of an indicator function in the PS term complicates the computation of derivatives in the NN framework using Monte Carlo simulation.

5.1 Indicator Function and Sigmoid Function

According to the $EP_{t_0}(\kappa)$ (2.6.2), the most challenging part in optimizing this objective function is the probability of shortfall (PS) term. We recall from Section 2.4, the indicator function in the PS term is:

$$\mathbf{1}_{W_T<W^*} = \begin{cases} 1 & \text{if } W_T < W^*, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1.1}$$

which is discontinuous when $W_T = W^*$. So mathematically, we cannot calculate its derivative. We approximate the indicator function by using the sigmoid function as follows:

$$\sigma(x) = \frac{1}{1 + e^{-\rho x}}, \tag{5.1.2}$$

where x represents $W^* - W_T$ and ρ is the parameter. The sigmoid function can approximate the indicator function by increasing the parameter ρ, which is shown in Figure 5.1.1.

It is clear that sigmoid function can approach the indicator function when the sigmoid parameter ρ becomes very large.

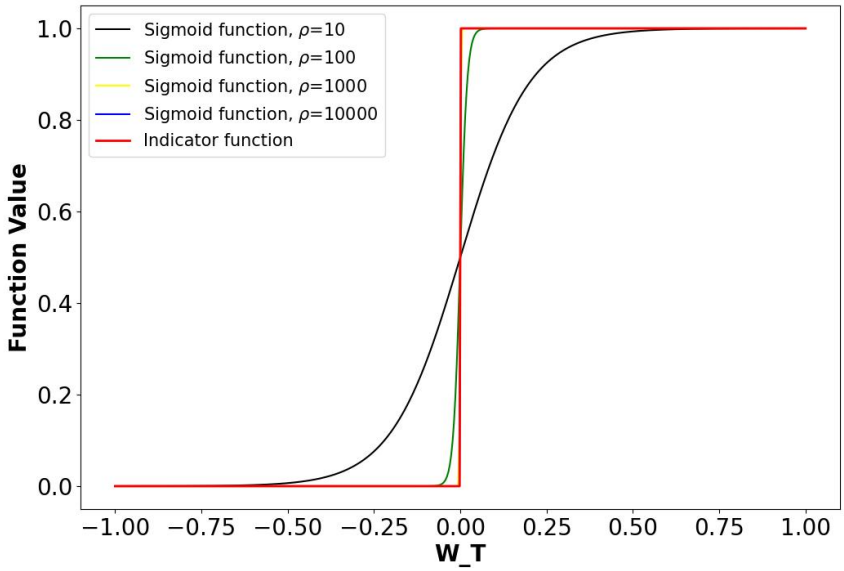


FIGURE 5.1.1: *Plot of the indicator function and the sigmoid function with different ρ values.*

After introducing the sigmoid function as a differentiable approximation for the indicator function, we analyze the derivative of the PS term in the following section. The PS term can be expressed in two ways: one using the indicator function directly and another by integrating the density function of W_T:

$$E[\mathbf{1}_{W_T<W^*}] = P(W_T < W^*) = \int_{-\infty}^{W^*} g(W')dW', \tag{5.1.3}$$

where $P(W_T < W^*)$ is the probability of $W_T < W^*$; $g(W_T)$ is the density function of W_T.

5.2 Simplified One-Period Portfolio Question

To better understand the challenges of the PS problem, this section compares the derivative of the PS term computed using the exact density function and Monte Carlo simulation (em-

ployed in neural network optimization) within a simplified portfolio management setting. To facilitate this analysis, we introduce a one-period portfolio model.

We consider an investment in a portfolio consisting of a stock S and a bond B. The investment is held for a single period, after which the portfolio value W_T is determined by the weighted combination of stock and bond returns. Assuming the stock price follows a Geometric Brownian Motion (GBM) and the bond price evolves at a constant risk-free rate, the portfolio value at the end of the period can be expressed as follows:

$$\begin{aligned} \frac{dS}{S} &= \mu\ dt + \sigma\ dZ, \\ \frac{dB}{B} &= r\ dt. \end{aligned} \tag{5.2.1}$$

The portfolio dynamics can be expressed as follows:

$$\begin{aligned} \frac{dW}{W} &= p\frac{dS}{S} + (1-p)\frac{dB}{B} \\ &= [(1-p)r + p\mu]\ dt + \sigma p\ dZ, \end{aligned} \tag{5.2.2}$$

where p is the allocation in stock, r is the interest rate in bond, μ is the stock return, σ is the stock volatility and dZ is the increment of a Wiener process.

This leads to the terminal wealth W_T as:

$$\frac{W_T}{W_0} = \exp\left[\left((1-p)r + p\mu - \frac{1}{2}p^2\sigma^2)T + \sigma p(Z(T) - Z(0))\right)\right]. \tag{5.2.3}$$

Given the two equivalent formulations of the PS term in Equation (5.1.3), we proceed to compute the derivative using both the exact density function and Monte Carlo simulation. The following sections present these calculations and highlight the challenges associated with each approach.

5.2.1 Derivative Calculation using the Density Function

To derive the density function of W_T, we take the logarithm of Equation (5.2.3):

$$\ln\frac{W_T}{W_0} = \left[\left((1-p)r + p\mu - \frac{1}{2}p^2\sigma^2)T + \sigma p(Z(T) - Z(0))\right)\right]. \tag{5.2.4}$$

Since $Z(T) - Z(0) \sim \mathcal{N}(0, T)$, we have the distribution of $\ln W_T$ as below:

$$\ln W_T \sim N\left(\ln W_0 + \left((1-p)r + p\mu - \frac{1}{2}\sigma^2 p^2\right)T, \sigma^2 p^2 T\right). \tag{5.2.5}$$

Then we have the closed form density function of W_T:

$$g(W_0, W_T, p, T) = \frac{1}{W_T \sigma p \sqrt{2\pi T}} \exp\left(-\frac{\left[\ln W_T - \ln W_0 - \left((1-p)r + p\mu - \frac{1}{2}p^2\sigma^2\right)T\right]^2}{2\sigma^2 p^2 T}\right). \tag{5.2.6}$$

The PS term can then be expressed by the explicit density function:

$$V(W_0, p, T) = E[1_{W_T < W^*}] = \int_0^{W^*} g(W_0, W', p, T) dW'. \tag{5.2.7}$$

This integral calculates the probability that W_T is below a threshold W^*. Considering W_T and W^* are positive under this scenario, we can convert it into a standard normal form:

$$V = P(W_T < W^*) = P(\ln W_T < \ln W^*). \tag{5.2.8}$$

Using the normal distribution transformation, we can have:

$$V = P\left(\frac{\ln W_T - \ln W_0 - [(1-p)r + p\mu - \frac{1}{2}p^2\sigma^2]T}{\sigma p\sqrt{T}} < \frac{\ln W^* - \ln W_0 - [(1-p)r + p\mu - \frac{1}{2}p^2\sigma^2]T}{\sigma p\sqrt{T}}\right). \tag{5.2.9}$$

If we define y as below:

$$y = \frac{\ln W^* - \ln W_0 - [(1-p)r + p\mu - \frac{1}{2}p^2\sigma^2]T}{\sigma p\sqrt{T}}, \tag{5.2.10}$$

then the PS term can be defined as:

$$V(y) = P(Y < y) = \Phi(y), \tag{5.2.11}$$

where $\Phi(y)$ is the cumulative distribution function (CDF) of the standard normal distribution, which is directly related to the error function:

$$\Phi(y) = \frac{1}{2}\left(1 + \text{Erf}\left(\frac{y}{\sqrt{2}}\right)\right), \tag{5.2.12}$$

where error function (Erf) can be expressed as $\text{Erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$.

