Probabilistic Forecasting with Neural Networks Applied to Loss Reserving

School of Risk and Actuarial Studies
UNSW Business School
University of New South Wales

Muhammed Taher Al-Mudafer

Under the supervision of:

Prof. Benjamin Avanzi

Prof. Greg Taylor

Prof. Bernard Wong

November 23, 2020

A thesis submitted in partial fulfilment of
the requirements for the degree of
Bachelor of Actuarial Studies (Honours)

DECLARATION

I hereby declare that this thesis submission is my own work and, to the best of my knowledge, contains no materials previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and concept or in style, presentation and linguistic expression is acknowledged.

Signed:

Date: 23/11/2020

ABSTRACT

The application of Neural Networks to Loss Reserving has seen rising popularity over the last two years. They have showcased their versatility, flexibility and accuracy. However, there has been little focus on distributional forecasting with Neural Networks on Loss Triangles. Accurately quantifying the volatility of Outstanding Claims (OSC) is essential for efficient capital allocation, liability reporting, and pricing.

This thesis applies the Mixture Density Network (MDN), a Network which focuses on probabilistic forecasting by fitting a Mixed Gaussian to Incremental Claims data. The MDN achieves outstanding results, outperforming the ccODP in central estimate, distributional and quantile estimate accuracy. This model was applied to a wide range of datasets of varying complexity and specifications, excelling in all environments.

A challenge with using Loss Triangles is the little data available to partition into training, validation and testing sets. In a machine learning framework, this thesis performed sequential data partitioning of the Loss Triangle, using the Rolling Origin Method. This partition allowed different Network hyper-parameters to be assessed based on their projection accuracy, allowing for the implementation of a model selection algorithm that provided accurate and stable OSC forecasts.

The black-box nature of Neural Networks causes a risk of inaccurate projections and a lack of justifiability of results. A GLM - MDN hybrid is developed to address this issue, the ResMDN, which is an adaptation of the Combined Actuarial Neural Network architecture prevalent in the Actuarial Neural Network literature. The ResMDN successfully boosted the ccODP's projections by correcting its structural errors, however, it under-performed the MDN.

ACKNOWLEDGEMENTS

I would firstly like to extend my sincere gratitude to my supervisors, Professor Benjamin Avanzi, Professor Greg Taylor and Professor Bernard Wong, for their invaluable guidance throughout the year. This thesis would not be possible at all without their input and support throughout the year.

I would like to thank Professor Hazel Bateman for running the Research Classes, and for her continuous effort in helping us research students succeed. I would also like to thank the Academics at the School of Risk and Actuarial Studies for their advice and support, in particular, Senior Lecturer Andrés Villegas, Associate Professor Anthony Asher, Associate Professor Jonathan Ziveyi, Dr Yang Shen, Senior Lecturer Fei Huang, Alan Xian, Guillaume Boglioni Beaulieu and Michelle Vhudjizena.

I would like to thank the Risk and Actuarial Sctudies Administration team for their support in enrolment, data access and managing campus study, in particular, Josette Milord, Virginia Hine and Helen Karakontis. I would also like to thank my fellow Honours Students, Bofang Tan, Mark Lui, Yuhao Liu and Zhen Dong Chen, for their support this year.

Finally, my deepest thanks go to my parents, Mustafa and Soha, for their unconditional love and support. Nothing reaches fruition without their blessings.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Often for insurers, claims can be reported and settled long after they have been incurred for the policyholder. Hence, in many insurance industries, the claim losses incurred by the company for a given period does not reflect the actual total of claims incurred, as there are some claims that have been incurred but yet to be reported, called IBNR claims, and claims which have been incurred and reported to the insurer, but not fully settled, called RBNS claims (Taylor, 2000). The sum of IBNR and RBNS claims make up the Outstanding Claims (OSC) for an insurer.

