Long-term Life Insurance Valuations Meet Deep Learning

Author:
Mr J.M. Blomerus

Supervisors:
Prof A. Ring
Prof K. ESSEL-MENSAH

A thesis submitted to the Faculty of Natural and Agricultural Sciences, University of the Free State, in fulfilment of the requirements for the degree of

Doctor of Philosophy

February 2026

Declaration of Authorship

I, Jan Marthinus Blomerus, Student Number: 2013173550, declare that this thesis titled, 'Long-term Life Insurance Valuations Meet Deep Learning' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except such quotations, this thesis is entirely my work.
- I have acknowledged all main sources of help.
- Any unintentional mistakes made are my own.

Signed: 

Date: 16 February, 2026

"To be, or not to be, that is the question."

William Shakespeare

"Everything should be made as simple as possible, but not simpler... If you can't explain it simply, you don't understand it well enough."

Albert Einstein

"An actuary who is only an actuary is not an actuary."

Frank Mitchell Redington

Abstract

Traditional actuarial methods for valuing insurance portfolios, while established, are often time-consuming, complex, and prone to manual error. This study investigates the potential of machine learning techniques to enhance and streamline these traditional methods, offering improved efficiency and accuracy.

Utilising a dataset from a commercial European life insurance company, this research designs and implements a deep neural network to predict policy reserve values. A combination of actuarial pricing and valuation bases is employed to prepare the data for training and evaluation, focusing on developing models capable of accurately predicting expected present values.

The results demonstrate that machine learning models can effectively predict policy reserve values, providing valuations comparable to those obtained through traditional actuarial methods. Notably, the developed "Midway" model consistently predicts accurate and efficient reserve estimates. These models demonstrate an ability to capture complex relationships between inputs and policy reserve values, even with combinations of previously unseen data within valid ranges, conforming to the 99.5% accuracy required by Solvency II.

This research has significant implications for the insurance industry, offering the potential for improved efficiency, enhanced risk management, and more informed decision-making. The ability to rapidly and accurately value large policy portfolios can lead to improved pricing strategies and investment decisions. Furthermore, machine learning techniques can reduce the time and resources required for traditional valuations and provide an independent check on accuracy for auditors and regulators.

Beyond its practical applications, this study contributes to the machine learning community by demonstrating a novel combination of methods for fitting supervised regression models and establishing a framework for data preparation, model training, and validation. This work provides valuable guidance for future research in applying machine learning to diverse categories of life insurance products.

In conclusion, this study provides a compelling proof-of-concept for leveraging machine learning techniques to value a commercial book of life insurance policies, highlighting the potential for significant improvements in risk management within the insurance industry.

Acknowledgements

I express my sincere gratitude to Professors Arne Ring and Kojo Essel-Mensah for their expert guidance and support throughout this research journey.

I extend my deepest thanks to my wife, Susan, for her unwavering patience and understanding during the many years dedicated to this study. While I cannot fully compensate for the time commitment, I am committed to cherishing and strengthening our relationship in the years to come.

I acknowledge the generous support of the University of the Free State, whose commitment to excellence provided an ideal environment for this research.

I am grateful for the indirect influence of the many students at the University of the Free State, whose intellectual curiosity and dedication inspired me to pursue this research and achieve my academic goals. I wish them all continued success in their future careers.

I extend my thanks to my colleagues and friends within the actuarial profession. While data restrictions prevented direct collaboration, I hope this work contributes to our collective knowledge and facilitates future collaboration.

I acknowledge the debt I owe to the researchers who laid the groundwork for this study, providing a solid foundation for my work. I aspire that future researchers will build upon these foundations and reach even greater heights.

Finally, I wish to express my gratitude to the developers and communities behind the following open-source software, which proved essential to this research. Their dedication to freely available tools significantly eased the challenges of this project and enabled me to focus on the core research objectives:

- Flux (Innes, 2018) for building working machine learning models,
- Julia (Bezanson et al., 2017) for creating super fast high level code,
- Makie (Danisch and Krumbiegel, 2021) for creating beautiful plots,
- Latex[1] for creating a tool that is making the structure of this thesis so much easier, and
- Visual Studio Code [2] for providing the interface that brings all of this together.

[1] https://www.latex-project.org/
[2] https://code.visualstudio.com/

Contents

List of Figures

List of Tables

Abbreviations

ADAM Adaptive Moment Estimation. 35, 69, 70, 74, 79– 81, 86, 87, 176– 179

AE Autoencoder. 22, 111, 127

AF Activation Function. xiii, 21, 25, 26, 28, 29, 32, 35, 37, 38, 47, 59, 60, 66, 72, 73, 76, 79, 80, 83– 88, 112, 113, 121, 147, 180– 182, 184, 186

AI Artificial Intelligence. 14, 24, 41, 42, 89, 112

ALM Asset-Liability Matching. 18

ANN Artificial Neural Network(s). 21, 22, 81, 82, 84, 87, 112

AOS Analysis Of Surplus. xiv, 14, 15, 18, 46, 118, 119, 151

AUC Area Under Curve. 80, 81, 92

BE Best Estimate. 10, 16, 42, 57

BFGS Broyden–Fletcher–Goldfarb–Shanno. 78, 81, 179

CANN Combined Actuarial Neural Network(s). 21, 22, 24, 84, 86

CAS Casualty Actuarial Society. 84

CEIOP Committee of European Insurance and Occupational Pensions Supervisors. 54

CELU Continuously Differentiable Exponential Linear Unit. 25, 26, 38, 72, 81, 112, 113, 120, 121, 174, 186

CIR Cox-Ingersoll-Ross. 56

CNN Convolutional Neural Network(s). 21, 24, 70, 72, 87

COC Cost Of Capital. 13, 57

CPU Central Processing Unit. 63, 95, 120

DBN Deep Belief Network(s). 22, 24

DCS Discounted Cashflow System. 12

DNN Deep Neural Network(s). xiv, 5, 10, 11, 14– 16, 21– 25, 28, 30, 33, 35, 37, 38, 40, 45, 46, 57, 60, 61, 65, 66, 74, 78– 81, 83, 86, 87, 89– 92, 94, 95, 101, 110, 111, 116, 125, 127, 128, 131, 141, 146, 147, 150– 154, 156, 174, 182, 183, 186

DT Decision Tree(s). 23, 40, 127

EA Endowment Assurance. ix, xiii, 6– 8, 53, 79, 81, 115– 117, 119, 124, 129, 137– 140, 143

EIOPA European Insurance and Occupational Pensions Authority. 15– 17

ELU Exponential Linear Unit(s). 79, 80, 84, 86, 87, 113, 121, 185, 186

ETL Extract Transform Load. 12

FFNN Feed Forward Neural Network. 21, 32, 34, 76, 83

GA Genetic Algorithm(s). 75

GAN Generative Adversarial Network(s). 22, 23, 87

GELU Gaussian Error Linear Unit. 121, 184

GLM Generalised Linear Model(s). 21, 80, 82, 84, 86, 111

GPU Graphics Processing Unit. 94, 120, 146, 156, 160, 174

GRU Gated Recurring Unit. 22, 24, 127

IAA International Actuarial Association. 16, 17

IBNR Incurred But Not Reported. 82

ICAAP Internal Capital Adequacy Assessment Process. 12

ICR Individual Claims Reserving. 83, 84

IF In-Force. 10, 15, 42

IFRS International Financial Reporting Standards. 19, 56, 58, 130, 150, 151

LF Loss Function(s). 21, 24, 26, 35– 38, 59, 61, 63, 72– 74, 76, 77, 79, 82– 84, 113, 121, 147, 156, 158, 167– 169, 172, 174, 186, 191

LR Learning Rate(s). 26, 29, 36, 58, 59, 63– 65, 69– 71, 73, 74, 83, 84, 86, 87, 112, 120– 122, 147– 149, 156, 159– 166, 171– 176, 186

LRP Layer-wise Relevance Propagation. 89

LSTM Long Short-Term Memory. 22, 127

LTC Long Term Care. 81

MAE Mean Absolute Error. 25, 29, 30, 33, 40, 41, 62, 73, 74, 81, 113, 120, 121, 149, 156, 158, 169, 174, 186– 188, 190, 191

MAPE Mean Absolute Percentage Error. 83, 156

MC Monte Carlo (simulation). 44, 57, 78, 80, 87

MCEV Market Consistent Embedded Value. 56

ML Machine Learning. vii, xiv, 13, 18– 21, 23, 24, 26, 28– 30, 33, 36, 37, 41– 43, 46– 49, 57– 59, 63, 64, 76, 81, 83, 84, 90, 91, 95, 99, 107, 110, 124, 127, 130, 131, 145, 150– 152, 154, 182, 187

MLP Multilayer Perceptron. 21, 34, 64

MSE Mean Squared Error. 35, 68, 70, 76, 77, 79, 81, 113, 120, 121, 147, 167– 169, 187– 190

MSLE Mean Squared Logarithmic Error. 189, 190

NADAM Nesterov Adaptive Moment Estimation. 36, 61, 70, 72, 81, 83– 85, 112, 113, 120, 121, 160– 165, 172, 174, 177– 179

NAG Nesterov's Accelerated Gradient. 76, 81

NAS Neural Architecture Search. 75, 90

NAV Net Asset Value. 56, 57

NBS New Business Strain. 8, 99

NN Neural Network(s). 1, 20, 22– 24, 26, 28, 32, 37, 42, 44, 65, 72, 75, 78, 82– 84, 86, 87, 147

ORSA Own Risk Solvency Assessment. 57

P&C Property and Casualty. 13, 82

PB Pricing Basis. xii, xv, 102, 104– 108, 116– 119, 122, 127, 131– 133, 140

PCA Principal Component Analysis. 107– 109

PSO Particle Swarm Optimisation. 75

PV Present Value. 17, 42, 46, 52, 53, 102, 111

PVFP Present Value of Future Profit. 13

RBM Restricted Boltzmann Machine(s). 22, 24, 182

RELU Rectified Linear Unit. 26, 32, 38, 46, 66– 68, 72, 78, 81, 112, 113, 121, 156, 181– 186

RF Random Forest(s). 23, 24, 40, 80, 127

RMSE Root Mean Squared Error. 188, 189

RMSPE Root Mean Squared Percentage Error. 83

RNN Recurrent Neural Network(s). 21, 22, 87, 127

SA Sum Assured. xii, 27, 39, 44, 45, 96– 98, 101, 103, 104, 106, 109, 115, 118, 126, 127, 129

SCR Solvency Capital Requirement. 54, 55, 57, 76, 87

SELU Scaled Exponential Linear Unit(s). 185, 186

SGD Stochastic Gradient Descent. 63– 65, 74, 77, 80, 81, 84, 120, 147, 176– 178

SRELU Shifted Rectified Linear Unit. 183

SVM Support Vector Machine(s). 22, 57

TA Term Assurance. ix, xii, xiii, 6– 8, 53, 81, 115– 117, 119, 121, 123, 124, 127, 137, 139, 140, 143

TANH Hyperbolic Tangent. 32, 38, 60, 72, 78, 83– 86, 112, 113, 121, 147, 182, 183

TIF Term In Force. xii, 51, 97– 99, 106, 109, 115, 118, 129, 168

UL Unit Linked. 81

US United States. 55, 56

VA Variable Annuities. 44, 61, 78, 81

VAR Value At Risk. 57

VB Valuation Basis. 99, 105, 106, 109, 115, 118, 119, 127, 132, 140, 164

WL Whole Life. xii, xiii, 6– 8, 14, 42, 53, 56, 81, 97, 103, 109, 114– 117, 119, 123– 125, 130, 132, 133, 137, 138, 140– 142

WP With Profit. 81

XAI Explainable Artificial Intelligence. 85, 88, 89, 91, 125, 153

XGB Extreme Gradient Boosting Algorithms. 23, 40, 82, 111, 127

Dedicated to my parents and teachers for ensuring a sound foundation,
and our children for ensuring a bright future...

Chapter 1

Introduction

1.1 Why I chose this study

From an early age, I was fascinated by the logic that drives both games and software. When I was eight, I was introduced to a simple "match-taking" version of tic-tac-toe: three-by-three grids of match boxes were used to encode the game state, and the number of matches in each box indicated the desirability of that move. A winning configuration accumulated more matches, a losing one lost them. Eventually, the most promising positions were the heaviest by weight. In that way, the gameplay became an early, intuitive form of reinforcement learning, where a simple rule set-"always pick the cell with the largest weight"-reduced the search space and guided the agent to a better strategy.

The analogy with modern Machine Learning (ML) is striking. In a Neural Network(s) (NN), weights are adjusted so that correct predictions receive higher signals, and the network settles on the weight configuration that maximises overall performance. However, the analogy also underscores a vital constraint: the input-output mapping must be compatible. A network trained on a 3×3 board will generalise only to that specific dimensionality. The resulting boxes containing many matches are marked with an X in Figure 1.1:

X	O	X
O	X	O
X	O	X

FIGURE 1.1: Tick-tac-toe 3x3 solution

Attempting to apply it to a 5×5 game would leave the model unable to anticipate moves outside the original 3×3 sub-grid. Likewise, in my work, the validity of a ML model for life-insurance valuation hinges on the alignment of the training data with the real-world actuarial processes it is meant to emulate. A properly trained model for the 5x5 game would be as follows, where the size of X is an indication of the importance thereof:

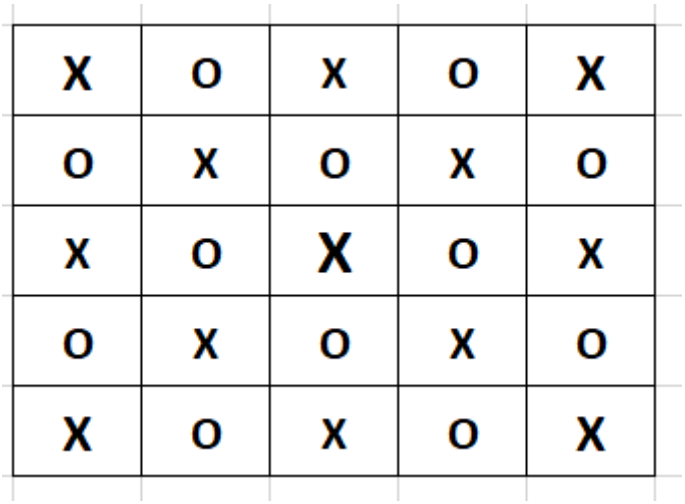

FIGURE 1.2: Tick-tac-toe 5x5 solution

In the match-box example, the model trained on a 3×3 board would exhibit peculiar behaviour when confronted with a 5×5 game. If the agent made the first move, it would invariably choose the centre cell (row 2, column 2), because that was the most heavily weighted in its training set. However, an optimal first move in a 5×5 game is the centre of the larger grid (row 3, column 3), and the four corner cells (1,1), (1,5), (5,1), and (5,5) offer even better long-term prospects. A human opponent who understands the training regime would exploit this defect: the agent would first secure the central 3×3 sub-grid, then follow with increasingly random plays-such as placing a piece in cell (1,3) instead of the strategically superior (5,5)-leading to sub-optimal outcomes.

In other words, the "match-box" system did not learn the rules of tic-tac-toe; it merely memorised a frequency distribution of moves. By contrast, a classification problem such as on the MNIST dataset (LeCun, Bottou, Bengio and Haffner, 1998) demonstrates how a network trained on 28×28 pixel images can correctly classify a 56×56 input, provided the new image is first rescaled to the network's expected input size. The challenge here is one of data conversion, not of discovering new underlying logic. The same principle applies to actuarial modelling: if we present an insurer's portfolio in the exact dimensions and feature set that our network was trained on, the model is expected to perform reliably. Deviations from that structure - new policy types, valuation basis, or tables like mortality tables - represent genuine logical extensions that the model must learn rather than merely rescale.

Several years later, I joined the actuarial profession, where my day-to-day responsibilities revolved around calculating present values for product pricing and reserving while simultaneously automating and maintaining valuation systems. Over a decade of hands-on work - cleaning and summarising data, updating actuarial bases and assumptions, executing model runs, and reviewing outputs-made the time-consuming nature of traditional actuarial pipelines all too familiar. Despite these efforts, I frequently observed situations in which months of manual preparation and calculation were required to generate a single high-level risk-adjusted result. The bottleneck stemmed largely from the need to rebuild spreadsheets, re-run deterministic calculations, and manually validate results, all while waiting for a single mostly deterministic model to finish.

This experience raised a clear question for me: could there be a more efficient route to the same high-level insights? I was convinced that a modern, data-driven approach could reduce the cycle time and increase repeatability, but practical exploration of this idea was postponed by the demands of daily actuarial work. Consequently, my research was motivated by the desire to replace the labour-intensive, manual workflow with an end-to-end deep-learning pipeline that can learn the entire valuation logic - from data pre-processing to final probability-weighted reserves - in a single, automated training loop.

My long-standing interest in applying ML to the valuation of life-insurance liabilities culminated in a decisive moment in 2017, when I first studied Goodfellow et al. (2016). The book's exposition of end-to-end learning convinced me that a fully neural architecture could, in principle, replace the manual, deterministic calculation chain that dominates actuarial practice today. Consequently, I set out to integrate deep learning with actuarial valuation workflows.

The main obstacle to this integration has been data availability. After 2016, I consulted colleagues, researchers, and industry partners in search of a usable dataset, but insurers consistently declined to share their data. Their reluctance stems from the competitive nature of life-insurance portfolios, the sensitivity of individual pricing decisions, and the perceived lack of incentive to invest resources in anonymisation or data-sharing agreements. Because such proprietary data are unavailable for research, standard ML approaches could not be directly pursued, and the field has largely relied on synthetic or highly aggregated data. This data paucity has constrained the depth and breadth of research on algorithmic valuation, underscoring the need for a novel modelling pathway that can operate effectively with limited or partially observable input.

This situation shifted dramatically at the outset of 2023 when a commercial life-insurance dataset, released in October 2022 for academic use, became publicly available. When I started this study, no peer-reviewed study had examined this data, rendering it a first-access opportunity for researchers. If the present work demonstrates how to exploit these data for accurate valuation, it may encourage insurers to release similar repositories, thereby accelerating research across the sector.

During a tenure in a corporate reserving department, I encountered a glaring audit issue: the code base contained a hard-coded adjustment of R40 million added to the reserves at inception ("Reserves at time zero = Reserves at time zero + R40 million"). This line was inserted as a temporary patch to reconcile a change in code which could not be reproduced, yet it remained embedded in the production system. The persistence of such ad-hoc fixes, buried among millions of lines of actuarial logic, underscores the fragility of manual, rule-based valuation systems. A disciplined, data-driven model-capable of learning the entire valuation process without reliance on fragile hard-coded corrections would mitigate this risk and enhance regulatory auditability.

Because actuaries routinely face the same problem: translating a finite set of observations and scenarios into predictions about future liabilities, my early curiosity about how simple heuristic strategies can evolve into powerful predictive tools has guided me toward deep learning as a means of modernising the actuarial valuation. This study, therefore, seeks to bridge the gap between the theoretical potential of neural networks and the concrete financial calculations required by insurance practice.

1.2 Problem statement, research aims and objectives

The conventional actuarial valuation workflow is time-consuming and requires a specialised team of senior actuaries. In contrast, modern deep-learning techniques offer a potential paradigm shift in terms of speed and scalability. The overarching question that guides this thesis is:

Can a data-driven neural-network framework, when supplied with a traditional actuarial basis reserve, provide accurate and regulatory-compliant life-insurance valuations?

To answer this question, the problem was decomposed into two levels of inquiry, discussed in the next two subsections.

1.2.1 Micro-level objectives

These objectives focus on the internal mechanics of the Deep Neural Network(s) (DNN):

1. Identify the most informative features within the valuation data (e.g., policy terms, demographic attributes).
2. Develop a mechanism for learning the complex inter-feature relationships that drive reserve values.
3. Choose the most appropriate hyperparameters for the DNN.

4. Design a data-grouping strategy (e.g., sampling and mini-batches) that maximises the utility of the training set for the DNN.

1.2.2 Macro-level objectives

These objectives address the practical deployment and industry impact.

1. Compile a tabular catalogue of critical features for any supervised regression model that could be applied to actuarial valuations.
2. Develop a transfer-learning framework that allows insurance organisations to build their own internal models with minimal bespoke coding.
3. Quantify the model flexibility with respect to:
    - Changes in valuation assumptions (e.g. discount rates, mortality tables),
    - temporal drift (e.g., new valuation rules), and
    - novel features in products.

1.2.3 Consolidated aim

The more robust a model can be made, the more it can be used as a standard model in the industry. What principles apply to ML models to provide alternative methods for calculating valuation reserves if a standard model can be developed? Would an insurance company and actuaries trust my 10101 methods?

My aim is to design, implement, and validate a baseline-aware DNN that can:

1. Accurately estimate reserves for life-insurance products across a broad policy space.
2. Provide per-policy values, when seen under several valuation bases, that are comprehensible to actuaries and regulators.
3. Adjust seamlessly to changes in actuarial assumptions and product inputs, enabling its adoption as an industry-standard tool.

By addressing both the micro-level (feature relevance, learning dynamics, data preparation) and macro-level (interpretability, transferability, flexibility) concerns, the thesis aims to demonstrate that deep learning can be a viable, responsible, and scalable alternative to traditional actuarial reserving methods. This dual focus ensures that the resulting model is not only technically sound but also practically deployable in real-world insurance settings. These are addressed in Chapter 5.

1.3 Life insurance products: background and context

1.3.1 Life insurance products

This research focuses primarily on non-profit Whole Life (WL) assurance contracts, with two additional product types treated as proof-of-concept cases: Endowment Assurance (EA) and Term Assurance (TA). The essential actuarial characteristics of these products are summarised in Table 1.1.

Product	Term	Benefit trigger	Typical use case
WL	Open-ended	Sum assured paid on death	Provides perpetual protection to a family or estate
EA	Fixed; up to retirement	Sum assured on death or at contract maturity (if survived)	Dual purpose: life protection and retirement-savings benefit
TA	Fixed, usually short to medium	Sum assured on death only	Cheaper, purely time-limited protection (e.g., covering a short-term loan)

TABLE 1.1: Characteristics of life insurance products

Non-profit in this context means that none of the profit made on the policies is shared with policyholders. The insurer's objective is to maximise profits while meeting the contractual guarantee (i.e., the sum assumed). The actuarial valuation of these contracts is driven mostly by mortality, investment, expense and surrender-frequency assumptions, making them well-suited to a data-driven learning framework.

The remainder of this section will elaborate on each product type's cash-flow structure, highlight the actuarial assumptions typically employed in their valuation, and explain why they constitute an appropriate testbed for a deep-learning valuation model.

1.3.2 Relative characteristics of the products

A table summarising policy attributes is provided in Table 1.2.

The ranking of premiums follows directly from the product duration: a TA contract covers a smaller time slice than a WL or EA contract and therefore requires a lower periodic premium for the same sum assured. Similarly, the likelihood of surrender increases as the benefits become less certain. Under an EA, the policyholder is guaranteed a benefit either upon death or at maturity, so voluntary termination is economically unattractive unless significant financial hardship occurs

Attribute	TA	WL	EA
Premium	Lowest (short coverage)	Intermediate	Highest (protection + savings)
Surrender propensity	Highest (short horizon and lower premiums)	Moderate	Lowest (guaranteed benefit either at death or maturity)
Reserve as % of sum assured	Lowest (predominantly expense reserve)	Intermediate	Highest (shorter payment horizon, higher mortality exposure)
Sensitivity to discount rates	Lowest	Intermediate	Highest
Underwriting intensity	Lowest (short-term, less risk)	Moderate	Highest (financial & health due to savings component)

TABLE 1.2: Policy attributes

- insurers therefore often offer a paid-up variant with a lower sum assured to accommodate this scenario. In contrast, a TA's brief coverage horizon means that, as the term nears maturity, the expected future death probability declines sharply, making surrender an appealing alternative - moreover, the underwriting intensity is substantially lower, so the market is more competitive, encouraging policyholders to shift to cheaper products.

Reserve sizing, expressed as a proportion of the sum assured, reflects these dynamics. An EA typically requires a larger reserve relative to its sum assured because the benefit can be paid sooner (on death or maturity). In a TA, mortality reserves are negligible due to the short exposure period; the bulk of the reserve is therefore allocated to cover policy expenses and commissions. Similarly, an EA's reserve demonstrates the greatest sensitivity, followed by WL and then TA.

Underwriting cost is incurred mainly at inception. Health underwriting ensures that premiums reflect the insured's risk profile, while financial underwriting validates that the sum assured aligns with the policyholder's actual protection needs and that the premium remains affordable. Underwriting intensity typically follows the order $EA > WL > TA$. The aggregate of underwriting expenses and any upfront commissions paid to distribution intermediaries is incorporated into the New Business Strain (NBS). NBS directly inflates the initial reserve and thus influences the day-zero profit figure (see Section 1.4.4).

1.3.3 Mortality risks

WL, TA and EA business is where, for a premium paid in advance, the insured will be able to receive a benefit if the insured event occurs. Premiums are paid upfront either as a single premium (rare for WL, EA and TA, and most common in annuities) or a recurring premium (monthly or annually). The insurer can make further investment profits on the net accrued premiums.

Typically, the sum of premiums is much less than the benefit promised. This creates a risk for the insurer, because if the insured event occurs before the anticipated date, the insurer will make a loss equal to the benefit paid out minus the accumulated value of net premiums received. In a portfolio of insurance products, the overall mortality risk is defined as more death benefits paid out than expected.

The larger the book of business, the more the law of large numbers can dampen deviations, assuming there is no systematic correlation between policyholder lifespans. However, correlated events - such as epidemics, natural catastrophes, or demographic shifts - can undermine diversification, creating large-scale surprises in claim frequency. A robust valuation model must therefore account not only for individual mortality probabilities but also for possible dependence structures across policyholders or under different scenarios. The Midway model investigates increases and decreases in mortality explicitly through sensitivity analysis.

1.3.4 Investment risk

The investment risk faced by a life insurance company is the potential shortfall between the realised return on the assets backing its liabilities and the assumed investment return used in valuation. In other words, the valuation investment risk is the event that the market discount rate applied to future cash flows falls below the actuarial assumed discount rate, thereby inflating the insurer's liability estimate.

Actuaries estimate the present value of future cash flows by applying a discount rate derived from an interest-rate model that reflects the yield curve of assets that are considered to back the liability profile. When a higher discount rate is assumed, the present value of future outgo - and consequently the reserve - decreases; conversely, a lower discount rate inflates the liability estimate. In practice, the discount rate is often impacted or derived from:

1. Market-based – the risk-free rate (e.g., government bonds of the insurer's domicile) adjusted for credit, liquidity and liquidity-risk premiums.
2. Regulatory or statutory - authorities typically require a solvency margin (i.e., a reduction of the market-based discount rate) to ensure that the insurer's reserves are conservative.

3. Strategic asset allocation – the insurer's own portfolio mix and investment policy, including required minimum liquidity and risk tolerance, and any mismatching.

The margin applied for regulatory or statutory purposes is therefore a deliberate discount- rate hedging technique: it reduces the assumed yield, increases the reserve, and thus mitigates the exposure to a future downturn in the yield curve. The interplay between discount rates and assumed investment returns is considered by insurers by having different valuation bases or by performing stochastic valuations. This study performs sensitivity analyses.

1.3.5 Expense risks

Expense risk refers to the discrepancy between the actual policy-related costs incurred by an insurer and those assumed in the valuation model. Two intertwined mechanisms generate this risk:

1. Expense inflation risk – the rate at which direct and indirect underwriting expenses grow relative to the assumed inflation path. For portfolios such as WL, where premiums are paid for the lifetime of the contract, any underestimation of expense inflation has a cumulative effect: the expense per policy can eventually exceed the premium, forcing the prospective reserve to rise above the sum assured and, over a prolonged horizon, resulting in a negative net premium.

2. Cost-allocation risk – the way fixed corporate and regulatory costs are apportioned to individual policies. A larger number of policies in force dilutes the per-policy fixed cost, whereas a reduction in In-Force (IF) volume concentrates these costs, thereby increasing the per-policy expense burden.

Because the insurance regulation framework in many European jurisdictions does not explicitly require the inclusion of expense inflation in reserve calculations, modelling this component is not always mandatory. Nevertheless, a comprehensive valuation model - particularly one designed to be generalisable across markets - must explicitly incorporate an optional expense-inflation factor $(1 + ei)$.

In later chapters, this parameter becomes an input that DNN models learn to handle, thereby enhancing their robustness across different market regimes. By setting expense inflation to zero, the model easily adapts to jurisdictions that do not mandate this assumption.

1.3.6 Surrender risks

In the traditional life-insurance model, surrender constitutes an optional, cash-in option for the insured. Because policyholders who are relatively healthy or who experience a large shortfall in the cost-to-benefit ratio are more likely to surrender, the surrender pool is typically adverse-selected. Consequently, the remaining IF population is on average sicker, which increases the insurer's mortality experience relative to a purely actuarial expectation.

When the insurer pays no surrender benefit, the surrender value is effectively zero. The insurer's profit from such a contract then consists of the accumulated net premiums.

From a valuation standpoint, the insurer must embed Best Estimate (BE) surrender assumptions into the present-value calculation. These assumptions determine the expected surrender timing and frequency, which in turn influence the projected loss reserve and the net-premium profit.

The valuation surrender risk is thus defined as the deviation between the actual surrender behaviour and the assumed surrender behaviour. If the actual surrenders are lower than expected, the insurer will over-reserve, leading to an understated actual profitability. Conversely, higher than expected surrenders may trigger a reserve shortfall and jeopardise solvency. Therefore, accurate modelling of surrender intensity constitutes a critical component of the DNN valuation architecture introduced later in this thesis.

1.4 Traditional actuarial valuations of life insurance business

When presented with identical data and a common problem statement, two actuaries applying the same valuation methodology will still, in practice, produce divergent results. The root cause is not the calculation algorithm itself but the sets of assumptions each actuary adopts about future mortality, experience, investment yield, and policyholder behaviour. A conservative actuary typically selects higher mortality tables, lower discount rates, and higher expense assumptions, yielding a larger reserve. Consequently, the residual sensitivity of the liability to these inputs is a major determinant of risk profile across the industry.

The heterogeneity of assumptions has prompted regulators, auditors, and capital-management boards to demand more rigorous, comparable valuation frameworks. The evolution of statutory and internal reporting standards - e.g., the transition from best-estimate to fair-value approaches, the introduction of Solvency II risk-based capital requirements, and the consolidation of data-capture protocols - has been largely driven by the need for transparency and comparability between insurers.

The quantitative principles that govern present-value estimation, reserve calculation, and profit -and-loss attribution are common to all life-insurance products. The regulatory backdrop for life

insurance is that a failure to meet contractual obligations would directly deprive policyholders of promised benefits. For this reason, insurers are required to perform statutory valuations regularly, certainly in advance of claim events, and those valuations are subject to audit by independent actuaries and external regulators. Statutory valuations are generally prepared on a more conservative basis than published valuation reports, ensuring that the institution remains solvent under adverse scenarios and that the solvency of each policyholder is maintained.

The remaining subsections of this section are divided into the sections as outlined by Table 1.3.

Subsection	**Description**
Requirements for Traditional Actuarial Reporting	Outlines the statutory and internal documentation standards that govern the preparation and disclosure of actuarial valuations.
Analysis of Surplus for Life-Insurance Companies	Examines the composition and drivers of an insurer's surplus, and how valuation choices influence capital adequacy.
The Regulatory Environment	Surveys the key legislation and supervisory regimes - such as Solvency II and Internal Capital Adequacy Assessment Process (ICAAP) - that shape valuation methodology.
Impact of Reserves on Insurance Companies	Discusses how reserve calculations affect profitability, solvency margins, and stakeholder confidence.
Actuarial Software	Reviews the main software platforms (e.g., Prophet, MoSes) and their limitations in the context of a data-centric valuation approach.

TABLE 1.3: Background considerations in actuarial valuations

These subsections collectively provide the theoretical and regulatory background against which the deep-learning framework of the thesis is introduced.

1.4.1 A deeper look into a traditional actuarial valuation

The following items all form part of a traditional actuarial valuation:

- Administrative system – Policy-administration and booking systems supply the granular policy-level inputs (premiums, lapses, benefits, etc.),
- Financial system – Consolidated financial statements and cash-flow information feed the discount-rate and expense assumptions,
- Asset register or information from asset managers – Portfolio-level asset allocations and yield curves provide the risk-adjusted investment basis,

- Extract Transform Load (ETL) or last-minute adjustments – Data-warehouse extracts, transforms, and, if necessary, manually adjusts the raw outputs (e.g., using Prophet Discounted Cashflow System (DCS)) to meet temporary regulatory or actuarial requirements,
- Data-audit sign-off – An independent audit trail (change log, version control) certifies that the inputs and any manipulations satisfy internal quality controls,
- Valuation model execution – The actuarial engine computes reserves, profits, and surplus for each product or portfolio,
- High-level analysis – The reserve-liability figures are aggregated to produce surplus statements and embedded-value metrics such as the Present Value of Future Profit (PVFP) and the Change in Cost Of Capital (COC),
- Conversion into statutory and published reports – The model outputs are reformatted to meet statutory filing requirements and to provide the company's published financial statements,
- Sign-offs – Final validation is performed by the statutory actuary, and signed off by the company's Chief Financial Officer, and external auditors, confirming compliance with the regulatory framework, and
- Backup and archiving – All raw data, models, output files, and audit logs are archived in a secure, tamper-free repository to satisfy both internal governance and regulatory audit trails.

In practice, the inputs fed into a commercial actuarial system, such as Prophet or MoSes, derive from the upstream systems above. The valuation model then performs cash flow projections to produce a liability sheet appropriate for statutory or internal reporting. As each step, there is usually some sign-off - a formal check that the inputs, manipulations, and resulting outputs comply with both internal quality-management standards and external regulatory expectations.

Albrecher et al. (2019) surveyed the current state of insurance modelling and data utilisation, concluding that the most promising directions for future work are client-centric rather than system-centric. In line with this observation, the application of large-language models has been explored in a small case study to address client-specific requirements: the model can generate personalised policy documents, facilitate online self-service portals, and recommend products that better match an individual's risk profile and financial goals. Across the life-insurance domain, the literature so far has focused largely on digitising paperwork and streamlining customer interfaces, but has offered little insight into the development of individualised valuation or underwriting models.

In contrast, the Property and Casualty (P&C) sub-sector has seen an uptick in ML approaches centred on aggregate loss data, notably claim-development triangles and loss-adjustment factors that will be discussed further in Section 2.4.2. While such models can capture overall loss trends, they typically rely on simulated data to overcome the scarcity of granular, insurer-level claim histories. This dependence on simulated inputs limits their transferability to real-world policy-level valuation tasks, where heterogeneity in underwriting, risk-exposure, and behavioural responses plays a critical role.

Thus, both the literature and industry practice reveal a gap: the need for Artificial Intelligence (AI) techniques that can produce tailored, policy-level outputs - whether via deep-learning valuation models or via generative natural-language interfaces - while remaining grounded in authentic data. The present study addresses this gap by developing an end-to-end DNN model that ingests genuine life-insurance data and outputs actuarial valuations that are accurate even on unseen bases.

For the deep-learning framework developed in this study, the same pipeline was re-imple- mented in a data-centric fashion. The model accepts structured, pre-processed datasets that have gone through the aforementioned stages, ensuring that its learning was conditioned on the same quality and consistency constraints that traditional actuarial valuations enforce.

1.4.2 Analysis of surplus for life insurance companies

At every statutory or internal valuation, an Analysis Of Surplus (AOS) is undertaken by life insurance companies. The primary objectives of an AOS are to:

1. Explain the movement in surplus between the preceding valuation and the current one.
2. Validate the reasonableness of the change through a systematic decomposition.
3. Allocate the change in surplus among the principal drivers of profit.

The three principal drivers of profit are:

- Experience changes (e.g., actual versus expected claims and investment returns),
- Basis changes (e.g., updates to mortality tables, discount rates, expense assumptions), and
- Model changes (alterations to the valuation methodology or model implementation).

When compared against AOS figures from historical valuation dates, this decomposition furnishes actuaries, regulators, and shareholders with a clearer, narrative view of the insurer's risk profile and performance dynamics.

For a WL insurance portfolio, the principal experience items include:

- Net investment gains or losses on the assets,
- Actual versus projected mortality experience,
- Actual versus projected expense trajectories,
- Actual versus projected surrender or lapse activity, and
- New business strain and other capital cost.

This study performs an AOS on traditional actuarial methods, as well as on a DNN model. Please refer to Chapters 3 and 4. The additional input and output required to perform an AOS are summarised in Table 1.4.

Item	From	Changes required
New business	Dataset	Identify from dataset New dataset for previous IF
Deaths and Withdrawals	Experience	Randomly select some policies that die or surrender during period
Interest and expenses	Asset manager Accounting department	Ignored for this study
Model changes	Audited model	A 2nd version of model was run
Basis changes	Actuary	Various small changes were investigated
Opening V Closing V	Audited model	Increased the output dimension to 11

TABLE 1.4: Changes required for AOS

By comparing the decompositions from the AOS approach, this study shows that a DNN model can reproduce and improve upon the analytical clarity traditionally afforded by actuarial valuations.

1.4.3 The regulatory environment (in Europe)

Because the data underpinning the empirical work in this thesis originates from the European market, it is convenient to frame the valuation discussion within the supervisory regime that governs life-insurance solvency and prudential reporting in that region. The principal body issuing the regulatory framework is the European Insurance and Occupational Pensions Authority

(EIOPA)[1], which promulgates the Solvency II directive[2]. Solvency II establishes a common accounting, risk-measurement, and reporting standard for insurers operating across the European Economic Area. Its core objectives are:

1. BE calculation – insurers must determine liability reserves under a scenario that optimises future cash flows whilst reflecting realistic pricing and underwriting assumptions.
2. Discount-rate determination – the market-consistent, stochastic discount rate must reflect assets backing the liabilities, adjusted for solvency margins.
3. Reporting consistency – insurers are required to disclose their risk profile, capital adequacy, and loss-and-expense drivers in a harmonised format for supervisory oversight.

The Solvency II regime, therefore, imposes stringent, yet somewhat flexible, assumptions on valuation bases. The exact set of nominal assumptions (e.g., particular mortality tables, discount-rate curves) is less crucial than the band of plausible alternative assumptions and the methodology used to calibrate them. This study demonstrates that, given the availability of a rich assumption space, a modern ML model can exploit this flexibility to produce valuations that are both compliant and tailored to each insurer's risk appetite. All subsequent analysis and model validation in this study explicitly respect the Solvency II requirements, ensuring that the proposed valuation framework can be integrated into an actual European practice setting.

The data underpinning the empirical component of this thesis pertain to the European market prior to the establishment of EIOPA in 2011. Between 2000 and 2010, the European insurance sector operated under a patchwork of national supervisory codes, with guidance from the International Actuarial Association (IAA)[3] providing a de facto standard. The IAA's Solvency I regime required insurers to present best-estimate reserves, but the specific margin added to account for prudential risk was left to the discretion of each statutory actuary.

To reconcile this historical context with a neutral, model-agnostic input space, the study adopts a generic 3% valuation interest rate - a value that reflects the low-yield, low-inflation environment prevailing in the early 2000s. By calibrating the discount-rate input in this way, the model is shielded from idiosyncratic standards while preserving the economic reality of the period.

The importance of the basis choice becomes clear in Section 3.11.2. Once the DNN is trained, the level of the discount rate (or any other valuation assumption) has a negligible impact on the model's predictive accuracy. What matters is that the assumption is consistently translated into the input features so that the network can learn the relationship between the underlying

[1] https://european-union.europa.eu/institutions-law-budget/institutions-and-bodies/institutions-and-bodies-profiles/eiopa_en
[2] https://www.bankofengland.co.uk/prudential-regulation/key-initiatives/solvency-ii
[3] https://actuaries.org/iaa

financial environment and the resulting reserve trajectory. Consequently, the study treats the historical basis as a parameter layer rather than a hard constraint, which allows the same model architecture to be transferred seamlessly to post-Solvency II contexts, without retraining.

However, since the data used in this study is from Europe and before 2010, reasonable pricing and valuation bases for 2000 to 2010 were designed. In 2010, EIOPA was not yet established. Various national actuarial organisations governed the actuarial principles in Europe, regulatory bodies, and international organisations such as the IAA[4].

1.4.4 Impact of reserves on insurance companies

The principal objective of an insurer is to maintain solvency while simultaneously optimising profitability. The profitability of a life-insurance business can be expressed as:

$$OP = P + I - E - C - \Delta V \tag{1.1}$$

where:

- P = Premiums received during the year,
- I = Investment profits from all sources (which would typically be split between investment returns belonging to policyholders and shareholders),
- E = Expenses over the year,
- C = Claims over the year, and
- ΔV = Change in reserves over the year.

Unlike the Present Value (PV) calculation that underpins reserve estimation, the profit equation employs actual year-to-year figures (premiums received, benefits paid, net investment returns). The magnitude and timing of the reserve component (ΔV) directly influence the available profits reported each year-larger reserves tie up capital, lowering cash available for investment or dividend distribution.

Many insurers, therefore, manage reserves strategically. By maintaining a comfort zone of surplus - including discretionary capital and reserves that may be released during periods of low performance - management can smooth the profit statement over time. This smoothing effect satisfies risk-averse shareholders, who typically constitute the majority of institutional investors

[4] https://actuaries.org/iaa

in insurance companies. The capacity to smooth profits is itself a function of the elasticity of reserves: a more elastic reserve system allows for larger releases in a single period, but may expose the company to higher regulatory scrutiny if releases are perceived as manipulative.

Beyond reserve management, the investment portfolio is the primary profit vector. Insurance firms engage in frequent Asset-Liability Matching (ALM), a dynamic optimisation exercise aimed at aligning the timing and magnitude of cash flows from assets with the liabilities derived from the reserve base. Accurate valuation of life-insurance liabilities as a critical input to the ALM problem (Albrecher et al., 2018). Since capital markets are highly volatile, insurers can only optimise their ALM strategy on a short-term horizon. Hence, conventional valuation tools that require weeks or months to compute full scenarios are inadequate.

Current approximations, also in the ML space, resort to summarised, aggregated data and simplified actuarial proxies (e.g., flat growth assumptions) to accelerate valuation cycles. The deep-learning framework presented in this thesis offers an alternative: by learning the mapping from policy-level features to reserve estimates in real time, the model can deliver high-fidelity valuations in the order of seconds, thereby enabling insurers to execute ALM with the granularity and speed required in a fluctuating market. Subsequent sections will demonstrate the empirical performance of this approach against conventional ALM drivers.

1.4.5 Actuarial software

The vast majority of the European life-insurance market is dominated by legacy insurers, each maintaining hundreds of product lines and millions of individual policies. The magnitude of the data-tens of thousands of cash-flow streams projected out to 50 years-requires a highly optimised, distributed computing infrastructure. Enterprise-level actuarial platforms (e.g., Prophet) therefore feature parallel processing over a computer cluster, often translated to low-level, highly efficient code (C/C++), and expose a comprehensive set of templates for group-level calculations, AOS, and embedded-value metrics.

Some of the principal vendors in the market are:

- FIS Prophet[5] – the most widely deployed solution, supporting International Financial Reporting Standards (IFRS) 17[6,7], Solvency II, and a plethora of industry-specific solutions,
- RiskAgility[8] – a cloud-based framework that emphasises regulatory compliance and transparent actuarial workflows,

[5] https://www.fisglobal.com/en

[6] https://www.insuranceerm.com/content/galleries/insuranceerm-awards-2017/best-actuarial-modelling-software.html

[7] https://www.prophet-web.com/

[8] https://www.wtwco.com/en-GB/solutions/products/riskagility-standard-formula

- R3[9] – an R-based platform that offers a modular approach with a focus on statistical analysis and data manipulation, and
- Mo.Net[10] – a .NET-based, workflow-oriented platform that provides pre-built valuation engines for Solvency II and IFRS 17.

Because packages can be heavily customised, integration of proprietary extensions (so-called "dirty" modules) can become costly and has the potential to introduce non-standardised behaviour when core modules are updated. A thorough review of these tools is given by Tucker (2018, p. 20-22), where the author discusses their respective strengths in data handling, computational speed, and regulatory alignment, while highlighting their limitations in extensibility and customising the valuation logic without disrupting standardised workflows.

In recent years, the Society of Actuaries has launched a Julia-based ecosystem[11] that includes packages for pricing, stochastic interest-rate modelling, and preliminary valuation logic. While these open-source tools are attractive for their flexibility and modern language features, the packages were still in early stages as of 2023 and lack the depth required for a full-scale, production-grade valuation routine.

Crucially, none of the mainstream actuarial platforms offer embedded ML models as part of their core product. Likewise, prior studies have not identified ML as a viable future solution within these systems. This absence is a significant gap, as the data-rich environment of modern insurers and the regulatory push for faster, more transparent valuations create a strong case for integrating data-driven methods - exactly the niche that this study addresses.

1.4.6 Deterministic and stochastic Valuations

Prior to the year 2000, deterministic methods were predominantly used for pricing and valuation within the life insurance industry. However, advancements in stochastic modelling and increases in computational power have led to a widespread adoption of stochastic methods for these calculations.

Deterministic methods rely on single-point estimates and fixed assumptions, while stochastic methods incorporate probability distributions and simulate a range of possible outcomes. This allows for a more comprehensive assessment of risk and uncertainty.

While stochastic valuation methods offer potential advantages, they are beyond the scope of this study. Future research will explore the application of these techniques to long-term insurance valuation.

[9] https://www.rnaanalytics.com/r3s-software-suite
[10] https://www.softwarealliance.net/
[11] https://juliaactuary.org/

1.5 Definitions of Machine Learning for actuarial life insurance valuations

1.5.1 Introduction

The terminology that underpins both actuarial practice and ML can diverge substantially, even when the same word is used. In this section, a few of the key terms that recur throughout this thesis are defined, explicitly indicating the actuarial versus ML conventions. These definitions will serve as a common reference for the chapters that follow. When a particular interpretation is required, the chapter will either refer back to this section or provide a specific footnote.

At first read, this section might seem like a duplicate of Section 2.3. It is not. This section discuss the ML methods in actuarial context, while Section 2.3 discuss ML methods from a computer science viewpoint.

The thesis targets actuarial accuracy: a model is acceptable only if it can be declared accurate by actuaries, regulators, and senior management. By establishing a common lexicon, we avoid "semantic drift" when presenting results to regulators, audit committees, or actuaries who might otherwise misinterpret an ML-specific phrase. In order for a model to be accurate, it needs to be accurate on every single input policyholder data, for all possible ranges of input valuation bases.

An excellent traditional example of jargon used with different meanings in different environments is "Discount Rate". The definition differs between accountants and actuaries. In formula terms, the discount rate for the accountant uses the symbol d, where (1-d) is used to discount cash flows paid in advance. Stated differently, for an accountant, a **discount rate d** implies a v of:

$$v = 1 - d \tag{1.2}$$

The actuary will speak of a **discount rate "v", calculated at interest rate i**. For an actuary, the discount rate (at) i is typically given the symbol v where

$$v = (1 + i)^{-1} = 1 - d \tag{1.3}$$

While previous authors have explored the integration of ML into insurance valuation, the fusion of actuarial intuition with NN infrastructure has not been addressed to the degree undertaken in

this study. Unlike a purely statistical study, this thesis interprets the output of the NN through an actuarial lens: it asks how a management committee would view the reserve produced by a hidden-layer representation? It also checks whether regulators could satisfy their solvency-oriented regimes with the model's output. Thus, the definitions are not merely pedagogical; they are pivotal to bridging the two disciplines in a meaningful way.

1.5.2 Definitions of Neural Networks

This subsection presents the ML families most frequently encountered in contemporary research, together with a concise actuarial interpretation adopted in this thesis. The notation is consistent with future sections and chapters.

A Generalised Linear Model(s) (GLM) (Nelder and Wedderburn, 1972) maps inputs to outputs through a single set of weights. It is structurally equivalent to an Artificial Neural Network(s) (ANN) lacking hidden layers or Activation Function (AF)s, or may be considered the final layer of a DNN as proposed in the Combined Actuarial Neural Network(s) (CANN) architecture.

ANNs (Rumelhart et al., 1986), also known as Feed Forward Neural Network (FFNN)s or Multilayer Perceptron (MLP)s, constitute a subset of ML and form the core of deep learning algorithms. Their name and structure are inspired by the biological neuron, mimicking the signalling pathways within the human brain. An ANN typically consists of one or two hidden layers and is referred to as an FFNN because activations propagate sequentially from one layer to the next during the calculation of the Loss Function(s) (LF). This is discussed in Section 1.5.5.

A DNN (Hinton and Salakhutdinov, 2006) is an ANN with more than two layers between the input and output layers that allows it to learn the features that optimally represent the given training data. The layers between the input and output layers are called hidden layers. The **depth** of the network is the number of layers, and the **width** of a layer is the total number of neurons in that layer.

A Convolutional Neural Network(s) (CNN) (LeCun, Bottou, Bengio and Haffner, 1998) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

A Recurrent Neural Network(s) (RNN) (Elman, 1990) is a class of ANN where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. This allows it to exhibit temporal dynamic behaviour. It is useful for analysing data of a sequential nature, for example, time series like stock prices or interest rates.

Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a variety of RNNs that are capable of learning long-term dependencies, especially in sequence prediction problems like speech training or stock market price predictions.

A Gated Recurring Unit (GRU) (Cho et al., 2014) is part of a specific model of RNN that intends to use connections through a sequence of nodes to perform ML tasks associated with memory and clustering, for instance, in speech recognition.

A Restricted Boltzmann Machine(s) (RBM) (Smolensky, 1986) is a generative stochastic ANN that can learn a probability distribution over its set of inputs. RBMs have found applications in dimensionality reduction, classification, collaborative filtering, feature learning, topic modelling, and even many-body quantum mechanics. They can be trained in either supervised or unsupervised ways, depending on the task.

Deep Belief Network(s) (DBN)s (Hinton et al., 2006) are formed by "stacking" RBMs and optionally fine-tuning the resulting deep network with gradient descent and backpropagation.

Support Vector Machine(s) (SVM)s (Cortes and Vapnik, 1995) are used heavily in the financial sector, as they offer high accuracy on both current and future data sets. The algorithms can be used to compare relative financial performance, value, and investment gains virtually. SVMs have been used in text, hypertext, and image classification. SVMs can work with handwritten characters, and the algorithms have been used in biology labs to perform tasks like sorting proteins. Supervised and unsupervised learning systems are used in chatbots, self-driving cars, facial recognition programs, expert systems, and robots, among other things.

A CANN is a term used for a regression model that fits the input to the output via a DNN as well as via skip-connections. Please see Gustafsson and Hansén (2021) for a review.

An Autoencoder (AE) (Hinton and Salakhutdinov, 2006) is a type of dual NN used to learn efficient codings of unlabeled data (unsupervised learning). The encoding is validated and refined by attempting to regenerate the input from the encoding. AEs can be used as a feature extractor for classification or regression tasks. AEs take unlabeled data and learn efficient codings about the structure of the data that can be used for supervised learning tasks.

Generative Adversarial Network(s) (GAN)s (Goodfellow et al., 2014) are deep generative models that have gained popularity. GANs have the ability to imitate data in order to model and predict. They work by essentially pitting two models against each other in a competition to develop the best solution to a problem. One neural network, a generator, creates new data while another, the discriminator, works to improve on the generator's data. After many iterations of this, data sets become more and more lifelike and realistic. Popular media uses GANs to manipulate images, video, and audio, creating what's commonly known as deepfakes. GANs

are also impactful for creating large data sets using limited training points, optimising models, and improving manufacturing processes.

A Decision Tree(s) (DT) (Quinlan, 1986) is a supervised learning technique that can be used for both classification and regression problems, but it is mostly preferred for solving classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome. It is the building block for Random Forest(s) (RF)s.

A RF (Breiman, 2001) is one of the most popular and commonly used algorithms by data scientists. RF is a supervised ML algorithm that is used widely in classification and regression problems. It builds DTs on different samples and takes their majority vote for classification and average in the case of regression.

Extreme Gradient Boosting Algorithms (XGB) is a highly popular subset of tree-ensemble models and is an algorithm in which new models are created from previous models' residuals and then combined to make the final prediction (Chen and Guestrin, 2016). It is highly recommended as it performs on par and in most cases better than DNNs (Katzir et al., 2021; Shwartz-Ziv and Armon, 2022). The solution space also includes some rules that were derived during training, which assist in explaining how the model works. The disadvantages of XGB are that almost the entire dataset has to be stored in memory, which may present us and insurance companies with a problem, given the large input datasets. This promising area is left for future research.

The TabNet NN architecture (Arik and Pfister, 2021) offers a compelling combination of high-performance prediction and transparent feature usage for tabular data, a problem setting that is central to actuarial science. TabNet implements a sequential attention mechanism that selects a small, instance-wise subset of the most informative features at each decision step. The resulting attention masks provide a natural local feature-importance signal (how the model uses the features for a given policy) while aggregating across all decision steps yields a global importance profile that aligns with the actuarial understanding of which underwriting or demographic factors dominate the reserve calculation.