Thus,

$$V = \frac{1}{2}\left(1 + \text{Erf}\left(\frac{\ln W^* - \ln W_0 - [(1-p)r + p\mu - \frac{1}{2}p^2\sigma^2]T}{\sigma p\sqrt{2T}}\right)\right). \tag{5.2.13}$$

To Compute $\frac{\partial V}{\partial p}$ from the integral definition, we have

$$\frac{\partial V}{\partial p} = \int_0^{W^*} \frac{\partial}{\partial p}[g(W_0, W', p, T)]dW'. \tag{5.2.14}$$

By differentiating V using the chain rule, we have

$$\begin{aligned} \frac{\partial V}{\partial p} &= \frac{\partial V}{\partial y}\cdot\frac{\partial y}{\partial p} \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y^2}{2}\right)\cdot\frac{\partial y}{\partial p}, \end{aligned} \tag{5.2.15}$$

where

$$\frac{\partial y}{\partial p} = \frac{(r + \frac{1}{2}\sigma^2p^2)T - (\ln W^* - \ln W_0)}{\sigma p^2\sqrt{T}}. \tag{5.2.16}$$

Thus,

$$\frac{\partial V}{\partial p} = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y^2}{2}\right)\cdot\frac{(r + \frac{1}{2}\sigma^2p^2)T - (\ln W^* - \ln W_0)}{\sigma p^2\sqrt{T}}. \tag{5.2.17}$$

It is clear that the derivative of the PS term calculated by the exact density function (5.2.17) is a smooth function.

5.2.2 Derivative Calculation using Monte Carlo Simulation

In the NN framework, we use Monte Carlo simulation to compute the derivative of the PS term. Since the indicator function is discontinuous and non-differentiable, calculating its derivative for each sample is challenging. To address this, we replace the indicator function with the sigmoid function, which, as previously discussed, serves as a smooth and effective approximation. Consequently, the PS term V can be approximated by $\tilde{V}$:

$$V = E[\mathbf{1}_{W_T<W^*}] \approx E[\sigma(W^* - W_T)], \tag{5.2.18}$$

$$\tilde{V} = \frac{1}{M}\sum_{i=1}^{M} \sigma(W^* - W_T^i), \tag{5.2.19}$$

where the sigmoid function σ referring to Equation (5.1.2) serves as a differentiable approximation of the indicator function and M represents the number of samples. The derivative of $\tilde{V}$ with respect to the portfolio allocation p is given by:

$$\begin{aligned}\frac{\partial \tilde{V}}{\partial p} &= \frac{1}{M}\sum_{i=1}^{M} \frac{\partial \sigma(W^* - W_T^i)}{\partial W_T^i}\frac{\partial W_T^i}{\partial p} \\ &= \frac{1}{M}\sum_{i=1}^{M} \Big(-\rho\sigma(W^* - W_T^i)(1-\sigma(W^* - W_T^i))\Big)\frac{\partial W_T^i}{\partial p},\end{aligned} \tag{5.2.20}$$

where $\sigma(W^* - W_T^i)$ refers to Equation (5.1.2) and using Equation (5.2.3) we have:

$$\begin{aligned}\frac{\partial W_T^i}{\partial p} &= W(0)\frac{\partial}{\partial p}\left[\exp\left((1-p)r + p\mu - \frac{1}{2}p^2\sigma^2)T + \sigma p Z_T^i\right)\right] \\ &= W(0)\left((-r+\mu-p\sigma^2)T + \sigma Z_T^i\right)\exp\left(((1-p)r + p\mu - \frac{1}{2}p^2\sigma^2)T + \sigma p Z_T^i\right),\end{aligned} \tag{5.2.21}$$

where Z_T^i is the i^{th} sample from a standard normal distribution.

Comparing the derivative calculated using the density function (5.2.17) with the derivative approximated via Monte Carlo simulation (5.2.20), it is evident that the indicator function does not affect the derivative computation when using the exact density function. However, in the Monte Carlo simulation, we must explicitly compute the derivative of the indicator function for each sampled value, leading to poor numerical performance. The following Section 5.3 presents a comparison of the results obtained from the exact density function and the sampling approach.

5.3 Results

Investment horizon T (years)	1
Initial portfolio value W_0	1
Threshold portfolio value W^*	0.9
Equity fraction range p	[0,1]
Stock return μ	0.05
Stock volatility σ	0.2
Interest rate r	0

TABLE 5.3.1: *One-period portfolio problem set up.*

We present the parameter setting for the one-period portfolio problem in Table 5.3.1. The derivative of the PS term computed using the exact density function (5.2.17) is shown in Figure 5.3.1. We observe that when the allocation of stocks is in the range $[0, 0.1]$, the derivative of the PS term is zero. This is because assigning such a small proportion to stocks implies that almost all funds are invested in bonds, which have a zero interest rate. As a result, $\text{Prob}(W_T < W^*) = 0$, meaning the derivative of the PS term is also zero.

Next, in Figure 5.3.2, we compare the derivative estimates obtained by Monte Carlo simulation (5.2.20) with those computed using the density function. To analyze the effects of different numbers of simulations (M) and different values of the parameter (ρ) of the sigmoid function, we conduct four experiments as follows:

- $M = 1{,}000$, $\rho = 100$ (black dashed line);
- $M = 1{,}000$, $\rho = 10{,}000$ (green dashed line);
- $M = 100{,}000$, $\rho = 100$ (blue dashed line);
- $M = 100{,}000$, $\rho = 10{,}000$ (purple dashed line).

The red line represents the derivative computed using the density function. The results show that for a fixed parameter (ρ) of the sigmoid function, increasing the number of simulations (M) leads to smoother and more accurate Monte Carlo estimates of the derivative of the PS term. Conversely, for a fixed number of simulations (M), increasing the parameter (ρ) of the sigmoid function makes the sigmoid function a closer approximation of the indicator function. However, this causes the derivative of the PS term to become more oscillatory and less accurate.

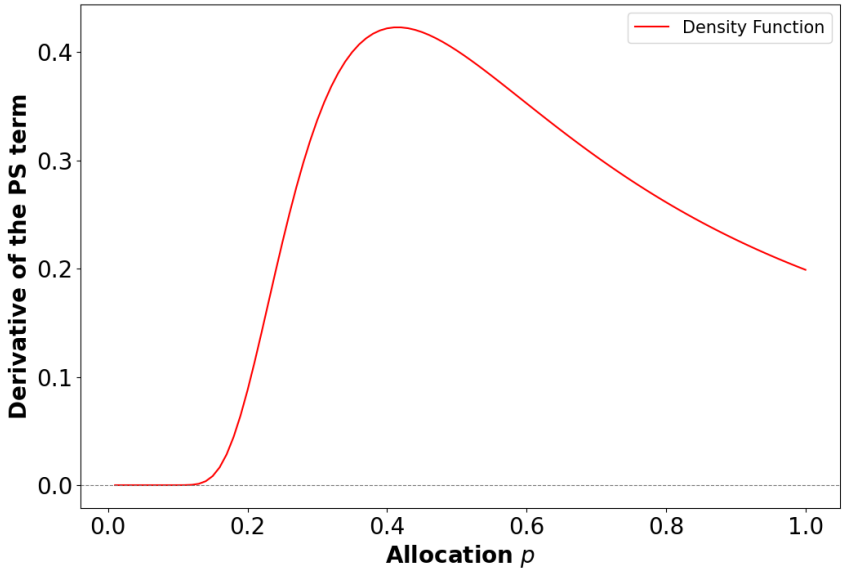


FIGURE 5.3.1: *Derivative of the PS term calculated by the density function (computed from Equation (5.2.17)). Scenario in Table 5.3.1.*