1.1.1 The Need to Model Outstanding Claims

Modelling and reserving for OSC are issues central to actuarial science, for many reasons. Firstly, insurers are required to hold reserves that meet Outstanding Claims when they come due. OSC often forms a large portion of an insurer's liabilities (Avanzi et al., 2016), hence failure to hold sufficient economic capital to meet these claims can cause liquidity issues. Due to the significance of this risk, modelling and holding sufficient reserves to meet OSC liabilities is subject to regulation. In Australia, APRA requires General Insurers to hold capital to meet OSC with a 75% probability of sufficiency. The Solvency II capital test requires an OSC reserve allocation that has a 99.5% probability of sufficiency (Radtke

et al., 2016). Hence, not only is it important to find a best central estimate, but modelling the distribution of Outstanding Claims will help gauge the volatility of these liabilities, allowing companies to effectively allocate capital to meet their risk appetite and satisfy regulatory requirements.

Secondly, Outstanding Claims are recorded as a liability on the balance sheet, hence an accurate forecast of these claims will better indicate the profit and loss faced by the insurer, as well as its solvency (Rossouw and Richman, 2019). Thirdly, modelling OSC is important for pricing, as premiums must be set to exceed claims according to a profit margin, meaning that the quantification of claims which have been incurred but not fully settled is essential in understanding the true claim dynamic.

In addition to forecasting the central estimate of Outstanding Claims, modelling the distribution is also important in liability reporting and pricing. To exercise prudence, liabilities are often reported on the Balance sheet with a risk margin on top of the central estimate. Quantifying the volatility of OSC will allow for these margins to be set more accurately. Similarly, understanding the distribution of claims will assist in setting appropriate profit margins when pricing and understanding pricing risk.

1.1.2 Traditional Reserving Methods

Many statistical models have been used today and in the past to model Outstanding Claims (Taylor, 2019). Most models, even today, have focused on the aggregate Loss Triangle, listing claims in a grid by accident and development period. Earlier models, such as the Chain Ladder, introduced in the 1960s, used a simplistic algorithm to provide a central estimate. With the lack of stochasticity provided by the Chain Ladder model an issue, a stochastic variant, the Mack Model, was introduced, helping to quantify the uncertainty of OSC. Other relatively simplistic models used in that time period include the Bornhuetter-Ferguson and Cape Cod models.

As computing power improved, GLMs became common in actuarial literature, starting in the 1990s. GLMs brought more flexibility in covariate, claim distribution and link function selection, which allowed for more accurate central estimate and probabilistic forecasting. Furthermore, interactions and external variables such as inflation could be incorporated into the model, improving its predictive accuracy.

1.1.3 Machine Learning Models - Neural Networks

With the continual improvement in computing technology and power, more powerful and complicated machine learning models (MLMs) have found recent applications in Loss Reserving, such as Regression Trees (Wüthrich, 2018a) and Neural Networks Kuo (2019). Out of these newly applied MLMs, Neural Networks (NNs) have found the most success, both in Loss Triangle and individual claims reserving (Richman, 2018). Since 2018, the application of Neural Networks to Loss Reserving has been gaining momentum. These new models have shown enormous potential, modelling complex interactions between variables, allowing them to model OSC with an accuracy that outperforms basic traditional models. The current Loss Reserving literature applies NNs in a variety of ways. Gabrielli et al. (2020) used a Residual Neural Network to boost a GLM model, Kuo (2019) and Poon (2019) jointly forecast the claims counts and amounts, which improved accuracy due to the interactions captured by these two variables. Wüthrich (2018b) used a Neural Network to estimate Chain Ladder development factors, modelling dependence between variables such as Line of Business, Labor Sector, age, etc. Delong et al. (2020) used six different Neural Network to model the micro-processes of Individual Claims. These various methods of applications have showcased the Neural Networks versatility and flexibility in design and modelling.