Classification problems are found where the output is a specific and known category. Examples include "Yes" or "No" and "Dog", "Cat", ..., "Elephant". It is the most popular field of study. The most popular NNs used for classification problems are CNNs, GRUs, RBMs, RFs, and DBNs.

Binary classification is the simplest kind of ML problem. The goal of binary classification is to categorise data points into one of two buckets: 0 or 1, true or false, to survive or not to survive. It is therefore a subset of classification problems.

The core innovation of this thesis is a DNN where the input vector x explicitly contains the policy data as well as the actuarial basis reserve (derived from the chosen mortality tables, discount curves, and other assumptions). In contrast to a residual-style CANN, no skip-connections are employed. Instead, policyholder and valuation basis inputs are concatenated. This approach has two advantages:

- Regulatory transparency – The baseline reserve is part of the observable input, so any deviation produced by the network can be immediately traced back to the actuarial assumptions, and
- Training efficiency – By feeding the baseline directly, the network can focus its representational capacity on learning residuals or corrections rather than having to learn a constant offset.

Regression problems occur where the output to be predicted is a continuous numerical value.

This study uses DNN to provide a solution for the regression problem of reserving in life insurance. Other regression approaches, like RFs, are left for future research.

1.5.3 Training of Machine Learning Models

ML is a subset of AI. Training an ML model entails the process of teaching the ML model the underlying function it is trying to estimate. Training ML models can be broken down further into supervised, unsupervised, semi-supervised, and reinforcement methods.

Supervised learning uses labeled datasets to train algorithms to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its model weights until the model has been fitted to the desired accuracy, by adjusting the weights recursively to minimise a specified LF.

Unsupervised learning uses unlabeled data. These algorithms discover hidden patterns in data without the need for human intervention and, in this way, try to make sense of the data.

Semi-supervised learning occurs when only part of the given input data has been labeled.

Reinforcement learning interacts with an environment (state, action, reward) to learn a policy to maximise cumulative reward.

The present study adopts a supervised strategy since both the policyholder details and basis items are provided as input x. The training pipeline, therefore, comprises:

- Data preparation – construction of policy-level feature and valuation basis vectors, including the basis reserve, and partition into training, validation, and test sets,
- Self-supervised pre-training – masked-reconstruction of input columns to learn robust embeddings,
- Supervised fine-tuning – minimising Mean Absolute Error (MAE) between the model output and the reference reserves, on an individual policy level, and
- Evaluation – measuring reserve error (actual vs expected).

1.5.4 Layer architecture

The DNN employed in this thesis comprises a compact sequence of fully-connected (dense) layers. The choice of a dense topology is motivated by the tabular nature of life-insurance data: each policy is represented by a fixed set of engineered features (demographic, underwriting, policy-level, and the valuation basis). Unlike image or text networks, there is no obvious spatial or sequential structure that would warrant convolutional or recurrent layers. Instead, a simple stack of dense units suffices to capture the non-linear interactions.

Empirical studies (Glorot and Bengio, 2010; Krizhevsky et al., 2012) have shown that DNNs can be trained effectively if the weight initialisation and gradient-flow control are carefully managed. In this study, initially all weights were initialised with the Xavier (Glorot) scheme (Glorot and Bengio, 2010) to keep the variance of AFs constant across layers. The network depth of the final DNN trained is restricted to four hidden layers (the number of layers chosen to balance bias and variance). Each layer contains 72, 36, 18, and 9 units, respectively, thereby gradually reducing dimensionality while preserving expressive capacity. The final layer is a linear unit that outputs the predicted actuarial reserve value per policy.

Continuously Differentiable Exponential Linear Unit (CELU) activations are applied at each dense layer to introduce non-linearity while preventing vanishing gradients. To further stabilise training, Batch Normalisation (Ioffe and Szegedy, 2015) is incorporated after every affine transformation. The combination of Rectified Linear Unit (RELU) (CELU-like) and Batch Normalisation has been empirically shown to accelerate convergence on tabular data.

1.5.5 Updating weights and bias

Every connection in a NN is assigned a weight, and each neuron in every layer has a bias value of that layer. The sum of each prior connection, plus the bias, of each neuron, is used in the calculation of the activation value of the neuron, and that value is passed on through multiple connections separately to each neuron in the next layer. The network updates weights and bias

through a two-step process, the first step being called feed-forward and the second step being called backpropagation.

During the feed-forward process, the network calculates the current loss from the first layer to the last layer. It calculates the activation (value) of each neuron by applying the current weight from each prior connection and the bias for the neuron. When the sum of activations from the second last layer reaches the output layer, the LF is used to measure how well the actual versus expected values compare, to give a score to the current set of weights and biases.

During the next phase, the backpropagation phase, the network performs partial differentiation of the LF with regard to the weights and biases, to find the direction of fastest change. This is performed consecutively from the last layer to the first layer. Where the partial derivatives are showing the most change per layer, the weights and bias for the applicable neurons are then updated by subtracting a proportion (called the Learning Rate(s) (LR)) of the partial derivative with regard to the LF from the previous weight and bias for that neuron. Once this is completed, a new loss is calculated with the new weights and biases through the next feed-forward phase.

In summary, a LF works on the output layer, which is the predicted values, and then iteratively steps backward along the negative gradient of the LF and AFs to update the weights and biases along the routes that lead to the minimisation of the LF. A single update of all the weights and biases (of all the batches) is called an epoch.

1.5.6 Datasets

There are three datasets in ML called:

1. the Training dataset,
2. the Validation dataset, and
3. the Test dataset.

The training dataset would typically constitute the bulk of the data (since training is the most important step) and constitute 50% to 80% of the data.

The validation set is used during training to test how well the model predicts while it is training. Ideally, the training loss and the validation loss should be of similar order, that is, if the validation set is a good representative of the training set data. The validation dataset is typically the smallest dataset of the three.

The test data is data that the model has never seen, and is used for the predictions by the final model. Typically, this prediction accuracy will be less than the training and validation accuracies

obtained during training, as there is no requirement for the testing data to be representative of the data used in training. It is this accuracy that depicts how well the model generalises.

In most studies to date, the researchers would take an input dataset and immediately start splitting it into training, validation, and testing datasets. Sometimes they will have spectacular results, but most of the time, the individual predicted values are far from the true values. Accuracy far from the truth would not entertain actuaries, auditors, shareholders, or regulators. When presented with new data with regions outside the regions trained on, models trained solely on the previous input data could fall over. Such models need retraining.

The final methodology employed deliberately avoids this pitfall. Drawing on my experience in pricing and reserving, a policy-attribute space is defined that captures the valid ranges for age, policy term, and Sum Assured (SA), as mandated by underwriting guidelines and typical valuation bases. Policy attributes are then grid-sampled uniformly across this space to generate a synthetic reference dataset that contains no direct reference to actual contracts. This reference set is used as the training material; it preserves the statistical properties of a typical European life-insurance book while preventing data leakage.

Finally, the policy input file - the raw, real-world portfolio is kept strictly for the testing phase. The best-found model (identified on the synthetic training set) is applied directly to this file, and its performance is recorded. By separating the synthetic training phase from the evaluation on authentic data, the reported accuracy truly reflects the model's ability to generalise to unseen business and valuation scenarios.

1.5.7 Normalisation of data

Machines interpret numerical values in binary, so the scale of each feature must be carefully controlled to accelerate convergence and to avoid numerical instabilities. Consequently, normalisation is performed individually on every continuous input feature, before feeding data to the DNN.

One method of normalisation is called Linear Normalisation, or MINIMAX, transforming the continuous input variables into the range [0,1] using the formula:

$$x_j = \frac{x_i - x_{MIN}}{x_{MAX} - x_{MIN}} \tag{1.4}$$

as the new input data point for x_j.

Another version is to take twice the formula and subtract 1, to obtain input variables in the [-1, 1] range, which should in theory, perform better as the data will be twice as far from each other.

Gaussian normalisation is transforming most of the variables into the typical range [-1, 1] (entire dataset range likely [-3.2, 3.2]) by applying the formula:

$$x_j = \frac{x_i - \mu}{\sigma} \tag{1.5}$$

where μ and σ are the mean and standard deviation of the (training) dataset.

The best method depends on the structure of the underlying data and what is passed into the model for training. It will become clear later on that the AF used in training the NN must be suitable for the normalisation method used; it must be a function that activates neurons over the same range as the normalisation range. Normalisation is not only important when comparing measurements that have different units or scales, but it is also a general requirement for many ML algorithms (Kevin and Kang, 2017).

Linear normalisation is preferred when the data is approximately uniformly distributed over the minimum and maximum ranges. Although the precise selection of a normalisation scheme can be data-driven, Gaussian normalisation was preferred for this study because the majority of continuous inputs, based on exploratory analysis, revealed a bell-shaped distribution around the mean. For the few highly skewed variables, the log-transform should be preferable. These transformations were computed solely on the training set. The same parameters (mean and standard deviation) were subsequently applied to the validation and test sets to preserve the statistical integrity of the data split.

Normalisation is therefore not a mere preprocessing convenience; it is a necessary prerequisite for stable and efficient learning in deep networks, and it directly influences the effective learning dynamics of each neuron. The impact of different normalisation schemes on the convergence trajectory is documented in Appendix B, confirming that the chosen strategy yields the lowest validation error across all hyper-parameter trials (Kevin and Kang, 2017).

1.5.8 Overfitting

In actuarial terminology, spurious accuracy - an implausibly low error on the calibration set that fails to hold up on unseen contracts - is synonymous with the ML concept of overfitting. Since this study focuses on the sum of absolute errors, or equivalently, the MAE per policy, it was monitored across training, validation, and test datasets to gauge generalisation.

During preliminary experiments, explicit regularisation techniques (e.g., dropout, L2 weight decay) were unnecessary. The key factors that mitigated overfitting were:

1. Careful network sizing – a modest architecture (four densely-connected layers with progressively halved widths: 72-36-18-9 units). An unnecessary increase in depth or width consistently led to a widening MAE gap, indicating memorisation of idiosyncratic policy features.

2. Early-stopping on validation MAE – Training was halted once the validation MAE plateaued for 20 consecutive epochs, preventing the model from over-fitting subsequent noise in the training data.

3. Robust data partitioning – The validation and test sets were sampled to preserve the statistical distribution of key underwriting attributes (e.g., age and policy term). By ensuring that each split truly represented the same underlying processes, the risk that the model would over-fit to idiosyncratic patterns present only in the training set was reduced.

4. Iterative model selection – a cohort of networks with varying hyperparameters (LRs, batch sizes, AFs) were trained. The final model was chosen based on the lowest validation MAE. Other models were discarded even if they produced a slightly lower training MAE, underscoring that optimisation focused on generalisation rather than training fit.

Because the DNN was trained on a synthetically generated reference set that already embodied the statistical regularities of a real European life insurance book, the risk of learning artefacts from the data was inherently limited. Consequently, the validation MAE and test MAE were similar, indicating a well-generalised model without the need for dropout, weight decay, or other explicit regularisers.

1.5.9 Bias

Bias in a ML system refers to systematic deviations that can cause the predictive model to deliver inaccurate or even unfair outcomes. In the context of actuarial valuations, bias may arise at several stages of the data-science pipeline: data collection, preprocessing, model design, or interpretation. The most common forms that pertain to life-insurance portfolios are given in Table 1.5.

By constructing a synthetic training portfolio that systematically covers the full spectrum of policy and valuation basis attributes, the impact on bias was mitigated.

According to Wüthrich (2020), a large bias manifests when a model exhibits a systematic failure to capture the underlying data distribution. The diagnostic signature is a high training error

Source	Typical manifestation in insurance	Impact on reserve estimation
Algorithmic bias	Numerical issues (e.g., under-flow in very small probabilities) or design constraints that restrict the model to a subset of the feature space.	Can produce under- or over-forecasted reserves for specific policy classes.
Sample (selection) bias	Training data that covers only a restricted subset of the insured population (e.g., only contracts issued to a single distribution channel).	Generates reserves that are not representative of the insurer's overall book.
Prejudice bias	Inclusion of implicit societal stereotypes (e.g., product usage patterns that differ systematically across age or gender).	Leverages spurious associations and jeopardises regulatory compliance (e.g., non-discriminatory underwriting).
Measurement bias	Systematic errors in recorded data (e.g., rounding of premium amounts, inconsistent coding of claim types).	Propagates into the estimated reserve surface, yielding systematic offsets.
Exclusion bias	Omission of a relevant factor (e.g., geographic region, policyholder occupation) from the input set.	Degrades the model's ability to adjust for local mortality or lapse patterns.
Recall bias	Estimates (e.g., underwriting grades) sourced via memory rather than objective measurement.	Inflates uncertainty in the learned mapping.
Observer bias	Variability in expert labels during data annotation (e.g., classification of benefit clauses).	Introduces inconsistency in the target variable (reserve benchmark).
Representation bias	Under-representation of minority policyholder groups in the training set.	Generates less reliable reserves for those groups, raising solvency-based requirements.

TABLE 1.5: Bias in insurance data

that is comparable to the test error, indicating that the model is under-fitting rather than over-fitting. In my experience with the reserve-prediction model, this situation arises when the network is too shallow or when the dimensionality of the input has been deliberately curtailed.

To reduce bias, the following measures were adopted:

1. Enlarge model capacity – Increasing the number of units per layer or adding additional hidden layers allows the network to represent more complex mappings.
2. Removed regularisation – Reducing dropout probability or the L2 penalty encourages the model to retain more information from the inputs.
3. Experiment with alternative architectures.
4. Enrich the feature set – Adding scenarios (more valuation bases) reduces the burden on the network to learn these relationships implicitly.

Cognitive biases that may infiltrate data-labelling or feature-engineering practices include stereotyping, bandwagon effect, priming, selective perception, and confirmation bias. These biases can produce label errors or systematic feature distortions that, once encoded into the training set, lead the network to learn spurious associations. Consequently, a robust modelling pipeline must include safeguards against such distortions, among them the explicit maintenance of a non-zero bias term in every affine transformation.

The following conditions motivate the inclusion of bias vectors in an FFNN:

1. Residual systematic error – When an ostensibly unbiased network consistently under- or over-predicts reserve amounts, the bias term can absorb the systematic shift.
2. Non-zero output on zero input – If the target reserve contains a constant component (e.g., a mandatory minimum), the NN must be capable of reproducing that component even when all input features are set to zero.
3. Decision surface outside the linear subspace – When the decision manifold is not confined to a linear subspace of the input space, the affine shift provided by the bias enables the network to shift the hyperplanes that separate different regions of the output space.

The necessity of biases is further elucidated by considering the role of AFs. For odd functions (e.g., the Hyperbolic Tangent (TANH) or scaled RELU), an affine shift can be absorbed by the symmetry of the activation, rendering the bias term redundant if the target function resides entirely within the subspace generated by the linear transformation. Conversely, when the AF is non-odd, the bias term cannot be discarded without altering the shape of the decision surface; it directly affects the gating of the activation and, consequently, the output magnitude. Therefore, if employing RELU, bias matrices should be retained, and could be dropped if non-RELU functions are used.

While bias parameters are mathematically necessary, they also serve as a potential conduit for hidden biases if not carefully regularised. Unconstrained bias terms can absorb systematic errors

introduced by data imbalances or measurement inconsistencies, thereby masking underlying issues. To mitigate this risk, the biases were initialised to zero and the optimisation process was allowed to adjust the bias gradually, while monitoring their magnitude during training. If a bias term grows excessively, it signals the presence of a systematic anomaly in the data or in the target benchmark, and training should be stopped.

Unchecked biases in a valuation model can lead to sub-optimal solutions, misleading solvency capital estimates, and, ultimately, to regulatory non-compliance. Moreover, the model's outputs must be trustworthy at each model point, thereby necessitating a controlled bias structure. The subsequent chapters detail how these biases are monitored and constrained to ensure that the network remains faithful to actuarial principles while benefiting from data-driven learning.

By combining these diagnostic indicators with the mitigation strategies above, the DNNs trained remained both accurate (low bias) and generalisable (low variance) across the diverse policy classes encountered in the European life insurance portfolio.

1.5.10 Variance

Beyond bias, the stability of a ML model is measured by its variance - the degree to which small perturbations in the training data induce large changes in the learned mapping. In actuarial practice, variance manifests as a model that achieves low error on its training portfolio but exhibits substantially higher error on unseen test data, a scenario that can jeopardise regulatory capital adequacy and confidence.

A high-variance model is often the consequence of:

- Over-parameterised networks, whereby the model has sufficient capacity to fit both the underlying signal and the idiosyncratic noise present in the training set,
- Noisy data-measurement errors, imprecise coding of claim severity, or inconsistent premium-recording introduce fluctuation that a low-variance model would ideally ignore, and
- Insufficient training volume relative to model complexity, preventing the network from observing a representative sample of the full policy-space distribution.

The goal in model construction is to position the model on the bias–variance continuum such that the total expected loss is minimised. In practical terms, this entails:

- Controlling model capacity (e.g., choosing an appropriate network depth and width) to curb excessive variance while retaining sufficient expressive power,

- Sufficient data coverage-by sampling the synthetic reference portfolio uniformly across the valid policy-attribute space, and
- Early-stopping and validation monitoring - by halting training when the validation MAE ceases to improve, the network is prevented from fitting to stochastic fluctuations.

Although bias and variance are conceptually distinct, they are intricately linked. A model that is heavily biased (i.e. under-fit) may exhibit low variance because it ignores much of the data variability; conversely, a model that is low-bias (i.e. over-fit) typically suffers from high variance. This study minimises both.

1.5.11 Treatment of categorical data

Categorical predictors (e.g., gender and product) require special treatment because conventional normalisation techniques assume a continuous scale, whereas categorical values are intrinsically discrete. It is hard for the model to deal with categorical variables as these are not on the same scale as numerical data, even after normalisation. The categorical variables used in this study are discussed in Section 3.4.1. Some common treatments for categorical variables are:

1. One-Hot Encoding (Dahouda and Joe, 2021) - Each category becomes an independent binary column.
2. Dummy coding scheme - similar to one-hot encoding. This categorical data encoding method transforms the categorical variable into a set of binary variables (also known as dummy variables). In the case of one-hot encoding, for N categories in a variable, it uses N binary variables.
3. Target (mean) encoding - replaces each category with the mean of the target variable for that category, which is helpful when the target (reserve magnitude) is strongly correlated with the categorical variable, but requires careful smoothing to prevent over-fitting.
4. Embedding - learns a dense vector representation for each category. The most scalable approach for high-cardinality features, as it reduces dimensionality while retaining semantic proximity between similar categories. Embeddings are particularly attractive for actuarial variables such as policy code, insurer, and region.

Brouwer (2004) investigated how to handle categorical inputs in FFNN models. They observed that a standard MLP implicitly assumes a continuous mapping from every input dimension to the output. When a categorical variable (e.g., gender) is encoded as a raw numeric value, the resulting mapping becomes discontinuous and often ill-posed because the numeric value carries

no intrinsic ordinal information. To remedy this, the authors proposed a hybrid network that merges an MLP (operating solely on the continuous features) with a separate encoder that maps each distinct categorical value to a dedicated output unit. By feeding the output of the encoder into the MLP's hidden layers, the network preserves the continuity of the numeric sub-space while allowing the categorical information to influence the hidden representation in a controlled manner. Empirical results showed that this separation of numerical and categorical inputs led to a consistently improved predictive accuracy compared with naïvely treating categorical values as continuous. Entity-embedding paradigms (Guo and Berkhahn, 2016) resolve these issues by learning, jointly with the main network, a low-dimensional vector representation for each category.

In this study, I do not have much categorical data and adopt one-hot encoding for low-cardinality nominal predictors. The one-hot encoding strategy keeps the input dimensionality manageable and allows the neural network to discover relationships that might be invisible to a purely linear encoding. The above principles are therefore left for future research.

By implementing the chosen encoding strategy in a preprocessing pipeline, I ensure that all categorical variables are incorporated into the DNN, preserving the full breadth of information that is essential for accurate reserve estimation.

1.5.12 Local minima and optimisation pitfalls

In an optimisation problem, the global minimum corresponds to a parameter configuration that yields the lowest achievable loss over the entire parameter space. For a well-specified reserve-prediction problem, the target loss would be zero if the model could perfectly reconstruct the functional relationship between underwriting inputs and the true reserve values. In practice, due to the highly non-convex loss surface induced by the network's multiple layers and multiplicative AFs, the optimisation trajectory frequently settles at a local minimum or traverses flat saddle regions.

Theoretical work by Auer et al. (1995) establishes that, even for a single hidden layer with Sigmoid AFs and a Mean Squared Error (MSE) LF, the number of local minima grows exponentially with the number of neurons. This combinatorial explosion means that for the deep architectures adopted in this thesis, the optimisation landscape is rife with sub-optimal valleys. Consequently, gradient-based methods such as Adaptive Moment Estimation (ADAM) can converge to solutions that are sub-par relative to the global optimum.

It was observed during the study that the network began predicting the average reserve for every policy, achieving a superficially low loss when evaluated on the training set but failing miserably on any hold-out data. This state - characterised by a flat output regardless of the

inputs - signifies that the network's weights had collapsed to a configuration where the last dense layer's bias dominated the output, and all preceding layers were effectively neutralised.

The global minimum is where the model fits the data perfectly and accurately predicts the results. If that can be attained, the model learned a weighted distribution of the underlying function. However, the solution space, especially where the underlying structure of the data is complex (see Section 1.7.2), is not smooth, and the solution space may contain many local minima or flat regions.

To resolve a local minimum, several techniques can be used, including:

1. Feature scaling: Feature scaling involves transforming the input features of the model so that they are on a similar scale. This can help the optimisation algorithm converge to a better solution by reducing the impact of outliers and improving the overall signal-to-noise ratio in the data.
2. Regularisation: Regularisation involves adding a penalty term to the LF used in the optimisation algorithm that discourages overfitting. Common regularisation techniques include L1 and L2 regularisation.
3. Model selection: Selecting a different model architecture, such as a deep neural network, can help resolve a local minimum by changing the shape of the LF and allowing the optimisation algorithm to converge to a better solution.
4. Ensemble methods involve combining multiple models to produce a more accurate prediction. This can help resolve a local minimum by providing a more robust prediction by considering the strengths and weaknesses of multiple models.
5. Hyperparameter tuning: Hyperparameter tuning involves adjusting the model's parameters, such as the LR, to improve the performance of the optimisation algorithm. This can help resolve a local minimum by enabling the optimisation algorithm to find a better solution.

Despite these precautions, I observed that the optimisation occasionally plateaued near saddle points-configurations where some directions in parameter space exhibit positive curvature and others negative. In such scenarios, the gradient magnitude tends to zero, causing the optimiser to stall. To address this, I monitored the curvature via the Hessian approximation implied by Nesterov Adaptive Moment Estimation (NADAM)'s second-moment estimates.

By using these techniques, a researcher can help reduce the impact of a local minimum and improve the performance of a ML model. However, it is essential to remember that there is no

guaranteed solution to a local minimum and that finding a global minimum can be difficult and time-consuming.

When using feature scaling, it is important to use a scale that is of the same range as the AF. And that only the Training dataset is used to perform the scaling. If the input data is not uniform or Gaussian, then another form, such as a log-transformation, should be used.

1.5.13 Hyperparameter optimisation

Hyperparameters in ML are those parameters that are explicitly defined by the user to control the training process. These include hyperparameters for the model and hyperparameters for optimisation:

- type of model (e.g. a DNN),
- number and depth of layers,
- train-test split ratio,
- batch size,
- choice of AF,
- number of Epochs,
- LF for training a NN,
- choice of optimiser,
- use of momentum,
- choice of LF,
- choice of regularisation, and
- to use ensemble learning.

The best values for hyperparameters are found by trial and error. In this study, various approaches were used. I first employed a grid-like search in the optimiser space, running hundreds of models to discard combinations of hyperparameters that did not work efficiently. Of the remaining candidates for hyperparameters, I ran a more in-depth analysis to find the combinations that are converging correctly. My approach is discussed in Chapter 3.

Akiba et al. (2019) produced a tool called Optuna[12] that can help researchers find the best optimisation hyperparameters. I did not use this tool, but it could be worthwhile to explore for other researchers having trouble with modelling.

1.5.14 Vanishing gradient

In a DNN, the gradient of the LF with respect to the weight and bias parameters is propagated backwards through each layer (back-propagation). When the AF is Sigmoid or TANH, the derivative of these functions is bounded. Consequently, when a network contains many layers, the product of successive derivatives often shrinks exponentially, a phenomenon known as the vanishing gradient problem (Bengio et al., 1994). The risk is that gradient updates for early layers become numerically negligible, preventing those layers from learning useful feature representations.

Using the RELU as the principal AF, the derivative of the AF is either 0 (for negative pre-activations) or 1 (for positive pre-activations). RELU eliminates the saturation problem for positive activations, thereby maintaining a constant gradient flow in those regions. Nevertheless, for inputs less than zero, the gradient indeed vanishes, which can lead to dying neurons if a unit's activation is permanently stuck at zero.

In general, this problem can be minimised by employing bias and with weight initialisation. This scaling reduces the probability that the initial pre-activations are predominantly negative, thus limiting the number of units that start in the inactive region.

The DNNs developed in this study employed the CELU AF with normally distributed weight initialisation and decreasing neurons per subsequent layer in the model architecture to handle this problem efficiently.

1.5.15 Curse of dimensionality

The curse of dimensionality refers to the combinatorial explosion in parameter space that occurs as the number of input features, hidden units, or network depth increases. In such situations, the optimisation landscape becomes increasingly rugged, making it difficult for gradient-based algorithms to locate the vicinity of a global optimum and often leading to slow convergence or outright failure to converge (Poggio et al., 2020, p. 30040–30041). The problem is exacerbated when the underlying mapping between inputs and outputs is highly non-smooth, because the required number of neurons grows with the complexity of the decision boundary.

The curse was mitigated in two ways:

[12] https://github.com/pfnet/optuna/

1. Controlled model capacity – The network architecture is deliberately compact. This choice balances the expressive power needed to capture the nonlinear relationships in a life-insurance portfolio against the risk of over-parameterisation.

2. Rich, full-coverage training data – The synthetic reference portfolio is constructed to span the entire admissible space of key variables (age, term, SA, policy type). By providing examples for every corner of the input domain, the target function behaves smoothly across the feature manifold. Consequently, the network learns a stable mapping, and the optimisation trajectory exhibits mostly convergence.

Under these conditions, no evidence of the curse manifesting was observed, as oscillatory loss curves or persistent spikes in gradient norms. The smoothness of the target reserve surface, coupled with the restrained parameterisation, ensures that the optimiser is able to navigate the loss surface efficiently.

1.5.16 Ensemble learning

Ensemble learning is a well-established technique in which several diverse models are combined to produce a single predictive output (Dong et al., 2020). In the remit of reserve valuation, this typically entails training multiple instances-often with different initialisations, architectures or data subsets-that all map the same set of underwriting features and basis reserve to an estimated reserve value. The ensemble can then aggregate the individual predictions by simple majority vote (for classification) or by weighted averaging (for regression).

For this thesis, the following motivations for adopting an ensemble paradigm were considered:

1. Variance suppression – Averaging over several models reduces the stochastic noise inherent to a single network. As different initialisations explore distinct local minima, the dispersion of their outputs tends to cancel out, yielding a smoother, more stable reserve estimate.

2. Uncertainty quantification – The spread of the ensemble's predictions can be interpreted directly as a confidence interval for the reserve, a feature that is particularly compelling for regulatory reporting and risk-based capital calculations.

3. Model robustness – Ensemble predictions are less sensitive to over-fitting or to idiosyncratic artefacts that may afflict a single network.

While the current work concentrates on a single, optimised DNN, a full exploration of ensembles is deferred to future research. In preliminary trials with two replicated DNNs whose weights were randomly initialised, the aggregated MAE did not improve materially over the best individual

model, suggesting that not using any ensemble learning is sufficient. A richer ensemble - either built from heterogeneous base learners (e.g., XGB, RF, or DT) or constructed via resampling strategies - will be investigated in future research, leveraging the insights from the ensemble taxonomy presented by Dong et al. (2020).

1.5.17 Reproducibility

Given the stochastic elements intrinsic to deep learning-random weight initialisation, data shuffling, and optimised hyperparameters - perfectly reproducible results are not guaranteed. This uncertainty was mitigated by controlling the random number generator: every experiment is run with a fixed seed (10101) in all libraries that expose seeding. With the seed fixed, the sequence of weight initialisations, minibatch orderings, and dropout masks becomes deterministic, enabling a third party to regenerate approximately the same loss trajectory and convergence behaviour.

Nevertheless, minor discrepancies are inevitable. Even with identical seeds, subtle differences in library versions, hardware floating-point precision, or underlying compiled routines can produce divergent floating-point trajectories that, over many iterations, lead to distinct local minima. To quantify the inherent variability, the full training procedure was repeated more than once for the final network configuration. The resulting MAEs on the hold-out set varied by a mean absolute difference of 0.08%, well within the tolerance required for actuarial reporting. Thus, while the exact numerical values of the weight vectors may not be identical, the predictive performance remains stable.

It is also noteworthy that DNNs possess a non-unique set of optimal solutions. The optimisation landscape comprises a continuum of parameter vectors that yield the same empirical loss. Consequently, the path taken by gradient descent determines which representative of this continuum is ultimately selected. Researchers attempting to reproduce the study should therefore focus on replicating the training protocol (architecture, loss, optimiser, hyperparameters) rather than on matching the exact weight tensors.

In summary, the methodological transparency embedded in this thesis - through detailed code, exhaustive documentation of data preprocessing steps, and controlled seeding - renders the results reproducible to within statistically acceptable bounds. Future work might explore deterministic training regimes or formal variance-reduction techniques to further close the reproducibility gap.

1.5.18 Evaluation of model performance and accuracy

This study utilises a peer review methodology, as proposed by Richman et al. (2019), to ascertain model performance and accuracy. Consequently, conventional metrics such as R^2 (Wright, 1921) and Area Under Curve (AUC) (Fawcett, 2006) are deemed unnecessary. Instead, a comparison is made between the outputs of a traditional actuarial valuation model and those from the trained DNNs. The output from the traditional actuarial model is designated as the "Actual" values, while the output from the DNNs is referred to as the "predicted" values. This approach is maintained throughout the this study, both in table formats and in graphs of actual versus predicted values. The scatter plots provide a detailed description of the output per policy, while the tables present the total accuracy for the entire portfolio as a summary.

This approach – comparing actual versus predicted values – is consistent with common model evaluation techniques such as MSE or MAE analysis, and represents a robust measure of model performance in actuarial contexts. By directly assessing the alignment between model outputs and observed values, this methodology provides a clear and interpretable indication of the model's predictive capability and its ability to accurately reflect underlying risk. It moves beyond simple statistical metrics to focus on the practical implications of model outputs for real-world valuation and risk management purposes.

This methodology directly addresses the research objectives outlined in Section 1.2, specifically the aim to evaluate the performance and accuracy of deep learning models in actuarial contexts while prioritising interpretation and practical application. By focusing on a direct comparison of model outputs with observed values, this approach provides a clear and quantifiable assessment of model performance, aligning with the emphasis on practical relevance and risk management outlined in the research methodology. The chosen methodology also facilitates a transparent and readily understandable evaluation, fulfilling the requirement for a robust and interpretable model assessment as outlined in Section 2.5.1.

1.6 Measuring the accuracy and regulatory compliance of a Neural Network

Evaluating the predictive accuracy of the developed DNN model is essential to ensure that the estimated values correspond closely with the observed, or actual, data. In this work, the **accuracy** of prediction was assessed quantitatively and graphically.

1.6.1 Quantitative accuracy

To measure the agreement between the total actual values (A) and the total estimated or predicted values (E) produced by the DNN, the accuracy (ratio) is defined as the Estimated-to-Actual (E/A) percentage, expressed as:

$$Accuracy = \begin{cases} \frac{E}{A} * 100 & \text{if } A \geq E \\ (2 - \frac{E}{A}) * 100, & \text{otherwise} \end{cases} \tag{1.6}$$

Values close to **100%** indicate near-perfect prediction accuracy, whereas larger deviations suggest under- or over-estimation.

1.6.2 Graphical accuracy

To visually validate the predictive performance, each individual model point i was plotted (as a scatter plot) with the predicted value (E_i) on the y-axis and the corresponding actual value (A_i) on the x-axis.

An ideal model would produce all points lying precisely along the 45-degree reference line, where $E_i = A_i$.

The closer the data points cluster around this line, the greater the model's fidelity in reproducing the actual data. Systematic deviations above or below the line provide an intuitive diagnostic of consistent over-prediction or under-prediction, respectively. If a DNN's predictions closely follow the reference trend, it display satisfactory overall model accuracy.

1.6.3 Regulatory compliance

As stated in Section 1.4.3, EIOPA requires 99.5% accuracy when calculating reserves. Therefore a model must be able to calculate reserves that are at least 99.5% accurate. Therefore the MAE, E/A and scatter plots were investigated to ensure that no individual reserve value was less than 99.5% accurate, in order for the model to comply with the Solvency II requirement.

1.6.4 Interpretation and use in model validation

The E/A-based numerical analysis and the 45-degree graphical test together provide a balanced assessment of model performance:

- Quantitative metrics (total E/A) offer numerical precision for comparative purposes among alternative architectures, and
- Visual inspection of the scatter plot confirms the consistency of model behaviour across the full range of observed data.
- Regulatory compliance is when all the individual model points are within 99.5% of the 45-degree line.

This dual approach ensures that both statistical reliability and interpretative transparency are achieved when evaluating the DNN's predictive capability.

1.7 Contribution to the field

The thesis builds a proof-of-concept that a single deep-learning architecture - named Midway - can serve as a fully compliant, transparent actuarial estimator for long-term life-insurance reserves. This contribution is dual-faced: it delivers a practical solution that can be deployed in a regulated environment, and it proposes a new research direction that merges actuarial theory with modern supervised learning.

1.7.1 Empirical proof and its wider impact

The Midway network, trained on a synthetic reference portfolio that mirrors the admissible range of life insurance products, achieves an MAE below 0.08% of the reserve values on an out-of-sample set that was not used during any phase of training. The model's predictions can be interpreted at both the global and local levels, thereby satisfying the requirement of explainability imposed by Solvency II. This result supports three important implications:

1. Catalyst for actuarial data-science – By demonstrating that a regulator-compliant valuation can be expressed as a supervised regression problem, the study invites more actuaries to experiment with ML techniques.
2. Operational audit support – Insurers and external auditors can adopt the Midway model as an independent, verifiable check on hand-calculated reserves, reducing manual effort and audit friction.
3. Regulatory evolution – Regulators may view the model as a tool for stress testing and solvency assessment, acknowledging that modern AI can coexist with traditional actuarial principles.

1.7.2 Complexity of traditional actuarial present-value calculations

Actuarial reserve calculations are inherently high-dimensional: they involve

- Demographic risk (mortality, surrender) derived from product-specific tables,
- Investment risk via time-varying discount factors, and
- Product-structure risk (term, benefit triggers, policy-holder behaviour).

Mathematically, the reserve for a WL contract is the sum of a series of cashflow elements, each assigned a probability of occurring and discounted at some discount rates under the chosen basis assumptions. Regulators require that the reserve be prudential - i.e., it should be larger than the BE reserve to provide a safety buffer. This leads to a margin that is calculated at each valuation date, with the excess potentially released as profit. The margin is a direct reflection of the insurer's risk appetite and is tightly controlled by Solvency II.

The actuarial PV of any insurance business is calculated by discounting future cash flows, allowing for the probability of them occurring. The demographic probabilities depend on the mortality and surrender tables used for the particular product. Policyholders not dying or surrendering are part of the IF book of the WL business, for which reserves need to be held, depending on the product and contract terms as specified.

Because of the sequential and probability-driven nature of these calculations, the loss surface that a learning algorithm would have to fit is highly non-convex and sparsely informed. Conventional practitioners, therefore, use deterministic, highly engineered models that are transparent but computationally intensive. By contrast, the Midway architecture embeds the actuarial basis reserve explicitly as an input, learning only the residual that captures the complex, non-linear adjustments. This design respects the same prudential standards while exploiting the statistical efficiency of NNs.

In what follows, the empirical evaluation confirms that this hybrid approach does not compromise regulatory compliance while delivering superior predictive performance. The work thereby illustrates a viable pathway for integrating modern deep learning into the actuarial valuation toolbox, thereby addressing the dimensional complexity that has historically limited AI adoption in this domain.

1.7.3 Challenges presented to Machine Learning adoption in the insurance industry

ML pipelines traditionally presume that, once a model has been trained, the training data no longer participates in further computation. In the life-insurance domain, this assumption is violated for several reasons as presented in Table 1.6:

Challenge	Description	Implication for ML
Dynamic book of business	Existing policies age out, new contracts are written, and existing contracts change status (e.g., lapses or surrenders).	The distribution of input features shifts over time, rendering a model trained on the historical book progressively mis-aligned with the current valuation universe.
Evolving pricing and product design	Premium bases, product features, and underwriting guidelines are regularly revised to maintain competitiveness and manage risk.	Model inputs (e.g., product code, underwriting score) acquire new semantics, potentially invalidating learned embeddings or feature interactions.
Economic and regulatory drift	Discount rates, mortality assumptions, and solvency regulations are subject to periodic revision.	The target reserve function changes, so the mapping from inputs to reserves is no longer stationary.
Regulatory constraints on transparency	Regulators demand that reserve calculations be auditable and traceable.	Black-box models, or models with opaque feature importance, face increased scrutiny and may be rejected or require extensive justification.

TABLE 1.6: Challenges in training ML models for insurance business

These dynamics mean that a model deemed accurate at the time of training may quickly become obsolete. Traditional actuarial practice handles this by reevaluating the book at each reporting date, but an ML system would have to be retrained or recalibrated with each shift - a costly endeavour.

Furthermore, actuarial expertise - particularly in constructing appropriate economic bases, in understanding how mortality and surrender tables interact with investment assumptions, and in interpreting profitability metrics - is not readily encoded in general-purpose ML libraries. This lack of domain transfer limits the effectiveness of off-the-shelf ML solutions.

Consequently, it is imperative to establish a principle-driven framework for developing ML models in this context:

1. Data spectrum training – Training should occur on the entire spectrum of possible policies.
2. Versioning of economic assumptions – A clear audit trail linking each model version to the specific discount rates and actuarial tables used in its training.
3. Hybrid architecture – Explicitly feed the actuarial basis reserve and other regulatory quantities as structured inputs so that the learned mapping focuses on residuals.
4. Regulatory compliance audit – Provide formal documentation and deterministic validation tests that demonstrate adherence to prudential standards.

By adhering to these principles, the insurance industry can harness the statistical power of deep learning while ensuring that the resulting models remain valid, auditable, and responsive to the ever-changing environment of life insurance.

1.7.4 Challenges posed by data

The data-intensive nature of actuarial reserving introduces a distinct set of obstacles that are amplified when deep learning is applied.

A life-insurance policy is described by dozens of policy-level variables (e.g., product, age, term, SA, premium), hundreds of demographic attributes, and a host of derived actuarial features (e.g., mortality tables, discount rates). As cited by Goodfellow et al. (2016), the curse of dimensionality makes generalisation exponentially harder; the conventional assumption that simple linear models suffice falls apart when underlying relationships are highly non-linear (Bengio et al., 2013).

Hynes et al. (2017) demonstrates that over 90% of real-world datasets contain at least one error (referred to as "lints"). In the actuarial context, this means inconsistent premium coding, missing underwriting scores, or duplicated policy entries. Noisy inputs lead the network to learn spurious associations that do not translate to unseen data, thereby damaging generalisation.

As detailed in the preceding subsection, the statistical distribution of policies shifts over time. A model trained on a snapshot can quickly become mis-calibrated when the frequency of certain product lines or risk scores changes. This distribution shift is especially problematic when the network has memorised highly specific input patterns (overfitting).

The evaluation of a NN with several million parameters still incurs more computational overhead than a deterministic actuarial formula or a Monte Carlo (simulation) (MC) for a single contract. Doyle and Groendyke (2018) demonstrated that, for Variable Annuities (VA) guarantees, a well-trained neural model can reduce runtime by an order of magnitude compared to MC, but

only after a significant upfront training cost. When considering large portfolios, the cost of repeatedly retraining the network to accommodate new data may become non-negligible.

1.7.5 Comparing processes when calculation mistakes are found

Traditional actuarial valuations benefit from well-established procedures for error detection and correction, underpinned by rigorous documentation and auditable processes. However, the application of DNNs in actuarial science, particularly for complex tasks such as reserve calculation, presents new challenges to maintaining this level of reliability and transparency. Unlike traditional methods, DNNs operate as 'black boxes,' lacking inherent traceability and making error identification more difficult. This absence of established validation procedures poses a significant risk to the accuracy and auditability of DNN reserve valuations.

While DNNs offer potential gains in efficiency and predictive power, the inherent complexity of these models necessitates a robust error detection framework. The retraining requirement following error identification can be computationally expensive and time-consuming. Furthermore, the potential for errors to propagate through a DNN and compound across a portfolio of policies demands a proactive approach to error mitigation. Currently, there is a lack of standardised procedures for validating DNN-driven reserve calculations, creating challenges for both internal control and external audit.

This research addresses this gap by including valuation bases with policy data, inspired by established actuarial validation checklists. This algorithm provides a structured and systematic approach to identifying discrepancies between DNN-calculated reserve values and expected traditional calculations, based on established valuation principles. The algorithm focuses on pinpointing the source of these discrepancies, whether originating from data input, valuation bases, model parameters, or inherent model behaviour.

The algorithm combines the following metrics:

1. Policy data: Verifying the accuracy and completeness of input data, including individual policy characteristics, such as SA, premiums, age, and term. In particular, outliers are included during training.
2. Valuation Bases: Ensuring that the model utilises the full spectrum of possible valuation bases and assumptions, and that these bases are consistently applied across all calculations. This includes verifying the accuracy of input parameters and performing sensitivity analysis to assess the impact of changes in assumptions.

A crucial aspect of validation involves comparing multiple valuation runs. While each run typically utilises the same or very similar policy data, variations are introduced through differing

valuation bases items – such as interest rates, mortality assumptions, or expense loadings. By systematically altering a single basis item between runs, the resulting difference in reserve values directly quantifies the impact of that specific assumption change. This comparison serves as a powerful verification step, ensuring the model's sensitivity aligns with expected actuarial principles. Discrepancies in the expected direction and magnitude indicate potential errors in either the model implementation or the input basis values. This run-to-run comparison is a standard procedure within AOS.

Beyond verifying basis changes, this comparative approach is also fundamental to identifying errors within a single valuation run. By comparing the PV of cash flow elements across the policy portfolio, discrepancies can pinpoint specific areas requiring further investigation.

In a typical reporting valuation, numerous runs are performed to assess the impact of various scenarios. By systematically comparing the outputs of consecutive runs, discrepancies can be quickly identified. Once corrected, the affected run and all subsequent runs must be recomputed to ensure data integrity throughout the entire valuation process. This run-to-run comparison, therefore, serves as a critical safeguard against error propagation and ensures the accuracy and reliability of the reported reserve values.

This research introduces a structured algorithm, drawing parallels with established actuarial validation checklists, to facilitate systematic error detection and correction in ML model outputs. This algorithm will be applied to the DNN model to improve the reliability of reserve value calculations. The methodology presented offers a framework for translating established actuarial principles to the domain of ML, contributing to a more robust and auditable application of these technologies.

1.7.6 Issues in regression Machine Learning itself

While ML is no longer an unknown technology, significant theoretical gaps remain, particularly within the field of deep learning. Despite demonstrated empirical success, a comprehensive understanding of why DNNs function as they do remains elusive. Many studies offer only partial insights, and the potential for unexpected behaviours and limitations remains largely unexplored. While theoretically capable of approximating any function given sufficient resources, DNNs may exhibit overfitting, limiting their generalisability to unseen data and hindering their practical application.

Shaham et al. (2018) highlights this paradox, noting that the success of DNNs occurs despite the absence of a robust theoretical foundation. Their work demonstrates that a two-layer DNN, under specific assumptions and utilising the RELU activation function, can indeed approximate

any regression function to a desired degree of accuracy. However, this result does not address the broader challenges of model stability, robustness, and generalisation.

Fernandez-Arjona and Filipović (2022) further emphasises the difficulties inherent in modelling high-dimensional functions, as commonly encountered in actuarial valuation. They propose a data-driven dimensionality reduction technique based on adjusting AFs. While data quality is undoubtedly critical to the success of any ML model, dimensionality reduction is not a panacea. Moreover, the authors point to a significant challenge within the life insurance industry: the lack of publicly available datasets for benchmarking and comparing advanced modelling techniques. Notably, their approach involves utilising the data itself to estimate optimal activation functions, raising questions about potential data dependency and bias.

Claims of superior performance among various ML models should be interpreted with caution. The brevity of typical ML research articles often limits the depth of analysis and hinders meaningful interpretation of results. Moreover, improvements in model performance can frequently be attributed to better data preprocessing and optimisation, rather than to architectural innovations (Shwartz-Ziv and Armon, 2022). Consequently, conclusions drawn from research papers should be viewed skeptically, recognising that a model validated on one dataset may not generalise effectively to others. This limitation is particularly relevant in the actuarial context, where data scarcity and the complexity of underlying risks necessitate a cautious approach to model deployment.

1.8 Research philosophy

This thesis adopts:

- A realist ontology (deterministic systems exist independently of observation)(Sacks et al., 1989),
- A post-positivist epistemology (knowledge is approximate and empirically tested)(Popper, 2005), and
- A pragmatically informed quantitative methodology(Creswell and Creswell, 2017).

This philosophical positioning is consistent with contemporary computational science, ML research, and the statistical design of computer experiments.

1.9 Overview of research design and ethical considerations

This study adopts a quantitative, computational research design grounded in controlled numerical experimentation. The central objective is to evaluate deep neural networks as surrogate models for deterministic systems. The research therefore follows an experimental modelling framework consisting of:

1. Specification of deterministic benchmark systems,
2. Construction of training and validation datasets via structured sampling,
3. Development and training of deep neural network surrogate models,
4. Comparative evaluation using predefined performance metrics, and
5. Sensitivity and robustness analysis.

The design is deductive in structure: hypotheses regarding surrogate performance, scalability, and approximation behaviour are evaluated through systematic experimentation. The research does not involve human participants, survey instruments, or qualitative data collection. Instead, it relies on computational data generated from deterministic simulators and publicly available benchmark datasets where applicable.

The study further incorporates controlled variation of architectural hyperparameters to ensure replicability and internal validity.

Methodologically, the research is situated within computational science and ML experimentation. The approach can be characterised as:

- Quantitative,
- Model-driven,
- Experimentally controlled,
- Reproducible.

DNNs are implemented and evaluated under consistent training protocols. Performance is measured using established regression metrics such as MAE.

Comparative analysis against alternative surrogate modelling techniques (where applicable) provides contextual benchmarking and strengthens the inferential basis of conclusions.

Although the research does not involve human participants, ethical considerations remain relevant in the context of responsible AI and scientific integrity. As the research focuses on deterministic numerical systems rather than social or demographic data, risks of algorithmic bias affecting protected groups are not directly applicable. However, the study recognises that surrogate modelling techniques may be deployed in sensitive domains, and therefore emphasises:

- Appropriate validation before deployment,
- Avoidance of over reliance on surrogate predictions,
- Transparency regarding epistemic uncertainty.

In summary, the research design is quantitative, experimental, and computational. It prioritises reproducibility, rigorous benchmarking, and transparent reporting. Ethical considerations primarily concern responsible AI practice, scientific integrity, and appropriate use of computational data rather than human subject protection.

1.10 Structure of the thesis

This research addresses a gap in the application of deep learning techniques to the valuation of long-term insurance business. While deep learning is a rapidly evolving field, its application to actuarial problems, particularly those involving long-term liabilities, remains largely unexplored. Consequently, this study adopts a broad framing of "... **Meet Deep Learning**" to encompass the novel methodology developed and presented herein.

The thesis is structured as follows:

Chapter 1 provides an introduction to the foundational concepts of life insurance, actuarial valuation, and deep learning, establishing the theoretical underpinnings of the research.

Chapter 2 reviews the existing literature on long-term insurance products, actuarial valuation, and deep learning techniques, culminating in a summary of current research and identifying the specific challenges addressed by this study.

Chapter 3 details the methodology employed in this research, outlining the development of the proposed deep learning model and the data preparation techniques utilised.

Chapter 4 presents a comparative analysis of the models' performance, visually and statistically, against traditional actuarial valuation approaches. This chapter highlights the strengths and limitations of the proposed methodology.

Chapter 5 summarises the key findings of the research, draws conclusions regarding the applicability of deep learning to long-term insurance valuation, and proposes avenues for future research.

The appendices A, B, and C contain pseudo-code diagrams, details on some interim models trained towards the final models, and finer details on ML methods, respectively.

Chapter 2

Research landscape

2.1 Introduction

As noted in Chapter 1, traditional actuarial valuation of the life insurance business is very complex and time-consuming. However, ML methods are just as complex. For example, Norgaard et al. (2013) gives the following examples of complexities: implementation, model setup, training and estimation of parameters. Furthermore, the use of ML methods to date has been restrictive and not focused on the life insurance field.

This chapter provides a contextual overview of both traditional actuarial valuation practices and the emerging application of ML techniques within the life insurance industry. It explores the inherent complexities of both approaches, highlighting the challenges and opportunities that motivate this research.

2.2 Evolution of actuarial valuations

The calculation of appropriate premium rates and the accurate estimation of future liabilities have been fundamental to insurance since its inception. Insurers must balance profitability and market competitiveness while maintaining sufficient reserves to meet future claim obligations. These reserves, representing the present value of expected future payouts less expected future income, form a critical liability on insurers' balance sheets.

While the history of actuarial science extends back centuries, this study focuses on the evolution of modern valuation techniques. For a comprehensive historical overview, readers are directed to Haberman (1996) and Bulinskaya (2017).

Actuarial valuations can be performed on a deterministic or stochastic basis, the latter providing a range of potential outcomes. For the purposes of this study, a deterministic basis is employed for pricing and valuation, allowing for a focused analysis of core principles. The specific formulae used for present value calculations are detailed in the next subsection, aligning with the derivations presented by Dickson et al. (2020) - specifically, those concerning:

- Present value of a whole life contract (page 109),
- Premium calculation (pages 182-191), and
- Reserve calculation (pages 219-231).

Beyond the core valuation calculations, the effective management of an insurance company requires numerous strategic and operational decisions. Wüthrich and Merz (2013) provides a comprehensive overview of these broader considerations.

2.2.1 Actuarial notation and calculation formulae

Standard International Notation was used (The Faculty of Actuaries and The Institute of Actuaries, 2002). For more details than provided in this Subsection, please refer to Macdonald (2006) or Slud (2012).

For the economic variables, the notations used in this study are:

- i, the interest rate compounded per annum,
- v, the discount rate, where $v = \frac{1}{1+i}$,
- e, the expense, expressed as an amount per month, and
- ei, the expense inflation rate, assumed to be annually compounded.

The following notation for decrements is used, which is the standard actuarial notation as taught to all actuaries, and that can be found in The Faculty of Actuaries and The Institute of Actuaries (2002), which is briefly repeated here for completeness or as used in this study.

- l_x the number of people alive aged exactly x years,
- d_x the number of deaths aged x last birthday (i.e. dying between ages x and x+1), and calculated as $l_x - l_{x+1}$,

- $q_x^{(d)}$ the probability of dying over the next year for a person aged exactly x years, calculated as $\frac{d_x}{l_x}$,
- w_t is the crude surrender rate over the next year, for a policy with a Term In Force (TIF) of t,
- ap_x is the dependent probability of survival to the next year for a person aged exactly x, calculated from the independent decrement probabilities expressed as forces of interest, as $e^{-(u_d+u_w)}$, where u_d and u_w are the forces of death and surrenders, respectively,
- aq_x is the dependent probability of not surviving the next year for a person aged exactly x, calculated as $1 - ap_x$,
- $aq_x^{(d)}$ is the dependent probability of dying over the next year for a person aged exactly x, calculated as $\frac{q_x^{(d)}}{q_x^{(d)}+w_t} * (1 - ap_x)$,
- a_x is the present value of an annuity of 1 payable annually in arrears for life for a life aged exactly x years, and
- A_x is the present value of a death benefit of 1 payable at the end of the year of death for a life aged exactly x years.

When a subscript t is put in front of ap_x or aq_x, it denotes the probabilities, not for one year, but up to the time $x + t$.

For this study, ω, the limiting age, is set at 108, which reduces calculation time and implies that all people die when they reach age 108.