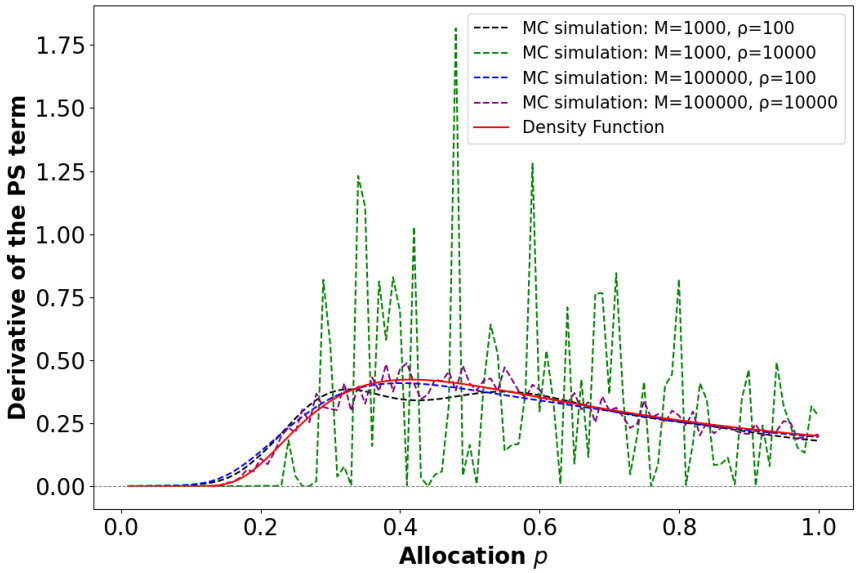


FIGURE 5.3.2: *Comparison derivative of the PS term calculated by the density function (computed from Equation (5.2.17)) and Monte Carlo simulation (computed from Equation (5.2.20)). Scenario in Table 5.3.1.*

5.4 Conclusion

We aim to understand the challenges of using the NN framework to produce reliable results for the PS problem. After introducing the sigmoid function as a differentiable approximation of the indicator function, we examine how the indicator function in the PS term affects the results obtained through Monte Carlo simulation.

In the NN framework, the derivative of the PS term is computed using Monte Carlo simulation. In contrast, the PDE method does not use a sampling method to approximate the PS term. Mathematically, we observe that due to the nature of Monte Carlo simulation, the derivative of the indicator function must be computed for every sample. In contrast, the density function approach avoids the discontinuity and non-differentiability introduced by the indicator function, leading to a more stable and accurate derivative calculation.

This distinction has important implications when solving $EP_{t_0}(\kappa)$ (2.6.2). While the PDE method provides accurate results by avoiding sampling, Monte Carlo simulation struggles with the non-differentiability of the indicator function, leading to instability. To mitigate this issue, we approximate the indicator function with a sigmoid function ($\rho = 100$), offering a smooth and differentiable alternative that enhances the stability of the optimization process.

Chapter 6

Impact of Hyper-parameters

In this section, we explore the impact of NN hyper-parameters on optimization performance, including the number of iterations, batch size, learning rate, and learning rate decay strategies (cyclic and multistep). The results in this paper are obtained by repeating experiments five times and calculating the mean and standard deviation to obtain valid and reasonable conclusions.

6.1 The Number of Iterations

The number of iterations refers to how many times the model updates its parameters using gradient descent. It depends on the number of epochs and the batch size:

$$N_{\text{iter}} = \frac{\text{Total Training Samples}}{\text{Batch Size}} \times \text{Number of Epochs} .$$

The number of epochs represents how many times the entire training dataset is passed through the model during the training process. In this paper, we do not explicitly fix the number of epochs; instead, we look for a good choice of the number of iterations and batch size.

The batch size defines the number of training samples used in each gradient update. In theory, we could use the entire training dataset as the batch size, or alternatively, a subset of the training samples, known as a mini-batch. In practice, mini-batch training is always used for neural networks. Although reducing the batch size increases the number of

iterations for a fixed number of epochs, each iteration requires less memory using mini-batch compared to using the entire dataset. Additionally, smaller batches can speed up training by enabling more frequent gradient updates. Considering our computational constraints, we experimented with batch sizes of 1,024 and 2,048 and found that a batch size of 2,048 yielded better optimization results.

In general, increasing the number of iterations improves convergence and accuracy. Recalling the one-period portfolio problem, Figure 5.3.2 shows that for a fixed parameter of the sigmoid function, the derivative of the PS term becomes smoother and more accurate as the number of simulations increases. However, due to computational limitations, a balance must be struck between accuracy and efficiency. To assess this trade-off, we conducted experiments with iteration counts of 50,000, 100,000, and 150,000. The results indicate that 100,000 iterations provide a satisfactory level of accuracy while maintaining reasonable computational efficiency.

6.2 Learning Rate

The learning rate determines the step size in the gradient descent:

$$\theta_p^{(t+1)} = \theta_p^{(t)} - \eta \nabla V_{NN}(\theta_p^{(t)})$$

$$\theta_q^{(t+1)} = \theta_q^{(t)} - \eta \nabla V_{NN}(\theta_q^{(t)})$$

where θ_p and θ_q refers to Section 3 (representing NN weights and biases); η is learning rate that controls the step size of the update.

A high learning rate may cause the model to overshoot optimal solutions, leading to instability, whereas a low learning rate results in slow convergence. A good η achieves a balance between convergence speed and stability.

Since the learning rate significantly impacts training performance, we carried out extensive experiments with η values ranging from 0.01 to 0.5 across different κ values, where κ represents the weight of the PS term in (2.6.2). Our experiments determine the good learning rates η for different κ values, as summarized in Table 6.2.1.

κ	η
10	0.09
100	0.01
1,000	0.01
3,000	0.01
5,000	0.04
7,000	0.03
15,000	0.01
20,000	0.008
100,000	0.005

TABLE 6.2.1: *Learning rates η for different κ.*

6.3 Learning Rate Decay Schedule

A learning rate decay schedule adjusts η over time to improve training efficiency and stability. We compare two common strategies:

- Cyclic Learning Rate Decay:

  Cyclic learning rates periodically increase and decrease within a range:

  $$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos(\frac{\pi t}{L})\right) \ ,$$

  where $\eta_{\min}$ and $\eta_{\max}$ are the minimum and maximum learning rates, L is the cycle length. This helps escape local minima and improves convergence.

- MultiStep Learning Rate Decay:

  The learning rate is reduced at predefined epochs:

  $$\eta_t = \begin{cases} \eta_0, & t < L_1 \\ \gamma\eta_0, & L_1 \leq t < L_2 \\ \gamma^2\eta_0, & L_2 \leq t \end{cases} \ ,$$

  where γ is a decay factor usually lying in (0,1), and L_1, L_2 are predefined steps.

Experimental results show that the multistep method consistently produces lower withdrawal values, while the cyclic method achieves higher ones. We hypothesize that this occurs because the multistep method reduces the learning rate in the final stage to fine-tune parameters more carefully, leading the network to converge more cautiously to a local minimum, which results in lower withdrawals. In contrast, the cyclic method periodically varies the learning rate between a lower and upper bound, helping the model escape local minima. However, this also means the model does not fine-tune its parameters as precisely in the later stages, leading to higher withdrawal values.

Overall, our results suggest that, in most cases, the cyclic learning rate schedule outperforms the multistep schedule. By allowing the model to explore different regions of the optimization landscape, the cyclic approach helps prevent premature convergence and enhances overall performance.

6.4 Hyper-parameters for NN

In conclusion, the hyper-parameters used in training the NN are listed in Table 6.4.1. The training loop tracks the minimum loss function value as training progresses and selects the model that had given the optimal loss function value based on the entire training dataset by the end of the specified number of training epochs.

NN framework hyper-parameter	Value
Hidden layers per network	2
# of nodes per hidden layer	10
Nodes have biases	True
# of iterations (#itn)	100,000
SGD mini-batch size	2,048
# of training paths	2.56×10^5
Optimizer	Adaptive Momentum
Initial Adam learning rate η for $(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p)$	See Table 6.2.1
Adam learning rate decay schedule	(Adam_eta$*$0.01, Adam_eta)
Adam_eta	0.1
Adam β_1	0.9
Adam β_2	0.998
Adam weight decay (L2 Penalty)	0.0001
Take running minimum as result	True

TABLE 6.4.1: *Hyper-parameters used in training the NN framework for numerical experiments presented in this paper.*

Chapter 7

Computational Results

We now compare the performance of the optimal control derived from the PDE and NN methods on synthetic data, using the investment specifications in Table 7.0.1. The effectiveness of each strategy is evaluated based on the objective function in 2.6.2, which balances reward (EW) and risk (PS). To construct the efficient frontier in the (EW, PS) plane, we vary κ, where each point on the curve represents the (EW, PS) performance at a computed optimal Pareto solution.