Despite showing promise, several gaps have become apparent in Neural Networks and their current implementation in the Loss Reserving literature. Firstly, there is little focus on distributional forecasting of claims, instead primarily focusing on central estimates. Secondly, Neural Network models are considered a black box due to their complexity, hence are much less interpretable than traditional GLMs. As a result of their lack of interpretability, NNs face a higher risk of unstable extrapolation (Rossouw and Richman, 2019). This weakness is exacerbated under volatile conditions, as interactions may not continue into the projection period (Rossouw and Richman, 2019). Thirdly, the Neural Network Loss Reserving literature doesn't explicitly outline a methodology for testing different models within the Upper Triangle. This risks fitting a Neural Network model with an un-verified design, leading to a sub-optimal fit.

1.2 Motivation

The recent successful applications of Neural Networks in Loss Reserving literature makes an exciting development in Outstanding Claims modelling. As mentioned earlier, NNs have demonstrated their accuracy and flexibility in modelling, showing an ability to capture

complex interactions between variables and accommodate large granular datasets. A downside of using GLMs is their user-intensive nature (McGuire et al., 2018), as it takes time to find the right distribution, link function and covariate structure. In addition to their potential to produce more accurate forecasts, Neural Networks can potentially save modelling time through their flexible structure and optimisation algorithms which automatically search for the minimum loss in a wide function space.

However, the practical and technical obstacles mentioned earlier have hindered the acceptance and use of Neural Networks by the actuarial community (Richman, 2018; Wüthrich and Merz, 2019). Analysing and fixing these shortcomings and expanding the implementation of Neural Networks in the literature will increase their reliability and practicality. This will encourage the NNs modelling power to be applied in practical Loss Reserving, adding more models available to the actuary and improving the reserving process.

This thesis will primarily focus on distributional forecasting in a **Loss Triangle** setting. In the current Actuarial literature, NNs are predominantly being used to provide central estimates of Outstanding Claims. Providing distributional estimates of OSC is important for allocating prudential economic and regulatory capital, setting appropriate risk margins when valuing liabilities on the Balance Sheet, setting appropriate profit margins and understanding pricing risk. Just as NNs have shown promise in providing central estimates of Outstanding Claims, focusing their modelling power on accurate probabilistic forecasting will potentially yield 'state of the art' results in that regard (Richman, 2018). Apart from Kuo (2020), no paper to my knowledge has focused on attaining probabilistic forecasts. Despite Delong et al. (2020) producing a stochastic individual model, the focus was on attaining central estimates. However, both these articles used Individual Claims data, hence there has been no focus on Loss Triangle distributional forecasting. As insurers may still be restricted from accessing reliable Individual Claim records, developing a NN model that applies to Loss Triangles is highly relevant.

Furthermore, NNs are considered a black box due to their lack of interpretability (Gabrielli, 2019). The model fits a highly complex function to the data, hence the main factors driving a forecast, while accurate, cannot be analysed as closely as traditional GLM models. As modelling OSC is important for reserving, analysing solvency and pricing, decisions in these actuarial areas should be well justified to company stakeholders, such as the board and consumers. The black box nature of NNs have slowed down their acceptance in the Actuarial field. To increase their acceptance, Wüthrich and Merz (2019) developed the Combined Actuarial Neural Network (CANN), which embedded a GLM into the Neural

Network architecture. This hybrid model has been applied successfully by Gabrielli et al. (2020); Poon (2019) and Gabrielli (2019), showcasing the improved interpretability and stability of Neural Networks. The CANN architecture acts as a bridge between GLMs and Machine Learning models, hence it is a valuable modelling framework which will be considered in this thesis.

Due to the NNs lack of interpretability, they are at risk of unstable extrapolation (Rossouw and Richman, 2019). This issue is important in reserving, since the key objective is to estimate future claims in the Lower Triangle given the Upper Triangle of claims data. The function fit by the NN in the Upper Triangle is highly complex and as a result, the function's behaviour in the Lower Triangle may be unreasonable.

Adding to the volatility in Neural Network fitting, no model testing framework has been outlined in the Neural Network Loss Reserving literature, which focuses on a Loss Triangle setting. Neural Networks are flexible in the hyper-parameter values they can take, hence it is vital to have a clear methodology for testing between these different models. A lack of methodology can lead to poor hyper-parameters being set for the Network, causing a sub-optimal fit and jeopardising the rising credibility NNs have recently built.