The whole formula for calculating a_x is therefore the sum of possible benefits payable from now until ω:

$$\ddot{a}_x = 1 + v * ap_x + v^2 * ap_x * ap_{x+1} + ... \tag{2.1}$$

$$\ddot{a}_x = \frac{N_x}{D_x} \tag{2.2}$$

where (for an annuity due):

$$D_{x+t} = v^{x+t} * l_{x+t} \tag{2.3}$$

$$N_x = \sum_{t=0}^{\omega} D_{x+t} \tag{2.4}$$

For A_x, the calculation is from the current year of age until ω. The whole formula is, therefore:

$$A_x = v * ap_x * aq_x^{(d)} + v^2 * ap_x * ap_{x+1} * aq_{x+1}^{(d)} + ... \tag{2.5}$$

$$A_x = \frac{M_x}{D_x} \tag{2.6}$$

where (if paid at the end of year of death):

$$C_{x+t} = v^{x+t+1} * d_{x+t} \tag{2.7}$$

$$M_x = \sum_{t=0}^{\omega} C_{x+t} \tag{2.8}$$

$$A_x = \frac{M_x}{D_x} \tag{2.9}$$

For premiums and death benefit paid with compound j% increases per annum, the above formulae are calculated at and i'% interest, where:

$$i' = \frac{1+i}{1+j} - 1 \tag{2.10}$$

For premiums and death benefits paid with a constant amount K of increase per annum in arrears, the formulae for calculating the PV of only the additional K is:

$$S_x = \sum_{t=0}^{\omega} N_{x+t} \tag{2.11}$$

$$R_x = \sum_{t=0}^{\omega} M_{x+t} \tag{2.12}$$

$$\mathrm{K} * Ia_x = \mathrm{K} * \frac{S_x}{D_x} \tag{2.13}$$

$$\mathrm{K} * IA_x = \mathrm{K} * \frac{R_x}{D_x} \tag{2.14}$$

where the letter I denotes that the a_x and A_x is of such an increasing nature.

For EA business of term n the formulae for calculating the PV of premiums and benefits are given by:

$$\ddot{a}_{x:\overline{n}|} = \frac{N_x - N_{x+n}}{D_x} \tag{2.15}$$

$$A_{x:\overline{n}|} = \frac{M_x - M_{x+n} + D_{x+n}}{D_x} \tag{2.16}$$

Finally for TA business of term n the formulae for calculating the PV of premiums and benefits are given by:

$$\ddot{a}_{x:\overline{n}|} = \frac{N_x - N_{x+n}}{D_x} \tag{2.17}$$

$$A^1_{x:\overline{n}|} = \frac{M_x - M_{x+n}}{D_x} \tag{2.18}$$

2.2.2 Life Insurance Products

Traditionally, life insurance products have fallen into one of the following categories:

- TA,
- WL,
- EA, and
- Annuities and Retirement Benefits.

With advancements in technology and market innovation, several more modern products have been developed, including:

- Universal Life Insurance,
- Unitised With-Profit Insurance, and
- Equity Insurance.

A comprehensive overview of these products and their underlying mechanisms can be found in Dickson et al. (2020). While access to data limited this study to Whole Life insurance policies, a comparative analysis of multiple product types would have provided valuable insights. In Section 3.9.2, a model will be trained on three distinct classes of protection business, and the results discussed in Section 4.4.

2.2.3 Methods used for calculations

The deterministic present value calculations used for annuity valuation, and employed in this study for all life insurance products, are detailed in Pitacco et al. (2009, p. 33-34). While Pitacco et al. (2009) also explores various stochastic approaches and the impact of mortality improvement, this study utilises a deterministic framework.

Regulatory standards, such as those outlined by the Committee of European Insurance and Occupational Pensions Supervisors (CEIOP), allow insurers to employ either standardised formulas or develop internal models for valuation. However, the imprecise language within these standards often presents implementation challenges for insurance companies (Hejazi, 2016, p. 76).

Specifically, this study incorporates a valuation calculation similar to the Solvency Capital Requirement (SCR) under Solvency II, which can be summarised as:

- the higher of two capital requirements,
- the amount of capital that insurers need to ensure they can meet their obligations over the next 12 months with a 99.5% confidence level, and
- can be calculated either with a standard formula or with an internal model.

Furthermore, Hejazi (2016, p. 81) highlights that the innovative and complex structure of insurance products does not allow for a straightforward calculation of liabilities. In practice, insurance companies often have to calculate the liabilities of insurance products by direct valuation of the cash flows associated with them. Hence, the difficulty in the calculation of SCR is primarily associated with the difficulty in the calculation of liabilities.

Further details on stochastic mortality modelling can be found in Plat (2009).

2.2.4 History of actuarial valuations in life insurance

Babbel and Merrill (1999) provides many of the general issues that are still applicable today, in particular, that any model used to calculate the value of assets and liabilities should be implementable and not exceed its capacity. Although the study is mainly focused on market-consistent valuations, they warn in general against developing theoretical models that are not practical or implementable. Models should be able to be created within a reasonable timespan and not exceed available computer technologies, which was a concern in 1999 but is not a real issue today. Furthermore, they state that a model should be developed on well-established valuation practices by professionals.

In Section 4 of their study, they focus specifically on insurance liabilities. They highlight that an indirect approach, albeit quick, is inefficient, and they show this by providing an example. For this study, an indirect approach is seen as a high-level analysis based on summarised data.

For a direct valuation approach, they advocate:

- taking interest rate as the current best market interest rate (for the United States (US) at that time, it was the US Treasury rate),
- mortality and morbidity on an expectation basis, with some sensitivities included, and
- to apply some margin or cushion for prudence, especially if any of the estimates are uncertain.

They highlight the problems in regards to expense inflation risk for insurers no longer writing new business.

Section 5 of their study presents arguments for and against using stochastic valuations over deterministic assumptions. The benefits of the stochastic model are that a range of outputs can be obtained, which can help the insurer see the distribution of outputs, as well as spot particular onerous scenarios that may be highly unlikely to occur, but can have significant financial consequences. Developing and maintaining a stochastic model comes at a cost; such a model takes more time to set up, more care is required to determine the input distributions and parameters, and it needs more expert knowledge to interpret the results. A deterministic valuation provides a single value, but different valuation bases can be used to perform scenario analysis and are much faster to implement.

They propose that a unified model for the US insurance market would benefit from a stochastic interest rate model to promote consistency and address differing levels of prudence applied by individual insurers. They also underscore the importance of open architecture and auditor acceptance for the successful implementation of new approaches.

Biffis (2005) extended the application of stochastic methods to mortality assumptions, demonstrating a practical implementation using pensioner mortality data. Their rationale stemmed from increasing uncertainty surrounding future mortality trends, particularly the unprecedented increases in longevity. They argued that changes in mortality estimates could significantly impact long-term business lines such as WL and annuity products. This approach was further developed by Dahl and Møller (2006), who first proposed utilising a time-inhomogeneous Cox-Ingersoll-Ross (CIR) model to model mortality stochastically – an approach that remains prevalent today.

Biffis (2005) utilised the Italian Life Tables SIM92 in their analysis. Consistent with this, this study also employs the SIM92 tables, which extend to age 108. To maintain consistency in the

mortality input for the ML models, the mortality calculations for the AM92 table were similarly limited to age 108, ensuring a consistent age range of 17 to 108 for all mortality inputs.

2.2.5 Solvency II and Regulatory Framework

Insurance companies are required to report valuation results to regulators and stock exchanges (if listed), adhering to standards such as IFRS 17[1].

While this study focuses on valuation reserves (contributing to liabilities and therefore the Net Asset Value (NAV) of a company), historically, insurance companies also reported their Market Consistent Embedded Value (MCEV). The formula for MCEV, prescribed by CFO Forum (2008), is:

$$\text{MCEV} = \text{ANAV} + \text{PVFP} - \text{COC} \tag{2.19}$$

where:

- ANAV is the Adjusted NAV, and
- COC is the sum of the frictional cost of required capital and the cost of residual non-hedgeable risks.

The Solvency II directive (European Commission, 2009) – effective January 1, 2016 – mandates the use of a risk-free interest rate term structure for valuations (Article 77.2). Furthermore, Articles 77.1 and 77.3 stipulate that technical provisions (reserves) must equal the sum of best estimate liabilities and a risk margin, representing the cost another insurer would charge to take over those liabilities in an arm's length transaction. Solvency II requires the industry to compute ANAV based on a probability distribution, using either the standard model or a regulator-approved internal model.

Solvency II comprises three pillars:

- Pillar I: Quantitative requirements encompassing the sum of BE and a SCR measured at the 99.5% Value At Risk (VAR) over one year.
- Pillar II: Own Risk Solvency Assessment (ORSA), a company's assessment of an optimal solvency constraint.

[1] https://www.ifrs.org/issued-standards/list-of-standards/ifrs-17-insurance-contracts/

- Pillar III: Qualitative requirements, including documentation.

Pillar II allows companies greater flexibility in defining models and adapting to their specific circumstances (Laurent, 2016, p. 106-107).

The implementation of Solvency II significantly increased the computational burden on life insurance companies and their software providers, necessitating stochastic calculations to produce the required 99.5% measures. While Tucker and Bull (2014) suggested simplified approaches to stochastic mortality calculations, software development companies eventually provided the necessary capabilities.

Castellani et al. (2018) investigated the potential of ML tools to facilitate compliance with Solvency II, highlighting the trade-off between timely submissions and regulatory adherence. They proposed High-Performance Computing and ML as potential solutions, comparing MC to DNNs and SVMs for with-profit life insurance policies. They observed that increasing model complexity (number of layers) could increase training accuracy but decrease testing accuracy. In contrast, this study suggests that testing accuracy is more strongly influenced by the similarity between training and testing datasets – complexity does not necessarily jeopardise accuracy if datasets are comparable. This study agrees with their conclusions regarding LR:

- A high training rate can prevent convergence to the global minimum.
- A low training rate can significantly slow down training.

Wüthrich et al. (2016) provides a comprehensive overview of running an insurance company on market-consistent actuarial principles, offering a detailed review of current practices under Solvency II, utilising a stochastic approach.

Recent advancements in life insurance valuation methodology include n-step actuarial valuations (Barigou et al., 2022). Exploration of these developments is left for future research.

From an accounting perspective, the implementation of IFRS 17 represents a significant shift for insurance companies, detailed in the next Section 2.2.6.

2.2.6 IFRS 17

IFRS 17 was developed to address the accounting complexities arising from the timing difference between premium payments and claim settlements. It prescribes that profit be recognised evenly over the policy term, significantly impacting profit reporting for long-term insurance contracts, and replacing IFRS 4.

Palmborg et al. (2021) discusses a method for companies to comply with IFRS 17 reporting, providing examples of calculations for single-premium annuities and pure endowments. They also raise criticisms regarding the risk margin calculation within Pillar 1 of Solvency II. They highlight the need for insurers to project the run-off of their reserves for each future valuation date, which is consistent with the limited proof-of-concept presented in this study to output reserves with the Midway model for each of the next 10 years.

A key requirement of IFRS 17 is the ability to project future income statements and balance sheets, for which an AOS is recommended to verify the accuracy of all reserves.

2.3 Machine learning methods

Two foundational books for understanding ML and deep learning are Murphy (2012) and Goodfellow et al. (2016). These resources provide a statistical framework and delve into relevant metrics. This study builds upon the fundamental principles discussed therein.

A valuable overview of the field as of 2019 is provided by Shrestha and Mahmood (2019).

For further details on the mathematics underpinning this section and Appendix C, please refer to Murrell and de Freitas (2019).

After reading this section, the reader is advised to read Section 5.2 as an overview of what this study found. For specific implementations and case studies, please refer to Section 2.4 of this study.

2.3.1 The mathematics behind machine learning

This section outlines the mathematical framework underpinning the proposed ML approach.

The following are the core definitions of the variables used in this study:

- a_j is the "activation" of neuron j,
- x_i is the input from the i^{th} variable x from the input dataset,
- w_{ij} is the weight applied to x_i by neuron j, and
- b_j is the bias of each neuron in layer j.

The activation of neuron j is calculated as a weighted sum of its inputs plus a bias:

$$a_j = \sum_{i=1}^{n} w_{ij} * x_i + b_j \tag{2.20}$$

Weights and biases are iteratively updated to minimise the error. The update rule is as follows:

$$[w(t+1); b(t+1)] = [w(t); b(t)] + \eta[\partial w(t); \partial b(t)] \tag{2.21}$$

Where η is the LR. This rule adjusts the weights and biases in the direction of the negative gradient of the LF.

The activations are then transformed into non-linear outputs using a differentiable AF $h(\cdot)$ to give $z_j = h(a_j)$.

These non-linear outputs are then linearly combined to compute the activations of the output units:

$$a_k = \sum_{i=1}^{n} w_{ik} z_i + b_k \tag{2.22}$$

Finally, the output unit activations are transformed using an appropriate AF to give a set of network outputs y_k.

The network's prediction for input X can be expressed as:

$$f(X) = \sigma_n(W_n \sigma_{n-1}(...(W_1 \sigma_1(W_1 X + b_1) + b_2) + ...) + b_n) \tag{2.23}$$

Where σ_n represents the AF of layer n. The vector $B = [b_1, b_2, ..., b_n]$ contains all the biases in every layer and is unique. A potential issue arises when using AFs such as TANH or Sigmoid. In such cases, if the bias vector B is initialised to zero, the network may suffer from vanishing gradients or divergence from the true values.

While a simple approach of repeatedly applying gradient descent from random starting points might seem intuitive, Bishop (2008) demonstrates that it is often a poor algorithm. They highlight the need to explore multiple starting points and validate performance on an independent dataset to find a sufficiently good minimum.

Furthermore, online methods offer advantages over batch methods, particularly in handling redundant data. As Bishop and Nasrabadi (2006) explains, duplicating data points simply scales the error function, increasing computational effort for batch methods but leaving online methods unaffected. This also allows for a potential escape from local minima, as a stationary point for the entire dataset may not be stationary for individual data points.

To leverage this property, a momentum method was employed in this study, inspired by Hejazi (2016), which uses a velocity vector to accelerate learning in the direction of persistent reduction in the objective error function.

However, Hejazi (2016) also points out that non-gradient-based methods generally require more iterations to reach a minimum, making them less efficient for calibration processes. Consequently, gradient-based methods for training the DNNs were chosen for this study. While incorporating Hessian information can further improve convergence, it significantly increases computational cost, scaling quadratically or even cubically with the number of weights and biases.

Given the abundant data available for this study and limited input features, the momen- tum-based method, with the NADAM optimiser, was chosen for this study.

2.3.2 Surrogate modelling

Surrogate modelling aims to approximate an expensive deterministic simulator

$$y = f(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d, \tag{2.24}$$

with a computationally efficient approximation $\hat{f}(\mathbf{x})$ constructed from a finite design set

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}. \tag{2.25}$$

In deterministic systems, repeated evaluations at identical inputs yield identical outputs. Consequently, the surrogate modelling problem is fundamentally one of function approximation rather than statistical estimation under observational noise.

While Gaussian Process (GP) models have traditionally dominated deterministic computer experiment literature (Sacks et al., 1989; Santner et al., 2003; Williams and Rasmussen, 2006),

DNNs have emerged as competitive alternatives, particularly in high-dimensional settings (Goodfellow et al., 2016; Lu et al., 2017). This section critically examines the theoretical, methodological, and practical implications of using deep neural networks as surrogate models for deterministic regression.

In stochastic regression, the data-generating process is typically written as:

$$y = f(\mathbf{x}) + \varepsilon, \tag{2.26}$$

where ε represents irreducible noise. In deterministic systems, however,

$$y = f(\mathbf{x}) \tag{2.27}$$

with no aleatory uncertainty.

This distinction has important implications:

- Overfitting must be reinterpreted: perfect interpolation of training data does not imply poor generalisation,
- Classical bias–variance trade-offs are altered because observational noise is absent, and
- Regularisation is motivated by limited sampling density rather than stochastic perturbations.

DNN surrogates therefore operate in the regime of deterministic function approximation, where generalisation error arises from sparse sampling of $\mathcal{X}$ rather than noise contamination. DNNs possess universal approximation properties: a feedforward network with sufficient width can approximate any continuous function on a compact domain (Hornik et al., 1989). More recent results demonstrate that depth can provide exponential representational efficiency relative to shallow architectures (Lu et al., 2017).

DNNs provide significantly greater representational flexibility than polynomial response surfaces or kernel methods with fixed bandwidth when deterministic simulators exhibit:

- Strong non-linearities,
- High-dimensional interactions, and
- Localised sharp gradients.

However, universal approximation is an asymptotic result. In finite data regimes - typical in expensive simulation contexts - network capacity may exceed information content, resulting in unstable extrapolation and poor calibration outside sampled regions.

Modern DNNs are frequently overparameterised, often containing more parameters than training samples. In deterministic settings, this leads naturally to interpolation of training data. Empirical findings in deep learning suggest that interpolation does not necessarily degrade generalisation, a phenomenon associated with the "double descent" risk curve(Belkin et al., 2019).

However, for surrogate modelling of physical systems, this raises concerns:

- Interpolation may capture numerical artefacts of the simulator,
- Overparameterised networks may learn spurious oscillatory behaviour between design points, and
- Lack of structural constraints may violate known physical properties (e.g., monotonicity, convexity).

Thus, while interpolation is theoretically consistent with determinism, unconstrained interpolation may reduce reliability in scientific applications.

A central limitation of standard DNN surrogates is the absence of intrinsic uncertainty quantification. In deterministic modelling, predictive uncertainty reflects epistemic uncertainty due to sparse sampling rather than aleatory variability.

GP provide closed-form predictive variances (Williams and Rasmussen, 2006), whereas standard NNs produce point estimates. Bayesian neural networks (MacKay, 1992) and Monte Carlo dropout approximations(Gal and Ghahramani, 2016) attempt to recover uncertainty estimates, but these methods:

- Are computationally intensive,
- Depend strongly on prior assumptions, and
- May produce poorly calibrated uncertainty bands.

In deterministic surrogate contexts, uncertainty quantification remains one of the principal advantages of GP models over DNNs.

Unlike stochastic regression, replication is unnecessary in deterministic systems. All simulation budget can be allocated to space-filling designs such as Latin Hypercube Sampling(Santner et al., 2003).

The following design qualities impact the sensitivity of DNN surrogates:

- Sparse regions induce uncontrolled extrapolation,
- Non-uniform sampling may bias learned gradients, and
- Adaptive sampling strategies may favour local accuracy at the expense of global fidelity.

Active learning strategies tailored to neural surrogates remain an open research problem relative to well-developed acquisition criteria in GP frameworks. NNs are known to behave unpredictably outside the convex hull of training data. Unlike GPs, which revert toward prior means, DNNs may produce arbitrarily large outputs in extrapolation regions.

For deterministic scientific simulators, this lack of extrapolation control poses substantial risk, particularly in optimisation or risk-sensitive applications.

Architectural constraints, physics-informed neural networks, or monotonic network constructions may partially alleviate this limitation. GPs surrogates scale cubically with sample size, limiting their applicability in moderately large design sets. In contrast, DNN training scales approximately linearly in n per epoch and benefits from Graphics Processing Unit (GPU) acceleration.

Thus DNN surrogates provide superior computational scalability for:

- Large design sizes,
- High-dimensional input spaces, and
- Real-time deployment requirements.

This is similar to this study. However, this advantage is offset by:

- Hyperparameter tuning complexity,
- Lack of closed-form solutions, and
- Training non-convexity and reproducibility concerns.

Therefore, DNNs represent a powerful but methodologically complex class of surrogate models for deterministic regression. Their primary strengths lie in expressive capacity and scalability to high-dimensional problems. Their principal weaknesses involve uncertainty quantification, extrapolation instability, and interpretability.

In deterministic surrogate modelling, DNNs are most appropriate when:

1. Dimensionality is high,
2. Large training sets are available, and
3. Computational efficiency at inference time is critical.

Conversely, GPs may remain preferable when calibrated uncertainty and principled experimental design are essential.

Ultimately, the choice between DNNs and alternative surrogate classes should be guided by problem dimensionality, simulation cost, required uncertainty quantification, and domain-specific structural knowledge.

Gan (2013) pioneered an approach to train a DNN on a selected subset of model points, enabling efficient valuation of portfolios of VA contracts. This significantly reduces training time and is also applicable to the valuation of other products. I used a similar approach for the superior Midway model, but not for the Z1 model.

Bengio et al. (2013) argued that the input data is a key determinant of the success of training a DNN. They used two approaches discussed in the following paragraphs.

Initially, the approach involved partitioning model points based on demographic and monetary attributes, and categorical attributes, then applying the k-prototypes algorithm (Huang, 1998) to minimise the distance between attributes iteratively. However, this method proved slow and potentially unreliable if the number of clusters did not adequately represent the underlying data.

Subsequently, the Kriging method (Isaaks and Srivastava, 1989; Krige, 1951) was adopted, demonstrating significantly faster performance and improved predictive accuracy with an increased number of clusters.

A similar study by Gan and Lin (2015) further explored this approach, utilising an even larger number of clusters. The specific attribute ranges employed in their study are detailed in Table 2.1.

Attribute	**Range**	**Distribution**
Product	[1, 2]	50%, 50%
Gender	[0, 1]	50%, 50%
Age	[20, 60]	Uniform
Account value	[10 000, 500000]	Uniform
Withdrawal rate	[4%, 8%]	Uniform
Term	[10, 25]	Uniform

TABLE 2.1: Kriging example in Gan and Lin (2015)

The Midway model developed in this study utilizes the same Kriging approaches. Including model points on the boundaries were found to be critical to the accuracy of the Kriging method, as detailed in Section 4.2.

2.3.3 Training and validation methodology

Following the approach outlined by Hejazi (2016), a validation portfolio is employed to monitor and prevent overfitting during network training. This portfolio consists of a set of data selected randomly from the input dataset. The MAE of the validation set is tracked throughout training; initially, the MAE is expected to decrease as the network learns optimal parameters. However, as the network begins to overfit, the MAE will typically increase beyond a minimum value.

To mitigate overfitting, the MAE of the validation portfolio is evaluated every Ith iteration. A moving window of length W is applied to the recorded MAE values to identify an increasing trend after the minimum MAE has been reached. If an upward trend is detected within this window, training is halted. Both I and W are user-defined parameters, tailored to this specific application.

Due to the inherent volatility of the validation MAE, a simple moving average with a window size of W is applied to smooth the data. A polynomial of degree d is then fitted to this smoothed data, allowing for a clear identification of the increasing MAE trend after the minimum is reached. W and d are free parameters dependent on the application. This will be discussed further in Chapter 3.

2.3.4 Batch and mini-batch methods

Training techniques can be broadly categorized by how they utilize the training data in each iteration. Batch methods use the entire training set for each update. While conceptually straightforward, batch gradient descent can be computationally expensive and slow due to data redundancy.

To balance accuracy and computational efficiency, mini-batch methods are commonly employed. These methods split the training data into smaller subsets, or batches, and update the model parameters after processing each batch. Three primary variations exist: batch gradient descent, Stochastic Gradient Descent (SGD), and mini-batch gradient descent. Training the ML model in batches is a trade-off between accuracy and time (Ruder, 2016).

Batch gradient descent utilises a fixed partitioning of the data throughout training, performing optimisation on the same subsets for each epoch. In contrast, stochastic gradient descent randomly shuffles and splits the data at the beginning of each epoch. Mini-batch gradient descent

operates on each batch of the split data, offering a trade-off between computational speed and stability.

However, determining an optimal LR for mini-batch gradient descent can be challenging due to the inherent variance within each batch, making it difficult to decide how to adjust the LR during training and when to change it altogether. Furthermore, mini-batch gradient descent is more susceptible to getting stuck in local minima.

Therefore, this study employs SGD with momentum (discussed in Section 2.3.5) to accelerate training. This approach utilises a batch size less than the total number of policies. Batch size refers to the number of policies processed in each iteration. While it can range from 1 to the total number of model points, it is generally advisable to choose a value that is a power of 2 (e.g., 32). Smaller batch sizes increase Central Processing Unit (CPU) utilisation and prolong individual epochs, but can lead to more robust training over fewer epochs. Conversely, larger batch sizes accelerate training but may result in more erratic predictions.

Keskar et al. (2016) observed a 5% decrease in generalisation performance when utilising larger batch sizes of 10% of the input data, compared to a smaller batch size of 256, identifying this as a generalisation gap. Their findings suggest that models trained with large batch sizes exhibit less exploration of the solution space than those trained with smaller batch sizes. Evidence was found to support the use of smaller batch sizes initially, followed by larger batch sizes later in the training process, corroborating previous studies by Byrd et al. (2012) and Friedlander and Schmidt (2012). Qian and Klabjan (2020) subsequently demonstrated that the variance of the LF gradients with respect to the model weights is dependent on batch size, exhibiting higher variance with smaller batch sizes.

Various batch sizes were explored during the model development process detailed in Chapter 3 and Appendix B.

2.3.5 Stochastic Gradient Descent (SGD)

While not directly applicable as a training algorithm for modern neural networks, the Robbins-Monro algorithm (Robbins and Monro, 1951) provides the foundational theoretical basis for stochastic optimisation methods. This work established the principles of update rules incorporating noisy observations and varying step sizes, demonstrating convergence under specific conditions.

Building upon this foundation, Kiefer and Wolfowitz (1952) addressed the practical challenge of unavailable analytic gradients by approximating them using finite differences. This demonstrated that stochastic optimisation remains feasible even in the absence of direct gradient information.

Within the field of ML and pattern recognition, Rosenblatt (1958) presented an early application of incremental updates for adjusting model parameters following the observation of a single (or small batch) training example.

Furthermore, the work of Amari and Saito (1967), particularly his development of the first MLP trained using SGD, is frequently cited as a seminal contribution that explicitly framed parameter updates in an SGD style for models extending beyond simple linear architectures.

During the backpropagation process, the weights and biases are updated iteratively after each epoch. The general formula for updating the weight w and bias b following epoch $(t-1)$ is:

$$w_t = w_{t-1} - \eta_{t-1} * \frac{\partial loss}{\partial w_{t-1}} \tag{2.28}$$

$$b_t = b_{t-1} - \eta_{t-1} * \frac{\partial loss}{\partial b_{t-1}} \tag{2.29}$$

where η_t is the LR at time (t).

When training is conducted using mini-batches and stochastically selected batches, the optimisation procedure is referred to as SGD. This approach introduces variability into the gradient estimates, potentially accelerating convergence and escaping local optima.

To further enhance the optimisation process, momentum can be incorporated. By defining exponentially weighted averages of the gradients as follows:

$$V_{dw_t} = \beta * V_{dw_{t-1}} + (1-\beta) * \frac{\partial loss}{\partial w_{t-1}} \tag{2.30}$$

$$V_{db_t} = \beta * V_{db_{t-1}} + (1-\beta) * \frac{\partial loss}{\partial b_{t-1}} \tag{2.31}$$

where β is a smoothing parameter (typically between 0 and 1), the weight and bias updates can be expressed as:

$$w_t = w_{t-1} - \eta_{t-1} * V_{dw_t} \tag{2.32}$$

$$b_t = b_{t-1} - \eta_{t-1} * V_{dw_t} \tag{2.33}$$

This implementation of SGD with momentum leverages the historical gradient information to dampen oscillations and accelerate convergence towards the optimal solution, and was used in the DNN models derived in this study.

2.3.6 Batch normalisation

Batch normalisation was introduced by Ioffe and Szegedy (2015) to address the issue of training NN models with saturating nonlinearities, or what they called internal covariate shift. Batch normalisation refers to the process of normalising input layers by adding a layer after each connected layer to normalise the inputs passed on to the next layer, which is done for each training mini-batch. This allowed them to use higher LRs and take less care during model initialisation, as well as do away with regularisation and Dropout in some cases. In a practice run, they outperformed models for classification problems by a significant margin and trained in 7% of the number of epochs previously required.

A further study by Wu and He (2018) introduced Group normalisation, which is useful where, for smaller batches, variables are grouped together to determine the mean and standard deviation for a group of inputs rather than single inputs. This overcomes the increase in prediction error experienced when Batch normalisation is used for small batches. This approach was also only tested on classification problems and showed promising improvements over Batch normalisation.

These approaches seem useful, and both were tested during model building in this study. Unfortunately for this regression problem, these approaches did not produce any improvements. In fact, models incorporating these normalisation layers exhibited a larger number of parameters and tended to converge to local minima. A review of the literature revealed limited guidance on effectively employing batch or group normalisation for regression problems, other than another classification study by Zhou et al. (2020) where a further dimension is included in the normalisation and the method called Batch Group normalisation. The scale of differences is particularly critical in regression problems involving financial values. Therefore, this area is left for future research, and these normalisation techniques are not employed in the DNN models ultimately derived.

2.3.7 Layers and number of neurons

The Z1 model employed in this study utilize dense (fully connected) layers implemented using Flux.Dense, defined as:

$$Dense(in => out, \sigma = identity; bias = true, init = he)$$

$$Dense(W :: AbstractMatrix, [bias, \sigma])$$

These layers perform a standard forward pass:

$$y = \sigma.*(W*x.+bias) \tag{2.34}$$

where x is an input vector of length n (or a batch of vectors represented as an in × N matrix), W is the weight matrix, bias is the bias vector, and σ is the activation function.

Keyword arguments allow for disabling the trainable bias (bias=false) and specifying the weight initialisation function (init, defaulting to glorot_uniform for the Midway model). The weight matrix and bias vector can also be provided explicitly.

Determining the appropriate number of neurons per layer is crucial for model performance. As noted by Glorot et al. (2011), representing symmetric or antisymmetric behaviour in data effectively may require a network with twice as many hidden units as a network utilising symmetric or antisymmetric activation functions. This consideration informed the selection of layer sizes in the models developed for this study.

Several researchers have investigated the optimal number of layers and neurons in deep neural networks. Bengio et al. (2013) advocate for deeper networks than strictly necessary, citing the following benefits: increased abstraction from the data, enhanced reusability for other tasks, and the ability to learn higher-order relationships within the data.

Telgarsky (2015) demonstrated mathematically that, for classification problems utilising RELU AFs, a deep network with only two neurons per layer can achieve lower error rates than a shallow network with many neurons in a single layer.

Goodfellow et al. (2016) propose an empirical rule of thumb for determining the number of units in hidden layers: the first layer of a dense block should contain between 1/2 and 1/5 the number of units in the preceding convolutional block, with subsequent dense layers decreasing in size by a factor of 2 or 3. This approach aims to create a "funnel effect," forcing the network to synthesise information and create higher levels of abstraction. This approach can be seen as a form of regularisation before model training even commences (Changpinyo et al., 2017), as it forces the model to condense the problem into smaller dimensions. It therefore removes the need to apply dropout or other regularisation techniques. This was the approach followed in this study.

Mhaskar et al. (2017) studied the impact of network depth and width on binary classification problems. They found that, similar to this study, high-dimensional deep networks trained on large datasets do not necessarily overfit. This leads to an exponential reduction in the number of training parameters and overall model complexity.

Di Lorenzo et al. (2021) summarises the trade-off between shallow and deep networks, arguing that while a network with a single hidden layer can solve a given problem, it may require longer training times and slower convergence. Therefore, they advocate for deep networks, particularly for regression functions with complex features (Wuthrich and Buser, 2021). This is supported by Elbrächter et al. (2021), who demonstrate that increasing the number of layers leads to exponential approximation, rather than polynomial approximation.

2.3.8 Initialisation of model weights and bias

Proper initialisation of model weights and biases is critical for successful training. Several methods have been proposed, each with specific mathematical characteristics:

- Xavier (or Glorot): Normal(0,var) with $var = \frac{2}{n_{in}+in_{out}}$
- He (or Kaiming Uniform): Normal(0,var) with $var = \frac{2}{n_{in}}$
- SNN: Normal(0,var) with $var = \frac{1}{n_{in}}$

A common practice (Hejazi, 2016, p. 61) is to initialise both weight and bias parameters to 0. However, it is often beneficial to initialise biases to a small positive value, such as 0.1, to ensure initial activation of RELU for most inputs and facilitate gradient flow (Goodfellow et al., 2016).

Martens et al. (2010) proposed sparse initialisation, capping the number of non-zero incoming weights to maintain diversity and reduce the risk of saturation.

Glorot and Bengio (2010) cautioned against random initialisation with non-linear activation functions like sigmoid, introducing a new initialisation method to accelerate convergence. This method, known as "Xavier" initialisation, scales the uniform interval based on layer size to prevent sigmoidal saturation, but is only valid for linear activation functions (Glorot et al., 2011).

Sutskever et al. (2013) emphasises the critical importance of initialisation and momentum in deep neural networks, highlighting that random initialisation can negatively impact performance.

A major breakthrough was when He et al. (2015) developed a weight initialisation method, based on a Normal(0, $\frac{2}{n_l}$) distribution (where n_l is the number of neurons in layer l) and zero bias, to overcome convergence issues encountered when training convolutional neural networks with random Gaussian initialisation. They demonstrated that Xavier initialisation is not appropriate for RELU activation functions, leading to the development of "He" initialisation.

Klambauer et al. (2017) proposed SNN as a minor adjustment to He's initialisation.

Finally, Wilson et al. (2017) demonstrated that, for problems with multiple global minima, different algorithms can converge to entirely different solutions even with the same initialisation.

After numerous attempts to find the best method, this study used the Xavier method for the Z1 model and the Kaiming Uniform method for the Midway model.

2.3.9 Dropout

Dropout is a regularisation technique designed to improve generalisation in ML models by randomly excluding neurons (and their connections) during training. Developed by Hinton et al. (2012), this forces "sleeping" neurons to become active and prevents co-adaptation of units. Their initial study utilised 20% dropout in the input layer and 50% dropout in all subsequent layers. Subsequent research has explored varying dropout percentages, particularly in or near the input layer. They also advocated for training multiple networks on the same problem and averaging the results to improve prediction accuracy, especially when using MSE as the loss function, as the squared error of the mean prediction is consistently lower than the mean of the squared errors. They found dropout to be more effective than traditional regularisation techniques, reducing overfitting and accelerating training.

Srivastava et al. (2014) further detailed the dropout technique, demonstrating significant improvements in supervised learning tasks across various domains, including computer vision, computational biology, speech recognition, and document classification. They explain that dropout effectively approximates the averaging of predictions from an exponential number of smaller networks by using a single network with dropout, which leads to smaller (but non-zero) weights, thereby reducing overfitting and surpassing the performance of other regularisation methods.

In this study, despite employing a large network trained on extensive datasets, dropout did not yield any noticeable benefits. Lowering the initial LR and momentum proved to be more effective in achieving better overall performance. No dropout was employed in this study.

2.3.10 Learning rate

The LR dictates the speed at which the model adjusts its parameters in response to gradients during training. A high LR can lead to overshooting the optimal solution, requiring corrective steps and extending training time. Conversely, a low LR results in small steps, potentially prolonging training while cautiously approaching the solution. The goal is to identify an ideal LR, which may change throughout training as the magnitude of gradients varies at each epoch.

Liu et al. (2019) highlights the success of LR warmup heuristics in stabilising training, accelerating convergence, and improving generalisation with adaptive optimisation algorithms like RMSprop and ADAM. They identify that warmup functions as a variance reduction technique in the early stages of training and present both empirical and theoretical evidence to support this hypothesis. They further introduced Rectified ADAM, a variant of ADAM that rectifies the variance of the adaptive LR.

Passos and Mishra (2022) describe a technique for identifying a suitable LR range through testing. By monitoring validation loss over a range of LRs, one can determine the minimum and maximum rates that produce stable solutions-those that allow the model to converge. They suggest using the maximum LR from this range test as the initial LR for stochastic gradient optimisation algorithms like ADAM. They also advocate for employing an LR scheduler, such as "Reduce LR on Plateau," which iteratively adjusts the LR during training. This scheduler monitors validation loss and decreases the LR if no improvement is observed after a predefined number of epochs, facilitating convergence. They recommend using the minimum LR from the range test as the lower bound for the "Reduce LR on Plateau" scheduler, preventing the optimisation process from being hampered by excessively small LRs.

Their experiments, conducted with a defined CNN architecture, utilised an LR range test (for 450 epochs) to determine suitable minimum and maximum LRs. They observed that the model exhibited high validation loss at lower LRs, indicating an inability to learn. Improvement occurred as the LR increased, with validation loss decreasing at a rate of around 0.000001, reaching a stable valley between 0.0001 and 0.3. Values beyond this range resulted in instability and a sharp increase in error. They ultimately chose a maximum LR of 0.0025 and a minimum LR of 0.000001 for use in the "Reduce LR on Plateau" scheduler and the ADAM optimiser, respectively. They implemented a target objective function that combined RMSE and the absolute difference between tuning and calibration MSE, assigning equal weights (0.5) to each term, but acknowledged that other combinations are possible. The second term serves as an adjustment for overfitting.

This study considered various approaches, and the eventual models used are summarised in Section 3.13.

2.3.11 Momentum

Momentum controls the degree of caution the model exhibits when moving in the direction of the largest gradient changes. Typically, studies utilise momentum values between 90% and 99.9%. There are two parameters, one for each of the first and the 2nd derivative of the loss with the NADAM optimiser.

Momentum incorporates a portion of the previous update into the current update, mathematically expressed as:

$$v_t = \gamma v_{t-1} * \eta \nabla J(\theta) \tag{2.35}$$

where:

- v_t is the current update,
- γ is the momentum coefficient,
- η is the LR,
- J is the Jacobian matrix of gradients (∇J is the Hessian), and
- $\theta = \theta - v_t$

This can also be expressed as a change in weights:

$$\nabla w_t = -\epsilon \nabla_w E(w) + \gamma \nabla W_{t-1} \tag{2.36}$$

where γ is the momentum parameter. The weight vector update is therefore dependent on both the current gradient and the previous weight change (Qian, 1999), who mathematically determined the optimal value of γ to be:

$$\gamma = (1 - \sqrt{\epsilon * k_i})^2 \tag{2.37}$$

where k_i is the eigenvalues of a diagonal matrix with positive entries.

The best value of k_i is reached when $k_i = \frac{\mu^2}{4m}$

Sutskever et al. (2013) found it beneficial to reduce momentum to 0.9 during the final 1,000 parameter updates of the optimisation process without reducing the LR. This allows for finer convergence. They also observed that while it may be tempting to use lower momentum values from the outset or reduce them when progress slows, doing so was detrimental to achieving the lowest possible errors. They found that despite potentially appearing stagnant or non-monotonic, the optimisers were performing useful functions over extended periods with higher momentum values. They explain that reducing momentum prematurely can hinder progress

along low-curvature directions, as first-order methods are notoriously ineffective in these scenarios.

This approach leverages the history of movement, allowing for the adjustment of how much of that past movement to incorporate into the next step. Hejazi (2016) set a maximum momentum value of 0.99 and then refined it using their velocity algorithm.

This is further discussed in Chapter 3 and Appendix B.

2.3.12 Optimisers

Optimisers dictate how the model updates its weights during stochastic gradient descent. The gradient represents the first-order derivative of a multivariate objective function, while momentum incorporates information from the second-order derivative.

Appendix C offers further detail on popular optimisers, including their mathematical equations. For a more comprehensive analysis of the advantages and disadvantages of each optimiser, please refer to Ruder (2016), and for an in-depth study, see Kochenderfer and Wheeler (2019), who dedicate an entire book to their examination.

In summary, given that this study focuses on a regression problem, an ADAM-like optimiser (Kingma and Ba, 2014) is most appropriate. Coupled with the need for momentum, as discussed in the previous subsection, the NADAM optimiser (Dozat, 2016) emerges as the clear choice for implementation. Dozat (2016) also listed the following approaches that can be useful to improve model convergence and show that NADAM is superior to all other optimisers used in their study (on the MNIST dataset (LeCun, Cortes and Burges, 1998)):

- adding more layers,
- improving the initial weights in order for errors to be evenly distributed, and
- better regulation techniques.

2.3.13 Activation functions

This study investigated all of the AFs listed in Appendix C, and this section just presents a summary of the eventual AF used.

For a considerable period, the Sigmoid AF was the default choice in training NNs. Subsequently, the TANH activation function gained prominence. These functions made sense for classification problems. However, for modern deep learning neural networks, the RELU AF has become the standard (Goodfellow et al., 2016, p. 195).

RELU is the primary AF used in contemporary supervised regression models. Most research papers reporting state-of-the-art results describe networks utilising RELU. For example, in the landmark paper by Krizhevsky et al. (2012), the authors developed a deep CNN with RELU activations that achieved state-of-the-art results on the ImageNet photo classification dataset (Russakovsky et al., 2015).

This study found that CELU (Barron, 2017) prevented overfitting and trained significantly faster and with less bias than RELU and its variants, such as Exponential Linear Unit(s) (ELU). The models presented in 2.4.1 of this study were therefore trained with CELU AFs for all layers but the final output layer, which was trained as an identity.

2.3.14 Loss functions for regression

This section summarises my findings regarding regression LFs, while Appendix C provides more detailed information on the various LFs. Classical classification LFs are not relevant to this study.

The LF measures the discrepancy between actual values and predicted values, and therefore quantifies how well the model fits the input data and the predicted function. The LR and the derivatives of the AFs are used by the model to update the weights and bias, as outlined in Section 2.3.12. Therefore, it is crucial that the LF is differentiable. The choice of LF depends on the problem's nature and the expected distribution of the target variables. A good LF is suitable for the underlying distribution of the input data. The LF is also known as the objective function or cost function in some literature.

Fissler et al. (2022) aims to improve the accuracy and efficiency of model evaluation by aligning the LF with the underlying data distribution. The study provides insights into the convergence speed of different LFs and highlights the importance of considering feature information in research. They highlight the importance of appropriate LFs.

The MAE LF performed best for this study, meeting the requirement for the model to be accurate on every single data point, rather than, for example, prioritising larger values of y as a shortcut.

2.3.15 Regularisation

Regularisation employs a penalty on the LF to constrain weight changes during backpropagation, aiming to minimise weight magnitudes.

L1 regularisation (also called LASSO regularisation) is a relatively common form of regularisation, where for each weight w_{ij} the term $\lambda \sum_{j=1}^{n} |w_j|$ is added to the LF, where λ is the regularisation strength. It leads to sparse weight vectors, meaning neurons utilise only essential inputs. This makes it more suitable for feature selection, particularly in classification problems. L1 involves absolute values, resulting in a non-differentiable piecewise function without a closed-form solution, and is computationally expensive.

L1 regression in formula terms:

$$\sum_{i=1}^{n}(x_t - y_t)^2 = \sum_{i=1}^{n}(x_t - \sum_{j=1}^{n} w_j * x_j)^2 + \lambda \sum_{j=1}^{n} |w_j| \tag{2.38}$$

L2 regularisation (Ridge regression) is the most common approach, penalising the squared magnitude of all parameters in the LF. That is, for every weight w_{ij} in the network, the term $\lambda \sum_{j=1}^{n} w_j^2$ is added to the LF. It heavily penalises large weights, encouraging lower magnitudes and the use of all inputs, rather than prioritising a few. During gradient descent, it linearly decays each weight towards zero. If explicit feature selection is not needed, L2 regularisation generally outperforms L1.

L2 regression in formula terms:

$$\sum_{i=1}^{n}(x_t - y_t)^2 = \sum_{i=1}^{n}(x_t - \sum_{j=1}^{n} w_j * x_j)^2 + \lambda \sum_{j=1}^{n} w_j^2 \tag{2.39}$$

L2 regularisation will approach L1 regularisation (and result in no regularisation) as $\lambda -> 0$.

A combination of L1 and L2 regularisation is called Elastic net regularisation, adding $\lambda_1 \sum_{j=1}^{n} |w_j| + \lambda_2 \sum_{j=1}^{n} w_j^2$ to the LF.

Hinton et al. (2012) established an upper bound on the L2 norm, limiting weight updates during training. This enabled the use of a large initial LR, reducing it during training, which yielded better results than methods starting with a small, constant LR.

Loshchilov and Hutter (2017) highlighted that L2 regularisation and weight decay are equivalent for standard SGD $\left(\text{when rescaled by the LR}\right)$, but not for adaptive gradient algorithms like ADAM. They proposed decoupling weight decay from the optimisation steps to improve ADAM's generalisation performance and allow it to compete with SGD with momentum.

In this study, regularisation did not have a positive impact, potentially due to:

- few input variables (9 to 12 features),
- the funnel-like decrease in layers in the model architecture,
- batch normalisation with SGD, and
- the choice of the absolute MAE LF.

In particular, L1 regularisation can drive unimportant weights to zero, making it unsuitable for the first layers of DNNs, as all input variables are important. Similarly, the second layer shouldn't employ L1 as most input combinations are crucial.

Poggio et al. (2020, p. 30041) and Zhang et al. (2021, Section 3.1) also found no benefits from using regularisation. Therefore, no regularisation was used in this study. It is hypothesised that the models exhibited increased bias once the best convergence indicators, discussed in Chapter 3, were reached. This contrasts with studies like Kadra et al. (2021), who found regularisation assisted DNNs with tabular data, comparing 40 tabular datasets with modern techniques, and finding Bayesian optimisation was best - though their study focused on classification. The uncertain area of regularisation for regression problems is therefore left for future research.

2.3.16 Metaheuristics

Metaheuristic algorithms offer high-level strategies for hyperparameter optimisation, often inspired by natural processes. Early work in this area included the application of Genetic Algorithm(s) (GA)s to optimisation problems (Razali et al., 2011; Sampson, 1976). More recently, Particle Swarm Optimisation (PSO) has been enhanced with supervised learning techniques to improve search efficiency and prevent premature convergence (Dong and Zhou, 2016). This approach, termed Adaptive PSO with Supervised Learning and Control, models the optimisation process as a control system, dynamically adjusting PSO parameters and promoting diversity through feedback mechanisms.

The rise of Neural Architecture Search (NAS) reflects a broader trend towards automating model design, reducing reliance on human expertise (Ren et al., 2021). NAS algorithms systematically explore architectural configurations, driven by various search strategies and evaluated based on performance metrics. This automation aims to overcome the limitations of manual design and discover optimal network architectures for specific tasks.

These areas are left for future research.

2.4 Machine learning applications in insurance

The integration of ML techniques into the insurance industry has become increasingly prevalent in recent years, offering significant potential for improved risk assessment, pricing accuracy, and operational efficiency. A foundational understanding of the historical development and current trends in this field is crucial. A comprehensive bibliography of research up to 2002, related to the application of NNs, Fuzzy Logic, and GAs within the field of actuarial science and insurance, can be found in Shapiro (2002). The studies they considered encompass a broad range of applications, including risk assessment, pricing, underwriting (classification), claim prediction and analysis, fraud detection, and financial modelling and prediction, across various fields of insurance and settings. This work remains a valuable resource for anyone interested in the intersection of actuarial science, insurance, and soft computing techniques.

Building upon this earlier work, a more recent survey was undertaken by Wüthrich and Merz (2022), reflecting the accelerated development of ML methodologies in the past two decades. Further demonstrating the growing impact of these techniques, an excellent and comprehensive summary of recent developments can be found in a list of 103 ML models developed for the insurance industry by Owens et al. (2022). In Appendix B.1 of that study, spanning pages 35 to 41, the authors meticulously catalogue these models. A further summary of the landscape of ML applications in insurance can be found in De Virgilis et al. (2022).

This section provides a broad context for understanding the application of ML in the insurance industry. In the subsections that follow, the most important studies relevant to this thesis are highlighted, focusing on those that directly inform the methodological approach and research questions explored within this work.

2.4.1 Life insurance business on individual policies

This section focuses on ML models developed for individual life insurance policies. For a discussion of ML applications utilising summarised life insurance data, please see Section 2.4.4.

A significant contribution in this area is the development of a supervised FFNN for valuing the SCR for annuities under Solvency II (Hejazi and Jackson, 2017). This model demonstrates both accuracy and computational efficiency, achieved through a relatively simple architecture featuring a single hidden layer with multiple output layers. The input features were strategically grouped, with each group mapping to a specific neuron within the hidden layer. Several key design choices contributed to the model's performance, including:

- An exponential transpose function employed between the hidden and output layers,

- MSE as the LF, with some caveats as highlighted below,
- Mini-batch training, randomly selecting policies for each iteration to accelerate training time,
- implementation of Nesterov's Accelerated Gradient (NAG) – a technique that enhances learning by incorporating momentum, and
- a Softmax AF applied to the final output layer.

To facilitate faster convergence, the model input features were scaled to the range of 0 to 1. This normalisation addresses the potential for large weight and bias updates, which can occur when dealing with potentially large liability values. Consequently, all input values were normalised to the interval [0, 1] by dividing each value by the maximum observed value for that variable.

The authors successfully trained a convergent model, explicitly acknowledging that continued training beyond the point of convergence would lead to overfitting. A randomised testing set was used during each epoch to assess model generalisation.

A two-step verification approach was employed to determine the optimal stopping point for training. The first step relies on the following criteria:

- A substantial decrease in the MSE of the training data, or
- An initial reduction in MSE followed by an increase in the MSE of the test data, indicating potential overfitting. Training is then continued only until the mean of all predictions on the validation set falls within an acceptable range.

The second step involves monitoring the overall trend of the MSE. They observed that, despite fluctuations, the MSE generally exhibits a decreasing trend. To smooth the data and identify this trend, a simple moving average was applied over m periods, and a polynomial of degree d was fitted to the smoothed data. If the MSE of the validation set increased in $m - 1$ preceding periods after reaching a minimum, training was halted.

How the length of m and size of d were chosen is further detailed in Hejazi and Jackson (2016), which builds upon the findings presented in Hejazi (2016). The overarching conclusion remains that the optimal values for m and d are contingent upon the specific research question and the characteristics of the data. In Hejazi and Jackson (2016), $m = 10$ and $d = 6$ were employed, while Hejazi (2016, p. 137) suggests values of d ranging from 4 to 8, coupled with a value for m that is not excessively large, to mitigate training time.

The primary difference in the approach adopted by Hejazi and Jackson (2016) lies in two key architectural and training choices:

- The utilisation of a hidden layer comprising a number of neurons equal to the number of input features, and
- The implementation of SGD for batch training.

The authors explicitly highlight the absence of definitive guidelines for making crucial architectural decisions, selecting appropriate optimisers and LFs, and determining optimal stopping criteria for training. They reiterate that these choices are fundamentally driven by the research question and the nature of the data.

Furthermore, Hejazi (2016, p. 48) acknowledges that back-propagation is susceptible to the vanishing gradient problem, which can significantly impede the calibration process of multi-layer NNs. Consequently, increasing the number of layers also substantially increases the computational demands of the model.

Doyle and Groendyke (2018) employed DNNs to price four distinct types of VA products. Their approach involved first generating simulated prices using a MC. These simulated prices then served as training data to predict the annuity prices with a desired level of accuracy. The DNN architecture they utilised was as follows:

- 10 input features,
- a single hidden layer comprising 10 neurons,
- 1 output variable representing the price,
- the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimiser,
- the R^2 metric as the loss metric, and
- a training dataset of 2,530 model points, split into 80% for training and 20% for validation.

They found that increasing the number of neurons or layers did not yield significant improvements in performance. Furthermore, they developed a DNN with two outputs, specifically designed for asset hedging purposes.

The resulting models achieved accuracy levels ranging from 98.7% to 99.7%, depending on the specific VA product being evaluated. Notably, the DNN produced results within 1 minute, a substantial reduction in computational time compared to the 8 hours required by the MC. They also observe that employing Kriging, the approach followed by the Midway model developed in Section 3.11, requires longer training and prediction times. They conclude that a marginal loss in accuracy can be tolerated in exchange for a model trained with a simpler, and therefore computationally less demanding, dataset.

Castellani et al. (2018) employed a DNN to value with-profit life insurance business stochastically, utilising seven risk matrices as input features. They achieved accuracy levels exceeding 99.7% with the following DNN architecture, which demonstrated superior performance compared to an MC alternative on several fronts:

- RELU activation function, which outperformed Softmax and TANH AFs,
- Three hidden layers comprising 28, 6, and 7 neurons, respectively,
- A large input dataset consisting of 100,000 real-world stochastic simulations and 10,000 risk-neutral stochastic simulations,
- A training dataset comprising $\frac{2}{3}$ of the total data and a test dataset comprising $\frac{1}{3}$, and
- RMSE as the LF.

They found that Dropout regularisation (up to 20%) did not improve the DNN's performance, a finding consistent with experiences of this study. However, they observed that L1 and L2 regularisation techniques did enhance the DNN's performance. This is likely attributable to the relatively small size of their network, coupled with the multitude of stochastic inputs. The high-level nature of their study unfortunately limits further detailed comparisons, but they emphasise the need for future research in this field and related areas.

Barigou and Delong (2022) developed DNN models for pricing equity-linked life insurance contracts, traditionally valued using stochastic methods. These products feature both death and survival benefits, similar to traditional EA contracts. They employed three distinct DNNs to cater for varying product designs and economic conditions. Their pricing model outputs a range of prices based on premium rates, the number of policyholders, and the force of mortality, allowing for volatility in these parameters with potential 25% upward or downward movements. This resulted in a model requiring six input features.

The architecture of their DNN pricing model was as follows:

- Two hidden layers, each comprising 20 neurons,
- ELU AF, implemented to accelerate the learning process,
- Normalisation using absolute values, scaling the inputs to a range of [-1, 1],
- The ADAM optimizer (Kingma and Ba, 2014),
- MSE as the LF,
- A batch size of 200, and

- 200 epochs.

They explored various hyperparameters but found no improvement upon the aforementioned configuration.

They achieved model convergence with a dataset of 100 policyholders and low interest rates after approximately 100 epochs, after which the model stabilised. They obtained 99% accuracy and found that training the DNN was significantly faster than employing MC.