We present results from the NN framework described in Section 3 and validate its accuracy by comparing it to the PDE method as a ground truth. Furthermore, we conduct a detailed analysis of the efficient frontier's knee point at $\kappa = 7{,}000$, as it offers a favorable risk-reward trade-off. For this critical point, we examine control heat maps and CDF plots to assess whether PS serves as an effective risk measure.

7.1 Accuracy of Strategy Computed from NN framework

7.1.1 Results of NN method using Transfer Learning

Transfer learning is a machine learning technique where a pre-trained model is adapted to a new but related task. Instead of training a model from scratch, transfer learning leverages knowledge from a source domain to improve learning in a target domain. In the ES problem, the ES measure (CVaR) is influenced only by the sample paths below the 5th

Investment horizon T (years)	30
Equity market index	CRSP Cap-weighted index (real)
Bond index	30-day T-bill index (real)
Initial portfolio value W_0	1000
Threshold portfolio value W^*	-10.7
Stabilization parameter ϵ	-10^{-4}
Cash withdrawal times	$t = 0,1,\ldots,29$
Withdrawal range	$[30, 60]$
Equity fraction range	[0,1]
Borrowing spread μ_c^b	0.03
Re-balancing interval (years)	1
Market parameters	See Table 4.1.1

TABLE 7.0.1: *Problem setup and input data. Monetary units: thousands of dollars.*

percentile of terminal wealth. Since these critical paths are sparse, optimization can be challenging. Transfer learning helps stabilize training, reducing the risk of convergence to local minima, particularly for higher values of κ (Chen et al. (2023)). In the PS problem, we hypothesize that transferring from smaller κ values should improve performance for larger κ values. A lower weight on the PS term makes it easier for the neural network to find an optimal solution, whereas a higher weight increases the complexity of optimization. Therefore, transfer learning may provide an effective way to guide the model toward better solutions as κ increases.

To test this, we fix an initial value of κ and use the pre-trained model from this setting to transfer learning to other κ values. Specifically, we apply transfer learning to every κ value starting from $\kappa = 10$, leveraging the parameter and weight settings optimized for this baseline. Our hypothesis is that since the neural network model at $\kappa = 10$ prioritizes increasing withdrawals, this setting might help boost expected withdrawal values for larger κ values. The learning rates are set as shown in Table 6.2.1.

From Figure 7.1.1, we observe that transfer learning produces results that deviate significantly from the PDE benchmark. While we initially expect it to enhance withdrawals for larger κ values, the results indicate that, for smaller κ, transfer learning provides some improvement while for larger κ, transfer learning becomes counterproductive. Given this poor performance, we conduct additional experiments using different initial κ values—100, 1,000, and 3,000—to improve performance across other κ values. However, in all cases, transfer learning results are nearly five times worse than training from scratch, indicating

that transfer learning is ineffective or even detrimental in this context.

Unlike in the ES problem, where the learned policy follows a similar trend across different κ values, making transfer learning effective; the PS problem presents fundamentally different optimization landscapes for different κ values. When ρ is large, the non-linearity and non-convexity of the PS term make the optimization process highly sensitive to initialization. The sharp transitions introduced by the PS term create unstable learning dynamics, rendering transfer learning not only ineffective but sometimes even detrimental.

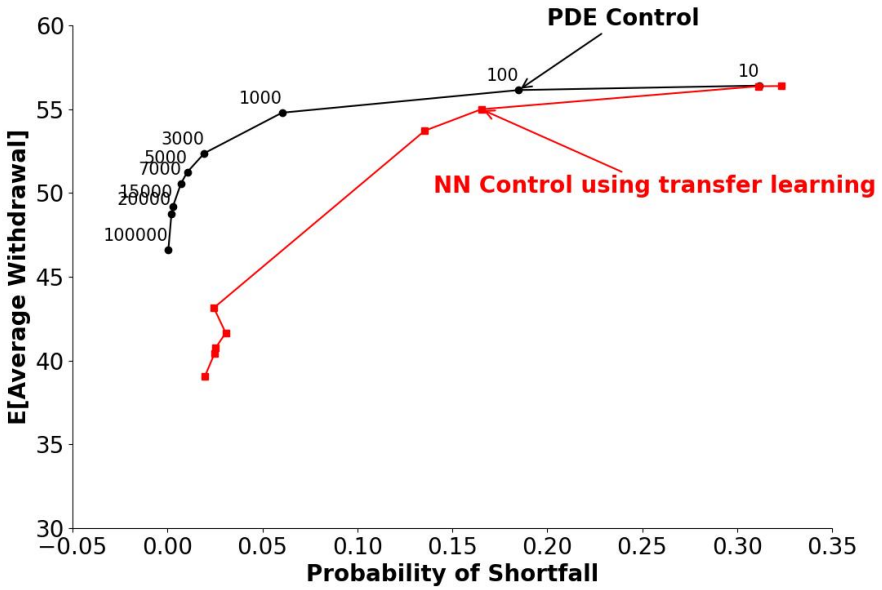


FIGURE 7.1.1: *EW-PS frontiers using transfer learning, computed from the problem (2.6.2). Note: Scenario in Table 7.0.1. Comparison of PDE solution performance. Both computed on* 2.56×10^5 *observations of synthetic data. NN used the mean value of five repeated experiments with different random seeds for every* κ *value to plot. Parameters for synthetic data based on cap-weighted real CRSP, T-bill (see Table 4.1.1).* $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. *Units: thousands of dollars. Labels on nodes indicate* κ *parameter values.*

7.1.2 Results of NN method without Transfer Learning

Considering the results of using transfer learning are too different from the PDE result, we conduct experiments without transfer learning. Table 7.1.1 represents the results of the

PDE method and Table 7.1.2 presents the results of the NN method. Figure 7.1.2 displays the EF plots for the NN and PDE results, it is clear that the curve of NN method without transfer learning is much closer to the curve of PDE method, compared to the curve of NN method using transfer learning (Figure 7.1.1). While $\kappa = 100$ is a very off point in this plot because NN framework cannot output probability value as low as PDE method.

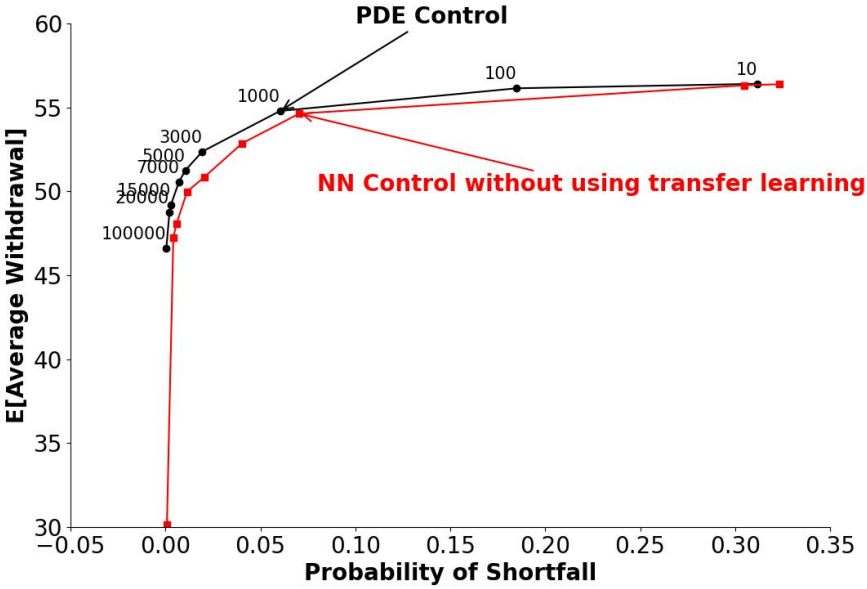


FIGURE 7.1.2: *EW-PS frontiers without using transfer learning, computed from the problem (2.6.2). Note: Scenario in Table 7.0.1. Comparison of PDE solution performance. Both computed on* 2.56×10^5 *observations of synthetic data. NN used the mean value of five repeated experiments with different random seeds for every* κ *value to plot. Parameters for synthetic data based on cap-weighted real CRSP, T-bill (see Table 4.1.1).* $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. *Units: thousands of dollars. Labels on nodes indicate* κ *parameter values.*