Combining the need for stable projections and a model testing framework, it is important to develop a partition of the Loss Triangle into training, validation and testing sets, which assesses models based on their projection accuracy. Such a focus will increase the probability of fitting Networks with hyper-parameters that produce stable forecasts, which is essential in Loss Reserving. This triangle partition must be accompanied with a systematic methodology for testing and selecting Network hyper-parameters. Such a methodology will make Neural Network fitting more methodical, assisting in popularising their implementation in practice.

1.3 Research Aims and Contributions

This thesis addresses a few of the gaps present in the implementation of Neural Networks in the Loss Reserving literature. The aims and contributions of this thesis are:

1. Applying probabilistic forecasting with Neural Network on Loss Triangle Reserving. This is done using Mixture Density Networks, a Neural Network which successfully focused its modelling power on produce accurate distributional forecasts of Loss Reserves.

2. Maintain the 'state of the art' results seen in the Neural Network Loss Reserving literature by ensuring that, despite the focus on probabilistic forecasts, the model also produces accurate central estimates of Outstanding Claims.

3. Develop a Machine Learning model selection framework by partitioning the Loss Triangle into training and testing sets in a way that suits the time-dependence structure of the Loss Triangle, allowing the framework to test and select model designs based on their projection accuracy.

4. Assist the implementation of Neural Networks in practice by adopting a more interpretable and justifiable model, the ResMDN, while maintaining the objectives outlines in points 1-3.

1.4 Outline of Proposal

After the Introduction in Chapter 1, Chapter 2 will explore the relevant literature in Loss Reserving, looking at current Neural Network applications, as well as literature on probabilistic forecasting with Neural Networks and time series model validation methodologies. With the knowledge obtained from the various fields in the literature, Chapter 3 outlines the modelling frameworks used to achieve the Research Aims outlines. Chapter 4 analyses the data used in this thesis, while Chapter 5 goes through the results obtained from the modelling, and their implications. Chapter 6 concludes by summarising the results, contributions, limitations and avenues for future research.

CHAPTER 2

LITERATURE REVIEW

This thesis focuses on probabilistic forecasting of Outstanding Claims using Neural Networks. This chapter explores the literature surrounding Neural Network in Loss Reserving and other fields. Section 2.1 will provide a brief analysis of the traditional Loss Reserving models, with Section 2.2 introducing the Neural Network. Section 2.3 explores in depth the current Neural Network applications in Loss Reserving, with a subsequent review of the strengths and gaps of these applications in Section 2.4. Building on the gaps in the Neural Network Loss Reserving literature, Section 2.5 will explore Neural Network models that perform probabilistic forecasting, while Section 2.6 concludes the Literature Review by analysing Model Validation methodologies specialised for time-series data.

2.1 Traditional Loss Reserving Models

Traditional claims reserving techniques have focused on using aggregate models, mainly the Loss Triangle (Taylor, 2019). Loss Triangles, visualised in Figure 2.1, record aggregate transactions on a specified accident and development period, and consists of a triangular grid of cells, with the accident period on the vertical axis and development period on the horizontal axis. A cell of coordinates (i, j) records aggregate claims data, usually claims paid or claims reported, for accident period i and development period j. For future analysis of triangles, we will assume a total of I accident periods and J development

periods, with $I \geq J$. Common period scales used are monthly, quarterly and annually.