They also performed sensitivity testing on the model's price outputs to ensure correct functionality, verifying that changes in input parameters (e.g., interest rates) resulted in corresponding changes in the output price, both in direction and magnitude. The model functioned correctly across all tested sensitivities.

Cummings and Hartman (2022) developed DNN models to model claims for long-term care products, comparing different hyperparameters and evaluating performance across various regional areas. They found that RFs produced the best overall results, but also that DNNs represent a significant improvement over GLMs. Notably, they discovered that the best-performing DNN architecture utilised a neighbour-based (or Kriging) approach, and that DNN performance improved with increased training data, albeit at the expense of longer training times. The DNN architecture employed was as follows:

- Implementation using Keras with TensorFlow (Chollet et al., 2015),
- A hyperparameter search algorithm for optimisation,
- Five hidden layers, each utilising the ELU AF,
- Twenty-six policy input variables,
- SGD with batch normalisation and Dropout regularisation,
- He initialization (He et al., 2015),
- A Sigmoid AF for the final layer,
- The ADAM optimizer (Kingma and Ba, 2014), and
- AUC scores as the reporting loss metric.

They justified the use of a Sigmoid AF in the final layer due to the binary nature of the claim modelling task (1 = Yes, 0 = No). For the RF models, they utilised Python with the Scikit-Learn package (Pedregosa et al., 2011).

The similarities between previous studies and this study are given in Table 2.2.

The gaps in the research and areas for future investigation are dealt with in Section 2.5.

Metric	Hejazi	Doyle	Castellani	Barigou	Cummings	Z1	Midway
Programming	Python	Python	Python	Python	R + Python	Julia	Julia
Product type	VA	4 VAs	With Profit (WP) WL	Unit Linked (UL) EA	Long Term Care (LTC)	WL	WL, EA, TA
Architecture	ANN	ANN	DNN	ANN	RF + DNN	DNN	DNN
Training/ Testing		80/20	67/33			75/25	27/3/70
Hidden layer neurons	[Input, Output]	[10,1]	[28,6,7,1]	[20,20,1]	5 total	[72,36, 18, 9,1]	[128,64, 32, 16,11]
Output	Classification	Pricing	Regression	Pricing	Classification	Regression	Regression
Data batching	Mini-batch				SGD	SGD	SGD
Data normalisation	[0,1] by division			Minimax		Gaussian	Gaussian
Input variables	Grouped					Ind Policy + Basis	Ind Policy + Basis
Loss function	MSE	MSE	RMSE	MSE	X-entropy	Abs. MAE	Abs. MAE
Activation	ELU/ Softmax		RELU		Sigmoid	CELU	glscelu
Optimiser	NAG	BFGS		ADAM	ADAM	NADAM	NADAM
Regularisation	Yes	Yes	No		Yes	No	No
Performance metrics	Confusion Matrix	R^2			AUC	Abs. MAE	Abs. MAE
Sensitivities test	No	No	No	Yes	No	No	Yes

TABLE 2.2: Comparison of DNNs over various previous studies and this study

2.4.2 General insurance studies

It should be noted that pricing and valuing general insurance is typically easier than with long-term life insurance. This is primarily due to the shorter term of most general insurance policies (typically less than a year). While the number of risk factors may be higher, these factors generally exhibit a readily observable relationship with premium rates, making them amenable to ML. Furthermore, reserves are often calculated on an Incurred But Not Reported (IBNR)

basis, which can be readily addressed with techniques like run-off triangles. There are still some principles that are important for this study, which will be highlighted below.

A review of ML applications in P&C insurance was published by Blier-Wong et al. (2020). They demonstrate how NNs, as developed in this study, can offer advantages over traditional GLMs and summarise likely future trends for model development in the industry. Section 1 of their study provides a useful history of methods developed over time. They also observe that ML has initially focused on pricing, due to its closer relationship with GLMs, with valuation models being developed more recently. They noted that XGB was the most popular technology employed for pricing, while published research on supervised regression NNs remains limited. Conversely, for reserving, NNs are the most popular approach (Blier-Wong et al., 2020, Figures 2 and 3, p. 5-6).

A critical consideration highlighted by Blier-Wong et al. (2020, p. 18) is the need to distinguish between variables that remain constant over time and those that change. In this study, age is the only variable that changes over time, but this is partially captured in the premium calculation, which is assumed constant, and the smoothness of the AM92 mortality table. A direct relationship between age and reserve will also be accounted for by the LF. Nevertheless, consideration should be given to how to deal with age as a dynamic variable. The authors conclude that implementing and developing these models requires a combination of ML and reserving expertise, which is a scarce skill set.

A survey by Taylor (2019, Section 6) considered the development of ANN models for valuing general insurance business reserves. They found that ANNs can perform well on large datasets with complex structures, noting that development in this area has accelerated in recent years, citing contributions from Wüthrich, Ahlgren, and Gabrielli. They also cautioned against the curve-fitting nature of ANNs, which could lead to extrapolation errors – a point to be carefully checked during model validation.

Addressing the issue of dynamic variables, Llaguno et al. (2017) suggested grouping data into similar clusters and building separate models for each cluster. However, this approach is not applicable in this study, as each age has a unique q_x (see Section 3.6), preventing the creation of meaningful clusters. This remains a topic for further consideration.

Models with a large number of neurons per layer are known to be difficult to calibrate, and it is often preferable to use more layers with fewer neurons each for better regression performance (Gabrielli and V. Wüthrich, 2018, p. 6-7). This reasoning underpinned the design of the network architecture employed in this study.

Finally, a creative approach to individual claims reserving was proposed by Baudry and Robert (2019). While not focused on ML, their methodology for building training and testing databases offers opportunities for future research utilising various ML models.

Wüthrich et al. (2016) provides a good introduction to individual claims reserving with ML. This paper presents a toy example using regression trees, with several simplifications, to introduce and demonstrate the concept of applying ML to individual claim reserving. The author notes that, despite presenting a simplified model, generalising it to a full real-world problem would not be difficult.

Kuo (2019) employed Mean Absolute Percentage Error (MAPE) and Root Mean Squared Percentage Error (RMSPE) as LFs, utilising the AMSGrad optimiser with a LR of 0.0005. Each model was trained for a maximum of 1000 epochs, but training was halted if no improvement was observed on the validation dataset for 200 epochs, selecting the model with the lowest validation loss. To mitigate the impact of random weight initialisation, all models were trained 100 times, and the average results were taken.

De Virgilis and Cerqueti (2020) analysed a dataset of fully developed claims to predict ultimate costs using various ML techniques. The authors highlight the strengths and weaknesses of each method, aiming to predict the ultimate cost of claims at the time of reporting, alongside intermediate cash flows before claim closure.

Wüthrich (2020) focused on general insurance pricing. They implemented a DNN with 2-dimensional embedding layers for categorical variables (brand and ct), modelling age, ac, power, dens, area, and gas as continuous variables. The network consisted of three hidden layers with 20, 15, and 10 neurons, respectively, employing TANH AFs. They used the R programming language. The network contained 780 parameters and was trained using the NADAM optimiser (Dozat, 2016), as further described in Sections 8.3.3 and 8.5.3 of Goodfellow et al. (2016).

Gabrielli (2021) explores the use of ML, specifically NNs, to improve the accuracy of Individual Claims Reserving (ICR) in the insurance industry. Traditional methods often focus on aggregate data, while ICR aims to predict the ultimate cost of each individual claim. This is crucial for more precise financial reporting and risk management. The paper contrasts traditional actuarial methods (chain ladder, etc.) with modern ML techniques. It argues that ML can provide more granular and potentially more accurate predictions, especially when dealing with complex claims data. They use several techniques: FFNNs, ResNets, and Dropout, while emphasising the importance of proper data preprocessing, including feature scaling and handling missing values, and claim that appropriately designed NNs can outperform traditional actuarial methods in ICR.

Schelldorfer and Wuthrich (2019) utilised a GLM as a starting point for a CANN on the Casualty Actuarial Society (CAS) motor insurance dataset, to reduce computational time. They discussed the use of embedding layers and representation learning as efficient methods for handling categorical variables. The network consisted of three hidden layers with 20, 15, and 10 neurons respectively, using 9, 11, or 40 input variables (including categorical dummy variables),

with a similar architecture to Noll et al. (2020). While they found embedding layers to be preferred over dummy and one-hot encoding, this came at the cost of longer running times, volatile frequency estimates, and fluctuations in bias. They reported substantial reductions in training time for the CANN model, achieving comparable accuracy to other models tested, and providing methods to systematically identify weaknesses in other regression models.

In a study on the motor claims on the CAS dataset by Noll et al. (2020), they ensured that the training and test datasets are similar with regard to the number of claims and claim frequency. This is another way to ensure your model generalises well. They chose the Poison LF since they used a Poisson GLM to model the number of claims as a Poisson distribution. They also used several categorical variables on which they used dummy coding for ML purposes. Details of their ANN:

- Single hidden layer with 20 neurons,
- Batch size of 10,000,
- Linear transformations (because their data is not distributed Gaussian) transformed to [-1, 1] range,
- SGD optimiser,
- Sigmoid AF, and
- LR scheduling (decreasing) with constant momentum.

Ferrario et al. (2020) extended this work using the same CAS dataset, incorporating an additional input variable for volume. Their study investigated network architectures with three hidden layers comprising 20, 15, and 10 neurons, respectively. Several improvements over Noll et al. (2020) were implemented, including the use of the TANH AF for the hidden layers - motivated by its larger gradient change near the origin - and the ELU AF for the output layer. Optimisation was performed using the NADAM optimiser.

Important findings from their study include:

1. The NADAM optimiser performed the best on the network with 1 hidden layer, and they then used it throughout.
2. The lowest loss was noticed using a batch size of 1,221 (over an input dataset of 610,212 policies).
3. Or a batch size of 6,103 when the run time was kept constant at +-90 seconds.

4. Their conclusion was that a batch size of 4.5% to 5.5% of the input dataset should be appropriate.

5. Using larger batch sizes leads to improved run time over the cost of accuracy, and vice versa.

6. They found small gains when using stratified batches and under- or over-sampling, regularisation, dropout and batch normalisation, although each incurred a computational cost.

The efficacy of the TANH AF was attributed to appropriate data normalisation and the exclusion of extreme values, given the Poisson nature of the problem. This allowed for the use of randomised Gaussian initial weights. The researchers also proposed a valuable heuristic: increasing the number of layers is beneficial for capturing sum-like feature interactions, while increasing the number of neurons within a layer enhances the modelling of product-like feature interactions. In their terms, network width relates to individual feature components, and depth reflects the complexity of interactions between those components. Given the prevalence of product-like terms in the underlying policies, an initial layer of 72 neurons was selected. Furthermore, they explored ensemble learning, which yielded slightly improved results.

Lorentzen and Mayer (2020) advocates for the use of Explainable Artificial Intelligence (XAI) techniques to understand and trust models used for insurance pricing, while also maintaining predictive accuracy and fairness. They stress the importance of not only evaluating the predictive performance of a model but also its interpretability and fairness. They, however, stopped short of finding, like this study did, that sensitivity is a valuable alternative to describe how a model works.

2.4.3 Mortality studies

Richman and Wuthrich (2019), Richman and Wüthrich (2021) and Perla et al. (2021) investigated the application of deep learning to multi-population mortality modelling, extending the traditional Lee-Carter model (Lee and Carter, 1992) using Keras-implemented NNs in R Keras. Their results demonstrated a significant performance improvement compared to conventional models, even when applied to sequential data. These works provide a relevant precedent for including a mortality basis in the DNN inputs when calculating reserves.

Gabrielli et al. (2020) investigated how NN features can lead to improvements in general insurance claims reserving. A classical actuarial regression model was embedded into a NN architecture. They were able to improve the prediction accuracy of the resulting architecture.

2.4.4 Tabular and summarised data studies

Several deep learning architectures have been applied to tabular data, including NODE (Popov et al., 2019), Net-DNF (Katzir et al., 2021), and Tab-Net (Arik and Pfister, 2021). Within actuarial science, Richman and Wüthrich (2022) developed localGLMnet, a model specifically designed for tabular claim data.

Net-DNF employs a fixed four-layer structure with a unique training approach. The first hidden layer is trainable, while the second is a binary layer, and the final layer is fixed. Training utilises the sign activation function with TANH for backpropagation. In classification tasks, the authors report optimal performance with a batch size of 2048, an LR of 0.05 employing an LR scheduler, a maximum of 1000 epochs with early stopping based on validation score over the last 30 epochs, SoftMax and Sigmoid cross-entropy AFs, and the ADAM optimiser.

Richman and Wüthrich (2022) presents a novel approach that integrates two fitted GLMs – utilising Uniform and Gaussian distributions – as input to a final output layer. This 'local-GLMnet' model builds upon previous work with CANN models. Designed for predicting total claim counts, its regressive nature lends itself to the use of the Poisson loss function and the ELU activation function, aligning with the desired distributional properties. A key strength of this approach is its interpretability, allowing for explanations of the model's predictions in GLM terms. However, given the primary goal of validating the predictive power of my models – rather than explaining existing reserve calculations – I deemed its incorporation unnecessary at this stage. While confirming model accuracy is valuable, it does not offer additional insight beyond the evaluation metrics already employed. Therefore, this area is a promising avenue for future research.

Bauer et al. (2012) proposed an MC approach for calculating SCR on summarised data for insurance business. The focus was on practicality, seeking to simplify the often-complex prescribed SCR calculations. The authors enhanced model reliability by restricting the simulation to likely outcomes, excluding extreme values, and thereby mitigating the impact of unreliable data points.

2.4.5 Applicable studies in other machine learning fields

The importance of this section and the previous one is to indicate that there are many diverse approaches in the NN field. These sections highlight some of the studies that have some similarity with the DNN models developed in this study.

Albelwi and Mahmood (2017) developed methods to design CNNs frameworks, selecting hyper-parameters, without requiring expert manual tuning. They also improved the loss function by

combining the error rate with information content of learned feature maps. By tuning these hyperparameters, they were able to obtain comparable results to the state-of-the-art CNNs at that time on the MNIST dataset and competitive results on the CIFAR-10/100 datasets. Their work inspired later works in NAS and XAI.

Cui and Fearn (2018) proposed an LR of $0.01 * \frac{BATCH}{256}$, clearly indicating the need for a relationship between the LR and batch size.

Olmschenk et al. (2019, p. 3) provide two semi-supervised GAN approaches. The techniques they employed demonstrate potential for generating synthetic data or augmenting existing datasets, which could be valuable for addressing data scarcity or improving model robustness.

Di Lorenzo et al. (2021) applied RNNs to value products for reverse mortgages, demonstrating their applicability to complex financial modelling tasks. Similarly, Lishner and Shtub (2022) successfully projected durations for engineering projects using an ANN, and even deployed a web application for instantaneous estimations based on the model's predictions.

Eckerli and Osterrieder (2021) and Anderson et al. (2021) represent additional areas of investigation. Anderson et al. (2021) recommended an ANN with a single hidden layer of five neurons, employing Sigmoid AFs and decay values between 0.01 and 0.0001.

Passos and Mishra (2022) details a comprehensive approach to model training, employing L2 regularisation, ELU AFs, the ADAM optimiser, and techniques for improved stability and reduced overfitting - specifically, the "Reduce LR On Plateau" scheduler and "Early Stopping". Their CNN architecture incorporates a single convolutional layer followed by three dense layers (128, 64, and 16 units, respectively), with a linear activation function in the output layer for regression tasks. They systematically optimised hyperparameters such as L2 regularisation strength, batch size, and LR, demonstrating a rigorous approach to model development.

This selective group of studies collectively demonstrate the versatility of neural networks across diverse applications, highlighting a range of architectures, training techniques, and optimisation strategies that assisted with the rigorous development of models for this study.

2.5 Research opportunities and motivation

2.5.1 Explainable artificial intelligence

Deep learning models often produce outputs as a series of weights (and output AFs), lacking the intuitive interpretability of traditional formulas or distributions. This opacity poses significant challenges regarding accountability, compliance, and auditability, particularly when decisions

yield adverse outcomes. The inability to reverse-engineer a model's reasoning hinders effective oversight and can be analogous to relying on "intuition" without a justifiable basis (Ming, 2017).

This research recognises the critical need for explainability in ML, particularly within the regulated insurance industry. While assessing the general performance of a model is important, understanding why a specific prediction was made is essential for auditability, regulatory compliance, and building trust in the model's outputs. The varying of policies and bases undertaken in this research aligns with the concept of 'local interpretability' – investigating model behaviour through targeted variations of input data to diagnose specific decisions.

The field of XAI has emerged in recent years (Carvalho et al., 2019; Du et al., 2019; Payrovnaziri et al., 2020) to address these challenges. Payrovnaziri et al. (2020) categorised existing XAI techniques into five key areas: knowledge distillation and rule extraction, intrinsically interpretable models, data dimensionality reduction, attention mechanisms, and feature interaction or importance. This research falls primarily under the category of 'data dimensionality reduction', as it aims to reduce a high-dimensional input space (multiple variables) into a single predicted reserve valuation.

Arik and Pfister (2021) demonstrate that a deep tabular data learning architecture, TabNet, outperforms other models on a variety of non-performance-saturated tabular datasets, yielding interpretable feature attributions and insights into its global behaviour.

Owens et al. (2022) provides a valuable benchmark, ranking 103 state-of-the-art insurance ML models based on their explainability and simplified representation. Their work also offers an updated definition of XAI, highlighting its importance in the insurance context and providing a foundation for evaluating the explainability of developed models. This study builds upon the foundation laid by Owens et al. (2022) and leverages their insights to enhance the transparency and trustworthiness of the proposed valuation model. This was highlighted in subsection 2.4.

A truly XAI system should disclose: its strengths and weaknesses; the criteria used to arrive at a decision; the rationale behind a specific prediction; the appropriate level of trust for various decisions; potential error types; and methods for error correction.

Techniques such as Layer-wise Relevance Propagation (LRP) (Montavon et al., 2019) offer promising approaches for achieving explainability in DNNs. LRP operates by propagating the prediction backwards through the network, highlighting the input features most influential in the decision-making process. Since we already know the exact way reserves are calculated, this is left for future research.

The increasing importance of XAI is further underscored by the emergence of regulatory bodies and academic conferences dedicated to AI ethics and transparency, such as the Association

for Computing Machinery Conference on Fairness, Accountability and Transparency[2]. This highlights the growing societal and regulatory demands for transparent and accountable AI systems.

Richman et al. (2019) states that DNNs are likely to be less transparent than traditional actuarial models, and DNNs may include unwanted bias. They suggest DNNs must undergo peer review and output from a DNN must be compared to the output of a traditional model, as an important risk control measure. This is exactly the approach followed by this study in Chapter 4.

The reliability of research findings is a crucial concern within the scientific community. Yang et al. (2020) highlights the significant issue of replicability, noting that many published studies fail replication attempts. This underscores the need for robust and verifiable methods for assessing model behaviour and ensuring the accuracy of predictions.

This study contributes a novel approach to address these concerns within the field of DNN explainability. By providing output based on different valuation bases, this methodology offers a clear and interpretable framework for understanding the model's inner workings. This allows for rigorous validation of the model's accuracy and reliability, ultimately facilitating approval by auditors and regulators. This provides a verifiable and transparent alternative to more opaque "black box" models, and directly addresses the need for robust, replicable results in ML research.

2.5.2 Little research in life insurance individual policies

As previously discussed in Chapter 1 and the preceding sections of this chapter, individual policy data is inherently scarce due to various privacy and competitive factors. Consequently, detailed academic research focusing specifically on this area remains limited. A search of prominent actuarial data science resources, including `actuarialdatascience.org`, `insurancedatascience.org`, and `actuarial-academy.com`, reveals a notable absence of publicly available datasets or published studies pertaining to individual life insurance policy valuation.

Beyond the studies outlined in this chapter, a comprehensive literature review has not revealed any other publicly available research specifically addressing the valuation of individual life insurance policies. This suggests a significant gap in the existing body of knowledge.

Furthermore, to the best of my knowledge, no research has been published employing DNNs on the specific dataset utilised in this study, which is detailed in the following chapter. This represents a novel application of DNNs to this particular data, potentially offering new insights and approaches to individual life insurance policy valuation.

[2] `https://facctconference.org/` (founded in 2018)[Accessed August 2025]

2.5.3 A rapidly evolving field

The application of ML within actuarial science is a rapidly evolving field, with an increasing number of resources becoming available to practitioners. The Swiss Association of Actuaries maintains a valuable online repository of resources for actuaries working with ML, including case studies, tutorials, and strategic guidance[3].

This dynamic nature is further underscored by Ren et al. (2021, p. 111-112), who notes that "Due to the deepening and rapid development of NAS-related research, some methods that were previously accepted by researchers have proven to be imperfect by new research, leading to the development of an improved solution". This highlights the iterative and constantly refining nature of ML research and the importance of staying current with the latest advancements.

2.6 Conclusion

The use of ML in insurance is a growing area of research, and there is an increasing body of literature in this field. This review has highlighted the potential benefits of ML techniques for various insurance applications, particularly in areas such as pricing, risk assessment, and claims prediction. However, it has also revealed significant gaps in the current research, specifically concerning the valuation of individual life insurance policies. While the broader field is evolving rapidly, detailed studies focused on individual policy data remain scarce due to inherent data limitations and privacy concerns.

The literature emphasises the critical need for explainability and transparency in ML models, particularly within the highly regulated insurance industry. The increasing focus on XAI reflects a growing recognition of the limitations of "black box" models and the importance of building trust and accountability into these systems. This research addresses this need by employing a novel approach to model validation - analysing output based on differing valuation bases - to provide a clear and interpretable framework for understanding model behaviour.

Furthermore, while a range of ML techniques have been explored within insurance, the application of DNNs to the specific dataset used in this study represents a novel contribution to the field. This research aims to bridge the gap in existing literature by providing a detailed exploration of DNNs for individual life insurance policy valuation, contributing to a deeper understanding of their potential and limitations in this context. This work contributes to that iterative process, offering new insights and paving the way for future advancements in the application of ML to individual life insurance.

[3] https://www.actuarialdatascience.org/

Chapter 3

Methodology

3.1 Introduction

This chapter details the methodology employed to construct and evaluate the DNN models utilised in this study. The process is structured into distinct stages: notation and definitions, experimental setup, data preparation, base valuation scenario development, model construction, model evaluation, and accuracy assessment. This chapter outlines the procedures used to train and validate the models, while the results obtained on the independent test dataset are presented in Chapter 4.

The overall goal is to have two different models - designated Z1 and Midway - by assessing their ability to predict actual reserve values. This comparative analysis will enable a ranking of the models based on predictive accuracy and generalisation performance on unseen policy data, varying valuation bases, and diverse product types.

The final architectures of the Z1 and Midway models were given in the last two columns of Table 2.2.

3.2 Hardware and software specifications

This study did not require extensive computational resources; however, more powerful hardware would be necessary for developing and training significantly more complex models.

The following hardware configuration was utilised for model development and execution:

- Intel Core i5 with 4 CPUs at 3.5Ghz,
- NVidia GeForce GTX 1650 with 4GB VRAM and 896 CUDA cores, and

- Dual RAM 8GB at 2.4GHz.

The following software tools were employed:

- Julia v1.8.3,
- Julia libraries, mostly Flux v0.13.13, CUDA v4.0.1 and Makie,
- Microsoft Office,
- Visual Studio Code 1.76.0, and
- Windows 11.

The selection of Julia as the primary programming language was deliberate, driven by its capacity for high-performance computing and its efficient utilisation of parallel processing capabilities through CUDA cores[1]. This allows for the efficient implementation of complex algorithms without requiring developers to resort to lower-level coding. Julia's compilation to machine code results in faster execution speeds and facilitates more extensive model investigations.

Furthermore, Julia represents a more recent advancement in programming languages, offering a cleaner and more flexible architecture compared to the established ecosystems of Python (Van Rossum and Drake, 2009) and R (Besard et al., 2018). While Python and R often rely on extensive legacy libraries, Julia enables easier implementation of custom algorithms from first principles. Additional discussion regarding the benefits of Julia can be found on the JuliaActuary[2] blog.

Although R was utilised during initial model development stages, the reliance on libraries such as Keras (Chollet et al., 2015), TensorFlow (Abadi et al., 2015), and H2O (Candel and LeDell, 2022; Candel and Malohlava, 2022; Landry, 2022; LeDell et al., 2023) introduced limitations on control and customisation. Moreover, R's execution speed proved comparatively slower than both Julia and Python.

All code developed for this study, beyond the utilisation of Julia libraries, is original and bespoke, designed to train DNNs incorporating the considerations outlined in Section 1.2. Two specific DNN architectures, designated Z1 and Midway, are presented in the body of this study. Z1 represents a model a practitioner would typically train using standard data and valuation bases, while the Midway model is trained using Kriging principles and incorporating additional pricing bases, 3 additional categorical variables, and thousands of additional valuation bases.

[1] https://www.trustedreviews.com/explainer/what-are-cuda-cores-4226433[Accessed 1 March 2023]
[2] https://juliaactuary.org/blog/julia-actuaries/[Accessed 1 March 2023]

The models developed are stored in BSON files, ensuring portability and interoperability with Python, R, and other programming languages. Utilisation of a GPU is highly recommended to accelerate model training.

The largest model, trained on 6.5 million policies, achieved convergence on a GPU within 30 minutes. Once trained, the models can be deployed on a CPU to predict reserves for millions of policies in under one second.

3.3 Investigative process

Developing accurate models necessitated a systematic and iterative process, often involving extended periods of investigation to optimise model training parameters. The challenges encountered and solutions implemented are detailed in this section and further documented in Appendix B.

A logical and incremental approach, analogous to traditional actuarial valuations, was adopted, modifying only one model hyperparameter at a time and carefully documenting the resulting changes in model performance. To facilitate comparison and validation, traditional actuarial valuations were performed alongside predictions generated by the DNN models, resulting in an 'Actual versus Expected' (scatter plot, see next section) analysis to draw accuracy conclusions.

3.3.1 Visualisation of Goodness-of-Fit metrics

To assess and illustrate the accuracy of the developed models, the following visualisation techniques were employed:

- Scatter plots (or called calibration graphs), and
- Performance metrics presented in tabular format.

Scatter plots display predicted values on the y-axis against true values on the x-axis. A high degree of accuracy is indicated by a strong correlation and clustering of points along a 45-degree angle.

Further issues addressed by scatter plots, typically ignored when not shown, include:

1. Do errors increase as actual values increase (suggesting heteroscedasticity)?
2. Systematic curvature (suggesting non-linearity).

3. If the scatter is widely spread, predictions may have large variance/errors even if mean error is low.

4. A scatter plot assesses whether a model systematically over- or under-predicts.

Quantitative measures of goodness-of-fit, specifically the percentage accuracy calculated as Actual values divided by the Predicted values, will be given as percentages in conjunction, to assess overall accuracy.

3.4 Data for training and testing overview

Section 2.4 provided context for the application of ML to actuarial valuations. The following sections detail the specific data utilised in this study. The primary objective is to develop a model capable of generalising to unseen data, requiring a training dataset that captures the underlying characteristics of the environment rather than containing every possible data point. A smooth output and accurate consideration of all input variables are critical for achieving this goal. Identified weaknesses in predicting reserves on the test data was addressed by augmenting the training set with additional examples from those specific areas, for the Midway model, but not for the Z1 model.

A range of valuation bases will be employed in both training and testing. The training and test datasets will comprise a diverse range of typical policies, ensuring a balanced distribution across all variables. The complete original dataset, with actuarial calculations performed across multiple pricing and valuation bases, will serve as the test data. A detailed description of this dataset is provided in the following subsections.

3.4.1 The dataset used

The data utilised in this study is sourced from Lledó and Pavía (2022), comprising 76,102 records of individual whole life insurance policyholders of a European commercial insurer from Spain as of December 31, 2009.

It is important to realise that the original dataset (after employing pricing and valuation bases, and splitting into training, validation and test datasets) was used for training of the Z1 model, but that it was used exclusive as the test dataset for the Midway model. The Midway model was training on Kriging principles. This is outlined in Sections 3.8 and 3.9.

For the purposes of this study, it was assumed that the SA for each policy remained constant throughout the policy term, and that monthly premiums, calculated as described in Section

3.6.5, also remained constant. Furthermore, it was assumed that the insurance company performed annual valuations as of December 31st of each year.

The insurance market environment in 2012, approximately one and a half years after the dataset's date, can be reviewed in a regulatory assessment titled "Spain: IAIS Insurance Core Principles: Detailed Assessment of Observance"[3]. The eventual pricing dataset, per policy, is available in Blomerus (2023).

3.4.2 Data cleaning

Prior to analysis, the data underwent a cleaning process to identify and correct any inaccuracies or inconsistencies. No missing values were detected, and no duplicated records were present. However, five policyholders were identified as being less than 17 years of age at inception, which posed a challenge as the mortality tables used for pricing and valuations only commence at age 17. To address this, the starting age of all policies with an inception age less than 17 years was adjusted to 17 years. This adjustment is considered negligible and reflects a reasonable approach that would be employed in practice. A consistent approach was followed throughout the data preparation process.

The quality and thoroughness of the data cleaning performed by the original data producers are commendable.

3.4.3 Data analysis and summary

The pricing and valuation of a whole life policy necessitate only the age, gender, SA, product code and TIF of the policyholder. The TIF is required to determine the inception date, which is crucial for establishing the premium based on the age at inception, and for calculating surrender rates, if applicable.

Following data cleaning, a comprehensive data analysis was performed, focusing on variables of particular relevance to this study. For further analysis of the dataset, please refer to Lledó and Pavía (2022).

Table 3.1 provides a summary of these variables provided in the data:

The youngest was aged 17, the eldest being 78 years and nine months. The SA values differed significantly, from very small policies to very large ones. Most of the policies were smaller than the average.

[3] `https://www.imf.org`[Accessed February 2023]

	Age	SA	TIF	Sex
Minimum	17	7 000	0	0
Maximum	78.75	3 010 000	118	1
Mean	44.36	91 452	41.5	0.63

TABLE 3.1: Data Summary

It can be seen that the business has written WL policies for almost ten years and is still writing new business.

About 63% of the policyholders are male, and 37% are female.

The distribution of the variables is given in Figures 3.1, 3.2, and 3.3.

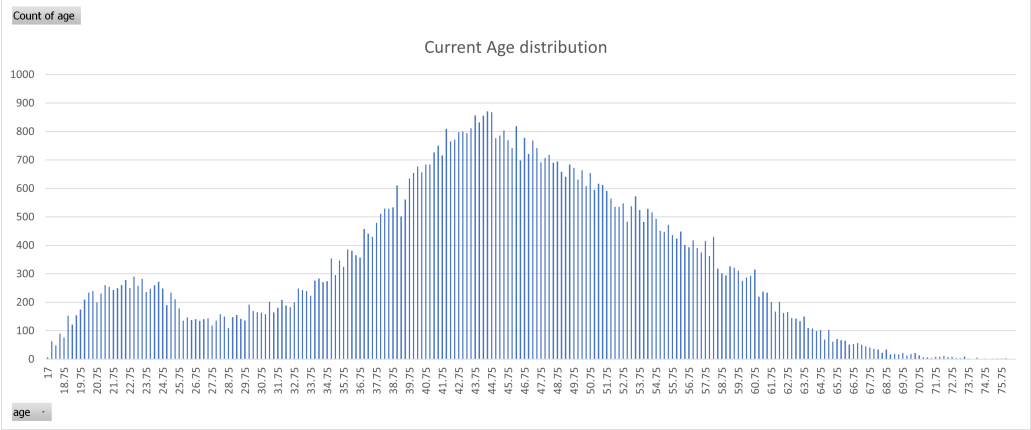

FIGURE 3.1: Age distribution of dataset

There is a slight hump at young ages. Therefore, this company may be successfully targeting younger policyholders who are less likely to die. There are very few older policyholders. Consequently, the book is not yet mature. In a statistical sense, more data is to the left of the average, whilst a Gaussian distribution could approximate the rest.

There is a wide range of SA, with most values between 50,000 and 110,000 Euros. It is essential to notice that the data is not skewed.

There are more recent policies than older policies. This indicates that the insurer is growing its book faster than ten years ago. Alternatively, there could have been surrenders of policies in force for extended periods, which skews this conclusion. A log-normal distribution would be a good fit after removing the few policies with short TIF.

A further comment is that there is very little business with terms of 4 months or less. The reasons might be any of the following:

- the insurer is still busy capturing policy details on the system, or

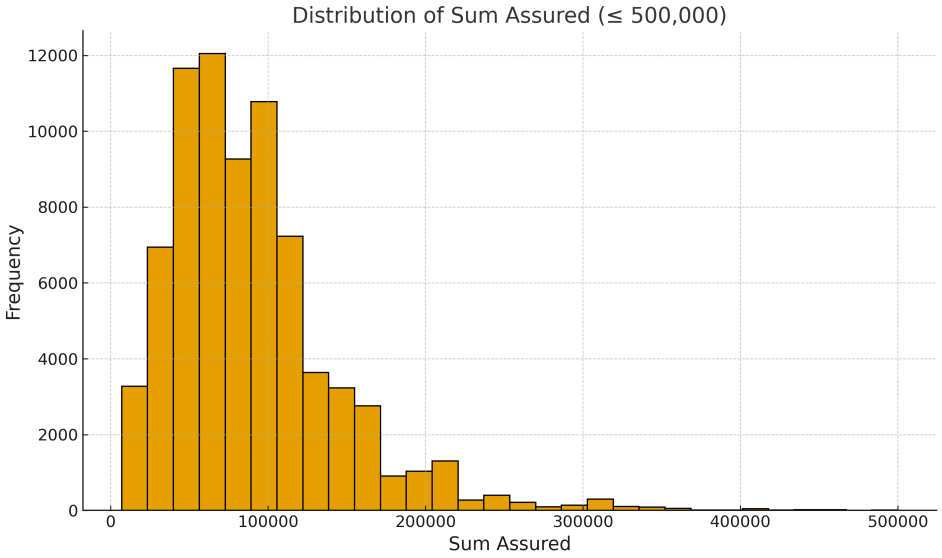

FIGURE 3.2: SA Distribution of dataset

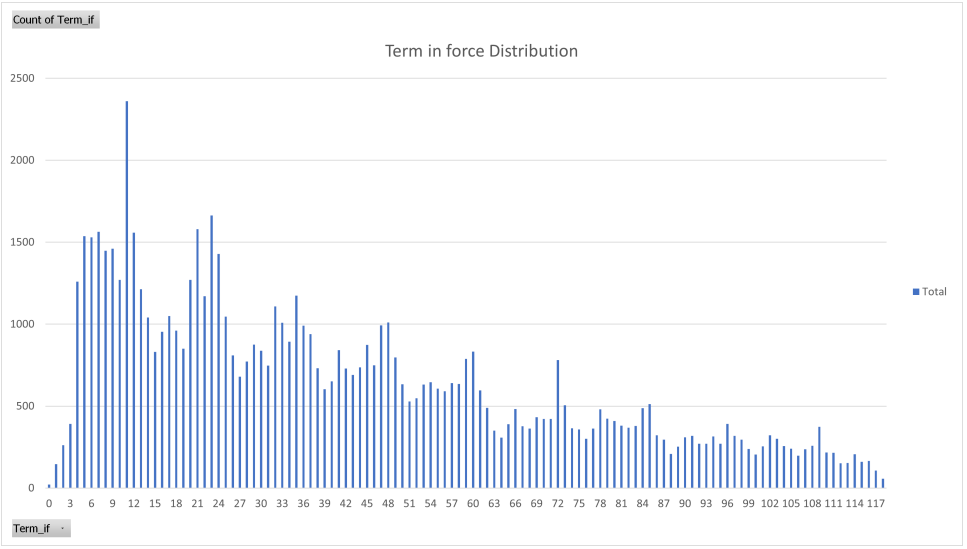

FIGURE 3.3: TIF distribution of dataset

- the insurer uses a cut-off period when performing annual valuations.

If the insurer is not using a consistent approach, the impact is that this insurer will likely be under-reporting its levels of new business and NBS.

Gender is a categorical variable, but since this study uses the AM92 and SIM92 mortality tables, all genders are considered equal. The distribution of gender is given in Table 3.2

Gender	Count	Percentage
Female	28,450	37%
Male	47,652	63%
Total	76,102	100%

TABLE 3.2: Gender distribution

3.5 Valuation bases

A Valuation Basis (VB) is the set of assumptions for mortality, investment, surrender, and expense assumptions.

For mortality, multiples of the AM92 and SIM92 tables were used. Graphically, the q_x values of these tables are shown in Figures 3.4 and 3.5.

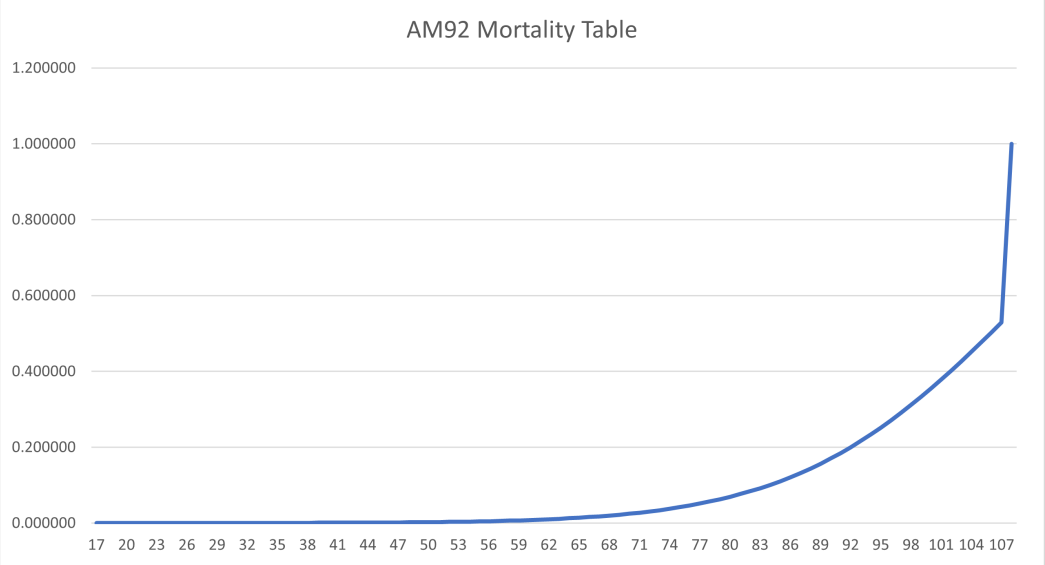

FIGURE 3.4: AM92 Mortality table

As can be seen, the mortality tables are smooth, which is a desired property for pricing, valuations, and ML, because a model can more easily pick up the interaction with age.

An arbitrary surrender table was used. Surrenders are either enabled or not, and when they are enabled, Table 3.3 contains the values for w_t, where t is measured in years:

t (from)	t (to)	w_t
0	1	10% per annum
1	2	5% per annum
2	8	2% per annum
8	15	1% per annum
15	end	0% per annum

TABLE 3.3: Surrender table

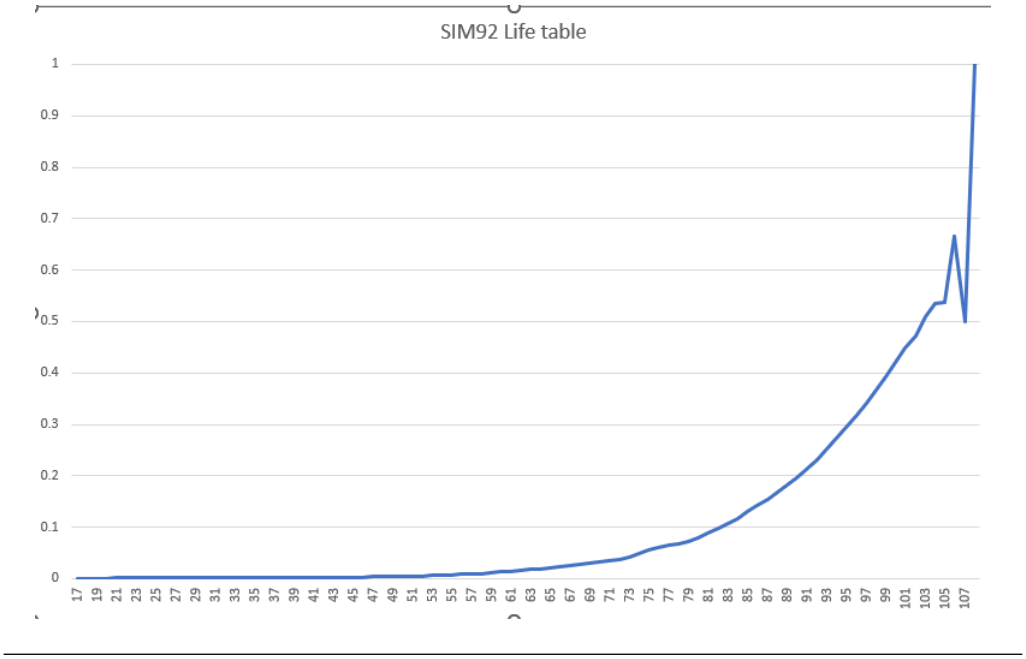

FIGURE 3.5: SIM92 Mortality table

If surrenders are ignored, then 0% per annum was used throughout. Enabling or disabling with these tables allows a DNN to pick up the relationship with the policy term.

Arbitrary expense per policy and expense inflation assumptions were assumed.

The actual values and impacts of these choices are discussed in the next section.

3.6 Actuarial pricing and valuation methods

This study discounts monthly future cash flows to obtain the present value thereof for both pricing and valuation purposes. Various deterministic approaches are used. This section discusses how the "Actual" reserve values were calculated using a traditional actuarial calculation approach.

3.6.1 Actuarial principles for premiums and reserves

When an actuary refers to pricing, contract premiums are calculated. Premiums are calculated using the following formula:

$$(\mathbf{P} - e'_t) * a_x = SA * A_x \tag{3.1}$$

where:

- P is the premium to be solved,
- e'_t is the expense at time t, after allowing for any expense inflation, and
- SA is the guaranteed benefit at death.

This is generally calculated at some optimistic interest rate and for each policy individually at inception. Pricing bases can be updated frequently as economic conditions change or during pandemics.

When actuaries refer to a valuation, it means reserves are calculated. For reserving, the following formula is used:

$$V_0 = \text{PV Outgo} - \text{PV Income} \tag{3.2}$$

where:

- V_0 is the reserve to be solved and held now at time 0,
- PV Outgo is the PV of all outgo, which is the sum of discounted benefits ($SA * A_x$) and expenses ($e'_t * a_x$),
- PV Income is the PV of premium income ($\text{P} * a_x$), and
- the return on reserves is implied by the discount rate v.

For valuation purposes, this is calculated at prudent interest and decrement assumptions, which typically leads to more significant and positive values when discounting the expected future cash flows. Valuation bases generally are updated annually, most likely if the experience indicates that the bases are incorrect or if changes in regulations are made.

3.6.2 Pricing and valuation bases

For this study, it was important to calculate individual monthly premiums on at least two different pricing bases for the machine learning models to gauge the impact that premium rates alone have on the reserve. The three premium bases used are given in Table 3.4.

Pricing Basis (PB)1 is more optimistic for all variables and leads to lower premiums for all policyholders. PB3 is the more pessimistic basis. For valuation purposes, PB1 and PB3, and PB2 is given for illustration purposes.

Assumption	**PB1**	**PB2**	**PB3**
Mortality	100% AM92 Ult	100% AM92 Ult	100% AM92 Ult
Interest rates	6%	**4%**	**2%**
Expenses	5 per month	**10 per month**	**20 per month**
Expense inflation	4% per annum	**6% per annum**	**8% per annum**
Surrenders	No	No	No
Total premiums	6.634b	9.223b	13.858b

TABLE 3.4: Pricing Bases

It is important to note that the AM92 table only has one mortality table for both genders. Therefore, no distinction was made for gender for both pricing and valuation exercises.

The six valuation bases used were:

Assumption	VB1	VB2	VB3
Mortality	100% AM92 Ult	100% AM92 Ult	100% AM92 Ult
Interest rates	4% per annum	4% per annum	**3% per annum**
Expenses	25 per month	25 per month	25 per month
Expense inflation	6% per annum	6% per annum	6% per annum
Surrenders	Yes	**No**	No

TABLE 3.5: Valuation Bases 1 to 3

Assumption	VB4	VB5	VB6
Mortality	**110% AM92 Ult**	110% AM92 Ult	110% AM92 Ult
Interest rates	3% per annum	3% per annum	3% per annum
Expenses	25 per month	**30 per month**	30 per month
Expense inflation	6% per annum	6% per annum	**7% per annum**
Surrenders	No	No	No

TABLE 3.6: Valuation Bases 4 to 6

3.6.3 Example calculations of premiums and reserves for individual policies

Consider a male policyholder aged 40 with a new WL policy with a constant SA of 80,000. Expenses are 100 per year and increase by a constant 4 per year. Using AM92 mortality and an interest rate of 6% per annum, the constant annual premium payable upfront is given by:

$$P_{40} = \frac{80000A_{40} + 96a_{40} + 4Ia_{40}}{\ddot{a}_{40}}$$

$$P_{40} = \frac{80000 * \frac{473.33}{2052.96} + 96 * \frac{41070.31}{2052.96} + 4 * \frac{643108.84}{2052.96}}{\frac{41070.31}{2052.96}}$$

$$P_{40} = \frac{21618.34}{20.00541} = 1\ 080.63$$

Consider a male policyholder aged 30 with a new WL policy with a SA of 80,000 increasing 2% per annum compound. Expenses are 100 per annum, increasing by 2% per year compound plus 1% of all premiums paid. Using AM92 mortality and an interest rate of 8% per annum, the annual premium payable upfront if increasing by 3% per annum compound is given by:

$$P_{30} = \frac{80000A''_{30} + 100a''_{30}}{0.99 * Ia'_{30}}$$

$$P_{30} = \frac{80000 * \frac{136.72}{1786.62} + 100 * \frac{29698.18}{1786.62}}{\frac{45850.46}{2394.108}}$$

$$P_{30} = \frac{7784.245}{18.95986} = 410.56$$

For valuations at an interest rate of 2%, the day 0 reserves for the two example policies are (policyholder aged 40 first):

$$V_0 = 80000A_{40} + 96a_{40} + 4Ia_{40} - 1080.63\ddot{a}_{40}$$

$$V_0 = 80000 * \frac{2070.869}{4463.818} + 96 * \frac{122040.4}{4463.818} + 4 * \frac{2269402.718}{4463.818} - 1080.63 * \frac{120040.4}{4463.818}$$

$$V_0 = 36488.12 - 33941.91 = 2\ 546.21$$

And for the policyholder aged 30:

$$V_0 = 80000A_{30} + 100\ddot{a}_{30} - 0.99 * 410.56 * Ia'_{30}$$

$$V_0 = 80000 * 1 + 100 * \frac{493844.1}{9925.209} - 0.99 * 410.56 * \frac{722937.3}{13299.99}$$
$$V_0 = 84975.65 - 26329.51 = 58\ 646.15$$

It is typical for policies with increasing premiums to have higher reserves per SA (more capital intensive), because the larger part of the premium income is paid later.

3.6.4 Resulting premiums

The calculated total monthly premiums per PB on the input dataset are shown in 3.7.

PB	#POLS	TOT SA	TOT PREM
PB1	76,102	6,960m	6.619m
PB2	76,102	6,960m	9.178m
PB3	76,102	6,960m	13.737m
PB4	76,102	6,960m	6.628m
PB5	76,102	6,960m	6.227m
PB6	76,102	6,960m	6.234m
PB7	76,102	6,960m	16.193m
TOTAL	**532,714**	**48,720m**	**64.816m**

TABLE 3.7: Premiums as calculated for Z1 and Midway models

This table also includes Pricing Bases 4 to 7, which will only be used for the Midway model from Table 3.14.

3.6.5 Resulting reserves

The calculated reserves per PB and VB are given in Table 3.8. In this table, and any further tables given, total premiums are the total monthly premiums payable immediately.

It is important to observe that reserves **increase** when:

- premiums are reduced,
- surrenders are reduced,
- interest rates are reduced,

PB/VB	#POLS	TOT SA	TOT PREM	TOT RES YR 0
PB1/VB1	76,102	6,960m	6.619m	936m
PB1/VB2	76,102	6,960m	6.619m	1,134m
PB1/VB3	76,102	6,960m	6.619m	1,671m
PB1/VB4	76,102	6,960m	6.619m	1,752m
PB1/VB5	76,102	6,960m	6.619m	1,859m
PB1/VB6	76,102	6,960m	6.619m	1,912m
PB3/VB1	76,102	6,960m	13.737m	-435m
PB3/VB2	76,102	6,960m	13.737m	-413m
PB3/VB3	76,102	6,960m	13.737m	-115m
PB3/VB4	76,102	6,960m	13.737m	-7m
PB3/VB5	76,102	6,960m	13.737m	100m
PB3/VB6	76,102	6,960m	13.737m	153m
TOTAL	**913,224**	**83.516b**	**122.132m**	**8.567m**

TABLE 3.8: Valuation reserves calculated

- mortality rates are increased,
- expenses are increased, and
- expense inflation rates are increased.

However, for policies recently written, with negative reserves (implying the premium is profitable), reserves will decrease when surrenders are increased, as can be seen for PB3/ VB1 to PB3/ VB2, which is a combination of decreases and increases in resulting reserves.

Table 3.9 provides key summary values of the dataset used for testing my models over the 12 combinations of PBs and VBs.

	Total
# Policies	913,224
Avg Age	44.36
Total SA	83,516m
Avg TIF	41.51
Total Premiums	122,132m
Total Reserves YR 0	8,567m

TABLE 3.9: Total Training and Test data

3.6.6 Comparison of present values between pricing and valuation output

The reserves for the first 10 years per run for each PB and VB are summarised in Figures 3.6 to 3.12.

As can clearly be seen, reserves increase over time, due to the benefit payment becoming nearer to being paid as policyholders age.

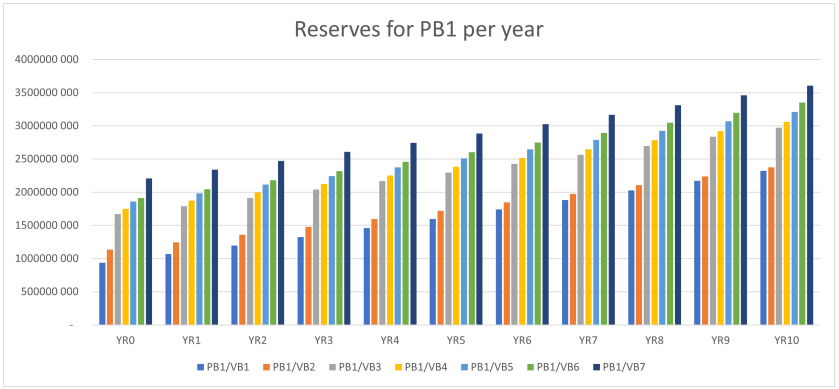

FIGURE 3.6: Reserves for PB1

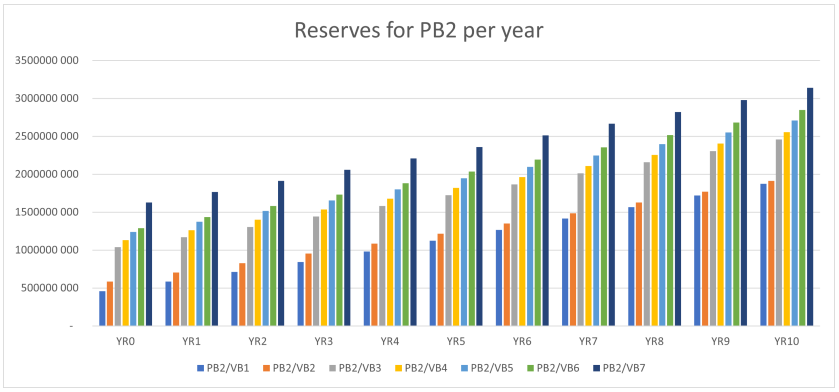

FIGURE 3.7: Reserves for PB2

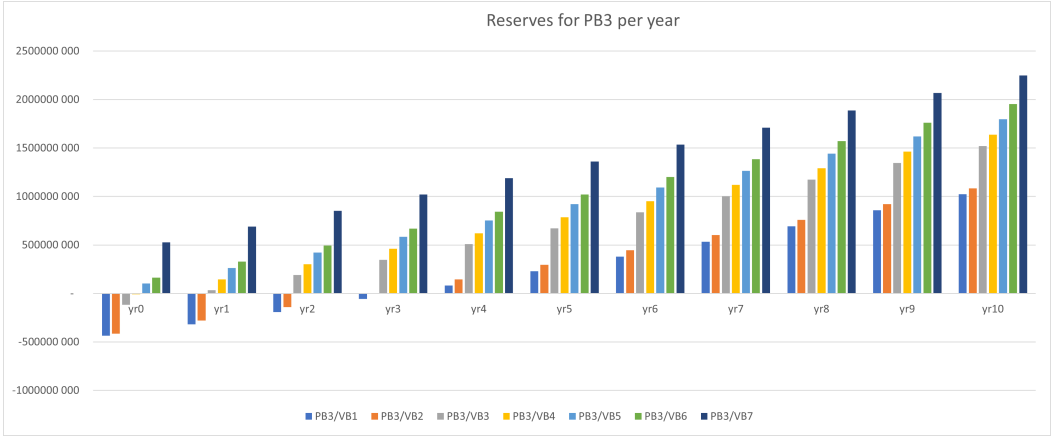

FIGURE 3.8: Reserves for PB3

For PB3 and PB7, reserves start negative and then become positive after 10 years. PB7 is the most conservative PB, and the results can be seen most clearly there.