7.2 Further Analysis for $\kappa = 7{,}000$

When selecting an appropriate κ value from Figure 7.1.2, we observe that for $\kappa > 7{,}000$, the investor must significantly reduce withdrawals to achieve only a marginal reduction in risk. Conversely, for $\kappa < 7{,}000$, the investor takes on substantially more risk for only a slight

κ	$E[\sum_i q_i]/M$	PS
10	56.405558	0.311655
100	56.139160	0.185008
1,000	54.787898	0.060345
3,000	52.373007	0.019439
5,000	51.249984	0.010477
7,000	50.574319	0.00716
15,000	49.208872	0.002925
20,000	48.751604	0.002118
100,000	46.620733	0.000484

TABLE 7.1.1: *Synthetic market results for PDE framework optimal strategies. This table presents the detailed results used to construct PDE efficient frontier in Figure 7.1.1. Note: Scenario in Table 7.0.1. Parameters for synthetic data based on cap-weighted real CRSP, T-bill (see Table 4.1.1). $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. Units: thousands of dollars. M is the number of re-balancing dates.*

κ	$E[\sum_i q_i]/M$	PS
10	56.381581($\pm$ 0.000)	0.323484
100	56.316276($\pm$ 0.000)	0.304625
1,000	54.628306($\pm$ 0.000)	0.070436
3,000	52.864045($\pm$ 0.001)	0.040467
5,000	50.857947($\pm$ 0.003)	0.020529
7,000	49.983293($\pm$ 0.001)	0.011426
15,000	48.083602($\pm$ 0.001)	0.005870
20,000	47.259942($\pm$ 0.001)	0.004175
100,000	30.151420($\pm$ 0.000)	0.000793

TABLE 7.1.2: *Synthetic market results for NN framework without using transfer learning optimal strategies. This table presents the detailed results used to construct NN efficient frontier in Figure 7.1.2. Note: Scenario in Table 7.0.1. NN method used the mean value of five repeated experiments with different random seeds for every experiment. The bracket contents imply the 95% CI. For the PS value, its 95% CI are all very small, so we don't present them in the table. Parameters for synthetic data based on cap-weighted real CRSP, T-bill (see Table 4.1.1). $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. Units: thousands of dollars. M is the number of re-balancing dates.*

increase in withdrawals. Since both extremes represent suboptimal trade-offs, $\kappa = 7{,}000$ emerges as a desirable balance between risk and reward (picking the exact κ will be a matter of investor preference, however).

In this section, we further analyze this choice by examining the control heat map and CDF plot for $\kappa = 7{,}000$. Additionally, we evaluate whether PS serves as an effective risk measure in this context.

7.2.1 Heat Map Analysis for $\kappa = 7{,}000$

In Figure 7.2.1, we observe that both heat map plots generally follow a similar allocation strategy.

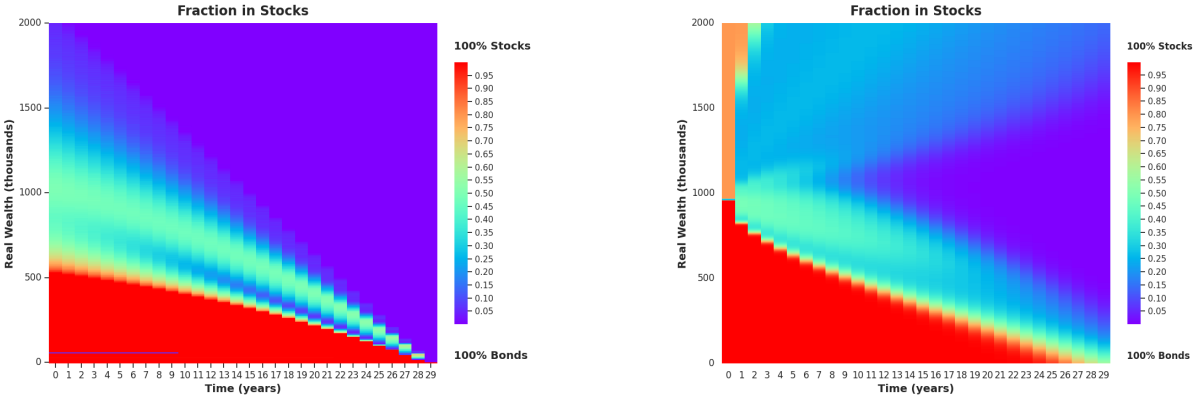
In the PDE control heat map (Figure 7.2.1(a)), we see that initially (in the early years and at low wealth levels), the allocation is heavily weighted toward stocks, suggesting an aggressive growth-oriented strategy early on. As wealth increases or as time progresses, the allocation gradually shifts toward bonds, indicating a more conservative approach focused on wealth preservation over time. This transition from high-risk to low-risk assets appears gradual, with intermediate shades indicating a balanced mix between stocks and bonds over time.

Looking at the NN control heat map (Figure 7.2.1(b)), a similar initial pattern emerges, with red at the bottom indicating a preference for stocks at lower wealth levels. The NN method allocates the portfolio to 75%-100% of the stocks during the first year (0-1 year), regardless of wealth. Over time, the NN model exhibits a more dynamic shift between stocks and bonds, with less smooth transitions compared to the PDE approach. The colors in the NN heat map transition more abruptly, suggesting distinct shifts in allocation rather than a smooth gradient.

Notably, in the mid-range of wealth (around 500 to 1,500), the NN control maintains riskier allocations for a longer period than the PDE control. For instance, while the PDE control predominantly shifts to bonds by years 20-30 (especially as wealth grows), the NN control retains a higher stock allocation until later years. Additionally, at higher wealth levels (around 1,500-2,000), the NN allocation strategy maintains a balanced mix of stocks and bonds instead of committing to a 100% allocation in a single asset.

7.2.2 CDF Plot Analysis for $\kappa = 7{,}000$

We plot the cumulative distribution function (CDF) of the NN method and the PDE method together in Figure 7.2.2. Although the cumulative probabilities for wealth in



(a) *Heat map under PDE control*

(b) *Heat map under NN control*

FIGURE 7.2.1: *Heat map of controls for* $\kappa = 7{,}000$*: fraction in stocks computed from the problem (2.6.2). Note: Scenario in Table 7.0.1. Comparison of PDE solution performance. Both computed on* 2.56×10^5 *observations of synthetic data. Parameters for synthetic data based on cap-weighted real CRSP, T-bill (see Table 4.1.1).* $q_{min} = 30$*,* $q_{max} = 60$*,* $\epsilon = -10^{-4}$*. Normalized withdrawal* $(q - q_{\min})/(q_{\max} - q_{\min})$*. Units: thousands of dollars.*

the range $[0, 300]$ are lower for the NN method compared to the PDE method, the two approaches closely align in other regions.

In Figure 7.2.2, we observe the following trends:

- In the low-wealth region ($W_T < 0$), both methods yield very small probabilities, indicating that extreme losses are unlikely.
- In the middle range ($0 < W_T < 300$), the PDE curve rises more steeply, suggesting that under the PDE approach, a larger proportion of samples cluster around this range.
- For higher wealth values ($W_T > 300$), the NN method increases more gradually, indicating a more dispersed distribution.

Overall, the PDE method exhibits a sharper transition than the NN method, implying that the PDE-based approach results in a more concentrated wealth distribution.

Comparing this to the CDF plot of the ES problem, we see that the ES curve rises steeply when wealth lies in $[0, 300]$, whereas the PS curve rises when wealth reaches 0.

An investor's trust in a risk measure depends on which aspect of risk they prioritize. The probability of shortfall (PS) is intuitive and easy to interpret, as it simply tells the investor the likelihood of wealth falling below a certain threshold W^*. Investors may initially trust PS because statements like "There is an X% probability of shortfall" are straightforward to understand.