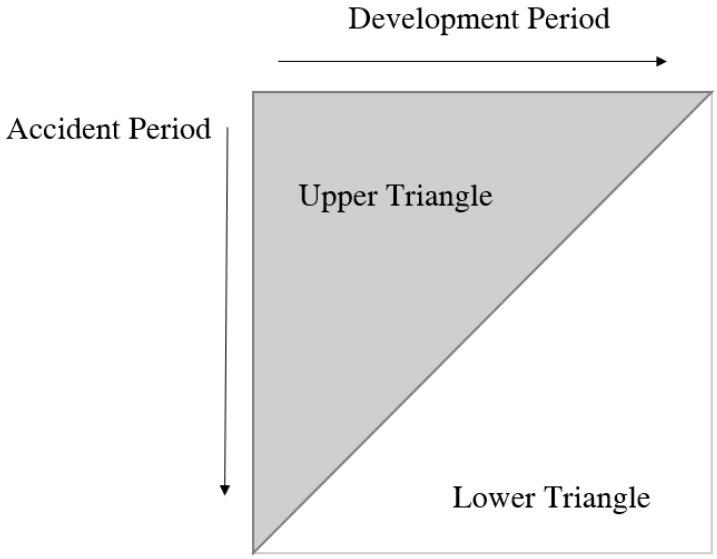


Figure 2.1: The Aggregate Loss Triangle used in modelling. Paid Claims are assorted in the grid, based on their Accident and Development periods. Given the upper triangle, the modelling objective is to accurately predict future Outstanding Claims, which make up the lower triangle

Popular models such as the Chain Ladder were computationally inexpensive and provided reasonable results, especially for short tail claims (Baudry and Robert, 2019). Technological improvements allowed more computationally expensive models, such as GLMs to arise. As Machine learning has increased in popularity, so too has Individual Claims modelling (Delong et al., 2020). Individual Claims modelling forecasts future claim payments by simulating micro-processes such as reporting delay, claim status, probability of payment, payment size, recovery payments, etc. These processes are modelled accounting for individual claim features such as the Line of Business (LoB), labor sector, age of injured, injured body part, etc.

2.1.1 Chain Ladder Model

The chain ladder model, applicable to Loss Triangles, is one of the earliest methods used in loss reserving, its use estimated to originate around the 1960s, or even earlier (Taylor, 2019). Let $C_{i,j}$ denote the cumulative payments for accidents that occurred in Accident period i and development period less than or equal to j. The key model assumptions, as listed by Wüthrich and Merz (2008), are as follows:

1. Claims across different accident periods are independent

2. For each accident and development period i and j, the cumulative paid claims for this category follows this formula:

$$E[C_{i,j}|C_{i,j-1}] = f_{j-1}C_{i,j-1},$$

where $f_{j-1} = \frac{\sum_{i=1}^{I-j+1} C_{i,j}}{\sum_{i=1}^{I-j+1} C_{i,j-1}}$

The intuition behind this expression of f_j is explained in Taylor and McGuire (2016), which represent the above formula as a weighted average of claim developments, for development period j. Let us, for the calculation of f_j, take $f_{i,j}$ as :

$$f_{i,j} = C_{i,j}/C_{i,j-1}$$

Then f_j is a volume-weighted sum of $f_{i,j}$, as such:

$$w_{i,j} = \frac{C_{i,j-1}}{\sum_{i=1}^{I-j} C_{i,j-1}},$$

$$\text{with } f_j = \sum_{i=1}^{I-j} w_{i,j} f_{i,j}$$

Today, the Chain Ladder model's simplicity and ability to often provide reasonably accurate results has seen it remain popular in practice (Wüthrich and Merz, 2008). The chain ladder will perform reasonably well with short tail claims, but struggle with long tail claims due to lacking data in later development periods (Baudry and Robert, 2019). Several downsides of the model firstly involve its inability to model diagonal effects, such as variable inflation. Changes in claims processing technology and legislation can change claim development patterns among accident years, which the Chain Ladder model will not capture. Secondly, the basic Chain Ladder algorithm described above isn't a stochastic model, providing only central estimates of future outstanding claims. However, several stochastic variants have been developed such as the Mack Model (Mack, 1993) .