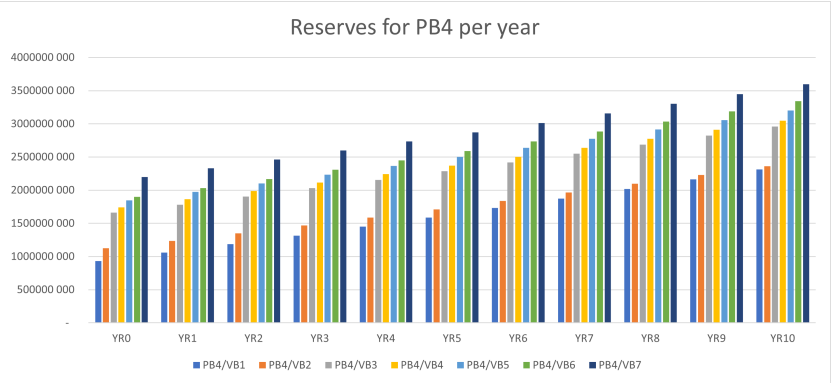

FIGURE 3.9: Reserves for PB4

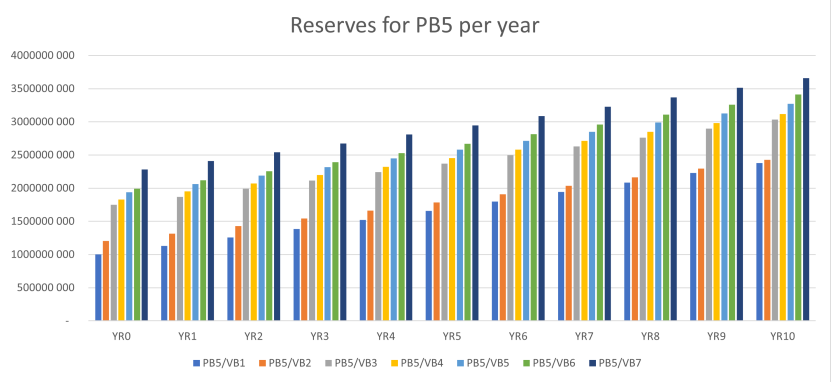

FIGURE 3.10: Reserves for PB5

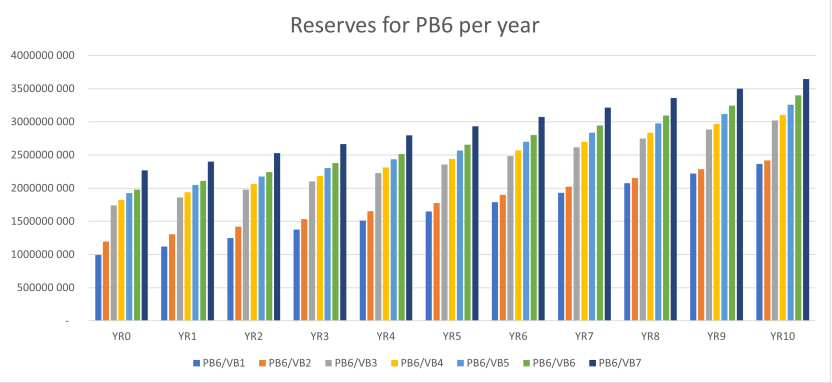

FIGURE 3.11: Reserves for PB6

3.6.7 Principal Components Analysis

After performing the pricing and valuation exercises, a Principal Component Analysis (PCA) was performed on the normalised 913,224 whole life policies of the ML input dataset used in the Z1 model. The PCA was repeated with only 5% of the input dataset. This was to determine if the data is reasonable. I found that limiting the number of principal components to 4 produced

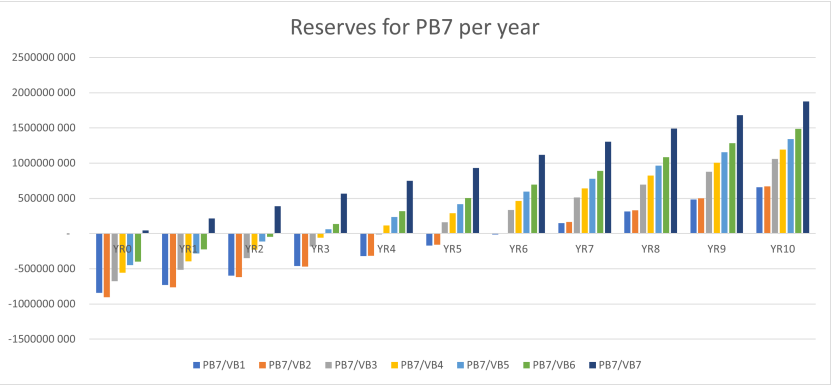

FIGURE 3.12: Reserves for PB7

a reasonable representation of the variance in the dataset. Of importance is therefore that it was also found that employing 4 hidden layers worked best, confirming the research of Rachmatullah et al. (2021).

The results of the PCA are given in Tables 3.10 and 3.11, with some discussion hereafter.

	PC1	**PC2**	**PC3**	**PC4**
SS Loadings (Eigenvalues)	3.09744	2.29922	1.58464	1.17332
Variance explained	0.28159	0.20902	0.14406	0.10667
Cumulative variance	0.28159	0.49061	0.63467	0.74134
Proportion explained	0.37984	0.28195	0.19432	0.14389
Cumulative proportion	0.37984	0.66179	0.85612	1.00000

TABLE 3.10: Principle Component Analysis

Variable	**PC1**	**PC2**	**PC3**	**PC4**
Age	0.05778	**0.90458**	-0.39175	-0.12355
SA	0.06088	0.36657	**0.87876**	0.22796
TIF	0.04089	0.17621	-0.29842	**0.66187**
Premium	0.02134	**0.65900**	**0.64183**	-0.23387
Term to 108	-0.05778	**-0.90458**	0.39175	0.12355
Mortality adjustment	**0.86951**	-0.06769	-0.01001	-0.08602
Valuation rate	**-0.82676**	0.05792	0.00603	-0.02397
Expenses	**0.81561**	-0.06759	-0.01193	-0.17268
Expense inflation	**0.64150**	-0.05639	-0.01139	-0.20904
PayM	0.00000	0.00000	0.00000	0.00000
Mort table	0.00000	0.00000	0.00000	0.00000
Surrenders	**-0.65147**	0.04713	0.00487	-0.05545
Product	0.00000	0.00000	0.00000	0.00000
Reserve	0.37974	0.21235	0.06366	**0.71659**

TABLE 3.11: Principle Components Variables

The most important component is the VB, with mortality, expenses, and expense inflation having a positive contribution and the valuation rate and surrenders a negative contribution.

This accounts for 38% of the variation in the data. The PCA therefore shows that VB is the most important consideration for determining reserves, and provides substance to employing more VB scenarios to the Midway model.

The second-most important component is the policyholder details, which include the age, with premiums being a function thereof. This accounts for 28.2% of the variation in the data.

The third-most important component is the policy details, which are the SA, with premiums being a function thereof. This accounts for 19.4% of the variation in the data.

The last important component is the reserve, with the expired term to date having a positive impact (on the current age and expected term until death). This accounts for the remaining variation in the data.

Since the WL policies are all of the same PayM and Product, and use the same pricing and valuation basis AM92, these variables had no impact on the PCA analysis.

In conclusion and as expected, the ML input dataset, where the calculation of the reserves employed traditional actuarial methods, is sufficient to enable the calculation of valuation reserves utilising ML methods.

3.7 Preparing data for machine learning

Having ensured the accuracy of the pricing and valuation exercises detailed in Section 3.6, the next step in the methodology involved preparing the data for input into the DNN models. This encompassed cleaning and transforming the data to ensure it was in a suitable format for modelling. The dataset consisted of valuation bases and policy details for a large number of life insurance products, with prices and reserves calculated using traditional actuarial methods as outlined in the preceding sections.

Data transformations were performed to ensure the data was in a suitable format for modelling. This included appending the valuation basis used for each policy to the corresponding data row, normalising continuous variables, and converting categorical variables into numerical representations. These transformations aimed to facilitate model training and improve algorithmic performance.

As discussed in Section 1.5.7, it is standard practice to normalise continuous variables to enhance interpretability in simple slope analysis. Continuous variables were normalised using the Gaussian method, by subtracting the mean and dividing by the standard deviation of each respective variable. This standardisation ensures that all variables contribute equally to the model and facilitates more efficient training. Categorical variables were encoded using binary

(0 or 1) representation. The following variables were treated as categorical: Surrender basis (Yes or No), PAYM (Pay at Maturity, Yes or No), and Mortality table (AM92 or SIM92).

As an example, the Surrender basis variable, being binary (enabled or not), was encoded using values of 1 and 0. No further transformation was required, as this single column can be readily translated into a computer language. Should additional categorical variables be incorporated, corresponding columns would be required for each category.

Two significant models, Z1 and Midway, were trained and are discussed in the next sections.

3.8 Building the Z1 model

Following data preparation, various DNN models were constructed. As discussed in Chapters 1 and 2, this study deliberately excluded commonly used ML methods such as GLMs, AEs, and XGB.

The initial Z1 model-building approach involved training the models on a dataset of 913,224 model points, constructed from the original dataset. However, limitations of this approach became apparent over time, and the following section highlights both the successes and challenges encountered.

A comparative analysis of several commonly used algorithms was conducted. The data was partitioned into a training set (75% of the total data), a validation set (a single policy), and a test set (the remaining 25% of the data) to assess the accuracy of the models on unseen data. Using 1 policy for the validation set is not standard practice, however the model training was stopped when high training accuracy was reached and no data was leaked from the test set to the training set. The motivation for only using 1 policy was that a model that was stopped training "incorrectly" could then be used to compare to a model (Midway) that was trained efficiently.

The models were constructed by defining the inputs (limited to demographic information and policy details) and outputs expected PV of the policies. The training data was then used to train the models and optimise their parameters to minimise prediction error.

Key performance metrics, including the number of epochs, training loss, training accuracy, validation loss, validation accuracy, test loss, and test accuracy, were recorded for each model.

To quickly identify promising model configurations, 300 different models were run using various architectures and hyperparameter values. The following dimensions were explored:

- Hidden layers of 1 - 6,

- increasing number of neurons per layer,
- decreasing number of neurons per layer,
- constant number of neurons per layer, and
- different combinations of the hyperparameters.

The process followed and some of the adjustments made during training are discussed in more detail in Appendix B. The following key findings emerged:

1. Models with hidden layers too low (< 3) or high (> 5) did not train effectively, even with varying LRs and batch sizes.
2. Increasing the number of neurons per layer did not consistently improve performance.
3. Maintaining a constant number of neurons per layer also did not yield satisfactory results.
4. RELU and its variants outperformed Sigmoid and TANH AFs.
5. CELU demonstrated the best overall performance as an AF.
6. NADAM proved to be the most effective optimiser throughout the experimentation.
7. Smaller batch sizes resulted in longer training times and were more prone to fluctuations.
8. Some models converged to a local minimum, producing reserves equal to the total actual value divided by the number of observations, indicating a highly flawed training process.

This approach mirrors the methodology employed in infrastructure project prediction, as demonstrated by Golizadeh et al. (2017). In that research, a novel automated approach for predicting the duration of dam construction projects was developed, utilising ANNs. Through literature review and expert interviews, they identified key physical, climatic, and predictable factors influencing project duration. Seven three-layer ANN models were then constructed, incorporating variables related to physical characteristics (e.g., spillway type, hydropower plant existence, dam height, and volume) and weather data (e.g., thunderstorm days, precipitation). Their study prioritised variables assessable during the pre-contracting stage and demonstrated the potential for AI-driven, more accurate time estimation in infrastructure projects. This highlights the importance of carefully selecting input variables and demonstrates a successful application of ANNs in a complex prediction task, informing our current exploration of model configurations

To identify the optimal AF, a comprehensive evaluation was conducted, testing all available options while maintaining consistent hyperparameters. This iterative process, repeated across numerous hyperparameter combinations, revealed the most effective AF for the model.

Potential vanishing gradients were closely monitored, as their presence would necessitate a simplification of the model or refinement of the training process. No neurons exhibiting vanishing gradients were identified, suggesting that the model architecture and hyperparameters were appropriately configured to facilitate effective training. Furthermore, the values of the weights and biases were monitored to ensure full connectivity and mitigate potential biases.

Different optimisation algorithms possess distinct strengths and weaknesses, and their performance can vary significantly depending on the neural network architecture and dataset characteristics. A thorough evaluation of various optimisers (as detailed in Section 2.3.13) led to the selection of NADAM, which demonstrated optimal generalisation performance without excessive training time or overfitting.

The combination of the CELU activation function with the NADAM optimiser proved particularly effective. While prior studies (as highlighted in Section 2.4) have reported successful implementations of NADAM with the ELU AF, this study clearly demonstrates the superior performance of CELU. Notably, RELU and TANH, commonly used AFs in the literature, did not generalise well to this dataset.

MSE is typically employed as the LF for supervised regression models involving monetary predictions. However, MSE exhibited scaling issues with policies having small or negative reserves and was prone to converging to a local minimum where the same reserve value was predicted for all policies. Given the importance of accurate reserve calculations for each policy due to portfolio fluctuations, MAE was selected as the preferred LF for both the Z1 and Midway models

Some performance metrics are given below. The performance of the Z1 model will be investigated in more detail in Chapter 4, where it will be compared against the Midway model.

Actual versus predicted reserves on the Z1 test dataset are given in Figure 3.13.

The Z1 model demonstrates a high degree of accuracy on unseen data, with the majority of predictions aligning closely with a 45-degree reference line. Outliers are largely attributable to policy characteristics significantly deviating from the mean. While a practitioner might reasonably consider such a model satisfactory, potentially citing an overall statistical accuracy, the question of further improvement through the inclusion of additional policy features and valuation bases remained a central focus. The pursuit of this optimisation ultimately culminated in the development of the Midway model, representing the pinnacle of this study's investigative efforts.

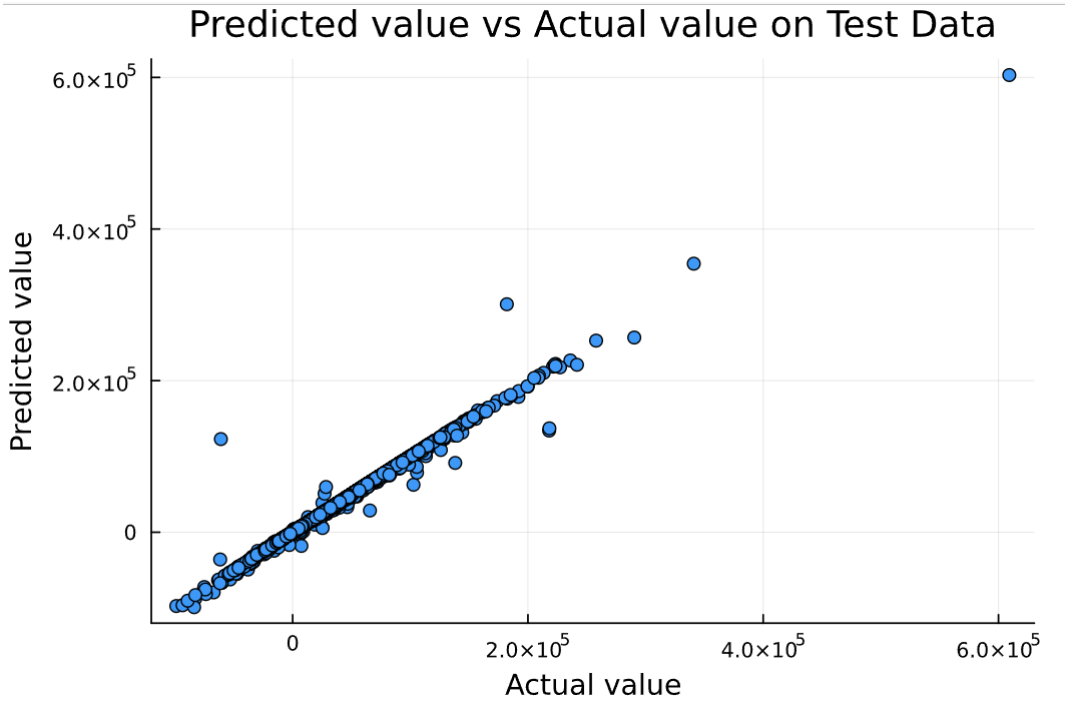

FIGURE 3.13: Z1 model: Actual vs predicted reserves on test data

3.9 Building the Midway model

The Kriging method, as detailed in Section 2.4.1, was employed to generate training data for the Midway model. This approach aimed to provide a more representative sample across the entire range of plausible policy and valuation basis characteristics.

3.9.1 WL data used for training the Midway model

In addition to EA and TA policies (discussed in subsection 3.9.2), additional and different WL policies were also incorporated into the training dataset. Model points for WL policies were generated utilising the ranges specified in Table 3.12. These model points were distributed uniformly across the likely distribution between the minimum and maximum values for each parameter.

Model points for WL policies were generated utilising the ranges specified in Table 3.12. These model points were distributed uniformly across the likely distribution between the minimum and maximum values for each parameter:

Variable	Minimum	Maximum	Difference	#Categories
Age	17	65	48	8
SA	4 000	5 00 000	496 000	9
TIF	0	480	480	7

TABLE 3.12: Policy distribution for training

3.9.2 EA and TA data used for training the Midway model

To demonstrate the model's adaptability, the introduction of EA and TA products was considered.

For TA, the maximum contract term is required. For EA, both the contract term and an indicator variable denoting benefit payment upon maturity are needed. These products can be integrated alongside WL products by introducing two additional columns (i.e., it is a substitute for product code):

- Term - the original term of the product in years (assumed to be an integer).
- PayM - A categorical variable, coded as 1 for EA and 0 for WL and TA.

The following values are assigned:

For WL, Term is calculated as 108 minus age, and PayM is set to 0.

For EA, Term is the contract term, and PayM is set to 1.

For TA, Term is the contract term, and PayM is set to 0.

Due to the absence of commercial data for EA and TA products, the training data utilise the same spectrum of policies used for the Midway model, with contract terms of 5 years for TA and a range of [15, 48] years for EA.

EA and TA policies were simulated from Kriging distributions derived in Table 3.12 subject to the following modifications:

- For EA and TA, all policyholders older than 50 at inception were adjusted to age 50 at inception.
- For TA, the maximum TIF was set to 3 years.

VB6 was utilised for reserving purposes.

Product	#Pols	TOT SA	PB4	PB5	PB6	PB7
WL	360	392.36m	370k	345k	320k	804k
EA	234	255.04m	330k	327k	326k	596k
TA	216	235.42m	21k	19k	18k	40k
TOTAL	**810**	**882.82m**	**721k**	**691k**	**664k**	**1.44m**

TABLE 3.13: Data inputs for Midway model

The Midway model was trained with the two additional columns, "Term" and "PayM," using the entire training dataset of 810 policies and 3,780 combinations of pricing and valuation bases. New pricing bases were also included, and are discussed in the next subsection.

A summary of all data input over all products is given in Table 3.13. Clearly, the sensitivity of WL and TA products to changes in pricing assumptions is correctly greater than that of EA.

The training results are presented in Table 3.19. The inclusion of additional products and the results on test data are discussed in Section 4.4.

3.9.3 Additional pricing bases for Midway

Four new pricing bases were introduced to enhance the model's understanding of the relationship between premiums and reserves. The first three bases incorporated minor variations from existing bases, providing increased diversity with regard to incremental premium changes across all contracts. The final base introduced a substantial, pessimistic shift, enabling the DNN to be trained on shock scenarios

The first three of these bases are given in Table 3.14.

Variable	PB4	PB5	PB6
Mortality	-10%	-20%	-30%
Interest rate	6%	6%	6%
Expenses	10	10	15
Exp Inflation	3%	1%	4%
Surrenders	No	No	No

TABLE 3.14: Pricing bases for training

One additional pessimistic PB7 was added. This was intended to provide the model with a greater opportunity to identify significant differences in pricing and also to incorporate the possibility of negative reserves, which, while not permitted under all supervisory regimes, are permissible in some actuarial regimes.

PB4 is given in Table 3.15.

Variable	PB7
Mortality	AM92+50%
Interest rate	2%
Expenses	36
Expense Inflation	4%
Surrenders	No

TABLE 3.15: Additional PB7 used

Consequently, 360 additional model points were generated, priced, and subjected to valuations across all valuation bases as outlined in Table 3.16, resulting in a total of 1,360,800 new model points for training. A larger majority of these points now incorporate negative reserves.

3.9.4 Additional valuation bases for Midway

The spectrum of possible valuations was expanded for the Midway model, by considering the entire spectrum of possible valuation bases.

For the valuation bases utilised in the training data, the Kriging approach was employed to determine bases as outlined in Table 3.16, assuming a uniform distribution between the minimum and maximum values, resulting in a total of 3,780 additional distinct combinations of bases.

Variable	Minimum	Maximum	Difference	#Categories
Mortality	-75%	75%	100%	7
Interest rate	0%	8%	7%	6
Expenses	0	60	60	6
Exp Inflation	0%	8%	8%	6
Surrenders	0	1	1	2
Mort Table	0	1	1	2

TABLE 3.16: Valuation bases for training

This process yields $360 * 4 * 3780 = 5,443,200$ WL model points for training on WL policies only. The subsequent addition of EA and TA policies results in a total of $810 * 4 * 3780 = 12,247,200$ model points.

The test dataset comprises the input dataset of 76,102 policies, as detailed in Section 3.4.1, which was priced and valued as previously for Z1 on 2 pricing bases and 6 valuation bases, resulting in a total of $76,102 * 2 * 6 = 913,224$ model points for testing, as described in previous and subsequent sections.

Error checking practices, following the approach outlined in Hynes et al. (2017), were employed. This included outlier detection and verification for duplicate data and missing values. No significant issues were identified. Table 3.17 provides a summary of the model points used as input to the Midway model.

Dataset	Percentage	#Model Points
Train	27%	3 306 744
Validation	3%	367 416
Test	70%	8 573 040
Total	**100%**	**12 247 200**

TABLE 3.17: Model points for Midway summary

Notably, a much lower percentage of data was used for training the Midway model. This was done to prevent overfitting.

3.10 Adding additional features for the Midway model

To develop a more versatile and detailed model with broader applications, several modifications were implemented, which are discussed in this section.

3.10.1 Preparation required for randomised data

Instead of utilising the provided dataset, a new dataset was simulated based on the best-fitted distribution of the original variables. The objective was to create a portfolio of policies with similar means and standard deviations to the original data, but with reserves decoupled from the interrelationships present in the original dataset. This decoupling was achieved through the following steps:

1. Data generation: The ages, SA and TIF values were randomly shuffled across the 76,102 original test input dataset to generate a new set of random model points.
2. Princing and valuation bases: PB1 and VB6 were employed.
3. Model architecture: No changes were required to the model architecture.

The results on randomised data are discussed in Section 4.3.

3.10.2 Preparation required for new life tables

To enable the Midway model to incorporate alternative mortality bases, a new categorical variable was introduced as an indicator of which mortality table to use. The AS variable was defined as 0 for the AM92 tables and 1 for the SIM92 mortality table.

As the SIM92 tables utilise a single q_x value for both males and females, no additional policy data variable for gender was required. However, should a different mortality table with gender-specific q_x values be selected, an additional categorical variable for "Gender" would need to be incorporated into the model inputs. The following changes were implemented:

1. Data Augmentation: An additional dimension was added to the Valuation Bases to accommodate the new mortality basis, resulting in an additional 1.524 million records for training purposes.
2. Pricing Bases: Pricing Bases 1, 2, 3, and 4 were retained.
3. Valuation Bases: A new VB7 was created, identical to VB6 except that the SIM92 mortality table was used. The outputs of VB6 and VB7 were then compared as of December 31, 2009.
4. Model Retraining: The model was retrained from scratch to incorporate the expanded solution space.

The results of introducing a new mortality table is discussed in Section 4.5.

3.10.3 Preparation required for training a model to perform pricing

As a final example, the pricing of WL, EA and TA products were considered. This requires the following changes:

1. Data: The same input datasets as for the final Midway model, but the reserve variable was removed altogether. The variable to be predicted will be the premium.
2. Pricing Bases: PB4 to PB7.
3. Models: Since the original Midway model predicts reserves, retraining is necessary to predict premiums.

The results on test data are discussed in Section 4.6.

3.11 Training of the Midway model

Once the models were built, their performance on the validation and test datasets was evaluated. Several commonly used performance metrics to assess the models, including the popular MAE

and MSE, were used. These metrics provide a measure of the accuracy of the models and allow comparisons between the performance of different models.

The following early stopping and updating of model parameters approaches were employed:

1. When a total training accuracy of more than 99.7% is reached, then perform the next steps.
2. Calculate the weighted mean of the last 20 training losses, where the weight at time t is equal to $(1-0.02*t)$. If the unweighted mean is more than this, then increase the learning rate and momentum.
3. Stop the training once the LR and momentum reach the maximum amounts.

The use of a mean versus a weighted mean calculation provides a more general indication of model improvement and effectively mitigates temporary fluctuations in performance. This approach also facilitates early stopping when the model begins to overfit.

3.11.1 Lessons learned when training the Midway model

Various model-building approaches were investigated during this research. The optimal model, trained on the large dataset of 4.082 million model points, is detailed in Table 3.18. A discussion of the key parameters and their impact on model performance follows.

Input type	Value
Hidden layers	4 (72,36,18,9)
Activation	CELU
Optimiser	NADAM
Loss function	MAE
Batch size	10 206
Reshuffle	Yes (SGD)
Learning rate	Schedule
Momentum	Schedule
Epochs	900
Runtime	30 min

TABLE 3.18: Best model for training

Increasing the number of hidden layers and neurons per layer was found to facilitate convergence to lower total training losses. However, computational limitations dictated that the model described in Table 3.18 represented the largest configuration that could be accommodated on the available GPU hardware. Extending the model further was not feasible due to prohibitive training times on CPU.

Exploration of various combinations of AFs and optimisers revealed that the combination of CELU and NADAM yielded the best performance for this particular problem. Testing different activation functions on the optimal model indicated that ELU, Gaussian Error Linear Unit (GELU), LeakyRELU, and Softplus all outperformed RELU in terms of achieving lower loss and reduced bias, as anticipated. In contrast, TANH and Sigmoid AFs exhibited slower training times, with TANH displaying greater bias and poorer accuracy, while Sigmoid demonstrated slightly better performance.

An initial attempt to employ MSE as the LF proved ineffective for accurately modelling small policies. The presence of a subset of policies with negative reserves and small values resulted in poor generalisation on the validation dataset when training was halted prematurely. Consequently, MAE was selected as the LF, as it demonstrated better performance across all classes of policies and valuation bases.

Initial investigations also indicated that batch size did not materially impact the training process. As a result, the focus was placed on reducing fluctuations in the gradient by maintaining a larger batch size. However, a trade-off existed between processing time and the accuracy of representing the underlying data. Consequently, a large batch size was employed in conjunction with a scheduled LR and momentum.

The introduction of additional products necessitated a re-evaluation of the batch size (from 200k to 10k). It was only then found that a smaller batch size was more effective, particularly given the characteristics of the TA business, which exhibited lower reserves, a smaller proportion of policies, and a shorter term. An even smaller batch size may have further improved the performance of the model in valuing TA products, but would have lead to longer training time and sharper updates to the gradient.

The training loss was found to fluctuate considerably when batch sized expressed a proportion of total model points of 5% or greater were used, while proportions below 0.5% failed to converge within a reasonable timeframe. The LR was adjusted in conjunction with changes to the momentum for the Midway model, only at points where training would have been halted for the Z1 model. Increasing the momentum allowed the model to better leverage the model points that were producing the largest loss. Therefore, a lower momentum was initially employed to facilitate model generalisation, followed by an increase in both momentum and LR to reduce the loss of most of the model points.

Finally, the early stopping rules were found to be suboptimal. It is proposed that an algorithm to identify over-fitting, rather than simply stopping when over-fitting occurs, would further enhance model performance. The updated early stopping algorithm was therefore:

1. A total training loss less than 0.1%, or

2. calculate the weighted mean of the last 20 training losses, where the weight at time t is equal to $(1 - 0.02 * t)$. If the unweighted mean is more than this, **then stop once the validation loss is less than 0.3%**.

When incorporating a new pricing basis, a constraint arises from the current model's capacity, which is limited to approximately 4.5 million model points. Several options were considered to address this:

- Removing some of the existing bases to create space and retrain the model from scratch, or
- Retraining the model from scratch on new data, but training each previously trained model on consecutive subsets of the total data at random, or
- Further training of the existing model using only the new premium data.

Caution is required here. Simply training the existing model on new data is insufficient, as the current mean and standard deviation do not include the new data points. To ensure accurate normalisation, the entire model must be retrained using the new mean and standard deviation, calculated on all policies in the training dataset only.

To address this, the dataset of 5,443,200 model inputs was randomly split into 80% training and 20% validation datasets, following the structure outlined in Section 3.11.1. The newly trained model was used to re-evaluate the 7 valuations on the test dataset of Z1 input policies, as previously described.

It was observed that the model continued to improve during training and may have benefited from a higher LR, given the increased number of model points. Applying the formula $(LR * \frac{batch\_size}{256})$, an increase of 33% was deemed appropriate. During training, it was noted that the model initially exhibited a positive bias, while the Z1 model had a small negative bias for the first four layers and a small positive bias for the last layer. Around epoch 500, the loss and accuracy were already comparable to those of the Z1 model. Over the subsequent 100 epochs, limited improvement was observed, with an accuracy of 99.8% achieved for both the training and validation datasets, representing a definite improvement. Furthermore, the final bias of the Midway model was found to be lower than that of the Z1 model. The early stopping rule, based on average loss, was triggered at epoch 620, with both training and validation losses were close to 0.2

Therefore, considering the options with regards to the above parameters, performing new training of the model with the policies from PB4 to PB7 appears to be the most straightforward

approach. This will facilitate the incorporation of the new data while leveraging the previously trained Z1 model's knowledge.

The resulting monthly premiums over the new Pricing Bases are given in Table 3.19.

Premium Basis	Total Premiums
PB4	5.047b
PB5	4.837b
PB6	4.648b
PB7	10.080b
TOTAL	24.612b

TABLE 3.19: Total premiums per Midway pricing bases

This model initially has a rapid decrease in total loss per training epoch, but the loss decreases very slowly from around epoch 800. Overall, the bias of all the layers is consistently extremely low. Previous models trained all had much higher bias, although the models had the same initial training characteristics.

The model produced a training accuracy of 99.9% and a prediction accuracy of 99.9% on the validation dataset. The final average bias per neuron in every layer was as follows:

- Layer 1: -0.77
- Layer 2: -0.68
- Layer 3: -0.62
- Layer 4: -0.06
- Output layer: -0.19

3.11.2 Accuracy of the Midway model on the validation dataset

Figures 3.14 to 3.19 explain how well the Midway model fits the validation dataset. The actual reserves are on the x-axis, and the model-predicted reserves are on the y-axis. The 45-degree line denotes that actual reserves are equal to expected reserves.

Figure 3.14 shows the current WL reserves in dark blue, and the WL reserves in 10 years' time in yellow. Figure 3.15 shows the current WL reserves in dark blue, and the WL reserves every year for the next 10 years in a different colour, where the last colour is pink. There are no outliers, and the model predicts the WL reserves accurately for all years considered.

Figure 3.16 shows the current TA reserves in dark blue, and the TA reserves in 4 years' time in yellow. Figure 3.17 shows the current TA reserves in dark blue, and the TA reserves every year

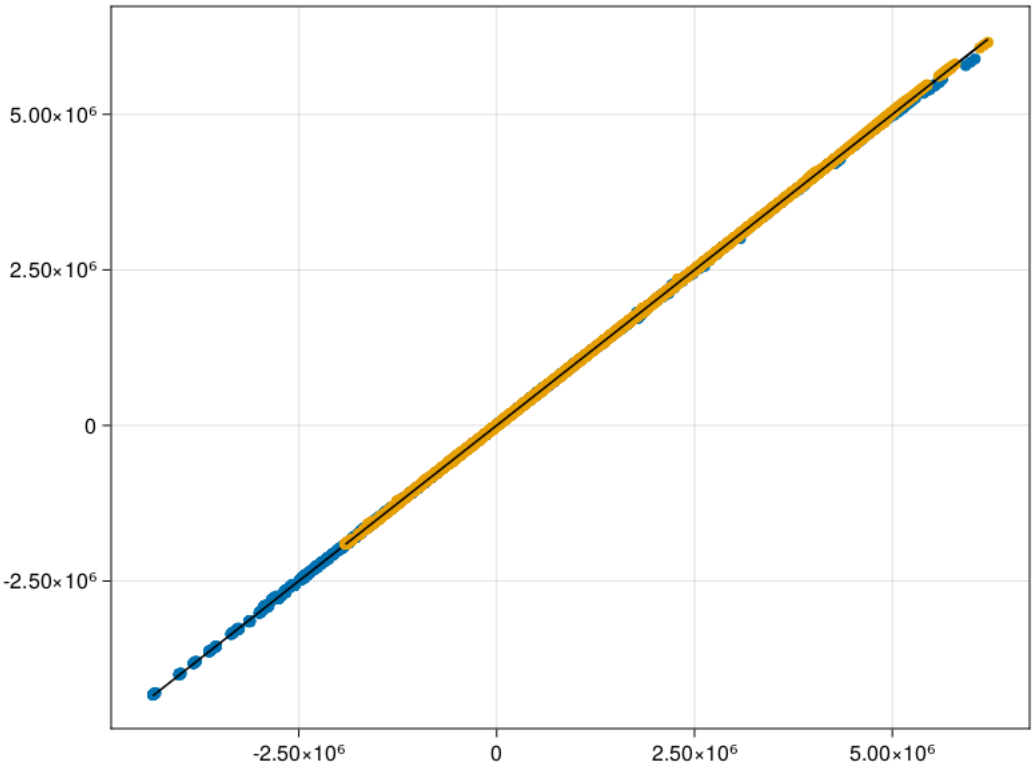

FIGURE 3.14: Midway model year 0 and year 10 WL reserves

for the next 4 years in a different colour, where the last colour is light blue. The model is not perfect for TA as the reserves are small, and absolute errors in the reserves will be swamped by errors on model points with larger reserves. TA also has a short term, unlike WL and EA, perhaps the model is not taking the term into account sufficiently. Lastly, the amount of model points for TA is only 25% instead of 33.3% of all policies, and this might have impacted the focus of the TA training.

Figure 3.18 shows the current EA reserves in dark blue, and the EA reserves in 10 years' time in yellow. Figure 3.19 shows the current EA reserves in dark blue, and the EA reserves every year for the next 10 years in a different colour, where the last colour is pink. There are no outliers, and the model predicts the EA reserves accurately for all years considered.

The accuracy of the Midway model on the original input data is considered in Section 4.2.

3.12 Limitations of the study

As with any ML project, particularly in scenarios where a direct relationship between model layers and established actuarial principles is absent, the interpretation of the model parameters and activation output is presented in a format not readily translatable to human understanding.

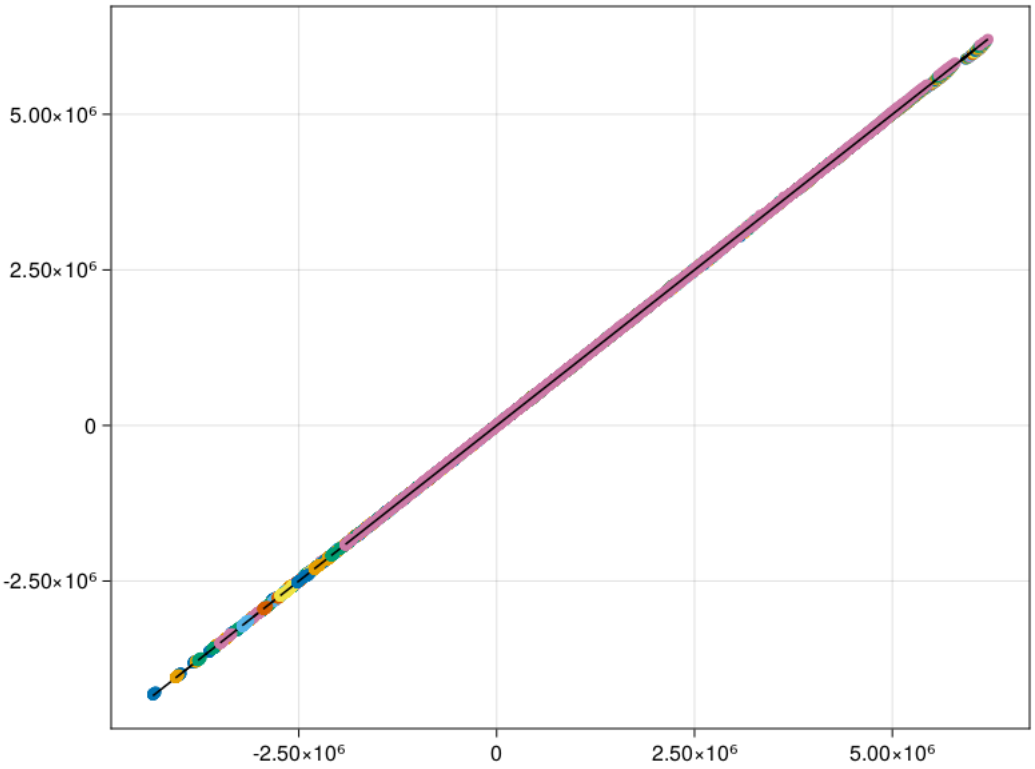

FIGURE 3.15: Midway model year 0 to year 10 WL reserves

This study does make an attempt to present the final output of DNN models in a language, namely, the actuarial reserve values, in comparison to traditional actuarial methods.

Results may appear sporadic and lack a clear rule-based foundation. Consequently, interpretation by human experts is required to convey the findings in an accessible manner, though perfect translation may not be achievable. Although progress has been made, as discussed in Section 2.5.1, full alignment between model outputs and human thought processes remains a challenge, and further research into XAI is warranted.

3.12.1 Limitation on categorical variables

The Midway model was designed using relatively simple and limited datasets of life protection business, with a limited number of categorical variables. Insurance companies typically require the following additional variables:

- Male and female mortality tables,
- select mortality,
- many more mortality tables,

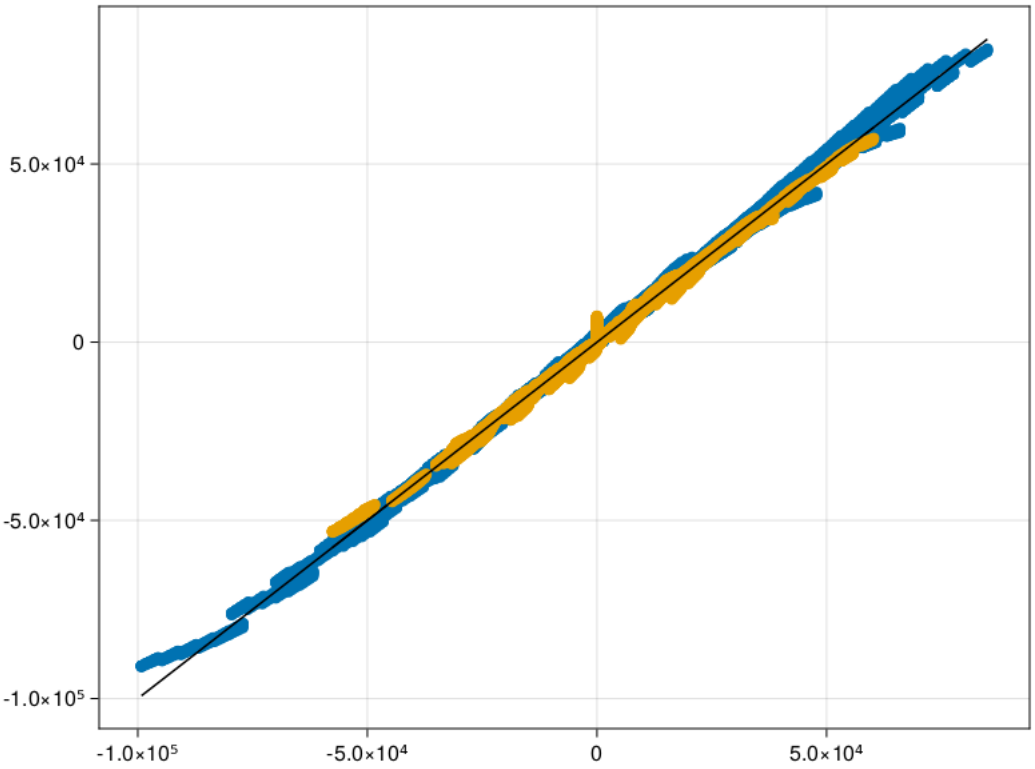

FIGURE 3.16: Midway model year 0 and year 4 TA reserves

- term structure for interest rates, and
- has more variance in products.

3.12.2 Limitations on products and features

Several limitations are noted, which can be readily addressed through further development:

- It is recommended that model developers cater for initial expenses (primarily commission and underwriting expenses) in the premium rates. This is not a reserving issue, and the Midway model will still calculate reserves correctly on higher premiums.
- It is recommended that model developers cater for termination expenses in the premium rates, as well as an increase in the SA. The Midway model will correctly calculate the revised reserves.
- Increases in premiums or SA at fixed intervals can be accommodated by incorporating additional numerical policy input variables.

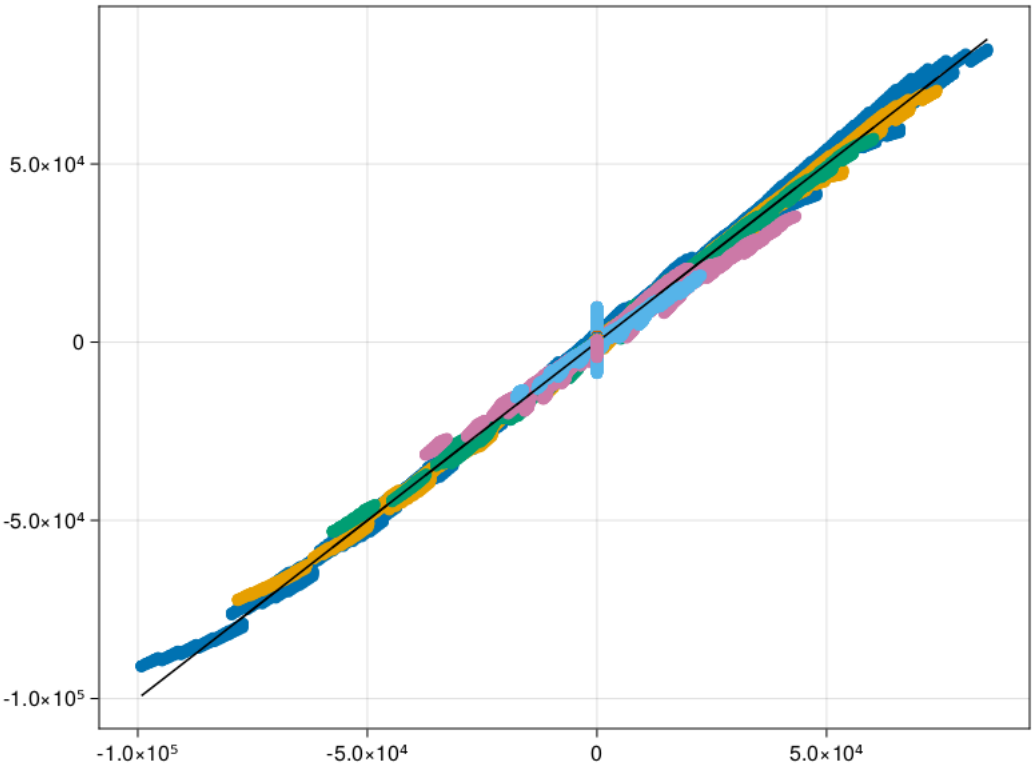

FIGURE 3.17: Midway model year 0 to year 4 TA reserves

3.12.3 Limitation on machine learning methods employed

Due to the attainment of strong results in terms of accuracy using a DNN, the following alternative ML models were not considered for this study: AE, DT, RF, XGB, RNN, GRU, and LSTM.

Further research exploring these methods in the context of ML for life insurance is recommended.

3.13 Summary of models used

A traditional actuarial pricing model was used, which takes policy data and PB as a dataset and, for each policy, calculates the premium for that PB.

A traditional actuarial valuation model was used, which calculates a reserve for each policy based on the input dataset, PB, and VB. Although the valuation model calculates the reserve for each policy at each duration, the reserves at the start and the end of the next 10 years are considered in this study. The combined input and output of the valuation model was passed to the DNN.

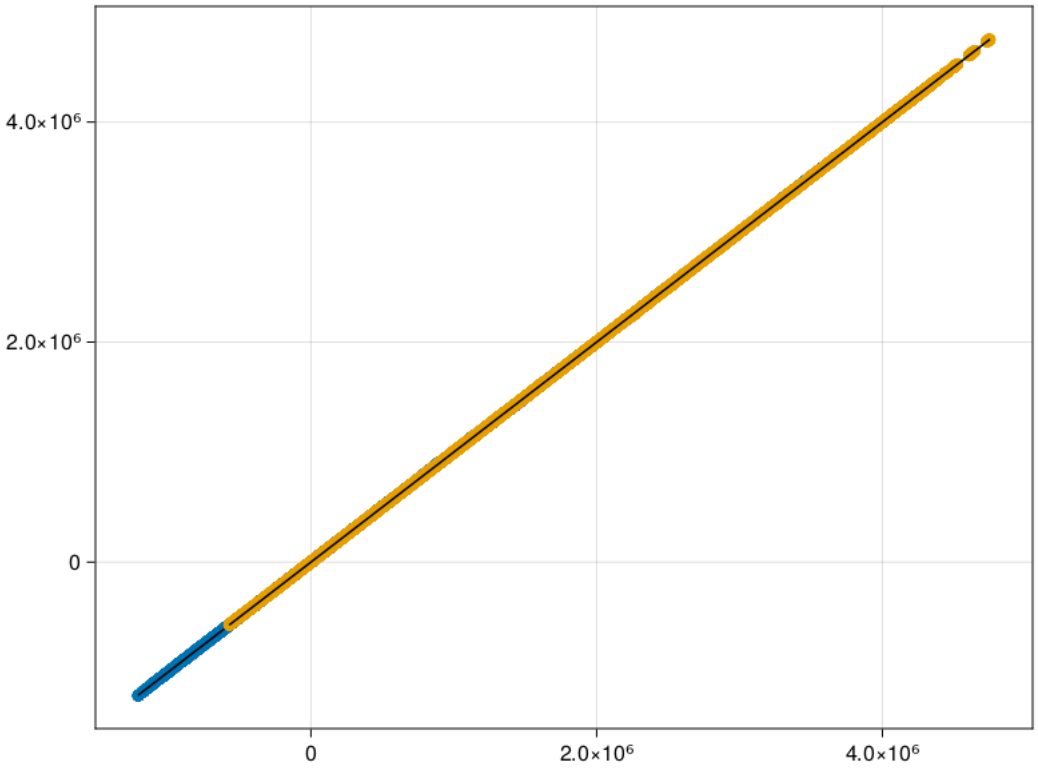

FIGURE 3.18: Midway model year 0 and year 10 EA reserves

The DNN model then takes the output from the valuation model, and creates a DNN as specified, which I then save and use to calculate reserves for the input dataset. The layers of the DNN model are of dimension 12 x 72 x 36 x 18 x 19 x Y. Y was limited to years 0 to 10. Calculations are performed once and simultaneously.

The reserves for years 0 to 10 were compared between the valuation model and the DNN.

The input dimension is 13, and a description of each input variable used is given in Table 3.20.

The first variable ID is ignored by the DNN. Variables number 2 to 10 are Gaussian normalised. Variables 11 to 13 are categorical variables and were not normalised.

The output of the Midway model is of dimension 11 and includes the reserve at the start of the year in [1, 11] for every policy. For example, Y_1 would be the base reserves now, and Y_{11} would be the reserve at the end of year 10.

A summary of the final trained models is given in Table 3.21.

In comparison, the Z1 model has been trained on the original dataset only with 12 combinations of PB/glsvb. The Midway model has been trained on 810 synthesised policies, selected over 4 product ranges, using Kriging methods. The Midway model, although trained on less policies,

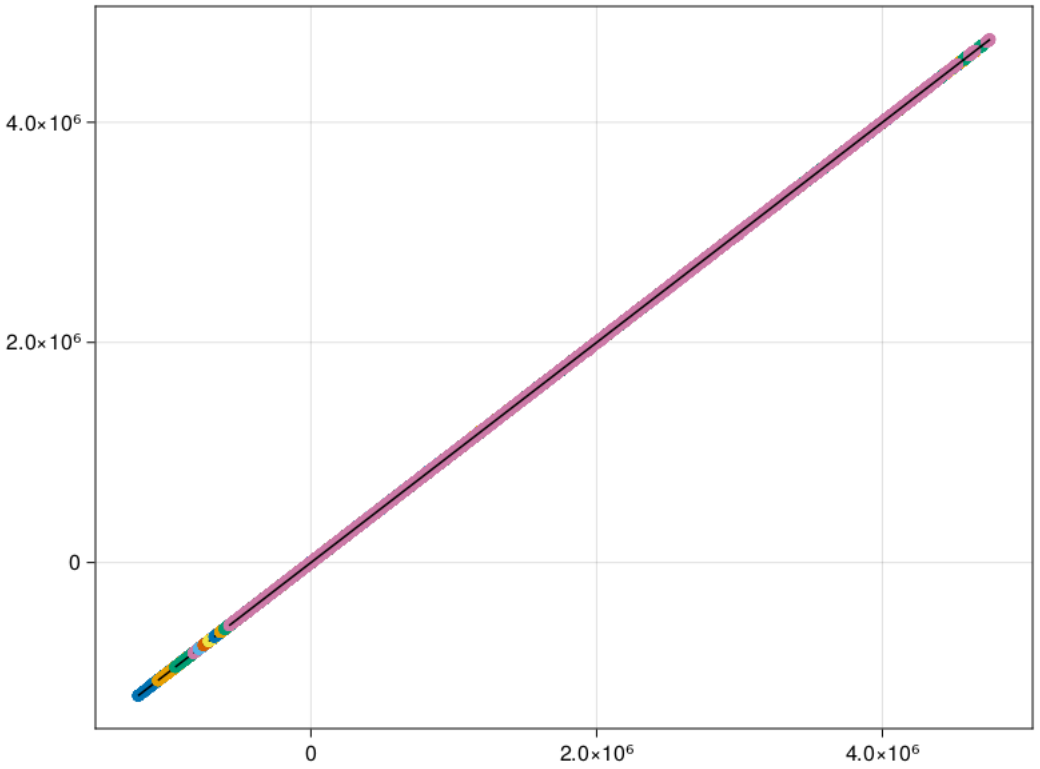

FIGURE 3.19: Midway model year 0 to year 10 EA reserves

#	Variable	Description	Range
1	ID	Identifier	[1, 50m]
2	Age	Current age	[17, 108]
3	SA	Benefit at death Benefit at maturity for EA	[0, 500m]
4	TIF	Current policy term	[0, 1092]
5	Premium	Contract premium per month	[0, inf]
6	Term	Original term	[0, 1092]
7	Mort Adj	Adjustment from standard mortality	[-100%, 100%]
8	Val Int	Valuation interest rate	[0%, 10%]
9	Exp	Expenses per month	[0, 5000]
10	Exp Infl	Expense inflation per annum	[0%, inf]
11	PayM	Benefit at maturity switch Yes = 0, No = 1	[0, 1]
12	MortT	Mortality table switch AM = 0, SIM = 1	[0, 1]
13	SurrIs	Surrender switch Off = 0, On = 1	[0, 1]

TABLE 3.20: Input data formats

have been trained on many more combinations of pricing and valuation bases. It can therefore

Model	Training input Accurary	Bases	Input size	Used for
Z1	76 102 policies 99.7%	2 PB 6 VB	913 224 unique model points	Valuations Random data IFRS 17
Midway	810 policies (100s each product) 100%	4 PB 3 780 VB	12.247m unique model points	Valuations Random data Pricing IFRS 17 New products New mortality table

TABLE 3.21: Summary of models

much more comfortably handle different basis inputs. The Midway model also has more input features than the Z1 model, as detailed in Table 3.20.