However, PS does not account for the severity of the shortfall. This means it could underestimate risk in extreme scenarios. In contrast, expected shortfall (ES) offers a deeper assessment by not only measuring the probability of falling below W^* but also quantifying the average loss when shortfall occurs. An investor might find the statement "On average, I will lose X when my wealth falls below W^*" more informative and trustworthy than a simple probability.

There are cases where probability of shortfall is sufficient:

- If an institution is legally required to maintain a "less than X% probability of ruin", then PS is adequate. For example, retirement planning often focuses on the probability of running out of funds rather than the magnitude of the shortfall.
- Some risk-averse investors may prefer a pass/fail metric for success. If the goal is simply to stay above a certain threshold, PS provides a clear-cut measure.

For risk-averse investors seeking a comprehensive view of downside risk, ES is more informative than PS. By incorporating the severity of losses, ES offers a better safety buffer for extreme market conditions. Investors with large portfolios or institutional investors often consider both probability and magnitude of losses. If only a rough, binary measure of risk is needed, PS suffices. However, if an investor wants a fuller picture of potential downside exposure, ES is more appropriate.

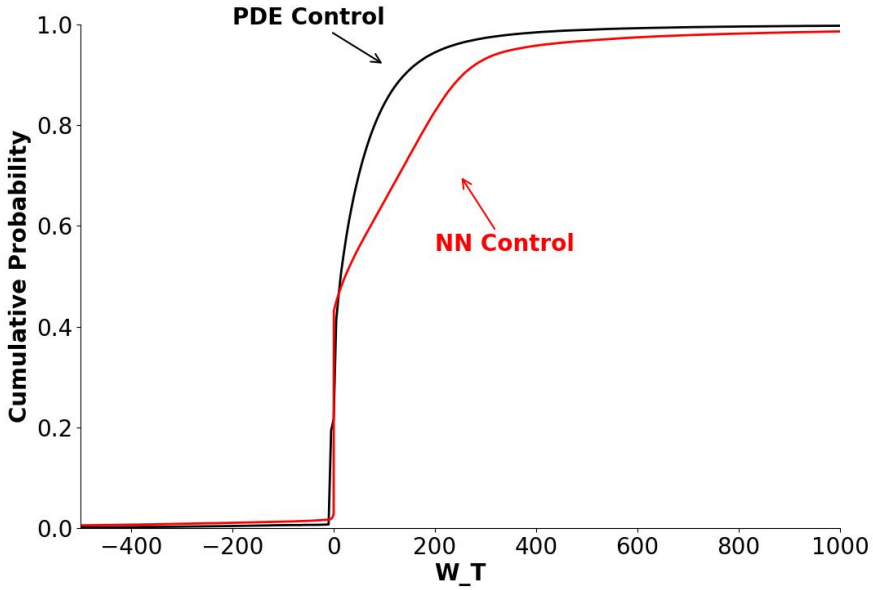


FIGURE 7.2.2: *Cumulative distribution function plot for* $\kappa = 7{,}000$*, computed from the problem (2.6.2). Note: Scenario in Table 7.0.1. Comparison of PDE solution performance. Both computed on* 2.56×10^5 *observations of synthetic data. Parameters for synthetic data based on cap-weighted real CRSP, T-bill (see Table 4.1.1).* $q_{min} = 30$*,* $q_{max} = 60$*,* $\epsilon = -10^{-4}$*. Units: thousands of dollars.*

Chapter 8

Model Robustness

A common pitfall of neural networks is overfitting to the training data. Neural networks that are overfitted do not have the ability to generalize to previously unseen data. Since future asset return paths cannot be predicted, it is important to ascertain that the computed strategy is not overfitted to the training data and can perform well on unseen return paths. In this section, we demonstrate the robustness of the NN model's generated controls.

We conduct three types of robustness tests: (i) out-of-sample testing, (ii) out-of-distribution testing, and (iii) control sensitivity to training distribution.

8.1 Out-of-sample Testing

Out-of-sample tests involve testing model performance on an unseen dataset sampled from the same distribution. In our case, this means training the NN on one set of SDE paths sampled from the parametric model, and testing on another set of paths generated using a different random seed. We present the efficient frontier generated by computed controls on this new dataset in Figure 8.1.1, which shows almost unchanged performance on the out-of-sample test set.

8.2 Out-of-distribution Testing

Out-of-distribution testing involves evaluating the performance of computed control on an entirely new dataset sampled from a different distribution. Specifically, test data are

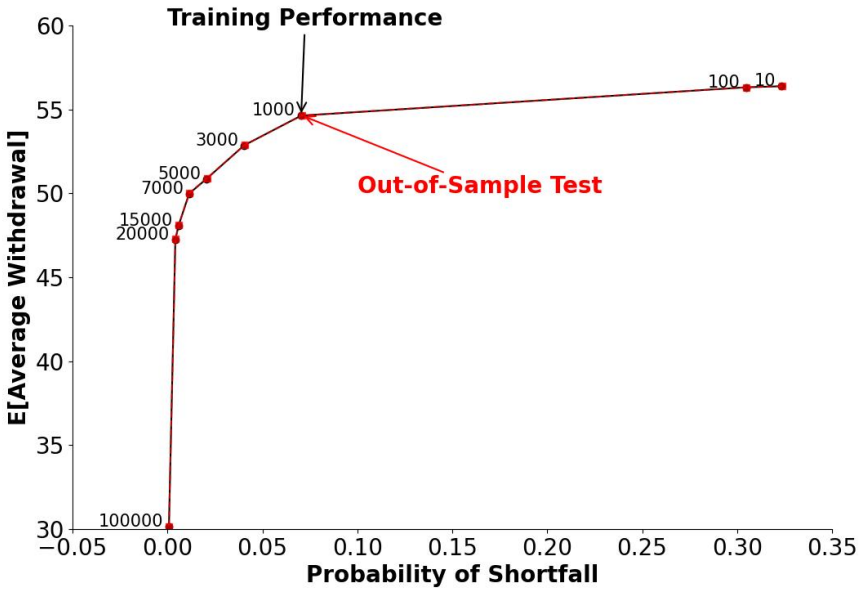


FIGURE 8.1.1: *Out-of-sample test. EW-PS frontiers, computed from the problem (2.6.2). Note: Scenario in Table 7.0.1. Comparison of NN training performance results v.s. out-of-sample test. Both training and testing data are* 2.56×10^5 *observations of synthetic data, generated with a different random seed. Parameters for synthetic data based on cap-weighted real CRSP, T-bill (see Table 4.1.1).* $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. *Units: thousands of dollars. Labels on nodes indicate* κ *parameter values.*

not generated from the parametric model used to produce training data, but are instead bootstrap resampled from historical market returns via the method described in Section 4. We vary the expected block sizes to generate multiple testing data sets of 2.56×10^5 paths.

In Figure 8.2.1, we observe that for each tested block size, the efficient frontiers remain fairly close, suggesting that the controls exhibit strong robustness.

Notably, the efficient frontier for test performance in the historical market with a block size of 12 months appears non-convex, indicating that the model trained on simulated data does not generalize well to historical test data when using a 12-month block size. Additionally, the efficient frontiers for block sizes of 1 and 3 months lie slightly above the synthetic market frontier. We hypothesize that this may be due to more pessimistic tail events in the synthetic market.

The out-of-sample and out-of-distribution tests confirm that the neural network is not

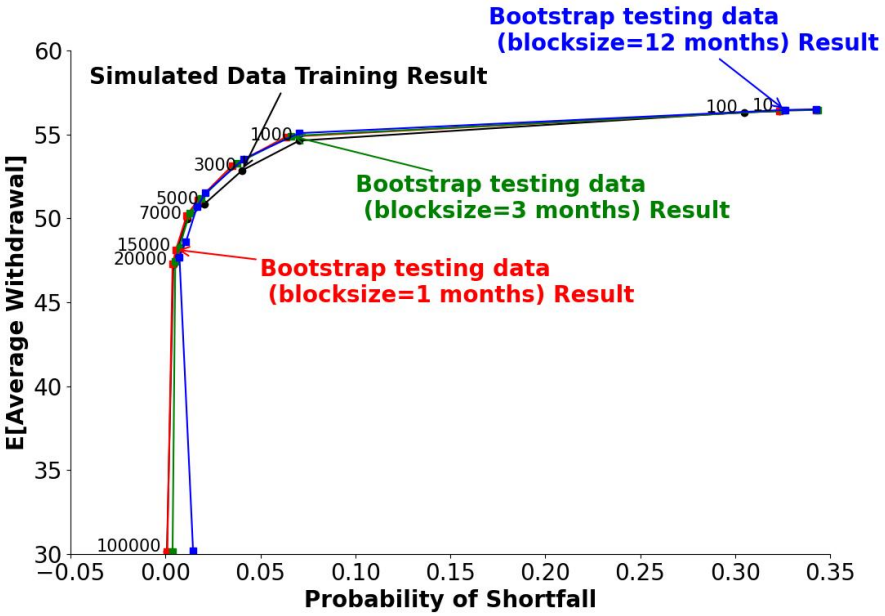


FIGURE 8.2.1: *Out-of-distribution test. EW-PS frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of synthetic data, tested on* 2.56×10^5 *observations of historical data with varying expected block sizes. Computed from the problem (2.6.2). Note: Scenario in Table 7.0.1. Parameters based on real CRSP index and T-bill (see Table 4.1.1). Historical data in range 1926:1-2022:12. Units: thousands of dollars.* $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. *Simulated training data refers to Monte Carlo simulations using the SDEs (2.2.3) and (2.2.4). Labels on nodes indicate* κ *parameter values.*

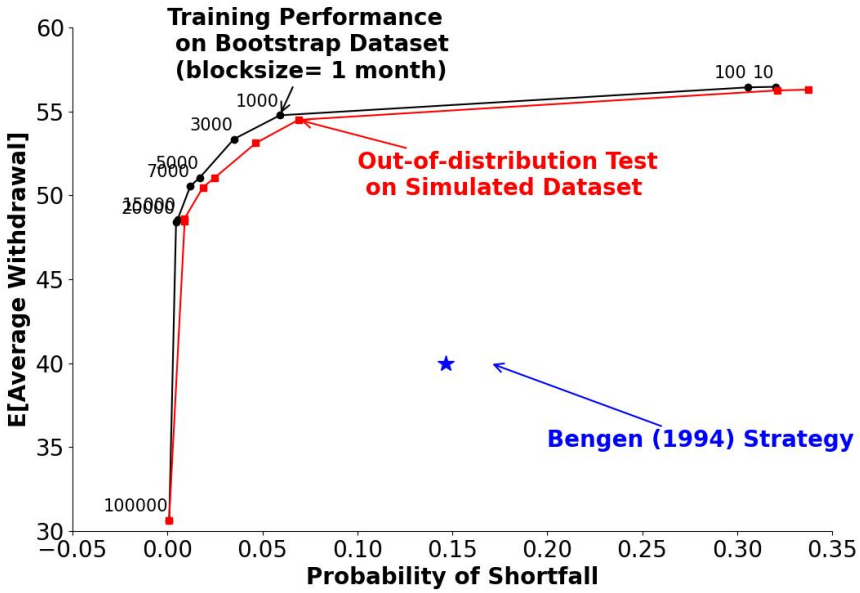


FIGURE 8.3.1: *Training on historical data. EW-PS frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of historical data with expected block sizes of 1 month, tested on* 2.56×10^5 *observations of synthetic data. Note: Scenario in Table 7.0.1. Parameters based on real CRSP index and T-bill (see Table 4.1.1). Historical data in range 1926:1-2022:12. Units: thousands of dollars.* $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. *Simulated training data refers to Monte Carlo simulations using the SDEs (2.2.3) and (2.2.4). The Bengen (1994) results are based on bootstrap resampling of the historical data. Labels on nodes indicate* κ *parameter values.*

overfitting to the training data and is successfully generating an effective strategy, at least within the block resampling framework.

8.3 Control Sensitivity to Training Distribution

To test the adaptability of the NN framework to other training data sets, we train the NN framework on historical data (with expected block sizes of 1 month, 3 months and 12 months) and then test the resulting control on synthetic data. In Figure 8.3.1, Figure 8.3.2 and Figure 8.3.3, we compare the training performance and the test performance. The EW-PS frontiers for the test results on the synthetic data are close to the results on

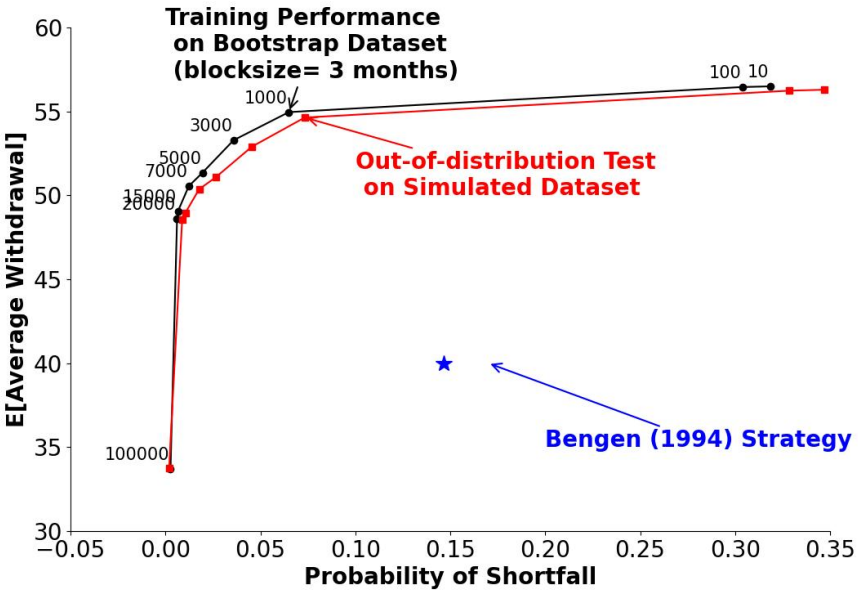


FIGURE 8.3.2: *Training on historical data. EW-PS frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of historical data with expected block sizes of 3 months, tested on* 2.56×10^5 *observations of synthetic data. Note: Scenario in Table 7.0.1. Parameters based on real CRSP index and T-bill (see Table 4.1.1). Historical data in range 1926:1-2022:12. Units: thousands of dollars.* $q_{min} = 30$, $q_{max} = 60$, $\epsilon = -10^{-4}$. *Simulated training data refers to Monte Carlo simulations using the SDEs (2.2.3) and (2.2.4). The Bengen (1994) results are based on bootstrap resampling of the historical data. Labels on nodes indicate* κ *parameter values.*

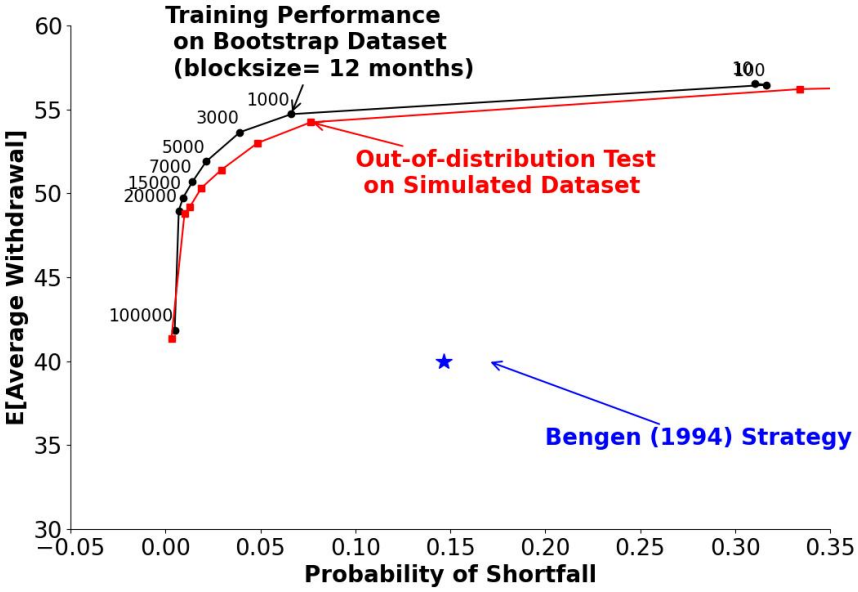


FIGURE 8.3.3: *Training on historical data. EW-PS frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of historical data with expected block sizes of 12 months, tested on* 2.56×10^5 *observations of synthetic data. Note: Scenario in Table 7.0.1. Parameters based on real CRSP index and T-bill (see Table 4.1.1). Historical data in range 1926:1-2022:12. Units: thousands of dollars.* $q_{min} = 30$, $q_{max} = 60$. $\epsilon = -10^{-4}$. *Simulated training data refers to Monte Carlo simulations using the SDEs (2.2.3) and (2.2.4). The Bengen (1994) results are based on bootstrap resampling of the historical data. Labels on nodes indicate* κ *parameter values.*