The Midway model is more accurate than the Z1 model (e.g., 100% vs 99.7% accuracy on the training datasets), and it is only versions of the Midway model that can handle pricing, new products and a new mortality table.

3.14 Conclusion

Chapter 3 detailed the methodology employed to build and evaluate ML models for valuing a commercial book of WL, EA and TA insurance policies. The process was designed to ensure model accuracy, efficiency, and generalisability to new data, ultimately identifying the most suitable model for this application and informing future research.

In addition to rigorous accuracy testing, a series of investigations was conducted to understand the strengths and weaknesses of the models. This involved analysing feature importances and decision rules to identify key drivers of policy value and assessing model sensitivity to changes in input data, including policy premiums and various actuarial bases.

A recurring challenge encountered throughout the study stemmed from data quality. As with traditional actuarial valuations, accurate and appropriately labelled data is critical; however, the risk is potentially amplified in an ML context. Care must be taken to ensure that policy data, valuation bases, and reserves are accurately captured for each policy and aligned with the chosen pricing and valuation bases.

The results of the accuracy testing and investigations were used to evaluate model performance and identify areas for improvement. This culminated in the derivation of the Midway model, which demonstrates a promising combination of accuracy and efficiency for valuing life insurance business.

The prepared data, bases, and models will be further investigated and applied in Chapter 4.

Chapter 4

Results

4.1 Introduction

This chapter presents the results obtained from the application of the DNNs developed in this study, with a primary focus on evaluating their overall accuracy and comparing their performance to that of a traditional actuarial valuation model. The findings detailed herein will demonstrate the potential of ML to achieve comparable or improved accuracy in valuing a commercial book of insurance policies. The chapter uses the key performance metrics to assess model accuracy, as outlined in Chapter 3, to perform a detailed comparison of the results obtained from the DNNs and the traditional cash-flow based valuation model. This comparison will highlight the strengths and weaknesses of each approach.

The architecture used in training the Midway model is given in Table 3.18.

The four tests employed on the Midway model are discussed in the following sections, but a summary is provided in Table 4.1.

Context	Actual	Predicted	Difference	Accuracy
Z1 data (PB1)	8 328m	8 383m	-55m	99.34%
Z1 data (PB4)	13 939m	13 936m	-4m	99.98%
Random data distributions	1 752m	1 767m	-14m	99.16%
Premiums	6 482m	6 480m	-2m	99.99%

TABLE 4.1: Summary of Midway accuracy

4.2 Accuracy of the Midway model on Z1 data

This section details the performance of the Midway model when applied to the original input dataset, a crucial step in validating its efficacy. The primary objective was to assess the model's

accuracy and reliability when predicting reserves for a book of WL insurance policies. To ensure a robust evaluation, the model was tested on the input dataset - containing 71,602 policy records - which had not been used during the training process of the Midway model. Prior to evaluation, the data were normalised using the mean and standard deviation derived from the training dataset, ensuring consistency and comparability.

The overall accuracy of the model on this unseen dataset was determined to be 99.7%. This result is particularly encouraging, indicating the model's ability to generalise beyond the training data and provide reliable predictions for new policies. A more detailed breakdown of the model's performance across some different pricing and valuation bases is presented in Table 4.2.

PB/ VB	Actual	Predicted	Difference	Accuracy
PB1/ VB2	1 134m	1 131m	3m	99.74%
PB1/ VB3	1 671m	1 685m	-14m	99.16%
PB1/ VB4	1 752m	1 767m	-14m	99.16%
PB1/ VB5	1 859m	1 875m	-16m	99.16%
PB1/ VB6	1 912m	1 925m	-13m	99.34%
Total	8 328m	8 383m	-55m	99.34%

TABLE 4.2: Midway reserves on PB1 input data

Table 4.2 reveals a consistently high level of accuracy across various combinations of pricing and valuation bases. The table displays the actual reserves as calculated by the traditional actuarial methods, predicted reserves as calculated by the Midway model, the difference between the two, and the resulting accuracy for each scenario. The results demonstrate that the model consistently produces predictions that closely align with the actual values, with minimal deviations observed across the different combinations. Notably, the overall difference between the actual and predicted reserves is relatively small, indicating the model's stability and consistency.

Furthermore, Table 4.3 presents a more detailed analysis of the model's performance on the test dataset, categorised by run. Each run represents a distinct scenario, and the table provides a comprehensive overview of the actual reserves, predicted reserves, differences, and accuracy for each scenario. The results consistently demonstrate a high level of accuracy across all runs, with a negligible difference between the actual and predicted reserves. This further reinforces the reliability and consistency of the Midway model.

The prediction accuracy on the test dataset was 99.98%. This clearly shows that the model can accurately predict on any unseen model point and reflects interpolation over a smooth valuation surface.

Visual representations of the model's performance on the original input dataset are presented in Figures 4.1 to 4.3. These figures display the relationship between the actual reserves and predicted reserves, providing a clear visual indication of the model's accuracy. The figures consistently demonstrate a strong correlation between the actual and predicted reserves per

Run	Actual	Predicted	Difference	Accuracy
0	851.081m	881.202m	-30.121m	96.5%
1	1 238.169m	1 258.150m	-19.981m	98.4%
2	1 503.735m	1 519.726m	-15.990m	98.9%
3	2 257.167m	2 233.008m	24.595m	98.9%
4	2 290.497m	2 276.326m	14.171m	99.4%
5	2 607.077m	2 597.510m	9.567m	99.6%
6	3 191.737m	3 169.982m	21.754m	99.3%
Total	13 939.463m	13 935.904m	3.560m	100.0%

TABLE 4.3: Midway on PB4 Z1 data

model point, with the predicted values closely aligning with the actual values. This visual confirmation further supports the quantitative findings presented in Tables 4.2 and 4.3.

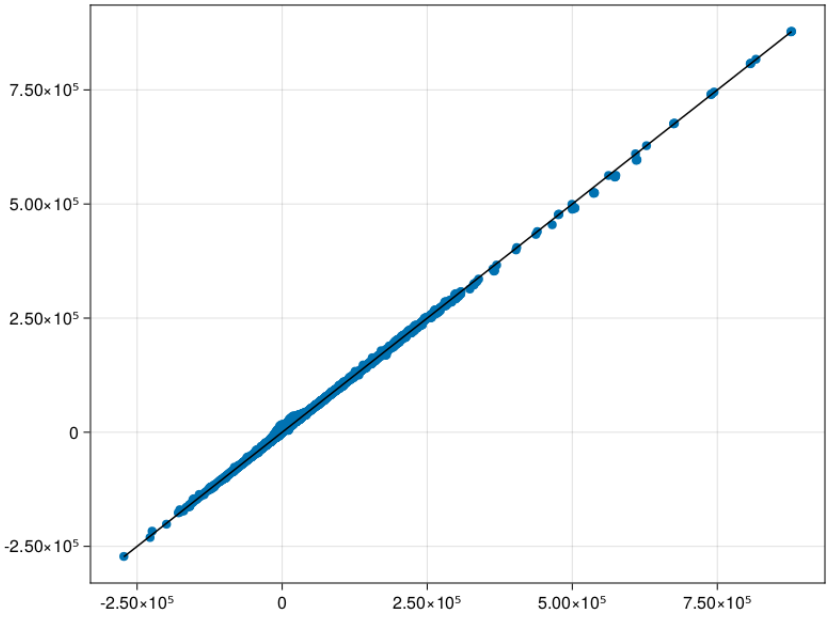

FIGURE 4.1: Midway AvE reserves day 0 on Z1 data

In conclusion, the results of this analysis demonstrate that the Midway model is capable of achieving a remarkably high level of accuracy in predicting reserves for a simplified WL insurance business. The model's ability to generalise beyond the training data and consistently produce reliable predictions makes it a promising tool for automating the valuation process. The following subsections will investigate more complex scenarios and explore the potential of this model in a wider range of applications.

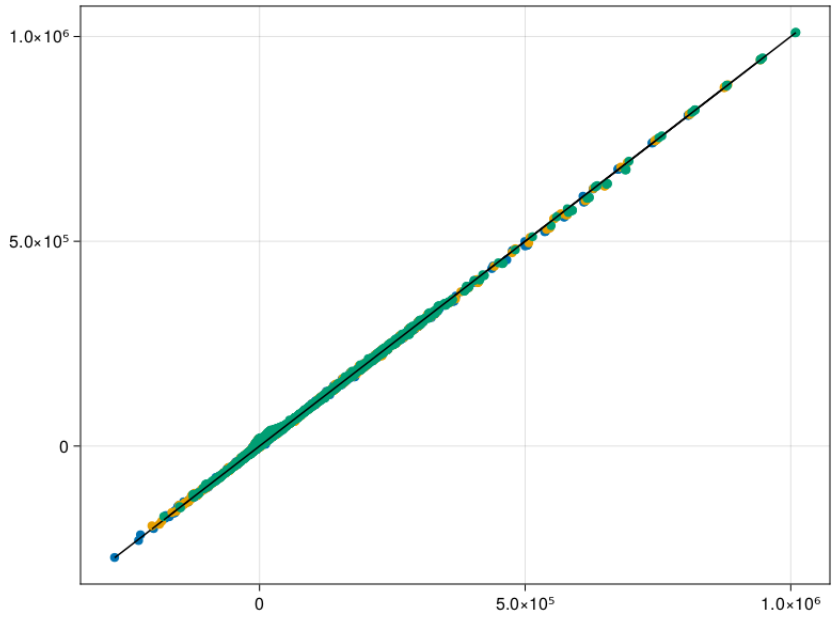

FIGURE 4.2: Midway AvE reserves years 0 to 2 on Z1 data

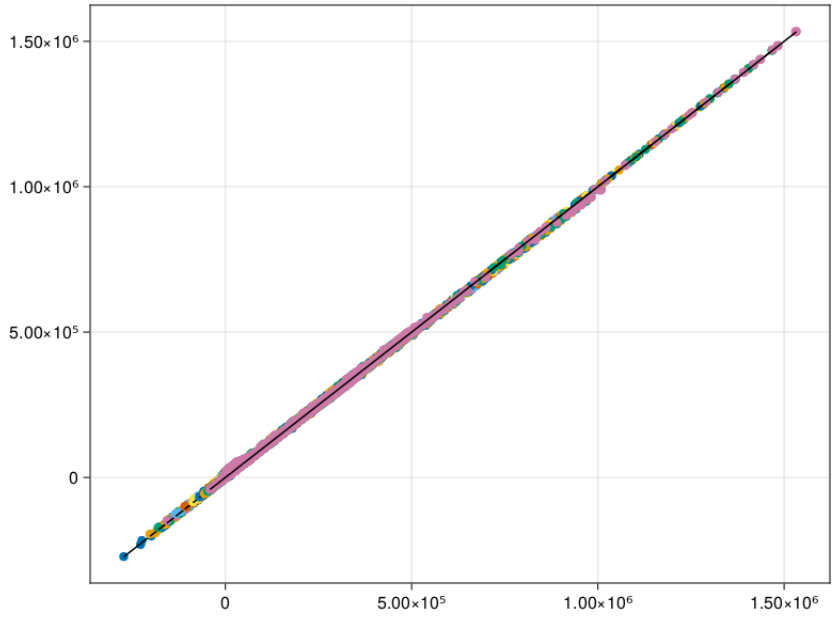

FIGURE 4.3: Midway AvE reserves all years on Z1 data

4.3 Comparing the Z1 and Midway models on random data distributions

This section details the comparative performance of the Z1 and Midway models when applied to a randomly generated dataset, intended to assess their robustness and ability to generalize

beyond the original training data. A dataset comprising 71,602 policy records was created with various distributions, while maintaining the total sum insured at a constant level. This approach ensures that the dataset reflects a realistic scenario while allowing for a rigorous evaluation of the models' predictive capabilities.

For clarity and detailed analysis, the results are presented in the form of Actual vs. Expected graphs, focusing on the individual data points and any deviations observed.

I then employed the Z1 and Midway models, as trained previously, on this new, unseen data. For the Z1 model, the results for the reserves are given in Figures 4.4 and 4.5.

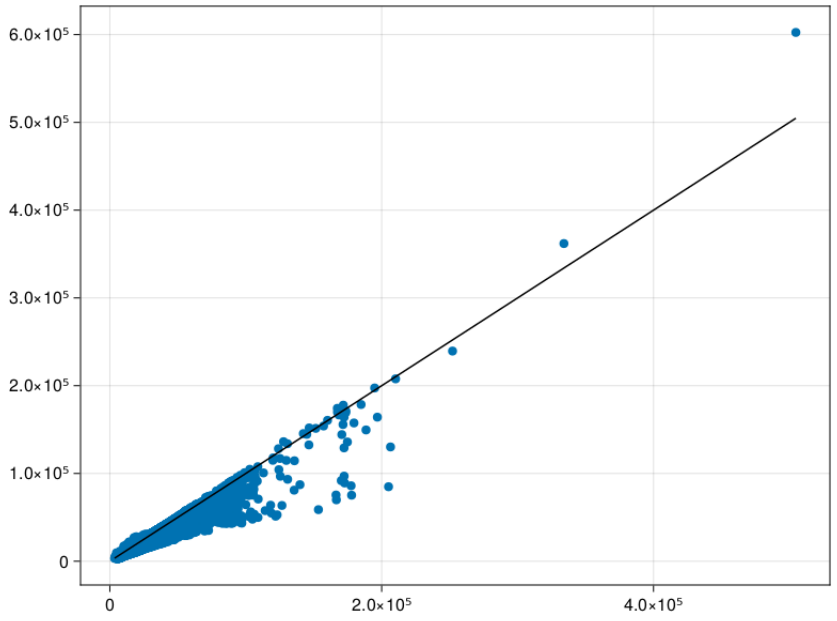

FIGURE 4.4: Z1 reserves day 0

As can be clearly observed, the model exhibited a diminished capacity to accurately predict the reserves for this randomised dataset, indicating a potential limitation in its ability to generalise to unseen data distributions. In contrast, the Midway model predicted the reserves accurately, as can be observed in Figures 4.6 and 4.7.

Notably, the Midway model maintained a high degree of accuracy in predicting the reserves, even when applied to this new and challenging dataset. This suggests that the Midway model possesses a greater capacity for generalisation and is less susceptible to the effects of data distribution changes.

This comparative analysis underscores the potential advantages of the Midway model in real-world scenarios, where the data distribution may vary from the original training data. The results suggest that the Midway model is a more robust and reliable option for accurately predicting reserves in a dynamic environment.

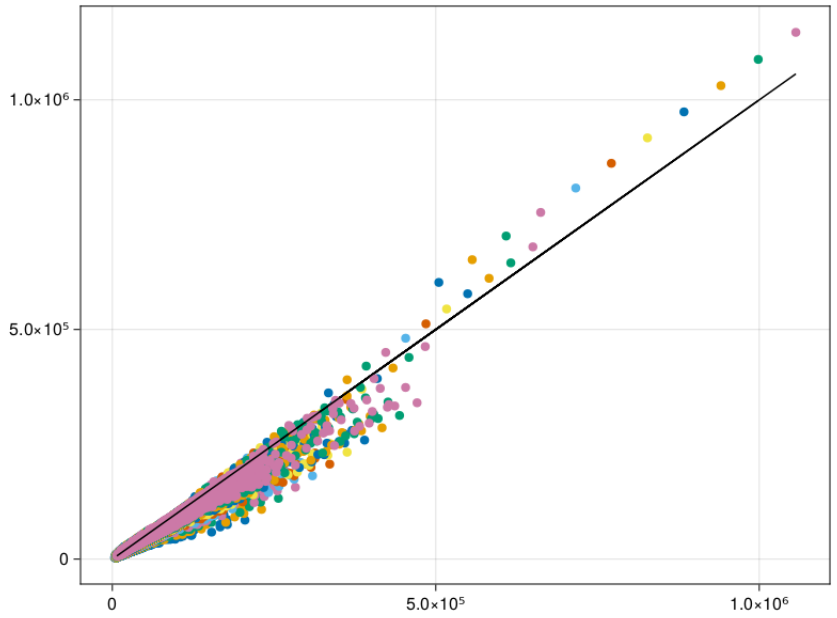

FIGURE 4.5: Z1 reserves years 0 to 10

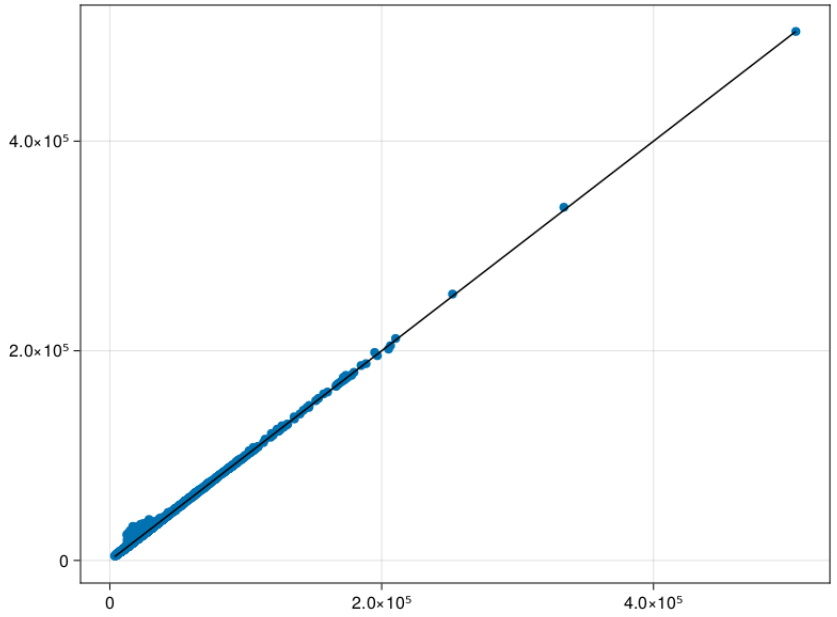

FIGURE 4.6: Midway reserves day 0

4.4 Results of Midway model including EA and TA business

This section presents a detailed analysis of the Midway model's performance when applied to a more comprehensive dataset, including WL, EA, and TA business. The goal is to assess the model's robustness and ability to accurately predict reserves across a wider range of insurance

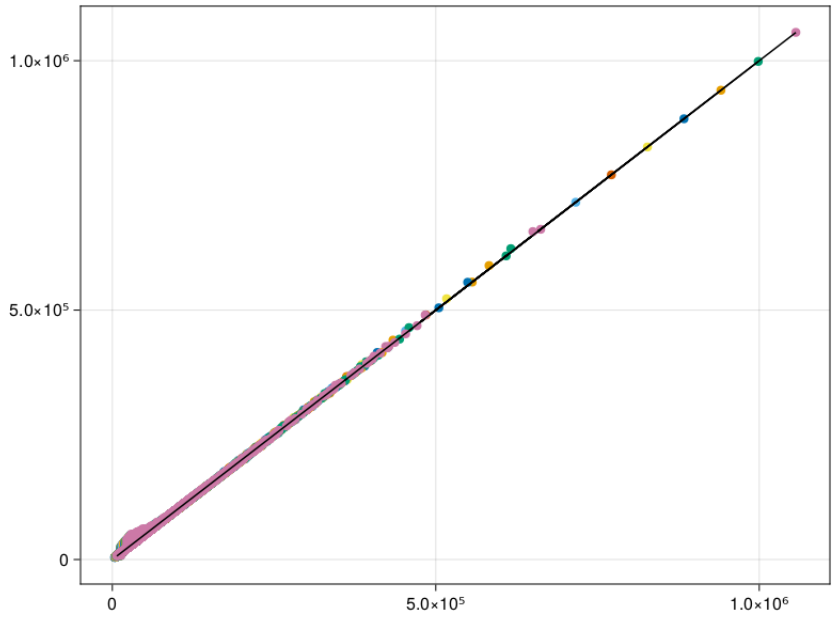

FIGURE 4.7: Midway reserves years 0 to 10

products. The analysis utilises a comprehensive dataset comprising 8 million unseen model points (70% of the data), allowing for a detailed evaluation of the model's predictive capabilities.

The overall accuracy of the Midway model on this previously unseen dataset was determined to be 99.7%, indicating a high level of predictive performance. To provide a granular understanding of the model's accuracy, the results are presented in Figures 4.8, 4.9, and 4.10, which depict the actual versus predicted reserves for each product line.

Figure 4.8 illustrates the model's performance on the WL dataset.

As can be clearly observed, the Midway model exhibits a strong correlation between actual and predicted reserves, with the points closely aligning along the diagonal line. This indicates a high degree of accuracy and reliability in predicting reserves for WL insurance policies.

Figure 4.9 presents the results for the EA dataset.

The Midway model's performance remains consistently high, with the points closely clustered around the diagonal line. This suggests that the model is capable of accurately predicting reserves for EA policies, even with the added complexity of the maturity benefit.

Figure 4.10 displays the results for the TA dataset.

While the Midway model generally exhibits good performance, a closer inspection reveals a slight deviation from the diagonal line. This indicates that the model may encounter some challenges in accurately predicting reserves for TA policies. The reduced accuracy can be

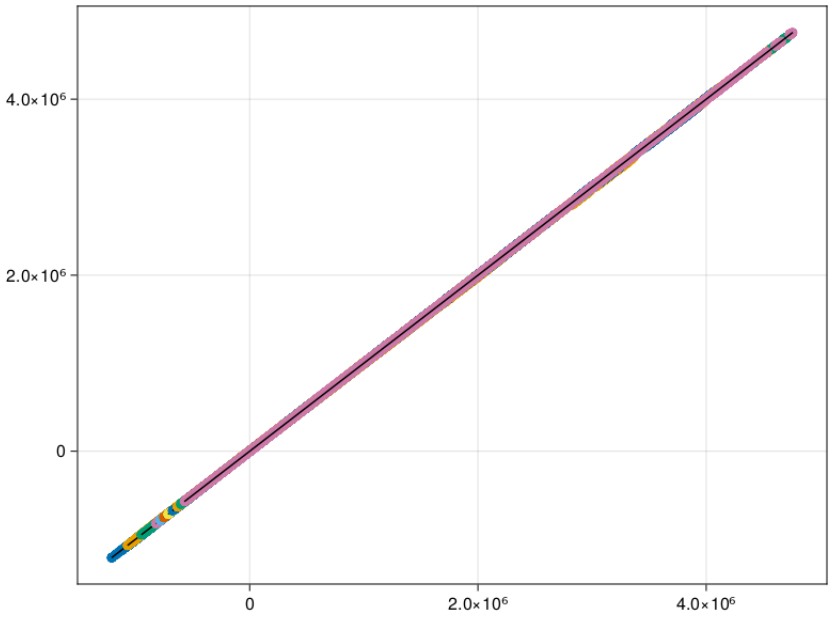

FIGURE 4.8: Midway AvE reserves on WL test dataset

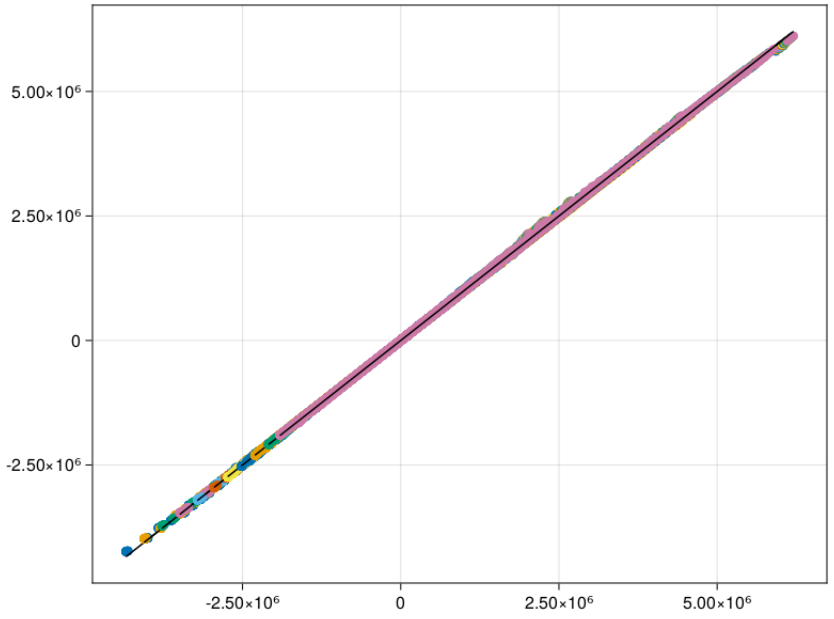

FIGURE 4.9: Midway AvE reserves on EA test dataset

attributed to several factors, including the lower reserves associated with TA and the relatively small proportion of model points in the dataset.

Noticeably, the Midway model had the most difficulty with TA products. It is due to the reasons mentioned previously: low reserves and proportion of model points. Importantly, the observed deviation in the TA dataset is not considered alarming. The overall accuracy remains at an

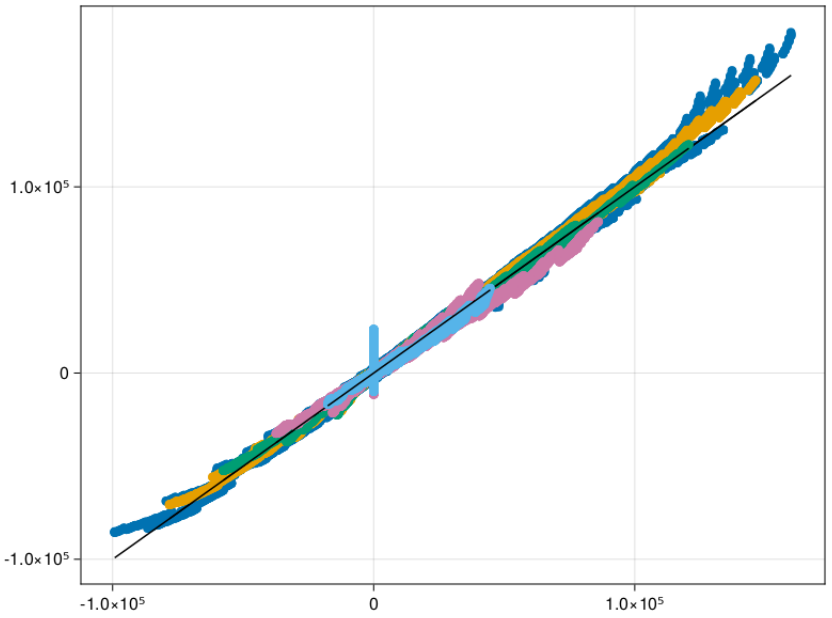

FIGURE 4.10: Midway AvE reserves on TA test dataset

acceptable level, and the results are consistent with the expected performance of the model. Further investigation may be necessary to identify and address any specific limitations or biases in the TA dataset at its extreme ends, especially.

In conclusion, the results demonstrate the robustness and versatility of the Midway model in predicting reserves across a wide range of insurance products. While the TA dataset may present some challenges, the overall accuracy remains high, indicating that the model is a reliable and potentially valuable tool for actuaries and insurance professionals.

4.5 Results when introducing the SIM92 mortality table

I introduced the SIM92 mortality table as a basis change and trained the model using an additional categorical variable only. I now test the Midway model using the test dataset only. This test dataset contains 2,858,566 model points for SIM92 mortality over WL, EA and TA products over all the PB and VB combinations. The actual versus predicted total reserves for all the years are given in Figure 4.11.

I conclude that the Midway model can easily incorporate different unseen demographic tables entered into the model via more categorical variables only.

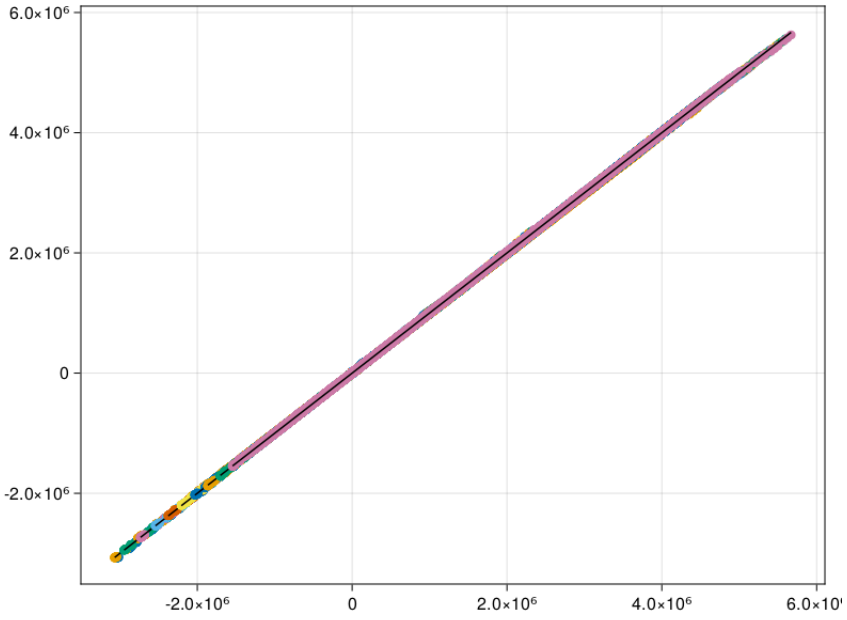

FIGURE 4.11: Midway reserves on test dataset for SIM92 mortality table

Dataset	Actual	Predicted	Accuracy
Train	4 670 414	4 669 387	100%
Validation	515 582	515 543	100%
Test	1 295 522	1 295 326	100%
Total	**6 481 518**	**6 480 256**	**100%**

TABLE 4.4: Midway model premium prediction on test dataset

4.6 Results of pricing for all products

This section presents the findings from applying the trained Midway model to predict monthly premiums, providing a final demonstration of its predictive capabilities. It is important to note that predicting premiums is a comparatively simpler task than calculating reserves, as the premium calculation directly serves as an input for the reserve calculation.

The model was trained using the 7 pricing bases on the Z1 dataset (comprising 532,714 model points), and the data was split into training (72%), validation (8%), and testing (20%) datasets. After 126 epochs, the model achieved 100% accuracy, indicating a robust and reliable predictive performance.

The results for each dataset are summarised in Table 4.4, which shows a near-perfect match between the actual and predicted values across all datasets. This demonstrates the model's ability to accurately predict premiums based on the input data.

Dataset	Actual	Predicted	Accuracy
Train	2 533 777	2 531 392	100%
Validation	269 697	269 587	100%
Test	713 914	712 792	100%
Total	**3 517 388**	**3 514 671**	**100%**

TABLE 4.5: Midway model premium prediction on PB4 to PB7 datasets

Additionally, Figure 4.12 visually depict the predicted premiums for the input dataset, further illustrating the high level of accuracy achieved.

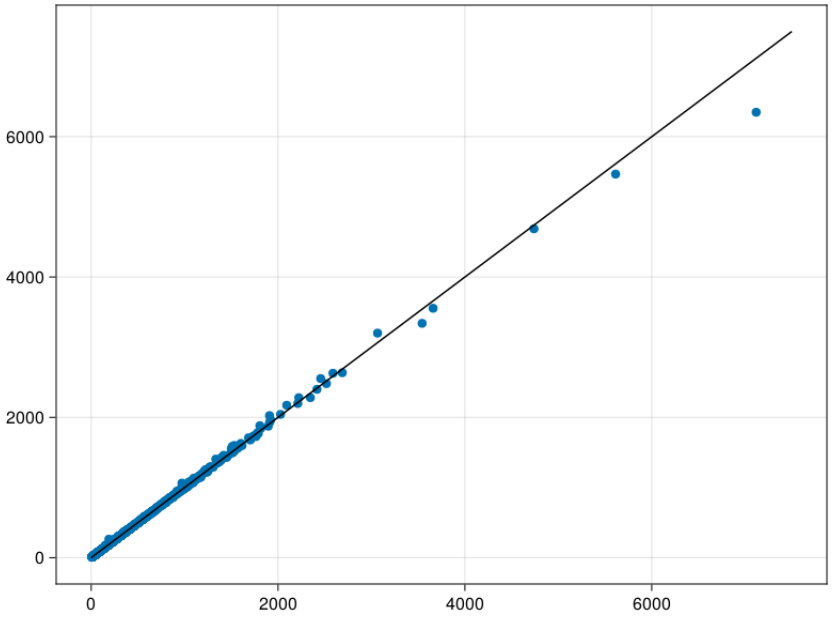

FIGURE 4.12: Midway predicted premiums on the test dataset

The DNN model had no issues in predicting premiums for the WL business; the outlier is a result of extreme values, in this case an area where not many inputs were provided.

To further validate the model's performance, the predictions were extended to include premiums for all products utilising pricing bases 4 through 7. This analysis encompassed a total of 3,240 model points, which were again split into 72%, 8%, and 20% train, validation, and test datasets. A summary is provided in Table 4.5.

Graphically, per product, the actual versus predicted premiums on the test dataset are shown in Figures 4.13 to 4.15.

This section concludes with a reiteration of the model's strong predictive capabilities. The consistent high accuracy across all datasets and the visual confirmation provided by the figures demonstrate the robustness and reliability of the Midway model in predicting premiums for these types of insurance products. The ability to accurately predict premiums is a crucial component

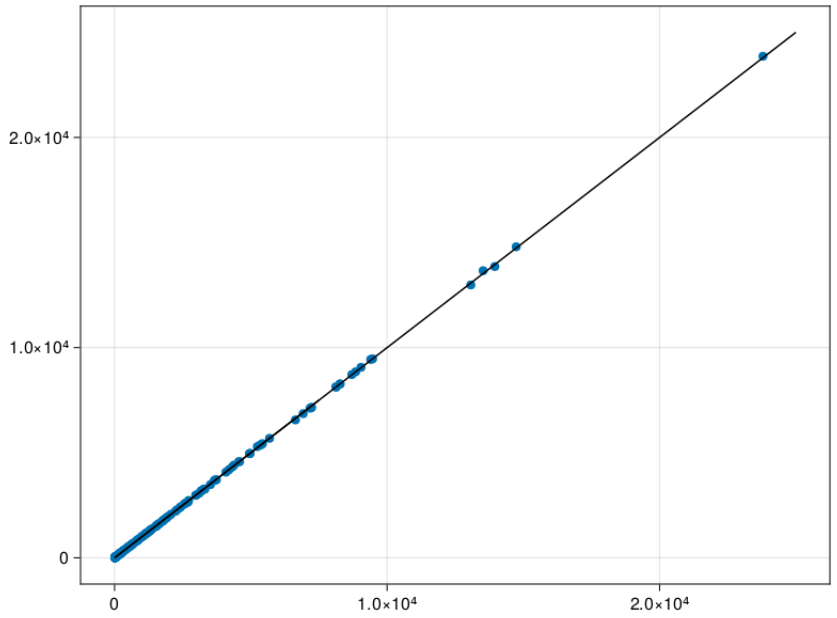

FIGURE 4.13: Midway predicted premiums for WL

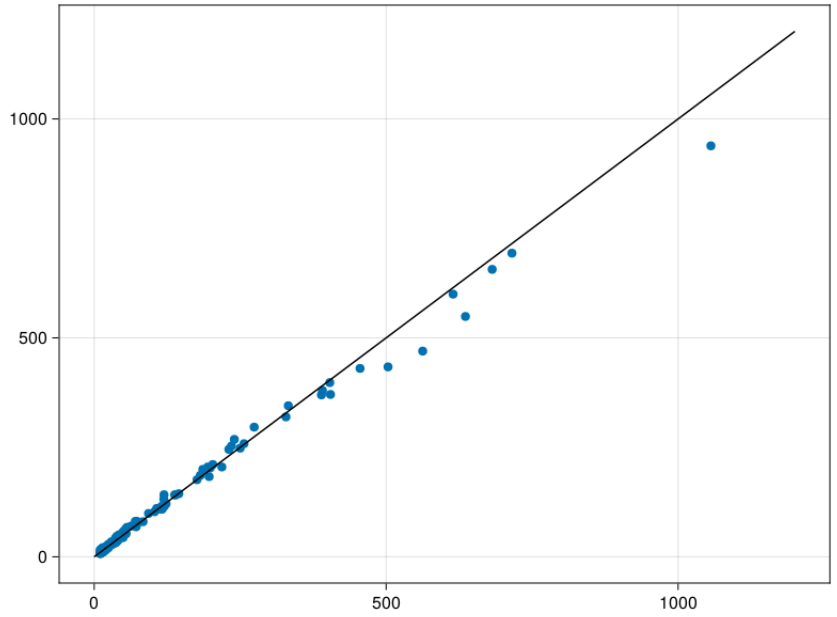

FIGURE 4.14: Midway predicted premiums for TA

of effective actuarial modelling and supports the broader application of the Midway model in various insurance contexts.

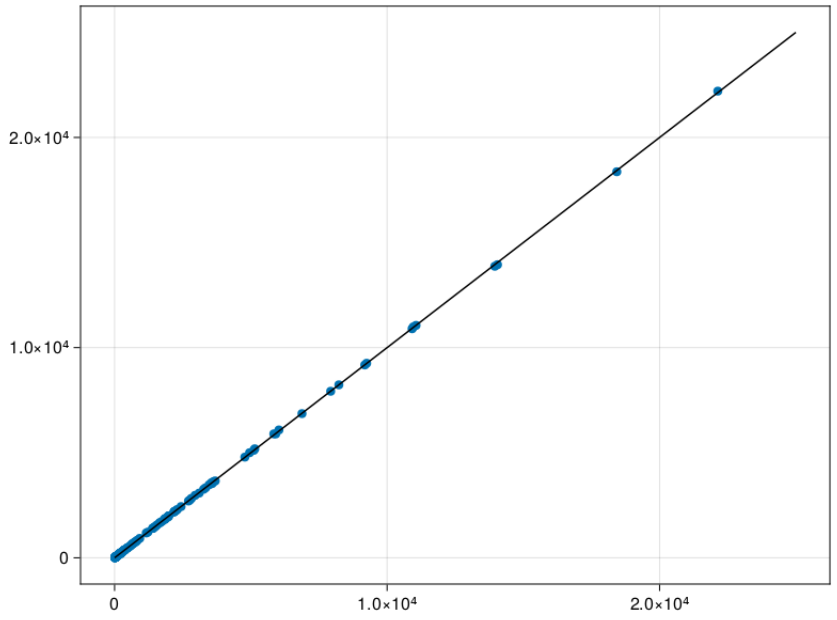

FIGURE 4.15: Midway predicted premiums for EA

4.7 Conclusion

Based on the results presented in this chapter, the following conclusions can be drawn regarding the performance of the Midway model:

1. Successful prediction of reserves and premiums: The Midway model consistently demonstrated a high degree of accuracy in predicting both reserves and premiums across all test datasets and criteria.

2. Robustness and generalisation: The model exhibited strong robustness and the ability to generalise to unseen data, consistently achieving near-perfect prediction accuracy across training, validation, and test datasets.

3. Superior performance compared to Z1: The results highlight the superiority of the Midway model over the Z1 model in generalising to new data, indicating that a focus on model architecture and broader data understanding leads to more reliable predictions.

4. Practical applicability: The consistent and accurate predictions achieved by the Midway model demonstrate its practical applicability as a viable tool for actuarial modelling and reserve calculation.

5. Potential for streamlined processes: The model's ability to produce fast and accurate predictions suggests the potential for streamlining actuarial processes and improving overall efficiency.

In conclusion, the results presented in this chapter strongly support the efficacy of the Midway model as a robust and reliable tool for predicting reserves and premiums. Its ability to consistently achieve high accuracy and generalise to unseen data makes it a promising candidate for further development and implementation in real-world actuarial applications. Further conclusions are drawn in the final chapter, Chapter 5.

Chapter 5

Conclusions and future work

5.1 Introduction

The data was analysed in Chapter 4, and the results of that analysis have informed the conclusions presented in this chapter. This research sought to explore the potential of ML techniques to address key challenges in the field of life insurance valuation, specifically focusing on the development of a robust and accurate model capable of pricing and valuing policies without relying on traditional actuarial formulas.

This chapter synthesises the key findings of the study, highlighting the implications for life insurance companies, ML methodologies, and the broader regulatory landscape. It also offers practical recommendations for auditors and regulators seeking to leverage these technologies to enhance their oversight of the insurance industry.

The following sections will delve into the specific contributions of this research, outlining the implications for data handling, model development, and the future of actuarial science. Ultimately, this chapter aims to demonstrate the transformative potential of ML in a field traditionally reliant on manual calculations and expert judgment, paving the way for an alternative yet efficient, accurate, and data-driven approach to insurance valuation.

5.2 Conclusions on the design of the models

An initial model, designated Z1, was developed to fit the input dataset. However, with broader research objectives in mind, a more versatile model, designated Midway, was also developed on a synthesised dataset via Kriging principles. The Midway model was used to predict reserves on the unseen original input data and demonstrated strong performance.

A summary of the critical analysis of the performance of neural networks to replicate deterministic calculations in a range of scenarios, is provided in the following list:

- Section 4.2 - the Midway model was able to produce similar results than the original dataset on PB1 and 5 different VBs (where it did not train on any of the data or assumptions),
- Section 4.3 - the Midway model produced near perfect results on random data, while the Z1 model struggled to obtain accurate fits on all model points,
- Section 4.4 - the Midway model produced near perfect results on an expanded test dataset for WL and EA business, but struggled a bit more with TA business due to it being under represented in the training data, and
- Section 4.5 - the Midway model had no problem when an additional mortality table was introduced.

From the above list, it is clear that a traditional NN model trained on an input dataset that is the raw data and using incorrect hyperparameters, will not be able to obtain the desired replication capabilities needed in a production environment.

Key lessons learned interpreting the output of the Z1 and Midway DNNs are presented in the next sub-sections. Further details regarding the development of the Z1 model are provided in Appendix B. Albeit that the Z1 model is inferior in all aspects to the Midway model, the models derived prior to the Z1 model are even worse. The Table 5.1 showcase the deficiencies of blindly trusting (parts of) the existing literature (when deriving the Z1 model), where the Model (number) refers to the model trained in Appendix B.

Eventually, a DNN, designated Midway, was trained to exhibit satisfactory performance. The following subsections summarise important lessons learned during this process.

5.2.1 Model infrastructure in Julia

For non-categorical variables, Float32 values were utilised. This not only improved memory efficiency compared to Float64 values but also resulted in an acceleration of training by approximately a factor of two. While a slight reduction in precision was observed, this was considered acceptable, given the rounding of premiums and reserves to the nearest second decimal place, which introduced a comparable level of imprecision.

Numerous reusable functions were developed in Julia to facilitate future model iterations and reproducibility. A powerful framework for building and training deep learning models, Flux.jl,

Model	Deficiencies
1	Limited architecture, incorrect optimiser, incorrect loss function, limited basis, RELU not as efficient as CELU
2	Incorrect architecture, incorrect optimiser, incorrect loss function, limited basis, RELU not as efficient as CELU, too small batch size
3+4	Incorrect architecture, incorrect optimiser, limited basis, RELU not as efficient as CELU, too small batch size
5	Incorrect architecture, limited basis, RELU not as efficient as CELU
6	Incorrect architecture, limited basis, RELU not as efficient as CELU, insufficient batch size
7+8	Incorrect architecture, limited basis, RELU not as efficient as CELU, incorrect batch size
9	Limited basis, RELU not as efficient as CELU, incorrect batch size
10	Limited basis, RELU not as efficient as CELU, limited batch size
11	RELU not as efficient as CELU, limited batch size
12+13+rest	Incorrect loss function, RELU not as efficient as CELU, limited batch size
14+15	RELU not as efficient as CELU, limited batch size
20	Z1 model

TABLE 5.1: Deficiencies in existing literature

was utilised. This framework provides tools for creating extensible training loops and implementing callbacks for common use cases, such as hyperparameter scheduling, metrics tracking, and checkpointing.

Early stopping mechanisms and patience parameters were implemented to improve training efficiency. These tools allowed for the monitoring of performance and the termination of training when improvement plateaued or overfitting commenced. Built-in utilities provided by Flux.jl, such as early_stopping and plateau, were leveraged for these purposes.

Several additional Julia utilities could have been employed to further enhance the model development process, but are left for future enhancements:

1. AdversarialPrediction.jl: to optimise generic performance metrics in supervised learning settings.
2. MLMetrics.jl: to score models and for evaluating performance.
3. ValueHistories.jl: for efficient tracking of optimisation histories and training curves.

An increase in layers leads to a more complex model, requiring a longer training period. It is important to note that a neuron of any prior layer can map directly to any neuron only in the subsequent layer. However, given the utilisation of a GPU for training, the training of complex models was significantly accelerated by the availability of multiple cores. Consequently,

a strategy of initially training more complex models - to identify at least a functional model - was adopted, followed by a reduction in layers to determine a simpler, more efficient model.

DNNs are prone to vanishing (or exploding) gradients due to the inherent method by which gradients (or derivatives) are computed layer by layer in a cascading manner, with each layer contributing to exponentially decreasing or increasing derivatives. Weights are adjusted based on gradients to reduce the cost function or error. Very small gradients can prolong the training process, whereas large gradients can lead to overshooting and divergence. This issue is exacerbated by some non-linear AFs, such as Sigmoid and TANH, which restrict outputs to a small range. Consequently, changes in weight can have a minimal effect on output, potentially extending training time. This problem can be mitigated through the use of linear AFs and proper weight initialisation.

By default, a NN is initialised with random uniform weights for its parameters. To increase randomness, initial values can be randomised with a Gaussian distribution. Especially when dealing with a large number of parameters due to a large number of layers and neurons per layer. Various distributions can be assigned to determine these values. In this study, weights were initialised using a Kaiming uniform adjustment (He et al., 2015). See Chapter 2 for a discussion of the other methods most commonly used.

5.2.2 Learning rate, batch size and momentum

It is not advised to utilise a constant LR. From time to time, the LR requires adjustment to facilitate faster model progress. For example, if minimal changes in the LF are observed over a period of 100 iterations, an increase in the LR would be appropriate. If model stagnation is detected, increasing the LR may allow the model to overcome this barrier.

It is recommended to begin with a lower LR. This allows the training algorithm to avoid moving too rapidly in a single direction, which can lead to a model that does not generalise well. In extreme cases, if the LR is too high, the model may effectively cheat by calculating average reserves and assigning that value to all policies - a simplification that bypasses the goal of accurate prediction. Employing the LF as MSE can also lead to the same conclusions.

The LR interacts with the batch size utilised in SGD. It was found that a lower LR performs better with larger batch sizes, and vice versa. The rationale is that a larger batch size is more likely to contain model points that significantly pull the model in the correct direction. If an overshoot occurs, a longer training period may be required.

The LR must also be considered in conjunction with momentum. Momentum determines the extent to which previous changes are incorporated into the current calculation and is influenced by the direction of the largest changes, which are loosely impacted by model points contributing

most to the losses. As the model converges towards a general solution, clusters of outlier policies can be identified more effectively with increasing momentum. Since these clusters become smaller as training progresses, a higher LR may be required to move predictions closer to actual values at a faster rate.

Therefore, LR and momentum were adjusted in combination; when momentum was increased, LR was also increased.

5.2.3 Batch sizes

Batch sizes of 512 or 1,024 were utilised for training the Z1 model. Due to the significantly larger input size of the Midway model-consisting of millions of model points - initial training employed a batch size of approximately 200,000. Following the introduction of TA and EA products, the batch size was decreased to 10,206. Further reduction in batch size resulted in increased bias and variance, consistent with findings reported by Qian and Klabjan (2020).

Consequently, a warm-up exploratory training schedule utilising a low batch size was investigated. However, consistent positive results could not be obtained with small batch sizes, as they tended to produce high bias and lead to convergence in local minima. Therefore, a constant batch size was maintained throughout the training process.

5.2.4 Confidence intervals

Due to the stochastic nature of weight initialisation, batch training, and gradient calculations, two models trained with identical infrastructure will not necessarily converge to the same solution. Training the same model multiple times, however, generates an ensemble of models. With an ensemble, the mean and standard deviation of reserve estimates can be calculated, providing a confidence interval for the reserve. Previous studies, highlighted in Chapter 2, discussed utilising the mean of multiple models to obtain a single estimate and the standard deviation of multiple models to obtain confidence intervals.

The Midway model demonstrated a degree of robustness to retraining, potentially attributable to the stricter stopping criteria defined in Section 3.11. Consequently, while the approach of training numerous models and predicting averages or confidence intervals may be viable when a model is trapped in a local minimum, it was deemed unnecessary for this study, as a clear absolute local minimum was attained.

5.2.5 Failure to converge

Several factors can contribute to a failure to converge. Insufficient nodes within the network may limit its capacity to adequately model the data, necessitating substantial architectural changes and potentially hindering convergence.

Stepwise removal of input variables is also a potential diagnostic technique. For example, a cohort of policies all of the same age could be used to assess whether the model can accurately calculate reserves without considering age as a predictor.

Low volumes of training data or the presence of corrupted or improperly collected data can also impede convergence.

The choice of activation function, while often yielding positive results, may be unsuitable for models with higher complexity, potentially leading to non-convergence.

Inappropriate weight initialization or application within the network can also contribute to this issue. Weights must be carefully calculated and aligned with the chosen activation function.

Finally, the LR parameter should be moderate, avoiding both excessively high and excessively low values.

5.2.6 Individual policy investigations

During the initial training of the Z1 model, it was consistently observed that high overall training accuracy did not necessarily correlate with high accuracy at the individual policy level, and vice versa. The models appeared to oscillate between these two extremes, preventing convergence.

A consistent relationship between overall and individual policy accuracy was sought, with the understanding that such models would function correctly and potentially be improved through architectural adjustments.

When the MAE was used as a loss function, instances were observed where large absolute errors existed despite a decreasing overall loss. This indicated that while some reserves were being accurately estimated, others were not.

Normalisation of continuous variables required ensuring that their sum equalled zero and that both the validation and test sets were normalised using the mean and standard deviation derived from the training data.

With these aspects addressed, attention turned to identifying areas where the proportion of incorrectly predicted reserves was high. An attempt was then made to establish any relationships or common characteristics among those model points, in order to determine whether specific

variable values might be contributing to incorrect calculations. This analysis led to the decision to incorporate additional premium and valuation bases and to apply Kriging to the policy data inputs. The identification of such variables would allow for targeted adjustments in how those values were incorporated into the model.

Once the above issues are resolved, hyperparameter tuning can be undertaken to achieve final convergence.

5.3 Important implications

5.3.1 Implications on life insurance companies

This research demonstrates the feasibility of constructing a DNN capable of accurately pricing and valuing life insurance policies, achieving this without reliance on explicit actuarial formulas, pre-defined relationships between variables and reserves, or access to traditional actuarial tables (mortality, surrender rates, etc.). The model achieves this with only minor adjustments to input variables (normalisation), indicating a potential for significant simplification in the valuation process.

This research suggests that a model trained on suitable intervals of policy and valuation basis features offers the optimal long-term solution for valuation purposes. Such a model not only benefits insurance companies but also provides a robust and transparent framework for auditors and regulators.

A proof-of-concept for forecasting future reserves has been presented, demonstrating a potentially valuable application for IFRS 17 compliance.

The ease with which additional products can be integrated and valued without model modification further underscores the potential for streamlining valuation processes and reducing operational costs.

It is anticipated that both auditors and regulators would benefit from a ML model capable of quickly assessing the sensitivity of reserves to changes in underlying assumptions and valuation bases. Such a tool would facilitate more efficient comparative analysis across different companies and valuation methodologies.

Finally, this research suggests the potential for insurance companies to develop internal models to enhance the accuracy of reserve calculations, improve the efficiency of AOS processes, and inform IFRS 17 strategies. Notably, the model's adaptability to diverse data sources minimises the challenges associated with data availability.

This work may contribute to the emergence of a specialised role - the ML Actuary - capable of leveraging internal data and valuation bases to develop and maintain sophisticated DNN models.

5.3.2 Implications on machine learning methods

This research demonstrates that a purely data-driven approach, relying solely on the input dataset for model training, may not be optimal. When faced with suboptimal predictive performance, a more effective strategy involves identifying areas of model weakness, augmenting the training data in those specific regions, and subsequently retraining the model.

The limitations of a purely data-driven approach become apparent when evaluating the model's generalisability. While high accuracy may be achieved on a dataset resembling the training data, performance can degrade significantly when confronted with unseen data or larger-scale test sets. This highlights the critical importance of addressing the representativeness of the training data and actively mitigating the risk of overfitting. Ultimately, the predictive power of the model is intrinsically linked to the extent to which the training data accurately reflects the underlying range of possible values.

Furthermore, this research emphasises the importance of reproducibility in ML research. The consistent use of fixed random seed values (10101 in this case) and custom-developed code facilitates independent verification and strengthens the reliability of the findings.

The study implicitly suggests the importance of thoughtful feature engineering. Identifying and incorporating domain-specific knowledge can significantly enhance model performance and interpretability.

Given the relatively nascent stage of applying ML techniques to actuarial science, this study aims to encourage further research in this field. A key principle underpinning this work is the notion of ***teaching the model the rules of the game, not the specific dataset.*** By focusing on developing models that capture the underlying principles governing actuarial data, rather than simply memorising specific examples, it is possible to achieve greater generalisability, robustness, and long-term performance. This approach encourages the development of models that are capable of adapting to changing circumstances and evolving data or changes in the external environment.