the bootstrap market data (training data set). Note that, in all cases, in the synthetic or historical market, the EW-PS control significantly outperforms the Bengen 4% Rule. [1] (Bengen, 1994). This shows the NN framework's adaptability to use alternative data sets to learn, with the added advantage of not being reliant on a parametric model, which is prone to miscalibration.

[1]The results for the Bengen strategy on the historical test data were computed with fixed 4% withdrawals and constant allocation of 60% in stocks for all cases. These were found to be the best performing constant allocations when paired with constant 4% withdrawals, in terms of PS efficiency.

Chapter 9

Conclusion

In this study, we used a neural network (NN) to determine the optimal decumulation strategy for retirees with defined contribution (DC) pension plans. Unlike traditional approaches that focus on expected shortfall, we reframe the risk of fund depletion as a probability of shortfall (PS) term. Our objective is to strike a balance between maximizing withdrawals and minimizing the probability that the retiree's terminal wealth falls below a predefined threshold.

We started by addressing the inherent challenges of the PS problem. Due to the non-differentiability of the indicator function, which introduces difficulties in calculating the derivative using Monte Carlo simulation, which hinders the NN's ability to update its weights using stochastic gradient descent (SGD). To overcome this limitation, we propose approximating the indicator function with a differentiable sigmoid function, allowing for smooth optimization in future research.

To ensure the NN's effectiveness, we conduct extensive numerical experiments to identify optimal hyper-parameters, including learning rate, batch size, number of iterations, and learning rate decay schedules. Given computational constraints, we evaluated each parameter using the mean and standard deviation of five experimental runs to ensure that the results and conclusions we obtained are valid. After tuning these parameters, we compare the NN's performance with PDE method, which served as a benchmark for accuracy and reliability.

Beyond standard validation, we analyze the control heat map and the cumulative distribution function (CDF) plot for $\kappa = 7{,}000$ to better understand the trade-off between risk and withdrawal value. We also explore the conditions under which PS can serve as a robust risk measure. Furthermore, to assess the generalizability of our approach, we test

the NN's performance on both synthetic and historical datasets, evaluating its sensitivity to different economic conditions.

Ultimately, our findings indicate that one of the main challenges in solving this problem is the non-differentiability of the indicator function, replacing the indicator function with a sigmoid function ($\rho = 100$) in the probability of shortfall term, combined with an NN trained using good parameters, produce results closely matching those of the PDE method. This highlights the NN's potential as an accurate and computationally efficient solution for optimizing decumulation strategies in DC pension plans. Compared to expected shortfall (ES), PS offers a more intuitive risk measure, as it quantifies the likelihood of wealth depletion. However, it does not capture the magnitude of the shortfall, which remains a key consideration for future research.

References

Bengen, W. (1994). Determining withdrawal rates using historical data. *Journal of Financial Planning 7*, 171–180.

Bernhardt, T. and C. Donnelly (2018). Pension decumulation strategies: A state of the art report. Technical Report, Risk Insight Lab, Heriot Watt University.

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

Calafiore, G. C. (2013). Direct data-driven portfolio optimization with guaranteed shortfall probability. *Automatica 49*(2), 370–380.

Chen, M., M. Shirazi, P. A. Forsyth, and Y. Li (2023). Machine learning and hamilton-jacobi-bellman equation for optimal decumulation: a comparison study.

Cont, R. and C. Mancini (2011). Nonparametric tests for pathwise properties of semimartingales. *Bernoulli 17*, 781–813.

Dang, D.-M. and P. A. Forsyth (2016). Better than pre-commitment mean-variance portfolio allocation strategies: a semi-self-financing Hamilton-Jacobi-Bellman equation approach. *European Journal of Operational Research 250*, 827–841.

Dichtl, H., W. Drobetz, and M. Wambach (2016a). Testing rebalancing strategies for stock-bond portfolios across different asset allocations. *Applied Economics 48*(9), 772–788.

Dichtl, H., W. Drobetz, and M. Wambach (2016b). Testing rebalancing strategies for stock-bond portfolos across different asset allocations. *Applied Economics 48*, 772–788.

Forsyth, P. A. (2022). A stochastic control approach to defined contribution plan decumulation: "the nastiest, hardest problem in finance". *North American Actuarial Journal 26*(2), 227–251.

Forsyth, P. A. and K. R. Vetzal (2019). Optimal asset allocation for retirement savings: deterministic vs. time consistent adaptive strategies. *Applied Mathematical Finance 26:1*, 1–37.

Gridin, V. N. and A. Y. Golubin (2023). Design of efficient investment portfolios with a shortfall probability as a measure of risk. *Automation and Remote Control 84*(4), 434–442.

Han, J. and W. E (2016). Deep learning approximation for stochastic control problems. *CoRR abs/1611.07422*.

Huré, C., H. Pham, A. Bachouch, and N. Langrené (2021). Deep neural networks algorithms for stochastic control problems on finite horizon: Convergence analysis. *SIAM Journal on Numerical Analysis 59*(1), 525–557.

Laurière, M., O. Pironneau, et al. (2021). Performance of a markovian neural network versus dynamic programming on a fishing control problem. *arXiv preprint arXiv:2109.06856*.

MacDonald, B.-J., B. Jones, R. J. Morrison, R. L. Brown, and M. Hardy (2013). Research and reality: A literature review on drawing down retirement financial savings. *North American Actuarial Journal 17*, 181–215.

Mancini, C. (2009). Non-parametric threshold estimation models with stochastic diffusion coefficient and jumps. *Scandinavian Journal of Statistics 36*, 270–296.

Ni, C., Y. Li, P. Forsyth, and R. Carroll (2022). Optimal asset allocation for outperforming a stochastic benchmark target. *Quantitative Finance 22*(9), 1595–1626.

Patton, A., D. Politis, and H. White (2009). Correction to: automatic block-length selection for the dependent bootstrap. *Econometric Reviews 28*, 372–375.

Pfau, W. D. (2018). An overview of retirement income planning. *Journal of Financial Counseling and Planning 29:1*, 114:120.

Politis, D. and J. Romano (1994). The stationary bootstrap. *Journal of the American Statistical Association 89*, 1303–1313.

Politis, D. and H. White (2004). Automatic block-length selection for the dependent bootstrap. *Econometric Reviews 23*, 53–70.

Ritholz, B. (2017). Tackling the 'nastiest, hardest problem in finance'. www.bloomberg.com/view/articles/2017-06-05/tackling-the-nastiest-hardest-problem-in-finance.

Tankov, P. and R. Cont (2009). *Financial Modelling with Jump Processes*. New York: Chapman and Hall/CRC.

Tsang, K. H. and H. Y. Wong (2020). Deep-learning solution to portfolio selection with serially-dependent returns. *SSRN 10.2139*.

U.S. Bureau of Labor Statistics (2024). Employee benefits survey: Latest numbers. https://www.bls.gov/ebs/latest-numbers.htm.