5.3.3 Valuation data for machine learning

This section is particularly relevant given the challenges encountered with data quality and representativeness during the training process! Effective handling of valuation data is critical for the successful implementation of ML models in this domain.

Summary of key considerations:

- Rigorous data cleaning and validation are essential to address inaccuracies, inconsistencies, and missing values.
- Ensuring the training data adequately represents the full spectrum of policyholder characteristics, economic conditions, and valuation bases is paramount.
- Careful selection and transformation of input variables are crucial for capturing the underlying relationships between policy features, risk factors, and reserve calculations.
- Techniques such as synthetic data generation can be employed to address data scarcity and improve model generalisation.

By addressing these considerations, it is possible to mitigate the risks associated with data quality and ensure the reliability and accuracy of ML models used for insurance valuation.

5.3.4 Advice for auditors and regulators

To enhance the efficiency and effectiveness of regulatory oversight, auditors and regulators should consider leveraging ML technologies to analyse insurance data. This involves gathering data from a diverse range of insurance companies, encompassing various policy types, valuation bases, and risk profiles.

This aggregated data can then be used to train a DNN capable of modelling the complex relationships between policy features, risk factors, and reserve calculations. By selecting data across the entire spectrum of insurance companies and valuation bases, the DNN can provide a comprehensive and unbiased assessment of reserve adequacy.

Specifically, DNNs can be used to:

1. Identify outliers and anomalies: Detect instances where reserves deviate significantly from expected values.
2. Assess the reasonableness of assumptions: Evaluate the validity of key assumptions used in reserve calculations.

3. Monitor trends and patterns: Track changes in reserve levels over time and identify emerging risks.

4. Automate regulatory reporting: Streamline the process of collecting and analysing insurance data.

5. Improve risk-based supervision: Focus regulatory resources on areas of highest risk.

By embracing these technologies, auditors and regulators can enhance their ability to effectively supervise the insurance industry and protect policyholders' interests. This proactive approach will contribute to a more stable and resilient financial system.

5.3.5 Qualification of scientific contribution

While the preceding chapters, sections and sub-sections emphasise the practical implications of the proposed framework for actuarial valuation, it is important to explicitly qualify the nature and scope of the scientific contribution made in this study. The contribution is primarily *applied* and *methodological* rather than purely theoretical. The value added by this work lies in the integration, adaptation, and empirical evaluation of existing theoretical constructs within actuarial science, rather than in the development of new abstract theory.

The methodological innovations introduced - particularly in the modelling workflow, the data-driven calibration procedures, and the demonstration of their viability across realistic actuarial scenarios - extend current practice by showing how established techniques can be systematically tailored to contemporary valuation challenges. These advances have direct practical significance for practitioners seeking robust and implementable approaches, but they also constitute a scientific contribution in their own right by clarifying how theoretical principles operate under real-world constraints.

5.4 Limitations of this study

With reference to Section 3.12, the following provide a summary of the limitations of this study:

- Categorical variables,
- Products and features, and
- ML methods employed.

If more categorical variables are included in the data, it increases the solution space with a factor of 2^n, where n is the number of categories for each categorical variable. If more products or features are included, this also increases the solution space by a similar factor. In both cases, the network architecture should be expanded by including more width and depth to the DNN architecture, and further training is therefore required.

This study only considered 3 distinct product ranges, where more product ranges are required, addition training on those ranges and their features will be required. This include expanding the width and depth of the DNN.

This study only considered DNN models (Z1 and Midway).

5.5 Possible future research opportunities

This study was based on a single dataset of commercial whole life business in Europe. It could be extended:

- to more insurers in different countries,
- for different periods and dates,
- Incorporating XAI techniques could help to shed light on the decision-making processes of the DNN, enhancing trust and facilitating regulatory compliance,
- on even more valuation bases, and
- on more diverse long-term insurance products.

This study can be repeated on reserves that were calculated on stochastic methods.

5.6 Mapping the aims and objectives to relevant sections in this study

To ease the burden on the reader, this section was included as a high-level map of where the research objectives are explicitly dealt with in this study. The aims and objectives are copied from Section 1.2 of this study.

The micro-level objectives are addressed as follows:

1. Identify the most informative features within the valuation data: Various model input was considered, and this was dealt with for both the Z1 and Midway models in Chapter 3. This was collaborated by a PCA in Section 3.6.7, where it was confirmed that the policy data and bases items are all important features when determining reserve values.

2. Develop a mechanism for learning the complex inter-feature relationships that drive reserve values: This was done in this study via a DNN named Midway.

3. Choose the most appropriate hyperparameters for the DNN: The eventual and most appropriate list of hyperparameters used are given in Table 3.18 and contrasted to other models in Table 2.2.

4. Design a data-grouping strategy (e.g., sampling and mini-batches) that maximises the utility of the training set for the DNN: The impact of batch size and using mini-batches or other related methods were discussed in Sections 2.3.4 and 3.11.1. Eventually, for the Midway model, a Kriging approach was preferred with eventual ranges disclosed in Table 2.1.

The macro-level objectives are addressed as follows:

1. Compile a tabular catalogue of critical features for any supervised regression model that could be applied to actuarial valuations: the data features and model infrastructures are developed throughout Chapters 2 and 3.

2. Develop a transfer-learning framework that allows insurance organisations to build their own internal models with minimal bespoke coding: I have explained in as much detail possible all the steps I took. Additional material is covered in Annexures A, B and C.

3. Quantify the model flexibility with respect to:

   - Changes in valuation assumptions (e.g. discount rates, mortality tables): introduced over 3000 different valuation bases, with different discount rates, and introduced a new mortality table SIM92 for the Midway model.
   - temporal drift: with the Midway model, developed a DNN that calculates reserves not only at the start, but also at the start of each year for the first 10 years.
   - novel features in products: considered different product ranges EA and TA.

I dealt with the aims of the study as follows:

1. Accurately estimate reserves for life-insurance products across a broad policy space: Calculated reserves using the Midway model is never less than 99.5% accurate.

2. Provide per policy values, when seen under several valuation bases, that are comprehensible to actuaries and regulators: This is provided through numerous per policy scatter plots in Chapters 3 and 4.

3. Adjust seamlessly to changes in actuarial assumptions and product inputs, enabling its adoption as an industry-standard tool: By training on 3000 plus valuation bases, the Midway model can easily handle any feasible valuation basis.

5.7 Final words

This study aimed to investigate the potential of DNNs to address key challenges in actuarial modelling, specifically focusing on the accurate prediction of reserves and premiums for life insurance products. Based on the findings presented, the following conclusions are drawn:

1. Successful Development of DNN Models: This study successfully developed DNN models capable of addressing all stated research questions with a high degree of accuracy and reliability. The models demonstrated the ability to predict reserves and premiums, offering a viable alternative to traditional actuarial methods.

2. Performance and Efficiency: The trained DNN models demonstrated the ability to produce fast, reliable, and comparable results to traditional actuarial methods. This highlights the potential for DNNs to significantly improve the efficiency of actuarial calculations and decision-making processes.

3. Generalisation and Robustness: While the Z1 model exhibited limited ability to generalise to unseen data, the Midway model demonstrated superior performance in predicting outcomes for previously unseen data within the trained range. This highlights the importance of developing models that prioritise generalisation and robustness over mere data fitting.

4. Advocacy for General Models: This research strongly advocates for practitioners to prioritise the development of general models, such as the Midway model, over models solely focused on fitting the training data. Developing models with strong generalisation capabilities is crucial for ensuring accurate predictions in real-world scenarios with evolving data patterns.

5. Analogy to Understanding the Rules of the Game: This approach can be likened to training a model in a larger, more diverse data environment - akin to understanding the underlying rules of the game rather than simply counting individual data points. By focusing on developing models that understand the fundamental principles governing the data, we can create more robust predictions in the future.

In conclusion, this research provides compelling evidence for the potential of DNNs to revolutionise actuarial modelling. The developed Midway model offers a viable alternative to traditional methods, providing the opportunity to improve efficiency and gain deeper insights into the complex dynamics of insurance risk. The findings presented in this thesis contribute to the growing body of knowledge surrounding the application of ML in the actuarial profession and pave the way for further research and innovation in this exciting field.

Appendix A

Pseudo Code

A.1 Introduction

This chapter gives high-level overviews of the logic behind various methods.

A.2 Finding the optimal model

This contains two parts, a structure to keep the hyperparameter variables stored, and loops to generate all the feasible candidates. This is employed recursively during training.

A.2.1 Structure for storing variables

```
abstract type ML_INPUT end
mutable struct MlInput1 <: ML_INPUT
    modno::Int64
    layer1n::Int64
    layer_act
    loss
    optimiser
    batchsize::Int64
    epochs::Int64
    opt_eta::Float64
    opt_beta::Tuple{Float64 , Float64}
    loss_f::Bool
    retrain::Bool
end
```

A.2.2 Looping through candidate models

Algorithm 1 Investigate Viable Models

Input: ModelSet $M = \{M_1, M_2, \ldots, M_n\}$, Dataset D, EvaluationCriteria C
Output: SelectedModels S
$S \leftarrow \varnothing$
for each model $m \in M$ **do**
    Train m using D
    Evaluate m using C, denote result r
    if r meets threshold **then**
        Add m to S
    end if
end for
Rank the models in S by performance
return top k models from S

A.3 Custom loss function

Algorithm 2 Custom Loss Function

function Loss(x, y)
    Input: input data x; target values y
    Output: loss value L
    $\hat{y} \leftarrow \text{model}(x)$
    $d \leftarrow broadcast\big(|\hat{y} - y|\big)$ ▷ element-wise absolute difference
    $L \leftarrow \sum d$
    return L
end function

A.4 Creating random data

Algorithm 3 Generate Random Dataset

Input: N (number of records), p (number of numeric variables), q (number of categorical variables), $D_{\text{num}} = \{ D_{\text{num},1}, \ldots, D_{\text{num},p} \}$ (numeric distributions), $L_{\text{cat}} = \{ L_{\text{cat},1}, \ldots, L_{\text{cat},q} \}$ (sets of categorical levels)
Output: Dataset D of size $N \times (p + q)$
Initialize $D \leftarrow$ empty dataset with N rows and $(p + q)$ columns
for $i \leftarrow 1$ to N **do**
    for $j \leftarrow 1$ to p **do**
        dist $\leftarrow D_{\text{num},j}$
        $D[i, j] \leftarrow$ SampleFrom($dist$)
    end for
    for $k \leftarrow 1$ to q **do**
        levels $\leftarrow L_{\text{cat},k}$
        $D[i, p + k] \leftarrow$ RandomChoice($levels$)
    end for
end for
return D

A.5 Working with categorical data

Example product where there are three types A, B and C.

Algorithm 4 Encode Categorical Feature - Products

Input: DataFrame D with a categorical column `product` whose values E $\{A, B, C\}$
Output: DataFrame D' with new binary columns `prod_A`, `prod_B`, `prod_C`
for each row i in D **do**
    $v \leftarrow D[i, \texttt{product}]$
    if $v = A$ **then**
        $D'[i, \texttt{prod\_A}] \leftarrow 1$; $D'[i, \texttt{prod\_B}] \leftarrow 0$; $D'[i, \texttt{prod\_C}] \leftarrow 0$
    else if $v = B$ **then**
        $D'[i, \texttt{prod\_A}] \leftarrow 0$; $D'[i, \texttt{prod\_B}] \leftarrow 1$; $D'[i, \texttt{prod\_C}] \leftarrow 0$
    else if $v = C$ **then**
        $D'[i, \texttt{prod\_A}] \leftarrow 0$; $D'[i, \texttt{prod\_B}] \leftarrow 0$; $D'[i, \texttt{prod\_C}] \leftarrow 1$
    else
        error: unknown category v
    end if
end for
return D'

A.6 Stop Training on Plateau

The following function is only called when the amount of history is more than 20.

Algorithm 5 Stop Training on Plateau — StopTrain

```
function StopTrain(history, incr)
    Input: history ← vector of loss or metric values (type Float32)
      incr ← decrement step (type Float32)
    Output: Boolean flag stop
    k ← 1
    hist1 ← 0
    hist2 ← mean( history )
    incrd ← 0
    for each value h in history do
        hist1 ← hist1 + h × k
        incrd ← incrd + k
        k ← k − incr
    end for
    hist1 ← hist1 / incrd
    if hist2 > hist1 then
        return true                ▷ Stop training: metric has plateaued/worsened
    else
        return false
    end if
end function
```

A.7 Training of the model

Algorithm 6 Train Neural Network

Input: Dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, number of epochs E, mini-batch size m, initial model parameters θ, learning rate η
Output: Trained model parameters θ
for epoch $= 1$ to E **do**
    Shuffle D randomly
    Partition D into $\lceil N/m \rceil$ mini-batches $\{B_1, B_2, \ldots, B_K\}$
    Initialize accumulated gradient $\Delta\theta \leftarrow 0$
    for each mini-batch B_j in the current epoch **do**
        $X \leftarrow$ inputs of B_j
        $Y \leftarrow$ targets of B_j
        $\hat{Y} \leftarrow \text{ForwardPass}(X;\, \theta)$
        $\mathcal{L}_j \leftarrow \text{ComputeLoss}(\hat{Y},\, Y)$
        $\nabla_j \leftarrow \text{BackwardPass}(\mathcal{L}_j;\, \theta)$
        $\Delta\theta \leftarrow \Delta\theta + \nabla_j$
    end for
    $\theta \leftarrow \theta - \eta \cdot (\Delta\theta / K)$ ▷ Update weights once per epoch
    Optionally: Evaluate validation set, adjust η or apply early stopping
end for
return θ

Appendix B

Model development and experimentation

This appendix details the steps taken in model development and experimentation.

B.1 Model 1

As the first step, a large model with 6 layers and low LR was tested.

Type: Supervised Regression DNN

Data Trained On: Single valuation run (Basis 2) (76,102 policies)

Layers: 1: 9x64, 2: 64x32, 3:32x32, 4: 32x16, 5: 16x8, 6:8x1 (these are the constraints or architecture)

Layer types: RELU

Run on: GPU

LF: MAE from Flux library (Percentages below reported as MAPE, amounts as absolute difference between actual and expected)

Optimiser: Nesterov, LR = 0.003, Decay = 0.97

Batch Size: 1,024

Epochs: 19,000

The following conclusions were reached:

- The model exhibited a slow convergence rate,
- The prediction accuracy on the single policy (validation dataset) starts at 0%, and after 5,000 epochs it reaches 37%,
- After 6,500 epochs it reached 40%, and I can see it is not improving as fast anymore, but it is still moving in the right direction, at around 0.01% increase in accuracy per epoch, and
- It reaches the maximum prediction accuracy on a single policy of around 42.37%.

In summary, the following was the sum of the absolute differences between actual and predicted, for each dataset:

Train Data: 5,426m (+-42%)

Validation Policy: 20,450.83 (42.36%)

Test Data: 5,405m (+-42%)

In conclusion, this is a model that does work correctly in theory, albeit very slowly and carefully, to reach the best prediction possible given the constraints in the design of the model. Running a model for so many epochs is not how long these models are normally run for. So what can be changed? Since it is not correctly modelling individual policies, the batch size will be reduced, and random batches will be used for each epoch.

B.2 Model 2

Changes from previous model:

Batch Size: **32**

Epochs: 375

The following conclusions were reached:

- The model is even slower than the previous model, but it is consistently moving in the right direction.
- The accuracy on the single policy is improving faster (initially, it seems, about 0.3% increase per epoch)
- The prediction accuracy on the single policy starts at 0%, and after **100 epochs** it reaches 29.6%, still learning at around 0.2% improvement per epoch,

- It reaches the 37% accuracy of the previous model after **155 epochs**, which is a much lower and acceptable number; the improvement was around 0.1% per epoch,

Summary of absolute differences per dataset:

Train Data: 5,405m (42.2%)

Validation Policy: 20,498.72 (42.36%)

Test Data: 5,426m (42.2%)

It should be noted that although this model reached the same accuracy, the time it took for 6,500 epochs with a batch size of 1,024, and training for 375 epochs with a batch size of 32, was similar. Therefore, changing the batch size made no difference in performance in terms of reducing training time.

In conclusion, this is a model that does work correctly, albeit very, very slowly and carefully, to reach the best prediction possible given the constraints in the design of the model. Changing batch size had no impact. So what can be changed next?

It is obvious when it was discovered that the absolute loss for all data, using the MAE LF, was less than that on a single policy. Here, the LF from the standard Flux library is working on the totals of the predictions and real values, not the individual values. Upon investigation for each policy in the training dataset, I found that the model "cheats" by setting the reserve for each policy equal to the total reserves divided by the number of policies, which it can only do if the total loss is equal to the sum of total reserves minus the sum of total predicted reserves.

Therefore, for the next model, and all models trained in the future, the LF was forced to manually calculate losses as the sum of individual losses.

B.3 Model 3

Changes from the previous model:

LF: **Custom element-wise MAE**

Epochs: 30

The following conclusions were reached:

- The model is consistently moving in the right direction,

- The accuracy on the single policy is improving faster, and it reaches 42.37% after 20 epochs!
- The prediction accuracy on the single policy starts at 10%, and after **7 epochs** it reaches 40%, and
- It reaches the maximum accuracy of around 42.4%, and was stopped after **30 epochs** as no noticeable improvement was observed.

In summary, the total absolute loss (and accuracy as a percentage) per dataset was:

Train Data: 5206m (42.28%)

Validation Policy: 20 452.72 (42.36%)

Test Data: 5426m (42.41%)

In conclusion, this is a model that does work correctly, and much faster reaches its plateau of 42.4%, to reach the best prediction possible given the constraints in the design of the model. The number of epochs is acceptable. The total training and test losses are also now calculated correctly, and the accuracy is low. Training was stopped early because it seemed that the LR is too low.

Tweaking the LR is considered in the next model, perhaps to overcome a local minimum.

B.4 Model 4

Changes from the previous model:

Optimiser: Nesterov, **LR = 0.03**, Decay = 0.97

Epochs: 60

For this run, the already trained Model 3, with 50 epochs, was used as a starting point.

The following conclusions were reached:

- The model started predicting at 42.24% for the first epoch,
- The model initially does not seem to make any progress from here,
- For some epochs the accuracy is up to around 42.6%, but then quickly moves back to 42.2%,

- No improvement can be seen, the model does not seem to be able to get out of the local minimum, and was stopped after 60 further epochs.

In summary, on the train and test data, the following absolute errors occurred per dataset:

Train Data: 5,205m (42%)

Validation Policy: 20,450 (42.4%)

Test Data: 5,205m (42%)

In conclusion, this model is stuck at a local minimum, which it reached after 1 epoch. A high LR seems to be fine here, as it reaches the limit quicker than the previous model.

The optimizer and batch size were changed for the next model.

B.5 Model 5

Changes from the previous model:

Optimiser: **NADAM**, **LR = 0.001**, **Decay = (0.88, 0.78)**

Batch Size: **1,024**

Epochs: 1,200

The following conclusions were reached:

- The model predicts the single policy between 80-100% correct for each epoch,
- The total loss, however, fluctuates around 50-60m with no clear direction,
- Training with 1,024 policies per batch again is resulting in very fast epochs on the GPU,
- The accuracy in training on the total is a factor of 5 better than the Nesterov optimiser, and
- the model never converges in total.

In summary, on the train and test data, the following absolute errors occurred per dataset:

Train Data: 55m (91%)

Validation Policy: 1,468 (91%)

Test Data: 55m (91%)

In conclusion, this model is working correctly and produces 91% accuracy. Increasing the batch size did not seem to matter; the large increase in convergence was a result of the NADAM optimizer. But this study requires more accuracy than 91%! However, in earlier trials, training runs fastest with a batch size of 512 (where the timing of showing results of epochs is just of the right speed to analyse during training runs), and this was used going forward.

B.6 Model 6

Changes from previous model:

Optimiser: NADAM, LR = 0.001, **Decay = (0.95, 0.95)**

Batch Size: **512**

Epochs: 3,600

The following conclusions were reached:

- Changing the decay did not seem to matter initially; the results are similar to those of Model 5,
- However, at around epoch 400+, the training loss started to decrease to less than 50m,
- And it was still decreasing after epoch 500, so the model was allowed to train further,
- It reached 40m at epoch 700, and was still decreasing, so the model was allowed to train further, and the accuracy was 93%,
- at epoch 780 it started to fluctuate, jumping up and down,
- at epoch 950, it jumped to less than 30m for the first time,
- at epoch 1,500, it seemed to stabilize between 22m and 28m, with accuracy of around 97%,
- at epoch 1,850, it reached training accuracy of 19m for the first time, but it was still fluctuating,
- at epoch 2,250, it reached a consistent training loss of less than 20m, and it was still improving, so the model was allowed to train further,
- at epoch 2,700, it reached a training accuracy of less than 10m, for the first time, and was fluctuating between 10m and 14m, and it became close to overfitting, and

- in hindsight should have stopped training there, as no noticeable improvement was made thereafter.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 10.658m (98.2%)

Validation Policy: 20.5 (99.94%)

Test Data: 10.873m (98.2%)

In conclusion, this model is working correctly and produces 98%+ accuracy - this is within a comfortable range for most actuaries. This combination of hyperparameters is a close candidate for being the architecture to select when more valuation bases are introduced. However, it did take a long time to train and is at risk of overfitting. Let's see first if more tweaks can get to the best possible solution faster. A more robust model is required (which implies reducing the batch size) and a higher LR to converge faster, as local minimum no longer seems to be an issue. The change in momentum solved the local minimum problem.

B.7 Model 7

Optimiser: NADAM, **LR = 0.005**, Decay = (0.95, 0.95)

Batch Size: **256**

Epochs: 1,350

The following conclusions were reached:

- This model had a training loss of 60m after 20 epochs already,
- It reached less than 50m after 125 epochs already,
- It reached 40m before epoch 200, but then started fluctuating like model 6,
- It did predict 30m loss consistently from epoch 300+, and it was going down,
- at epoch 450, it jumped to less than 20m for the first time, with little fluctuation,
- at epoch 600, however, the model seemed to fluctuate, but it did not yet reach less than 10m,
- at epoch 900, the model still did not reach a loss of less than 10m, but at least it was consistently between 11m and 15m, so it was allowed to train more,

- It did reach less than 10m at epoch 1,030, so was allowed to train more, however, it failed to reach less than 10m consistently thereafter.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 10.324m (98.3%)

Validation Policy: 1,169 (96.7%)

Test Data: 10.558m (98.3%)

In conclusion, this model is working correctly and is even better than Model 6 in terms of epochs. An even higher LR was attempted for the next model, but the risk is that such a model might run longer as it is continuously overshooting the minimum values.

B.8 Model 8

Changes from the previous model:

Optimiser: NADAM, **LR = 0.01**, Decay = (0.95, 0.95)

Epochs: 950

The following conclusions were reached:

- This model had a training loss of around 60m after 20 epochs (same as model 7),
- It reached a loss of less than 50m after 125 epochs (same as model 7),
- It reached 40m before epoch 200, but it did not fluctuate as much as model 7, although there were some fluctuations,
- It did predict 30m loss consistently from epoch 220, and it was going down,
- at epoch 350, it jumped to less than 20m for the first time, with little fluctuation, which is better than model 7,
- the model then fluctuated around 20m for a long time, up to epoch 550, when it started reaching less than 20m consistently,
- at epoch 800 the model still did not reach less than 10m, but at least it was consistently around 15m, so it was allowed to train more, and

- this model seems never to reach even 12m, which could imply that the learning rate is too high when values are getting small, it continuously overshoots and never finds the right path.

In summary, the following absolute errors (and accuracy as percentage) was noted per dataset:

Train Data: 20.607m (96.6%)

Validation Policy: 1 813 (94.9%)

Test Data: 21.109m (96.6%)

In conclusion, this model is working correctly and is initially training similarly to Model 7, but at later epochs, better than Model 7. A higher LR helped a bit, but something else should be tried. This model would work well on a single VB, but perhaps with a lower LR, and I might require more neurons to handle different valuation bases. Therefore another layer was added to the next model, and the LR was set back to 0.005.

B.9 Model 9

Changes from previous model: (Additional layer)

Layers: 1: **9x64, 2:64x64, 3: 64x32, 4:32x32, 5: 32x16, 6: 16x8, 7:8x1** (these are the constraints or architecture)

Optimiser: NADAM, **LR = 0.005**, Decay = (0.95, 0.95)

Batch Size: **256**

Epochs: 950

The following conclusions were reached:

- This model had training losses of around 60m after 10 epochs (Better), but had a poorer accuracy on the test data,
- It reached a training loss of less than 50m after 90 epochs (Better),
- There were a lot fewer fluctuations than any of the previous models,
- It reached a training loss of 40m before epoch 120, which is much better; however, then it started fluctuating, and accuracy was still a lot less than previously, fluctuating at about 90% only,

- at epoch 270, it jumped to a training loss of less than 20m for the first time, with little fluctuation, which is better than model 8,
- and at epoch 270, it also reached a training loss of less than 20m consistently,
- at epoch 350, the model still did not reach a training loss of less than 10m, but at least it was consistently around 15m, so it was allowed to train more, and the test loss was now consistently low,
- this model, however, also seems never to reach a training loss of even 12m, which could imply that the LR is too low or high when values are getting small; it continuously overshoots and never finds the right path.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 13.075m (97.9%)

Validation Policy: 412.45 (98.8%)

Test Data: 13.296m (97.8%)

In conclusion, this model is working correctly and is initially better than previous models, but not better than Model 6 at the end. To improve, the LR should be decreased even more, in order to try and beat Model 6. The batch size was also increased again.

B.10 Model 10

Changes from the previous model:

Optimiser: NADAM, **LR = 0.001**, Decay = (0.95, 0.95)

Batch Size: **512**

Epochs: 1,200

The following conclusions were reached:

- This model reached a training loss of 60m at epoch 60 only.
- It then seemed to hit a local minimum of 55m, where it got stuck even past epoch 300.
- It reached a training loss of less than 50m after 450 epochs (far worse), and it was not even improving,
- There were a lot fewer fluctuations, even less than model 9.

- It reached 40m at epoch 650, reasonably consistently, before it started a little fluctuation episode,
- at epoch 850, it jumped to a training loss of less than 30m for the first time, with little fluctuation, which is far worse than previous models, and
- this model however, also seems never to reach even 20m, which could imply that the LR is too low.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 31.171m (95%)

Validation Policy: 3959.71 (88.8%)

Test Data: 31.64m (94.9%)

In conclusion, this model is working correctly, and up to this stage, I developed mixed feelings about whether there might be a different combination that might perform better. Here I made the call to introduce the model to all 7 valuation bases, with the argument that the additional data should help the model find its way better.

B.11 Model 11

Changes from the previous model:

Data Trained On: **Include 7 valuation bases data**. The total reserves of the datasets are given in Table 3.9.

Epochs: 600

The following conclusions were reached:

- This model trains longer since it has 7 times more policies,
- The training loss starts at 2,500m,
- And by epoch 50 this is reduced to about 1,500m, with a lot of fluctuations,
- At around epoch 125, it reaches 1,250m for the first time, but accuracy on the test data is very low, (i.e. it is summing very high and low values),
- I started to suspect this will take a very long time to get the training loss and prediction accuracy to the same order as seen previously, as the better the training loss becomes, the worse the accuracy,

- Maybe this model is too complex regarding the data? Or too simple regarding the number of layers and neurons?
- at epoch 220, it jumped to less than 1,000m for the first time, with only small amounts of fluctuation, but the accuracy for test data was still dismal at only 29%,
- Were thought about the LF again, perhaps adding or subtracting something could result in training and test losses of the same magnitude, make the loss and accuracy better at the same time,
- but I still let the model run this way, until test accuracy eventually was similar to the train accuracy, and
- after noticing no improvement stopped training.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 895m (87.1%)

Validation Policy: 11,316 (45.2%)

Test Data: 898m (87.1%)

In conclusion, this model reached an accuracy of 87% would not be frowned upon. Some changes are called for. First, the batch size will increase again, so that the model produces output faster. The MSE loss will also be tried again, one that is custom-written and not the one from the Flux library.

B.12 Model 12

Changes from the previous model:

LF: **Element-wise MSE, custom written**

Batch Size: **1,024**

Epochs: 360

The following conclusions were reached:

- Since the loss is now the MSE, a quadratic function, the resulting output can now longer be compared as easily by just looking at the amounts[1],

[1] It was only much later that I realised this could have been done.

- However, the loss and the accuracy do seem to move in the same direction initially, and
- it became clear that it is also moving apart later on, and was stopped when no noticeable further improvements were seen.

In summary, the following MSEs were noted per dataset:

Total Loss Train Data: 7.14b (the loss is given in squared terms)

Total Loss Test Data: 7.11b (the loss is given in squared terms)

A check was performed on all policies individually, and it was clear there were many errors, especially on smaller and negative policies. But it is here that I realised that to date I had not normalised the age and TIF variables, since they are not monetary values. Perhaps after normalising those as well, I would get a more perfect fit?

B.13 Model 13

Changes from the previous model:

Data Trained On: All training and validation data **correctly Normalised**

Epochs: 500

The following conclusions were reached:

- The loss started at 48b and did not seem to move initially,
- then at epoch 50 it started going down fast, first to 30b and then 20b. Accuracy then got worse,
- Also, there were large discrepancies between the training and testing losses,
- after 200 epochs, the loss was still hovering around 20b, and fluctuating, but the testing loss was now closer to the training loss,
- there were some runs which were really good, but the model could not converge, so I let it run longer,
- until it reached the previous accuracy, when again no further improvements were noticed.

In summary, the following MSE losses were noted per dataset:

Train Data: 6.92b

Test Data: 6.93b

The problem is that it is not as easy to interpret the MSE loss output as it was for the MAE output. Perhaps, changing back to MAE will enable better management of the output.

B.14 Model 14

Changes from the previous model:

LF: **Element-wise MAE**

Epochs: 360

The following conclusions were reached:

- The loss started at 2.5b and again was stuck there for quite a while. Accuracy on the test dataset, however, was high, which is strange; it might have been a weights initialisation issue?
- Over time, the large discrepancies between the training and testing losses disappeared,
- after 107 epochs it reached 1,500m for the first time, but accuracy was only at 60%,
- it was clear again that there are large discrepancies between the training and test losses,
- the model seemed to reach its best accuracy at 77.8% on the training dataset.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 1,554m (77.8%)

Test Data: 1,552m (77.2%)

Test Policy: 6,151 (70.3%)

The problem that I saw in Model 11 persisted, and before I change something in the architecture, it dawned on us that perhaps the model would do better if I shuffle the data for each epoch, albeit that it might change how I should interpret the feedback per step, it might help the model to converge quicker if it has to find derivatives for different samples. And also, that this may help to make the training and testing losses more similar, as more generalisation would occur.

B.15 Model 15

Changes from previous model:

Data Trained On: All training and validation data **shuffled each epoch**

Epochs: 460

The following conclusions were reached:

- The loss started at 2.5b and again was stuck there for quite a while. Accuracy on the test dataset, however, was high,
- Now there was very little difference between total training and testing losses,
- after 24 epochs, it reached 2,000m for the first time, and total training and testing losses remained similar,
- after 114 epochs it reached 1,500m for the first time, however, it then started fluctuating, which it continued to do with marginal improvements until the training was stopped.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 1,589m (77.2%)

Validation Policy: 3,760 (81.8%)

Test Data: 1600m (77.0%)

The training went similarly to Model 13, but improvements were noticed for the accuracy on the test dataset. I started to suspect the model has too high dimensionality, because the model seems to pursue a low loss and high accuracy on two separate ends of the scale. So for the next run, all neurons per layer were reduced by a factor of 2, and then consideration was given to removing a layer as well.

B.16 Model 16

Changes from the previous model:

Layers: **1:9x32, 2:32x32, 3: 32x16, 4:16x16, 5: 16x8, 6: 8x4, 7:4x1**

Epochs: 200

The following conclusions were reached:

- The loss started at 7,000m and was stuck there for quite a while, slowly decreasing every step. Accuracy on the test dataset now correctly starts at 0%.
- After 100 epochs, the loss was 6,950m, but there were no fluctuations. I stopped training there.

In summary, the model did not predict well. Losses were billions and the accuracy was less than 0.2%.

It was clear that I needed to adjust the LR as this model was learning too cautiously. Or else that the model was now too simple? Also, for later, a higher initial LR might be required, and when the model starts fluctuating, a lower LR.

B.17 Model 17

Changes from the previous model:

Optimiser: NADAM, **LR = 0.01**, Decay = (0.95, 0.95)

Epochs: 650

The following conclusions were reached:

- The loss started at 2,500m. Accuracy on the test dataset started at 90%.
- After 20 epochs, the loss was 2,000m, and there were some fluctuations,
- after 220 epochs it briefly reached 1,500m for the first time, but then fluctuated again, and no improvements occurred, until the training was stopped.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 1,631m (76.6%)

Validation Data: 10,096 (51.2%) (probably as a result of overfitting)

Test Data: 1,633m (76.5%)

I wondered whether to test another LF, and the Chi-Squared test was proposed, because that will penalise more. Although this will again make it difficult to follow the progress, it was attempted next.

B.18 Model 18

Changes from previous model:

LF: **Element-wise Chi-Squared**

Epochs: 650

The following conclusions were reached:

- The loss started at 1.133m (squared).
- The loss came down in a logical way at each epoch, but very slowly initially,
- after 100 epochs it reached 1.124m, reducing by 0.001 at each epoch consistently, and was allowed to continue training until it either no longer came down, or started to fluctuate, but no further progress was made.
- I stopped the model after 650 epochs, as even though the model still made progress, it was much too slow, and not even close to 1% accuracy.

In summary, the model did a poor attempt at solving at a high LR. The accuracy was less than 1%. I discarded this model and used Model 16 with one less layer in the next run.

B.19 Model 19

Changes from the previous model:

Layers: 1: 9x32, 2:32x32, 3: 32x16, **4: 16x8, 5: 8x4, 6:4x1** (these are the constraints or architecture)

Epochs: 650

The following conclusions were reached:

- The loss started at 2,500m. Accuracy started at 90%.
- After 20 epochs the loss was 2,000m, and there was some fluctuations,
- after 220 epochs it briefly reached 1,500m for the first time, but then fluctuated again.

In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 1,631m (76.6%)

Validation Policy: 10,096 (51.2%)

Test Data: 1,633m (76.5%)

The approaches above were typical in training. I continued in this vain carefully keeping track of changes and where improvements were made. Until I reached the model below, which is called the Z1 model. The full architecture used in training of Z1, is given in section B.20.

B.20 Model Z1

I continued along the route of the analysis as set out by the preceding sections, and eventually arrived at the following best model:

Type: Supervised Regression DNN

Data Trained On: 75% of input data shuffled each epoch (the remaining data is the validation data)

Layers: 1: 12x72, 2: 72x36, 3: 36x18, 4:18x9, 5:9x1 (3 hidden layers)

Layer types: CELU

Run on: GPU

LF: Custom Element-wise MAE

Optimiser: NADAM, LR = 0.01, Decay = (0.975, 0.975)

Batch Size: 512

Epochs: 72

In summary, the model did work correctly and solved the problem satisfactorily in a very short time. In summary, the following absolute errors (and accuracy as a percentage) were noted per dataset:

Train Data: 2.04m (99.7%)

Validation Data: 678k (99.7%)

Appendix C

Machine Learning Methods

C.1 Optimisers

C.1.1 AdaGrad

Adagrad is the abbreviation used for adaptive gradient. It was designed by Duchi et al. (2011), noting that many online and stochastic learning applications involve high-dimensional input data where only a few features are non-zero. They further emphasise that infrequently occurring features are often highly informative and can significantly contribute to discriminative power.

Adagrad updates the LR at every epoch as follows:

$$\eta_t = \frac{\eta_{t-1}}{\sqrt{\alpha_t + e}} \tag{C.1}$$

$$\alpha_t = \sum_{i=1}^{n} (\frac{\partial loss}{\partial w_i})^2 \tag{C.2}$$

C.1.2 RMSProp

RMSProp was developed as an improvement on Adagrad to make it effective in a nonconvex problem space. Goodfellow et al. (2016) describes RMSProp as a modification of the AdaGrad algorithm designed to address challenges in non-convex optimisation problems. RMSProp replaces the cumulative sum of squared gradients in AdaGrad with an exponentially decaying moving average of the gradients, thereby reducing the influence of past gradients.

RMSProp updates the LR at every epoch as follows:

$$\eta_t = \frac{\eta_{t-1}}{\sqrt{w_{avg}(t) + e}} \tag{C.3}$$

$$w_{avg}(t) = \alpha * w_{avg}(t-1) + (1-\alpha) * \sum_{i=1}^{n} (\frac{\partial loss}{\partial w_i})^2 \tag{C.4}$$

Compared to AdaGrad, RMSProp gives more weight to recent moving averages.

C.1.3 AdaDelta

AdaDelta improves on AdaGrad's diminishing LR by using exponentially weighted past gradients. The formula for α_t is:

$$\eta_t = \frac{\eta_{t-1}}{\sqrt{S_{dw_t} + e}} \tag{C.5}$$

$$S_{dw_t} = \beta * S_{dw_{t-1}} + (1-\beta) * (\frac{\partial loss}{\partial w_{t-1}})^2 \tag{C.6}$$

where β is the momentum.

C.1.4 ADAM

Adaptive Moment Estimation is a combination of SGD and AdaDelta. Kingma and Ba (2014) introduced ADAM, an optimisation algorithm that builds upon concepts from RMSProp by incorporating both the first moment (mean) and second moment (uncentered variance) of the gradients, utilising exponential moving averages for both. It uses the SGD weighted average of past gradients and the AdaDelta average of past squared gradients.

To derive ADAM mathematically:

Let $f(\theta)$ be the loss function, where $\theta \in \mathbb{R}^d$ are the parameters. We denote by

$$g_t = \nabla_\theta f(\theta_{t-1})$$

the gradient at iteration t.

Notation:

α :	step size / learning rate
β_1 :	decay rate for first moment (momentum)
β_2 :	decay rate for second moment (squared gradients)
ε :	small constant for numerical stability
m_t :	first moment (exponential moving average of gradients)
v_t :	second moment (exponential moving average of squared gradients)
$\widehat{m}_t$, $\widehat{v}_t$:	bias-corrected versions of m_t and v_t
$\widetilde{m}_t$:	Nesterov-corrected first moment used in NADAM
θ_t :	parameter vector at iteration t

ADAM performs the following updates:

$$m_t = \beta_1\, m_{t-1} + (1-\beta_1)\, g_t, \tag{C.7}$$

$$v_t = \beta_2\, v_{t-1} + (1-\beta_2)\, g_t^2, \tag{C.8}$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \widehat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{C.9}$$

$$\theta_t = \theta_{t-1} - \alpha\, \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \varepsilon} \tag{C.10}$$

Another way to state the above formulae is:

$$\theta_{t+1} = \theta(t) - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t \tag{C.11}$$

with momentum η.

Kevin and Kang (2017) found that the ADAM optimiser produces better accuracy compared to the Momentum optimiser. Yet, Wilson et al. (2017), in a binary classification problem, concluded that AdaGrad, ADAM and RMSProp attained worse predictions than SGD (and gradient descent) when the models are over-parameterised. They define over-parameterised as when the parameters exceed the number of input model points. The trained models in such cases also had much worse generalisation. The total parameters trained by the models in this study are around 4,000, which is a fraction of the data points used by the Z1 and Midway

models. ADAM and NADAM were therefore always strong candidates for the final model architectures.

C.1.5 Nesterov momentum

Nesterov (1983) introduced a momentum-based optimisation technique that builds upon traditional momentum by incorporating a correction term based on the projected future position of the parameters.

As described by Goodfellow et al. (2016, p. 300), Nesterov momentum can be conceptualized as a four-step process:

1. Projection: Estimate the future position of the solution based on the current momentum.
2. Gradient Calculation: Evaluate the gradient at the projected position, rather than the current variable value.
3. Update Calculation: Determine the change in the variable using the calculated gradient.
4. Variable Update: Update the variable with the calculated change.

By calculating the gradient at the projected position, Nesterov momentum provides a more accurate estimate of the optimal direction of descent, leading to improved convergence characteristics, particularly in non-convex optimisation problems.

C.1.6 Nesterov's Accelerated Gradient (NAG)

Sutskever et al. (2013) first introduced Nesterov Momentum in the training of neural networks with SGD. Traditional momentum involves maintaining an additional variable that represents the last update performed to the variable, an exponentially decaying moving average of past gradients.

NAG is a momentum-based SGD optimiser that "looks ahead" to where the parameters will be to calculate the gradient after the fact, rather than before the fact. The intuition is that the standard momentum method first computes the gradient at the current location and then takes a big jump in the direction of the updated accumulated gradient. In contrast, Nesterov Momentum first makes a big jump in the direction of the previous accumulated gradient and then measures the gradient where it ends up and makes a correction. The idea being that it is better to correct a mistake after you have made it.

C.1.7 Nesterov momentum for ADAM

Dozat (2016) derived NADAM to provide a better optimiser than standard ADAM by stepping into both the direction of the gradient and the momentum, and also corrects the initial bias, which is a result of an initial momentum of zero. NADAM alters how the first moment of ADAM is used. Mathematically:

$$m_t = \beta_1 \, m_{t-1} + (1 - \beta_1) \, g_t, \tag{C.12}$$

$$v_t = \beta_2 \, v_{t-1} + (1 - \beta_2) \, g_t^2, \tag{C.13}$$

$$\widehat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{C.14}$$

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{C.15}$$

Then define the Nesterov-corrected first moment:

$$\widetilde{m}_t = \beta_1 \, \widehat{m}_t \; + \; \frac{(1 - \beta_1) \, g_t}{1 - \beta_1^t} \tag{C.16}$$

Finally, the parameter update is:

$$\boxed{\theta_t = \theta_{t-1} - \alpha \, \frac{\widetilde{m}_t}{\sqrt{\widehat{v}_t} + \varepsilon}} \tag{C.17}$$

C.1.8 Broyden, Fletcher, Goldfarb, and Shanno optimiser

The BFGS optimiser is a quasi-Newton optimisation method commonly used in numerical optimisation and occasionally in machine learning. It approximates the Hessian (second-derivative) matrix of the objective function using only gradient evaluations — so you get the benefits of Newton's method without the heavy computational cost of computing or inverting the Hessian directly. Mini-batch training common in deep learning conflicts with BFGS's reliance on accurate full-batch gradients — making stochastic BFGS variants more complex and less reliable, but a solution has been proposed by Bollapragada et al. (2018).

C.2 Activation Functions

C.2.1 Argmax

This function takes the activation with the largest value as 1, and sets the values for all the other activations to 0. It is used to build a confusion matrix for classification problems.

C.2.2 Unit step

The Unit step AF returns either a 0 or a 1. The formula is:

$$UnitStep(x) = \begin{cases} 0, & \text{if } x < 0 \\ 0.5, & \text{if } x = 0 \\ 1, & \text{otherwise} \end{cases} \tag{C.18}$$

This function maps input values to a range between 0 and 1, making it useful for binary classification problems.

C.2.3 Sigmoid

The Sigmoid AF has been in existence for a long time and it still used in classification problems today. It is also called the logistic function.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{C.19}$$

This function maps input values to a range between 0 and 1, making it useful for binary classification problems.

C.2.4 Softmax

The Softmax function is the multiple activation variant of the Sigmoid AF. It also acts as a "soft" approximation to the Argmax AF.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{C.20}$$

The term below the line is a normalisation term that ensures that the sum of the activations is equal to 1.

The Softmax AF works well when there are multiple variables in the last hidden layer for classification or reinforcement learning problems, and is used to scale the last hidden layer into probabilities.

C.2.5 Softplus

$$Softplus(x) = log(1 + e^x) \tag{C.21}$$

Softplus is an attempt to replace Sigmoid by the sum of Sigmoid functions over each dimension (Dugas et al., 2000). This can help prevent overfitting and improve model performance over traditional sigmoidal functions.

C.2.6 Swish

The Swish AF was developed by Google researchers (Ramachandran et al., 2017) to improve on the Sigmoid AF, as it does not always have a single continuous positive derivative everywhere (Kiliçarslan et al., 2021). In formula terms:

$$Swish(x) = \frac{x}{1 + e^{-x}} \tag{C.22}$$

The Swish AF has less of a problem with vanishing gradients than the Sigmoid AF, and also overcomes the negative region problem of the RELU AF, but it is still more widely used in classification problems for images.

C.2.7 Hyperbolic tangent (TANH)

This function maps input values between -1 and 1, making it a good choice for regression models where the target values are within a similar range. It is also used in classification problems. To

date, the TANH function has been used frequently in actuarial ML, and examples are provided in Section 2.4.

The formula is:

$$TANH(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{C.23}$$

TANH is preferred over Sigmoid AF because it pushes the mean activation faster to 0 when close to zero. However, it gives low attention to extreme values. In a way to overcome this, Sutskever et al. (2013) set the bias to 0.5 and rescaled weights to 0.25.

A graphical illustration of the above AFs is found in Figure C.1.

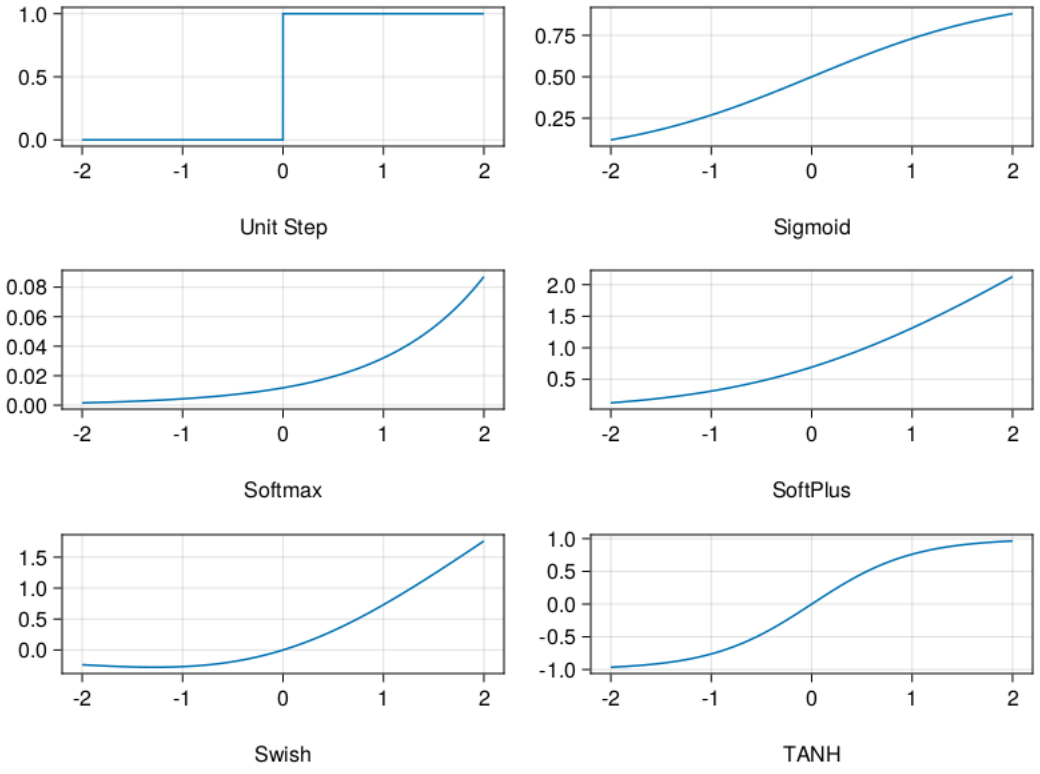

FIGURE C.1: Classification AFs

C.2.8 RELU

The RELU AF was first used in RBMs (Nair and Hinton, 2010) and subsequently used in DNNs as well. RELU is a non-saturated AF which shows gains over sigmoidal non-linearities

(or saturated AF) like TANH and Sigmoid. It has the following benefits over saturated methods (Xu et al., 2015):

- solves both the exploding and vanishing gradient problems, and
- faster model convergence.

The formula for RELU is:

$$RELU(x) = max(0, x) \tag{C.24}$$

Glorot et al. (2011) showed that RELU outperforms TANH, which outperforms Sigmoid, especially on sparse data. Contrary to prior beliefs, they found that RELU's hard activation at 0 is beneficial to supervised training. They employed L_1 regularisation on activation values, which also provided additional sparsity. They warned that RELU networks are subject to ill-conditioning of the initial weights and bias and suggested scaling, which does not impact the network function.

The gradient is therefore either 0 (when $x \leq 0$) or 1. It aims, therefore, to solve the vanishing gradient and exploding gradient problem, which are problems of the sigmoidal methods for multi-layer DNNs. It also has higher accuracy with large DNNs and with Dropout, than sigmoid methods (Dahl et al., 2013).

The potential issues with RELU are neurons never activating (because the activation is 0 for $x < 0$) and that model training is slow for small values of x. Some of this can be overcome by employing regularisation like Dropout.

The next Subsections provide improvements on RELU. It is important to note that all of these more complex forms of RELU can benefit from a non-zero bias term. RELU has positive activations, while the variants have negative activations. Therefore, RELU typically also has more bias than the variants.

C.2.9 SRELU

Shifted Rectified Linear Unit (SRELU) is just making the minimum value of the activation -1 instead of 0, in an attempt to remove some bias generated by RELU.

$$SRELU = max(-1, x) \tag{C.25}$$

C.2.10 GELU

The GELU is similar to the RELU function but is designed to add more Gaussian noise to the RELU input values, which leads to performance improvements in computer vision, natural language processing, and speech tasks (Hendrycks and Gimpel, 2016).

$$GELU(x) = xP(X \leq x) = x\Phi(x) \tag{C.26}$$

where $\Phi(x)$ is the standard Gaussian cumulative distribution function. The distribution is very similar to the RELU distribution, except that some noise is introduced. This helps prevent overfitting and improve model performance.

C.2.11 LRELU

AFs like LRELUs, PRELUs, and RRELUs do not ensure a noise-robust deactivation state (Clevert et al., 2015), which leads to issues when training.

$$LRELU(x) = \begin{cases} x, & \text{if } x > 0 \\ \frac{x}{\alpha_i}, & \text{otherwise} \end{cases} \tag{C.27}$$

where α_i is a fixed parameter in the range $[1, +\infty]$.

Leaky RELU has been developed by Maas et al. (2013) to overcome some of the pitfalls of RELU, and allows for small gradients even when a neuron has not been activated. LRELU, therefore, can generalise better than RELU. They suggested an α_i of 100.

C.2.12 RRELU

Randomised Leaky Rectified Linear Units (RRELUs) have found that using a non-zero slope for the negative part of RELU AFs could consistently improve results. They employed random sampling of the slope of the negative part, which raised the performance on image benchmark datasets and convolutional networks (Xu et al., 2015). They also disproved the then-common belief that sparsity is the key to good performance in RELU. They found that RRELU is better than RELU and is less at risk of overfitting on small input datasets.

$$RRELU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha_i * x, & \text{otherwise} \end{cases} \tag{C.28}$$

where α_i is randomly sampled from a Uniform distribution with range [l, u], with l and u in range [0, 1].

C.2.13 PRELU

Parametric Rectified Linear Units (PRELUs) improve on LRELUs by learning the slope of the negative part, which yielded improved learning behaviour on large image benchmark data sets (He et al., 2015). The resulting model also generalises better. The formula is the same as LRELU except that α_i is learned during the training through back propagation. The formula is:

$$PRELU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha_i * x, & \text{otherwise} \end{cases} \tag{C.29}$$

C.2.14 ELU

A further improvement on RELU is ELUs. It is a useful rectifier for constructing deep learning architectures, as it may speed up and otherwise improve learning by virtue of not having vanishing gradients and by having mean activations near zero.

$$ELU(x, \alpha) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases} \tag{C.30}$$

The advantage of ELU is that it has negative values, which allows batch training with less computational complexity and also removes the vanishing gradient problem. This leads to faster training and better model generalisation, especially for models with more layers (Clevert et al., 2015).

The downside of ELU is that when the shape parameter is equal to 1, the activation function becomes discontinuous at $x = 0$, which can lead to exploding gradients and compromise training (Barron, 2017).

C.2.15 SELU

A variant of ELU is the Scaled Exponential Linear Unit(s) (SELU), which was developed by Klambauer et al. (2017), which introduces a scaling parameter λ as follows:

$$SELU(x, \lambda, \alpha) = \begin{cases} \lambda x, & \text{if } x \geq 0 \\ \lambda\alpha(e^x - 1), & \text{otherwise} \end{cases} \tag{C.31}$$

SELU has been found to be highly robust for training DNN with many layers and can be used with higher LRs, clearly better than RELU (Kevin and Kang, 2017; Pedamonti, 2018; Sakketou and Ampazis, 2019). All these studies used $\lambda = 1.6733$ and $\alpha = 1.0507$ for classification problems. Kevin and Kang (2017) found the SELU AF works best on classification problems.

C.2.16 CELU

The CELU (Barron, 2017) was the best AF for this study. It is an improvement of ELU because the derivative with respect to the scale is bounded, and it holds its shape in scale.

The formula for CELU is given by:

$$CELU(x, \alpha) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(exp(\frac{x}{\alpha}) - 1), & \text{otherwise} \end{cases} \tag{C.32}$$

CELU and ELU are identical when $\alpha = 1$ and ELU converges to RELU as α approaches 0 from the right.

Graphically, the last group of AFs is displayed in Figure C.2. For an overview of more AFs, please see Kiliçarslan et al. (2021).

C.3 Loss Functions

The following subsections provide an overview of common LFs used in regression problems, and a schematic diagram for a single value is given in Figures C.3 and C.4. In terms of notation, we use x_t as the predicted value for the reserve of policy t, and y_t as the actual reserve of policy t, which was set to 1.5.

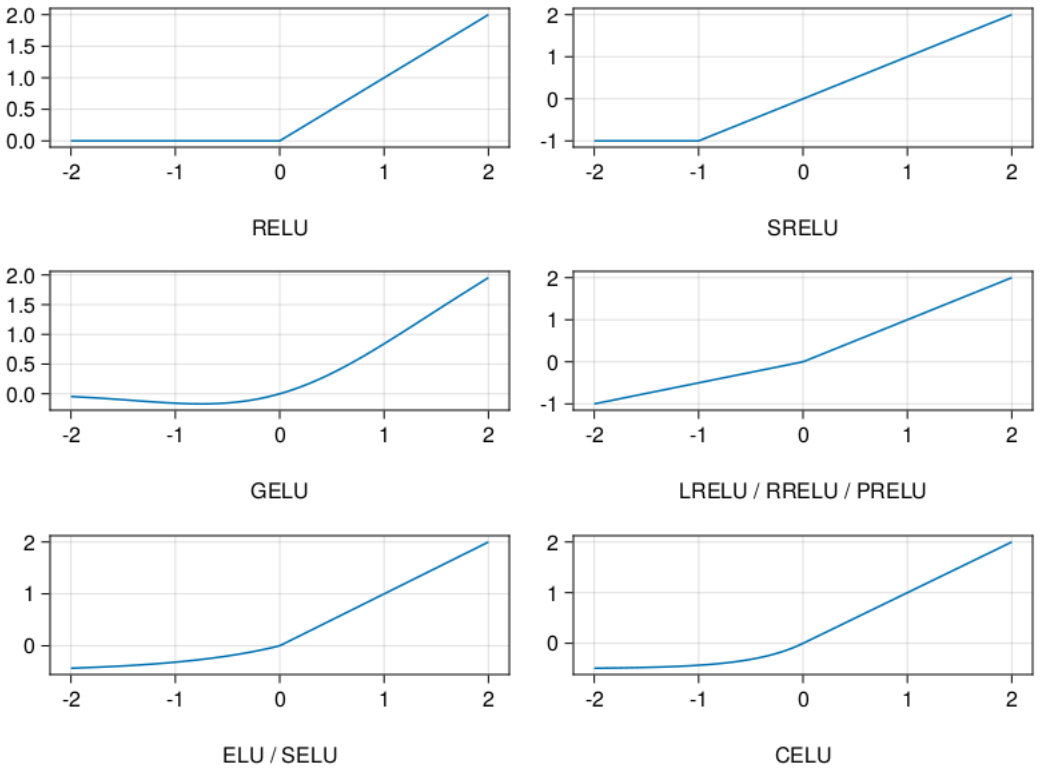

FIGURE C.2: Regression AFs

C.3.1 Mean Absolute Error (MAE)

MAE, also known as the L1 loss, is measured as the average of the sum of absolute differences between predictions and actual observations. This measures the magnitude of error without considering the direction. MAE needs more complicated tools, such as linear programming, to compute the gradients. The formula for MAE is (the total amount of policies is n):

$$MAE(x, y) = \frac{1}{n} \sum_{t=1}^{n} |x_t - y_t| \tag{C.33}$$

MAE penalises outliers less than MSE. This is much less common in the ML domain as compared to its counterpart. Clearly, there's a need for caution as positive and negative errors could cancel each other out, and still predict an overall total close to the actual total. Although less accurate in practice, the MAE is useful to determine whether the model has positive or negative bias.

C.3.2 Mean Squared Error (MSE)

MSE, also known as L2 loss, is measured as the average of the squared difference between predictions and actual observations. It's only concerned with the average magnitude of error

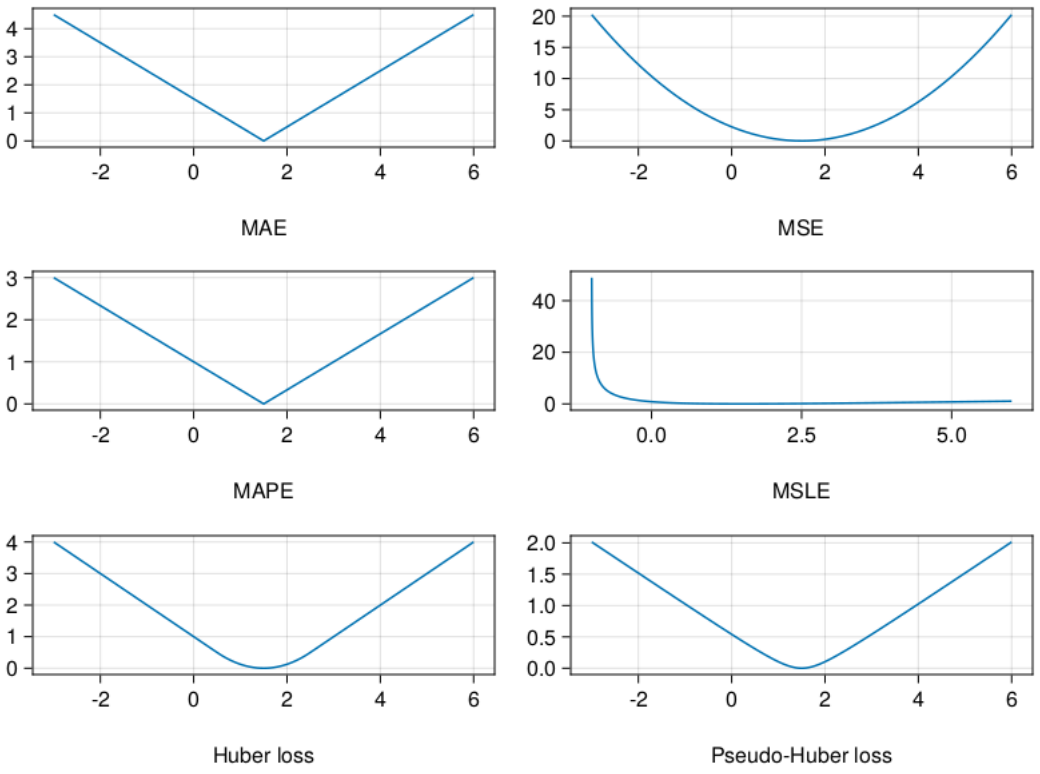

FIGURE C.3: Loss functions 1/2

irrespective of their direction. Predictions which are far away from actual values are penalised heavily in comparison to less deviated predictions. MSE has mathematical properties which make it simple to calculate gradients.

The formula for MSE is:

$$MSE(x, y) = \frac{1}{n}\sum_{t=1}^{n}(x_t - y_t)^2 \tag{C.34}$$

MSE is more suitable if there are many outliers because the derivative at large differences is more than at small differences, and will converge faster than MAE in total, but not for all individual records equally well.

C.3.3 Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) is the square root of the MSE. It has the advantage that it is of comparable scale to the values you are trying to predict, but it is still more difficult to interpret than MAE.

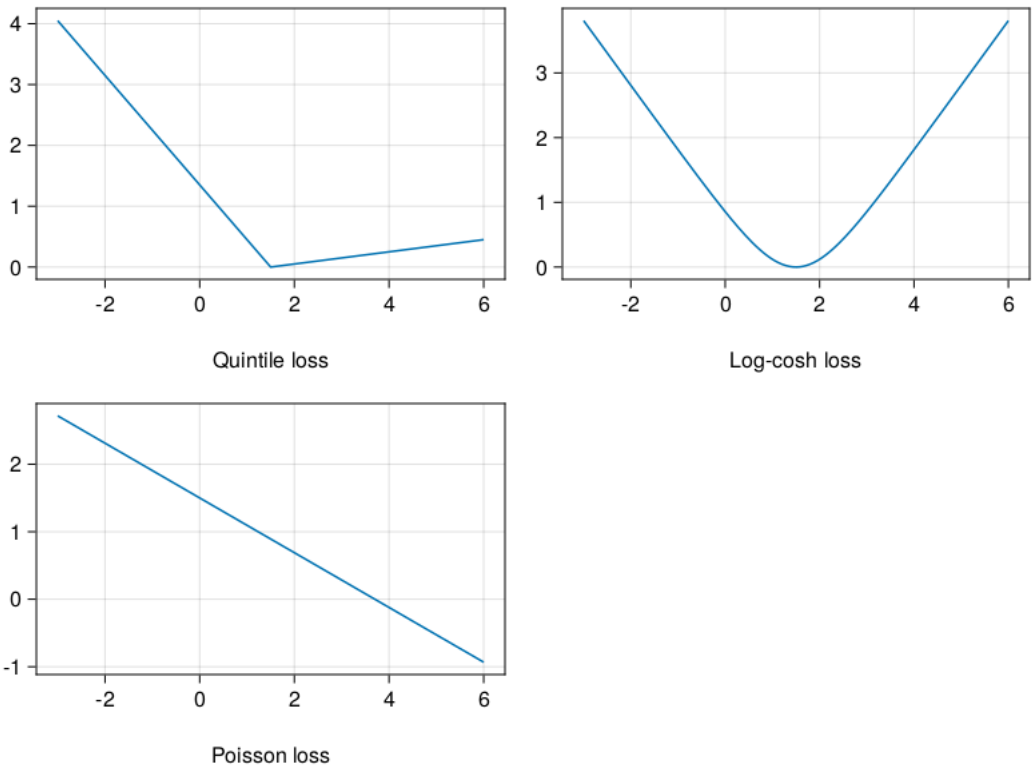

FIGURE C.4: Loss functions 2/2

The formula for RMSE is:

$$RMSE(x, y) = \sqrt{MSE} \tag{C.35}$$

For comparing the accuracy among different linear regression models, RMSE is a better choice than R^2.

C.3.4 Mean Squared Logarithmic Error (MSLE)

When confronted with regression problems where the target value has a spread of values, and when we try to predict a very large amount, we may not want to punish the model as much as we punish it with the MSE. The formula for Mean Squared Logarithmic Error (MSLE) is:

$$MSLE(x, y) = \frac{1}{n}\sum_{t=1}^{n}(log(1 + x_t) - log(1 + y_t))^2 = \frac{1}{n}\sum_{t=1}^{n}(\frac{log(1 + x_t)}{log(1 + y_t)})^2 \tag{C.36}$$

Small differences between small values are treated the same as big differences between big numbers. The downside with MSLE is that it penalises underestimates more than overestimates, which creates more bias, while training leads to a longer time for the model to converge.

C.3.5 Huber loss

The Huber loss, like the MSLE loss, is also less sensitive to extreme values in the data. It also takes an input parameter α, which can be set by the researcher to decide when to use the quadratic part or the linear part. The formula for the Huber loss is:

$$Huberloss(x_t, y_t, \alpha) = \begin{cases} \frac{1}{2}(x_t - y_t)^2, & \text{if } |x_t - y_t| \leq \alpha \\ \alpha|x_t - y_t| - \frac{1}{2}\alpha^2, & \text{otherwise} \end{cases} \tag{C.37}$$

An interpretation of the Huber loss is therefore that it is like MSE when losses are small, and like the MAE when losses are large.

C.3.6 Pseudo-Huber loss

The Pseudo-Huber loss is a smooth approximation of the Huber loss that ensures the loss is differentiable everywhere. The formula is given by:

$$PsuedoHL(x_t, y_t, \alpha) = \alpha^2 * (\sqrt{1 + (\frac{x_t - y_t}{\alpha})^2 - 1}) \tag{C.38}$$

For the Pseudo-Huber loss, the parameter α controls how steep the linear part is.

C.3.7 Quantile loss

Quantile regression is an extension of linear regression used when the conditions of linear regression are not met. The advantage of using the quantile loss rather than MSE is that the quantile regression estimates are more robust against extremes. The formula is:

$$QuintileL(x_t, y_t, q) = \begin{cases} q * (x_t - y_t), & (if) x_t - y_t \geq 0 \\ (q-1) * (x_t - y_t), & (otherwise) \end{cases} \tag{C.39}$$

where q is the quantile. For example for 10% quintiles q would be 0.9 and for the mean q would be 0.5, which is equal to the MAE.

C.3.8 Log-Cosh loss

The Log-Cosh LF is another variant of the Huber LF and compared to other methods it has the characteristics that it produces predictions with smaller variance than the MAE. The formula is:

$$LogCosh(x_t, y_t) = log(cosh(x_t - y_t)) \tag{C.40}$$

The Log-Cosh function is not smooth (has Non-Monotonicity) when $x = y$. For further discussion, please see Saleh et al. (2022).

C.3.9 Poisson loss

The Poisson LF is useful when the solution space is discrete and when we count items that are distributed exponentially. The formula is:

$$Poisson(x_t, y_t) = x_t - y_t * log(x_t) \tag{C.41}$$

The Poison LF can have both positive and negative gradients, depending on the relationship between x_t and y_t. When $x_t = y_t$, this LF can suffer from vanishing gradients.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015), 'TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/* Cited in section(s): 3.2

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. (2019), Optuna: A next-generation hyperparameter optimization framework, *in* 'Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining', pp. 2623–2631. Cited in section(s): 1.5.13

Albelwi, S. and Mahmood, A. (2017), 'A framework for designing the architectures of deep convolutional neural networks', *Entropy* **19**(6), 242. MDPI. Cited in section(s): 2.4.5

Albrecher, H., Bauer, D., Embrechts, P., Filipović, D., Koch-Medina, P., Korn, R., Loisel, S., Pelsser, A., Schiller, F., Schmeiser, H. et al. (2018), 'Asset-liability management for long-term insurance business', *European Actuarial Journal* **8**, 9–25. Springer. Cited in section(s): 1.4.4

Albrecher, H., Bommier, A., Filipović, D., Koch-Medina, P., Loisel, S. and Schmeiser, H. (2019), 'Insurance: Models, digitalization, and data science', *European Actuarial Journal* **9**, 349–360. Springer. Cited in section(s): 1.4.1

Amari, S. and Saito, H. (1967), 'Learning patterns and pattern sequences by self-organizing nets of threshold elements', *IEEE Transactions* . Cited in section(s): 2.3.5

Anderson, N., Walsh, K., Flynn, J. and Walsh, J. P. (2021), 'Achieving robustness across season, location and cultivar for a NIRS model for intact mango fruit dry matter content. II. Local PLS and nonlinear models', *Postharvest Biology and Technology* **171**, 111358. Elsevier. Cited in section(s): 2.4.5

Arik, S. Ö. and Pfister, T. (2021), 'Tabnet: Attentive interpretable tabular learning', *AAAI* **35(8)**, 6679–6687. Cited in section(s): 1.5.2, 2.4.4, 2.5.1

Auer, P., Herbster, M. and Warmuth, M. K. (1995), 'Exponentially many local minima for single neurons', *Advances in neural information processing systems* **8**. Cited in section(s): 1.5.12

Babbel, D. F. and Merrill, C. (1999), 'Toward a unified valuation model for life insurers', *Changes in the Life Insurance Industry: Efficiency, Technology and Risk Management* pp. 245–278. Springer. Cited in section(s): 2.2.4

Barigou, K. and Delong, Ł. (2022), 'Pricing equity-linked life insurance contracts with multiple risk factors by neural networks', *Journal of Computational and Applied Mathematics* **404**, 113922. Elsevier. Cited in section(s): 2.4.1

Barigou, K., Linders, D. and Yang, F. (2022), 'Actuarial-consistency and two-step actuarial valuations: A new paradigm to insurance valuation', *Scandinavian Actuarial Journal* pp. 1–27. Taylor & Francis. Cited in section(s): 2.2.5

Barron, J. T. (2017), 'Continuously differentiable exponential linear units', *arXiv preprint arXiv:1704.07483* . Cited in section(s): 2.3.13, C.2.14, C.2.16

Baudry, M. and Robert, C. Y. (2019), 'A machine learning approach for individual claims reserving in insurance', *Applied Stochastic Models in Business and Industry* **35**(5), 1127–1155. Wiley Online Library. Cited in section(s): 2.4.2

Bauer, D., Reuss, A. and Singer, D. (2012), 'On the calculation of the solvency capital requirement based on nested simulations', *ASTIN Bulletin: The Journal of the IAA* **42**(2), 453–499. Cambridge University Press. Cited in section(s): 2.4.4

Belkin, M., Hsu, D., Ma, S. and Mandal, S. (2019), 'Reconciling modern machine-learning practice and the classical bias–variance trade-off', *Proceedings of the National Academy of Sciences* **116**(32), 15849–15854. Cited in section(s): 2.3.2

Bengio, Y., Courville, A. and Vincent, P. (2013), 'Representation learning: A review and new perspectives', *IEEE transactions on pattern analysis and machine intelligence* **35**(8), 1798–1828. IEEE. Cited in section(s): 1.7.4, 2.3.2, 2.3.7

Bengio, Y., Simard, P. and Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks* **5**(2), 157–166. Cited in section(s): 1.5.14

Besard, T., Foket, C. and De Sutter, B. (2018), 'Effective extensible programming: Unleashing Julia on GPUs', *IEEE Transactions on Parallel and Distributed Systems* **30**(4), 827–841. IEEE. Cited in section(s): 3.2

Bezanson, J., Edelman, A., Karpinski, S. and Shah, V. B. (2017), 'Julia: A fresh approach to numerical computing', *SIAM Review* **59**(1), 65–98.
URL: *https://epubs.siam.org/doi/10.1137/141000671* Cited in section(s): 3

Biffis, E. (2005), 'Affine processes for dynamic mortality and actuarial valuations', *Insurance: mathematics and economics* **37**(3), 443–468. Elsevier. Cited in section(s): 2.2.4

Bishop, C. M. (2008), 'A new framework for machine learning', *IEE* pp. 1–24. Cited in section(s): 2.3.1

Bishop, C. M. and Nasrabadi, N. M. (2006), *Pattern recognition and machine learning*, Vol. 4, Springer. Cited in section(s): 2.3.1

Blier-Wong, C., Cossette, H., Lamontagne, L. and Marceau, E. (2020), 'Machine learning in P&C insurance: A review for pricing and reserving', *Risks* **9**(1), 4. MDPI. Cited in section(s): 2.4.2

Blomerus, J. M. (2023), 'Pricing of a dataset of an actual life-risk insurance portfolio'.
URL: *https://ufs.figshare.com/articles/dataset/Pricing_of_a_dataset_of_an_actual_life-risk_insurance_portfolio/22316941/?file=39704857* Cited in section(s): 3.4.1

Bollapragada, R., Nocedal, J., Mudigere, D., Shi, H.-J. and Tang, P. T. P. (2018), A progressive batching L-BFGS method for machine learning, *in* 'International Conference on Machine Learning', PMLR, pp. 620–629. Cited in section(s): C.1.8

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**(1), 5–32. Cited in section(s): 1.5.2

Brouwer, R. K. (2004), 'A hybrid neural network for input that is both categorical and quantitative', *International journal of intelligent systems* **19**(10), 979–1001. Wiley Online Library. Cited in section(s): 1.5.11

Bulinskaya, E. (2017), 'New research directions in modern actuarial sciences', *Modern Problems of Stochastic Analysis and Statistics: Selected Contributions In Honor of Valentin Konakov* pp. 349–408. Springer. Cited in section(s): 2.2

Byrd, R. H., Chin, G. M., Nocedal, J. and Wu, Y. (2012), 'Sample size selection in optimization methods for machine learning', *Mathematical programming* **134(1)**, 127–155. Springer. Cited in section(s): 2.3.4

Candel, A. and LeDell, E. (2022), 'Deep Learning with H2O'.
URL: *https://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/DeepLearningBooklet.pdf* Cited in section(s): 3.2

Candel, A. and Malohlava, M. (2022), 'Gradient Boosted Models'.
URL: *https://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/GBMBooklet.pdf* Cited in section(s): 3.2

Carvalho, D. V., Pereira, E. M. and Cardoso, J. S. (2019), 'Machine learning interpretability: A survey on methods and metrics', *Electronics* **8**(8), 832. MDPI. Cited in section(s): 2.5.1

Castellani, G., Fiore, U., Marino, Z., Passalacqua, L., Perla, F., Scognamiglio, S. and Zanetti, P. (2018), 'An investigation of machine learning approaches in the Solvency II valuation framework', *Available at SSRN 3303296* . Cited in section(s): 2.2.5, 2.4.1

CFO Forum (2008), *Market Consistent Embedded Value Principles*.
URL: *https://cfoforum.eu/downloads/CFO-Forum-MCEV-Principles-and-Guidance.pdf*
Cited in section(s): 2.2.5

Changpinyo, S., Sandler, M. and Zhmoginov, A. (2017), 'The power of sparsity in convolutional neural networks', *arXiv preprint arXiv:1702.06257* . Cited in section(s): 2.3.7

Chen, T. and Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining', pp. 785–794. Cited in section(s): 1.5.2

Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), Learning phrase representations using rnn encoder-decoder for statistical machine translation, *in* 'EMNLP', pp. 1724–1734. Cited in section(s): 1.5.2

Chollet, F. et al. (2015), 'Keras'.
URL: *https://github.com/fchollet/keras* Cited in section(s): 2.4.1, 3.2

Clevert, D.-A., Unterthiner, T. and Hochreiter, S. (2015), 'Fast and accurate deep network learning by Exponential Linear Units (ELUs)', *arXiv preprint arXiv:1511.07289* . Cited in section(s): C.2.11, C.2.14

Cortes, C. and Vapnik, V. (1995), 'Support-vector networks', *Machine Learning* **20**(3), 273–297. Cited in section(s): 1.5.2

Creswell, J. W. and Creswell, J. D. (2017), *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, Sage. Cited in section(s): 1.8

Cui, C. and Fearn, T. (2018), 'Modern practical convolutional neural networks for multivariate regression: Applications to NIR calibration', *Chemometrics and Intelligent Laboratory Systems* **182**, 9–20. Elsevier. Cited in section(s): 2.4.5

Cummings, J. and Hartman, B. (2022), 'Using Machine Learning to Better Model Long-Term Care Insurance Claims', *North American Actuarial Journal* **26**(3), 470–483. Taylor & Francis. Cited in section(s): 2.4.1

Dahl, G. E., Sainath, T. N. and Hinton, G. E. (2013), Improving deep neural networks for lvcsr using rectified linear units and dropout, *in* '2013 IEEE international conference on acoustics, speech and signal processing', IEEE, pp. 8609–8613. Cited in section(s): C.2.8

Dahl, M. and Møller, T. (2006), 'Valuation and hedging of life insurance liabilities with systematic mortality risk', *Insurance: Mathematics and Economics* **39**(2), 193–217. Elsevier. Cited in section(s): 2.2.4

Dahouda, M. K. and Joe, I. (2021), 'A deep-learned embedding technique for categorical features encoding', *IEEE Access* **9**, 114381–114391. IEEE. Cited in section(s): 1

Danisch, S. and Krumbiegel, J. (2021), 'Makie.jl: Flexible high-performance data visualization for Julia', *Journal of Open Source Software* **6**(65), 3349.
URL: *https://doi.org/10.21105/joss.03349* Cited in section(s): 3

De Virgilis, M. and Cerqueti, P. (2020), 'Estimation of Individual Claim Liabilities', *CAS Working Party* . Cited in section(s): 2.4.2

De Virgilis, M., Lupton, D., McGrath, L., Qazvini, M., Roby, S. and Vernon, L. (2022), 'Machine learning in insurance'. Cited in section(s): 2.4

Di Lorenzo, E., Piscopo, G., Sibillo, M. and Tizzano, R. (2021), 'Reverse mortgages through artificial intelligence: New opportunities for the actuaries', *Decisions in Economics and Finance* **44**, 23–35. Springer. Cited in section(s): 2.3.7, 2.4.5

Dickson, D. C., Hardy, M. R. and Waters, H. R. (2020), 'Actuarial Mathematics for Life Contingent Risks', **3**. Cambridge University Press. Cited in section(s): 2.2, 2.2.2

Dong, W. and Zhou, M. (2016), 'A supervised learning and control method to improve particle swarm optimization algorithms', *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **47**(7), 1135–1148. IEEE. Cited in section(s): 2.3.16

Dong, X., Yu, Z., Cao, W., Shi, Y. and Ma, Q. (2020), 'A survey on ensemble learning', *Frontiers of Computer Science* **14**, 241–258. Cited in section(s): 1.5.16, 1.5.16

Doyle, D. and Groendyke, C. (2018), 'Using neural networks to price and hedge variable annuity guarantees', *Risks* **7**(1), 1. MDPI. Cited in section(s): 1.7.4, 2.4.1

Dozat, T. (2016), 'Incorporating Nesterov Momentum into Adam'.
URL: *https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ* Cited in section(s): 2.3.12, 2.4.2, C.1.7

Du, M., Liu, N. and Hu, X. (2019), 'Techniques for interpretable machine learning', *Communications of the ACM* **63**(1), 68–77. ACM New York, NY, USA. Cited in section(s): 2.5.1

Duchi, J., Hazan, E. and Singer, Y. (2011), 'Adaptive Subgradient Methods for Oline Learning and Stochastic Optimization.', *Journal of Machine Learning Research* **12**(7). Cited in section(s): C.1.1

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C. and Garcia, R. (2000), 'Incorporating second-order functional knowledge for better option pricing', *Advances in neural information processing systems* **13**. Cited in section(s): C.2.5

Eckerli, F. and Osterrieder, J. (2021), 'Generative adversarial networks in finance: An overview', *arXiv preprint arXiv:2106.06364* . Cited in section(s): 2.4.5

Elbrächter, D., Perekrestenko, D., Grohs, P. and Bölcskei, H. (2021), 'Deep neural network approximation theory', *IEEE Transactions on Information Theory* **67**(5), 2581–2623. IEEE. Cited in section(s): 2.3.7

Elman, J. L. (1990), 'Finding structure in time', *Cognitive Science* **14**(2), 179–211. Cited in section(s): 1.5.2

European Commission (2009), 'Framework Solvency II Directive'. (Directive 2009/138/EC). Cited in section(s): 2.2.5

Fawcett, T. (2006), 'An introduction to ROC analysis', *Pattern recognition letters* **27**(8), 861–874. Cited in section(s): 1.5.18

Fernandez-Arjona, L. and Filipović, D. (2022), 'A machine learning approach to portfolio pricing and risk management for high-dimensional problems', *Mathematical Finance* **32**(4), 982–1019. Wiley Online Library. Cited in section(s): 1.7.6

Ferrario, A., Noll, A. and Wuthrich, M. V. (2020), 'Insights from inside neural networks', *Available at SSRN 3226852* . Cited in section(s): 2.4.2

Fissler, T., Lorentzen, C. and Mayer, M. (2022), 'Model comparison and calibration assessment: User guide for consistent scoring functions in machine learning and actuarial practice', *arXiv preprint arXiv:2202.12780* . Cited in section(s): 2.3.14

Friedlander, M. P. and Schmidt, M. (2012), 'Hybrid deterministic-stochastic methods for data fitting', *SIAM Journal on Scientific Computing* **34(3)**, A1380–A1405. SIAM. Cited in section(s): 2.3.4

Gabrielli, A. (2021), 'An individual claims reserving model for reported claims', *European Actuarial Journal* **11**, 541–577. Springer. Cited in section(s): 2.4.2

Gabrielli, A., Richman, R. and Wüthrich, M. V. (2020), 'Neural network embedding of the over-dispersed Poisson reserving model', *Scandinavian Actuarial Journal* **2020**(1), 1–29. Taylor & Francis. Cited in section(s): 2.4.3

Gabrielli, A. and V. Wüthrich, M. (2018), 'An individual claims history simulation machine', *Risks* **6**(2), 29. MDPI. Cited in section(s): 2.4.2

Gal, Y. and Ghahramani, Z. (2016), 'Dropout as a bayesian approximation: Representing model uncertainty in deep learning', *International Conference on Machine Learning* . Cited in section(s): 2.3.2

Gan, G. (2013), 'Application of data clustering and machine learning in variable annuity valuation', *Insurance: Mathematics and Economics* **53**(3), 795–801. Elsevier. Cited in section(s): 2.3.2

Gan, G. and Lin, X. S. (2015), 'Valuation of large variable annuity portfolios under nested simulation: A functional data approach', *Insurance: Mathematics and Economics* **62**, 138–150. Elsevier. Cited in section(s): 4, 2.3.2, 2.1

Glorot, X. and Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* 'Proceedings of the thirteenth international conference on artificial intelligence and statistics', JMLR Workshop and Conference Proceedings, pp. 249–256. Cited in section(s): 1.5.4, 2.3.8

Glorot, X., Bordes, A. and Bengio, Y. (2011), Deep sparse rectifier neural networks, *in* 'Proceedings of the fourteenth international conference on artificial intelligence and statistics', JMLR Workshop and Conference Proceedings, pp. 315–323. Cited in section(s): 2.3.7, 2.3.8, C.2.8

Golizadeh, H., Banihashemi, S., Sadeghifam, A. N. and Preece, C. (2017), 'Automated estimation of completion time for dam projects', *International Journal of Construction Management* **17**(3), 197–209. Taylor & Francis. Cited in section(s): 3.8

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep learning*, MIT press. Cited in section(s): 1.1, 1.7.4, 2.3, 2.3.2, 2.3.7, 2.3.8, 2.3.13, 2.4.2, C.1.2, C.1.5

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014), Generative adversarial nets, *in* 'Advances in Neural Information Processing Systems', Vol. 27. Cited in section(s): 1.5.2

Guo, C. and Berkhahn, F. (2016), 'Entity embeddings of categorical variables', *arXiv preprint arXiv:1604.06737* . Cited in section(s): 1.5.11

Gustafsson, A. and Hansén, J. (2021), 'Combined actuarial neural networks in actuarial rate making'. Cited in section(s): 1.5.2

Haberman, S. (1996), 'Landmarks in the History of Actuarial Science (up to 1919).', *Insurance Mathematics and Economics* **2**(18), 153–154. Cited in section(s): 2.2

He, K., Zhang, X., Ren, S. and Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on Imagenet classification, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1026–1034. Cited in section(s): 2.3.8, 2.4.1, 5.2.1, C.2.13

Hejazi, S. A. (2016), A Neural Network Approach to Efficient Valuation of Large VA Portfolios, PhD thesis, University of Toronto (Canada). Cited in section(s): 2.2.3, 2.3.1, 2.3.3, 2.3.8, 2.3.11, 2.4.1

Hejazi, S. A. and Jackson, K. R. (2016), 'A neural network approach to efficient valuation of large portfolios of variable annuities', *Insurance: Mathematics and Economics* **70**, 169–181. Elsevier. Cited in section(s): 2.4.1

Hejazi, S. A. and Jackson, K. R. (2017), 'Efficient valuation of SCR via a neural network approach', *Journal of Computational and Applied Mathematics* **313**, 427–439. Elsevier. Cited in section(s): 2.4.1

Hendrycks, D. and Gimpel, K. (2016), 'Gaussian error linear units (GELUs)', *arXiv preprint arXiv:1606.08415* . Cited in section(s): C.2.10

Hinton, G. E., Osindero, S. and Teh, Y.-W. (2006), 'A fast learning algorithm for deep belief nets', *Neural Computation* **18**(7), 1527–1554. Cited in section(s): 1.5.2

Hinton, G. E. and Salakhutdinov, R. R. (2006), 'Reducing the dimensionality of data with neural networks', *Science* **313**(5786), 504–507. Cited in section(s): 1.5.2

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. R. (2012), 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv preprint arXiv:1207.0580* . Cited in section(s): 2.3.9, 2.3.15

Hochreiter, S. and Schmidhuber, J. (1997), 'Long short-term memory', *Neural Computation* **9**(8), 1735–1780. Cited in section(s): 1.5.2

Hornik, K., Stinchcombe, M. and White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**(5), 359–366. Cited in section(s): 2.3.2

Huang, Z. (1998), 'Extensions to the k-means algorithm for clustering large data sets with categorical values', *Data mining and knowledge discovery* **2**(3), 283–304. Springer. Cited in section(s): 2.3.2

Hynes, N., Sculley, D. and Terry, M. (2017), The data linter: Lightweight, automated sanity checking for ML data sets, *in* 'NIPS MLSys Workshop', Vol. 1, p. 5. Cited in section(s): 1.7.4, 3.9.4

Innes, M. (2018), 'Flux: Elegant Machine Learning with Julia', *Journal of Open Source Software* . Cited in section(s): 3

Ioffe, S. and Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning', PMLR, pp. 448–456. Cited in section(s): 1.5.4, 2.3.6

Isaaks, E. H. and Srivastava, R. M. (1989), 'Applied geostatistics: Oxford University Press', *New York* **561**. USA New York. Cited in section(s): 2.3.2

Kadra, A., Lindauer, M., Hutter, F. and Grabocka, J. (2021), 'Regularization is all you need: Simple neural nets can excel on tabular data', *arXiv preprint arXiv:2106.11189* **536**. Cited in section(s): 2.3.15

Katzir, L., Elidan, G. and El-Yaniv, R. (2021), Net-dnf: Effective deep modeling of tabular data, *in* 'International Conference on Learning Representations'. Cited in section(s): 1.5.2, 2.4.4

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P. T. P. (2016), 'On large-batch training for deep learning: Generalization gap and sharp minima', *arXiv preprint arXiv:1609.04836* . Cited in section(s): 2.3.4

Kevin, P. and Kang, D.-K. (2017), 'The Effect of Hyperparameter Choice on RELU and SELU Activation Function', *International Journal of Advanced Smart Convergence* **6**(4), 73–79. The Institute of Internet, Broadcasting and Communication. Cited in section(s): 1.5.7, C.1.4, C.2.15

Kiefer, J. and Wolfowitz, J. (1952), 'Stochastic estimation of the maximum of a regression function', *The Annals of Mathematical Statistics* **23**(3), 462–466. Cited in section(s): 2.3.5

Kiliçarslan, S., Kemal, A. and Çelik, M. (2021), 'An overview of the activation functions used in deep learning algorithms', *Journal of New Results in Science* **10(3)**, 75–88. Cited in section(s): C.2.6, C.2.16

Kingma, D. P. and Ba, J. (2014), 'ADAM: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* . International Conference on Learning Representations. Cited in section(s): 2.3.12, 2.4.1, C.1.4

Klambauer, G., Unterthiner, T., Mayr, A. and Hochreiter, S. (2017), 'Self-normalizing neural networks', *Advances in neural information processing systems* **30**. Cited in section(s): 2.3.8, C.2.15

Kochenderfer, M. J. and Wheeler, T. A. (2019), *Algorithms for optimization*, Mit Press. Cited in section(s): 2.3.12

Krige, D. G. (1951), 'A statistical approach to some basic mine valuation problems on the witwatersrand', *Journal of the Southern African Institute of Mining and Metallurgy* **52**(6), 119–139. Cited in section(s): 2.3.2

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**. Cited in section(s): 1.5.4, 2.3.13

Kuo, K. (2019), 'Deeptriangle: A deep learning approach to loss reserving', *Risks* **7**(3), 97. Multidisciplinary Digital Publishing Institute. Cited in section(s): 2.4.2

Landry, M. (2022), 'Machine Learning with R and H2O'.
URL: *https://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/RBooklet.pdf* Cited in section(s): 3.2

Laurent, J.-P. (2016), *Modelling in life insurance–a management perspective*, Springer. Cited in section(s): 2.2.5

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324. Cited in section(s): 1.1, 1.5.2

LeCun, Y., Cortes, C. and Burges, C. J. C. (1998), 'The mnist database of handwritten digits', `http://yann.lecun.com/exdb/mnist/`. Accessed: YYYY-MM-DD. Cited in section(s): 2.3.12

LeDell, E., Gill, N., Aiello, S., Candel, A., Click, C., Kraljevic, T., Kurka, M., Fu, A., Wong, W. and Botzer, Y. (2023), *h2o: R Interface for H2O*. R package version 3.42.0.2.
URL: *https://CRAN.R-project.org/package=h2o* Cited in section(s): 3.2

Lee, R. D. and Carter, L. R. (1992), 'Modeling and forecasting u.s. mortality', *Journal of the American Statistical Association* **87**(419), 659–671. Cited in section(s): 2.4.3

Lishner, I. and Shtub, A. (2022), 'Using an Artificial Neural Network for Improving the Prediction of Project Duration', *Mathematics* **10**(22), 4189. MDPI. Cited in section(s): 2.4.5

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J. and Han, J. (2019), 'On the variance of the adaptive learning rate and beyond', *arXiv preprint arXiv:1908.03265* . Cited in section(s): 2.3.10

Llaguno, L. S., Bardis, E. T., Chin, R. A., Gwilliam, C. L., Hagerstrand, J. A. and Petzoldt, E. C. (2017), Reserving with Machine Learning: Applications for Loyalty Programs and Individual Insurance Claims, *in* 'Arlington: Casualty Actuarial Society Forum'. Cited in section(s): 2.4.2

Lledó, J. and Pavía, J. M. (2022), 'Dataset of an actual life-risk insurance portfolio', *Data in Brief* **45**, 108655. Elsevier. Cited in section(s): 3.4.1, 3.4.3

Lorentzen, C. and Mayer, M. (2020), 'Peeking into the black box: An actuarial case study for interpretable machine learning', *Available at SSRN 3595944* . Cited in section(s): 2.4.2

Loshchilov, I. and Hutter, F. (2017), 'Decoupled weight decay regularization', *arXiv preprint arXiv:1711.05101* . Cited in section(s): 2.3.15

Lu, Z., Pu, H., Wang, F., Hu, Z. and Wang, L. (2017), 'The expressive power of neural networks: A view from the width', *Advances in Neural Information Processing Systems* . Cited in section(s): 2.3.2, 2.3.2

Maas, A. L., Hannun, A. Y., Ng, A. Y. et al. (2013), Rectifier nonlinearities improve neural network acoustic models, *in* 'Proc. icml', Vol. 30(1), Atlanta, Georgia, USA, p. 3. Cited in section(s): C.2.11

Macdonald, A. S. (2006), 'Commutation functions', *Encyclopedia of Actuarial Science* **1**. Wiley Online Library. Cited in section(s): 2.2.1

MacKay, D. J. C. (1992), 'A practical bayesian framework for backpropagation networks', *Neural Computation* **4**(3), 448–472. Cited in section(s): 2.3.2

Martens, J. et al. (2010), Deep learning via Hessian-free optimization, *in* 'ICML', Vol. 27, pp. 735–742. Cited in section(s): 2.3.8

Mhaskar, H., Liao, Q. and Poggio, T. (2017), When and why are deep networks better than shallow ones?, *in* 'Proceedings of the AAAI conference on artificial intelligence', Vol. 31(1). Cited in section(s): 2.3.7

Ming, Y. (2017), 'A survey on visualization for explainable classifiers', *Hong Kong* . Cited in section(s): 2.5.1

Montavon, G., Binder, A., Lapuschkin, S., Samek, W. and Müller, K.-R. (2019), 'Layer-wise relevance propagation: An overview', *Explainable AI: interpreting, explaining and visualizing deep learning* pp. 193–209. Springer. Cited in section(s): 2.5.1

Murphy, K. P. (2012), *Machine Learning: A Probabilistic Perspective*, MIT press. Cited in section(s): 2.3

Murrell, H. and de Freitas, N. (2019), 'Deep Learning Notes using Julia with Flux'.
URL: *https://hughmurrell.github.io/DeepLearningNotes/deep.pdf* Cited in section(s): 2.3

Nair, V. and Hinton, G. E. (2010), Rectified linear units improve Restricted Boltzmann Machines, *in* 'Proceedings of the 27th International Conference on Machine Learning (ICML-10)', pp. 807–814. Cited in section(s): C.2.8

Nelder, J. A. and Wedderburn, R. W. M. (1972), 'Generalized linear models', *Journal of the Royal Statistical Society: Series A (General)* **135**(3), 370–384. Cited in section(s): 1.5.2

Nesterov, Y. E. (1983), A method for solving the convex programming problem with convergence rate o $(1/\kappa\hat{}\ 2)$, *in* 'Dokl. akad. nauk Sssr', Vol. 269, pp. 543–547. Cited in section(s): C.1.5

Noll, A., Salzmann, R. and Wuthrich, M. V. (2020), 'Case study: French motor third-party liability claims', *Available at SSRN 3164764* . Cited in section(s): 2.4.2

Norgaard, L., Lagerholm, M. and Westerhaus, M. (2013), 'Artificial Neural Networks and Near Infrared Spectroscopy - A case study on protein content in whole wheat grain', *Foss White Paper http://www. foss. dk/campaign/-/media/242657904D734CE9B0652C3D885776AE. ashx* . Cited in section(s): 2.1

Olmschenk, G., Zhu, Z. and Tang, H. (2019), 'Generalizing semi-supervised generative adversarial networks to regression using feature contrasting', *Computer Vision and Image Understanding* **186**, 1–12. Elsevier. Cited in section(s): 2.4.5

Owens, E., Sheehan, B., Mullins, M., Cunneen, M., Ressel, J. and Castignani, G. (2022), 'Explainable Artificial Intelligence (XAI) in Insurance', *Risks* **10**(12), 230. MDPI. Cited in section(s): 2.4, 2.5.1

Palmborg, L., Lindholm, M. and Lindskog, F. (2021), 'Financial position and performance in IFRS 17', *Scandinavian Actuarial Journal* **2021**(3), 171–197. Taylor & Francis. Cited in section(s): 2.2.6

Passos, D. and Mishra, P. (2022), 'A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks', *Chemometrics and Intelligent Laboratory Systems* p. 104520. Elsevier. Cited in section(s): 2.3.10, 2.4.5

Payrovnaziri, S. N., Chen, Z., Rengifo-Moreno, P., Miller, T., Bian, J., Chen, J. H., Liu, X. and He, Z. (2020), 'Explainable artificial intelligence models using real-world electronic health record data: A systematic scoping review', *Journal of the American Medical Informatics Association* **27**(7), 1173–1185. Oxford University Press. Cited in section(s): 2.5.1

Pedamonti, D. (2018), 'Comparison of non-linear activation functions for deep neural networks on MNIST classification task', *arXiv preprint arXiv:1804.02763* . Cited in section(s): C.2.15

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830. Cited in section(s): 2.4.1

Perla, F., Richman, R., Scognamiglio, S. and Wüthrich, M. V. (2021), 'Time-series forecasting of mortality rates using deep learning', *Scandinavian Actuarial Journal* **2021**(7), 572–598. Taylor & Francis. Cited in section(s): 2.4.3

Pitacco, E., Denuit, M. and Haberman, S. (2009), *Modelling longevity dynamics for pensions and annuity business*, Oxford University Press. Cited in section(s): 2.2.3

Plat, R. (2009), 'On stochastic mortality modeling', *Insurance: Mathematics and Economics* **45**(3), 393–404. Elsevier. Cited in section(s): 2.2.3

Poggio, T., Banburski, A. and Liao, Q. (2020), 'Theoretical issues in deep networks', *Proceedings of the National Academy of Sciences* **117**(48), 30039–30045. National Acad Sciences. Cited in section(s): 1.5.15, 2.3.15

Popov, S., Morozov, S. and Babenko, A. (2019), 'Neural oblivious decision ensembles for deep learning on tabular data', *arXiv preprint arXiv:1909.06312* . Cited in section(s): 2.4.4

Popper, K. (2005), *The Logic of Scientific Discovery*, Routledge. Cited in section(s): 1.8

Qian, N. (1999), 'On the momentum term in gradient descent learning algorithms', *Neural networks* **12**(1), 145–151. Cited in section(s): 2.3.11

Qian, X. and Klabjan, D. (2020), 'The impact of the mini-batch size on the variance of gradients in stochastic gradient descent', *arXiv preprint arXiv:2004.13146* . Cited in section(s): 2.3.4, 5.2.3

Quinlan, J. R. (1986), Induction of decision trees, *in* 'Machine Learning', Vol. 1, Springer, pp. 81–106. Cited in section(s): 1.5.2

Rachmatullah, M. I. C., Santoso, J. and Surendro, K. (2021), 'Determining the number of hidden layer and hidden neuron of neural network for wind speed prediction', *PeerJ Computer Science* **7**, e724. Cited in section(s): 3.6.7

Ramachandran, P., Zoph, B. and Le, Q. V. (2017), 'Searching for activation functions', *arXiv preprint arXiv:1710.05941* . Cited in section(s): C.2.6

Razali, N. M., Geraghty, J. et al. (2011), Genetic algorithm performance with different selection strategies in solving TSP, *in* 'Proceedings of the world congress on engineering', Vol. 2, International Association of Engineers Hong Kong, China, pp. 1–6. Cited in section(s): 2.3.16

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X. and Wang, X. (2021), 'A comprehensive survey of neural architecture search: Challenges and solutions', *ACM Computing Surveys (CSUR)* **54**(4), 1–34. ACM New York, NY, USA. Cited in section(s): 2.3.16, 2.5.3

Richman, R., von Rummell, N. and Wuthrich, M. V. (2019), 'Believing the Bot - Model Risk in the Era of Deep Learning', *Available at SSRN 3444833* . Cited in section(s): 1.5.18, 2.5.1

Richman, R. and Wuthrich, M. V. (2019), 'Lee and Carter go machine learning: recurrent neural networks', *Available at SSRN 3441030* . Cited in section(s): 2.4.3

Richman, R. and Wüthrich, M. V. (2021), 'A neural network extension of the Lee–Carter model to multiple populations', *Annals of Actuarial Science* **15**(2), 346–366. Cambridge University Press. Cited in section(s): 2.4.3

Richman, R. and Wüthrich, M. V. (2022), 'LocalGLMnet: Interpretable deep learning for tabular data', *Scandinavian Actuarial Journal* pp. 1–25. Taylor & Francis. Cited in section(s): 2.4.4

Robbins, H. and Monro, S. (1951), 'A Stochastic Approximation Method', *The Annals of Mathematical Statistics* **22**(3), 400–407. Cited in section(s): 2.3.5

Rosenblatt, F. (1958), *The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain*, Cornell Aeronautical Laboratory. Cited in section(s): 2.3.5

Ruder, S. (2016), 'An overview of Gradient Descent Optimization algorithms', *arXiv preprint arXiv:1609.04747* . Cited in section(s): 2.3.4, 2.3.12

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* **323**, 533–536. Cited in section(s): 1.5.2

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Li, F.-F. (2015), 'Imagenet large scale visual recognition challenge', *International Journal of Computer Vision* **115**(3), 211–252. Cited in section(s): 2.3.13

Sacks, J., Schiller, S. B. and Welch, W. J. (1989), 'Designs for computer experiments', *Technometrics* **31**(1), 41–47. Cited in section(s): 1.8, 2.3.2

Sakketou, F. and Ampazis, N. (2019), On the invariance of the SELU activation function on algorithm and hyperparameter selection in neural network recommenders, *in* 'Artificial Intelligence Applications and Innovations: 15th IFIP WG 12.5 International Conference, AIAI 2019, Hersonissos, Crete, Greece, May 24–26, 2019, Proceedings 15', Springer, pp. 673–685. Cited in section(s): C.2.15

Saleh, R. A., Saleh, A. et al. (2022), 'Statistical properties of the log-cosh loss function used in machine learning', *arXiv preprint arXiv:2208.04564* . Cited in section(s): C.3.8

Sampson, J. R. (1976), 'Adaptation in natural and artificial systems (John H. Holland)'. Society for Industrial and Applied Mathematics. Cited in section(s): 2.3.16

Santner, T. J., Williams, B. J. and Notz, W. I. (2003), *The Design and Analysis of Computer Experiments*, Springer. Cited in section(s): 2.3.2, 2.3.2

Schelldorfer, J. and Wuthrich, M. V. (2019), 'Nesting classical actuarial models into neural networks', *Available at SSRN 3320525* . Cited in section(s): 2.4.2

Shaham, U., Cloninger, A. and Coifman, R. R. (2018), 'Provable approximation properties for deep neural networks', *Applied and Computational Harmonic Analysis* **44**(3), 537–557. Elsevier. Cited in section(s): 1.7.6

Shapiro, A. F. (2002), 'The merging of neural networks, fuzzy logic, and genetic algorithms', *Insurance: Mathematics and Economics* **31**(1), 115–131. Elsevier. Cited in section(s): 2.4

Shrestha, A. and Mahmood, A. (2019), 'Review of deep learning algorithms and architectures', *IEEE access* **7**, 53040–53065. IEEE. Cited in section(s): 2.3

Shwartz-Ziv, R. and Armon, A. (2022), 'Tabular data: Deep learning is not all you need', *Information Fusion* **81**, 84–90. Cited in section(s): 1.5.2, 1.7.6

Slud, E. V. (2012), *Actuarial mathematics and life-table statistics*, Hapman & Hall/CRC. Cited in section(s): 2.2.1

Smolensky, P. (1986), 'Information processing in dynamical systems: Foundations of harmony theory', *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* **1**, 194–281. Cited in section(s): 1.5.2

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research* **15**(1), 1929–1958. JMLR. org. Cited in section(s): 2.3.9

Sutskever, I., Martens, J., Dahl, G. and Hinton, G. (2013), On the importance of initialization and momentum in deep learning, *in* 'International conference on machine learning', PMLR, pp. 1139–1147. Cited in section(s): 2.3.8, 2.3.11, C.1.6, C.2.7

Taylor, G. (2019), 'Loss reserving models: Granular and machine learning forms', *Risks* **7**(3), 82. MDPI. Cited in section(s): 2.4.2

Telgarsky, M. (2015), 'Representation benefits of Deep Feedforward Networks', *arXiv preprint arXiv:1509.08101* . Cited in section(s): 2.3.7

The Faculty of Actuaries and The Institute of Actuaries (2002), *Formulae and Tables for Examinations*, Cambridge. Cited in section(s): 2.2.1

Tucker, M. (2018), Applying High Performance Computing to profitability and solvency calculations for life assurance contracts, PhD thesis, The University of Edinburgh. Cited in section(s): 1.4.5

Tucker, M. and Bull, J. M. (2014), 'An efficient algorithm for the calculation of reserves for non-unit linked life policies', *Algorithmic Finance* **3**(3-4), 143–161. IOS Press. Cited in section(s): 2.2.5

Van Rossum, G. and Drake, F. L. (2009), *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA. Cited in section(s): 3.2

Williams, C. K. and Rasmussen, C. E. (2006), *Gaussian processes for machine learning*, MIT press Cambridge, MA. Cited in section(s): 2.3.2, 2.3.2

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N. and Recht, B. (2017), 'The marginal value of adaptive gradient methods in machine learning', *Advances in neural information processing systems* **30**. Cited in section(s): 2.3.8, C.1.4

Wright, S. (1921), 'Correlation and causation', *Journal of agricultural research* **20**(7), 557. Cited in section(s): 1.5.18

Wu, Y. and He, K. (2018), Group normalization, *in* 'Proceedings of the European conference on computer vision (ECCV)', pp. 3–19. Cited in section(s): 2.3.6

Wüthrich, M. V. (2020), 'Bias regularization in neural network models for general insurance pricing', *European Actuarial Journal* **10**(1), 179–202. Springer. Cited in section(s): 1.5.9, 2.4.2

Wüthrich, M. V., Bühlmann, H., Furrer, H. et al. (2016), *Market-consistent actuarial valuation*, Vol. 3, Springer. Cited in section(s): 2.2.5, 2.4.2

Wuthrich, M. V. and Buser, C. (2021), 'Data analytics for non-life insurance pricing', *Swiss Finance Institute Research Paper* pp. 16–68. Cited in section(s): 2.3.7

Wüthrich, M. V. and Merz, M. (2013), Financial modeling, actuarial valuation and solvency in insurance, Technical report, Springer. Cited in section(s): 2.2

Wüthrich, M. V. and Merz, M. (2022), Selected topics in deep learning, *in* 'Statistical Foundations of Actuarial Learning and its Applications', Springer, pp. 453–535. Cited in section(s): 2.4

Xu, B., Wang, N., Chen, T. and Li, M. (2015), 'Empirical evaluation of rectified activations in convolutional network', *arXiv preprint arXiv:1505.00853* . Cited in section(s): C.2.8, C.2.12

Yang, Y., Youyou, W. and Uzzi, B. (2020), 'Estimating the deep replicability of scientific findings using human and artificial intelligence', *Proceedings of the National Academy of Sciences* **117**(20), 10762–10768. National Acad Sciences. Cited in section(s): 2.5.1

Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O. (2021), 'Understanding deep learning (still) requires rethinking generalization', *Communications of the ACM* **64**(3), 107–115. ACM New York, NY, USA. Cited in section(s): 2.3.15

Zhou, X.-Y., Sun, J., Ye, N., Lan, X., Luo, Q., Lai, B.-L., Esperanca, P., Yang, G.-Z. and Li, Z. (2020), 'Batch group normalization', *arXiv preprint arXiv:2012.02782* . Cited in section(s): 2.3.6