2.1.2 Generalised Linear Models

Generalised Linear Models (GLMs) became popular in actuarial modelling in 1990, accommodated by the increase in computational power (Taylor, 2019). In application, they provide higher forecasting accuracy and model flexibility than traditional Chain Ladder

models. Applied on Loss Triangles, the GLM assumes incremental payments, which we will denote as $X_{i,j}$ for accident period i and development period j, follow an exponential dispersion family distribution, structured as such (Taylor and McGuire, 2016):

$$f_{X_{i,j}}(y) = exp\left[\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right]$$

Where θ is the location parameter, ϕ is the dispersion parameter and b is the cumulant function. Common distributions, such as Poisson, Gaussian, Log-Normal and Gamma fall under this family of distributions. A link function h of a GLM dictates the relationship between the mean of the distribution and the controlled covariates, as such:

$$\begin{aligned} E[X_{i,j}] &= b'(\theta) \\ &= h^{-1}\big(\sum_{k=1}^{p} x_{i,j,k}\beta_k\big) \end{aligned}$$

Where $x_{i,j,k}$ are covariates such as accident period, development period, inflation, operational time and interactions. Common link functions include the log-link ($h(x) = log(x)$), identity ($h(x) = x$) and reciprocal ($h(x) = 1/x$) functions. Common distributions used for Loss Triangle reserving are Poisson and Gamma (Zhou and Garrido, 2009). For example, we consider the cross-classified Over-Dispersed Poisson (ccODP) Model (England and Verrall, 2002). Each entry in the triangle is assumed to follow the ODP distribution as such:

$$\frac{X_{i,j}}{\phi} \sim Poi\big(\frac{A_i B_j}{\phi}\big),$$

where A_i and B_j are accident period and development period effects respectively. Maximum likelihood estimation is used to estimate the GLM parameters.

GLMs offer flexibility in modelling by allowing a wide range of distributions and link functions to fit the data. The model allows categorical and numerical covariates to be included, which can model externalities such as inflation or interactions. Where the Chain Ladder will fail due to factors such varying inflation, seasonality or a change in operational time, GLMs provide the ability to add covariates to allow for these factors, improving the fit (Taylor and McGuire, 2004). Fitting a distribution allows forecast error to be quantified by analysing the process and parameter error resulting from the stochasticity of the fitted distribution.

A downside of GLMs is their user intensiveness (McGuire et al., 2018). It can take many hours to find the right combination of covariates, distribution and interactions.

2.2 Neural Networks

With the further improvements in computing power, more complicated machine learning models (MLMs) have seen applications to loss reserving in recent years. These models have shown promising forecasting accuracy, an ability to deal with large complex data and model non-linear interactions between variables, which basic Chain Ladder and GLMs couldn't model. The most popular machine learning model in actuarial Loss Reserving literature is the Neural Network.

Neural networks (NNs) are models which attempt to simulate the human brain structure. Richman (2018) mentions early interest in simulating biological learning in the 1940s. Notable breakthroughs in model design came with Rosenblatt (1958) designing the artificial neuron and Rumelhart et al. (1986) developing the back-propagation algorithm which allowed more practical training of networks. The popularity of NNs was revived in 2006, and are now used in regression, image classification and speech recognition, just to name a few (LeCun et al., 2015).

Most NN applications to Loss Reserving have yielded high forecasting accuracy, better than or equal to basic Chain Ladder, GLM and other ML models (Kuo, 2019; Rossouw and Richman, 2019; Gabrielli et al., 2020). Many design variations of NNs exist, each with its own specialisation. We shall define a shorthand notation of representing the input/output of a Network. Let us say that a Network which takes input x and predicts the response y can be written in the shorthand:

$$x \mapsto y$$

2.2.1 Feedforward Neural Networks

Richman (2018) provides a thorough introduction to a standard Neural Network model, the feedforward NN. Figure 2.2 provides a visualisation of their basic structure.

They are composed of the input, hidden and output layers. Inside each layer is an array of neurons, with each neuron in a hidden layer being connected to each neuron in the preceding and subsequent layers by weighted synapses. As Figure 2.3 illustrates, each neurons takes a weighted sum of the previous output and passes it through an activation function f, before transferring that output to the next hidden layer.

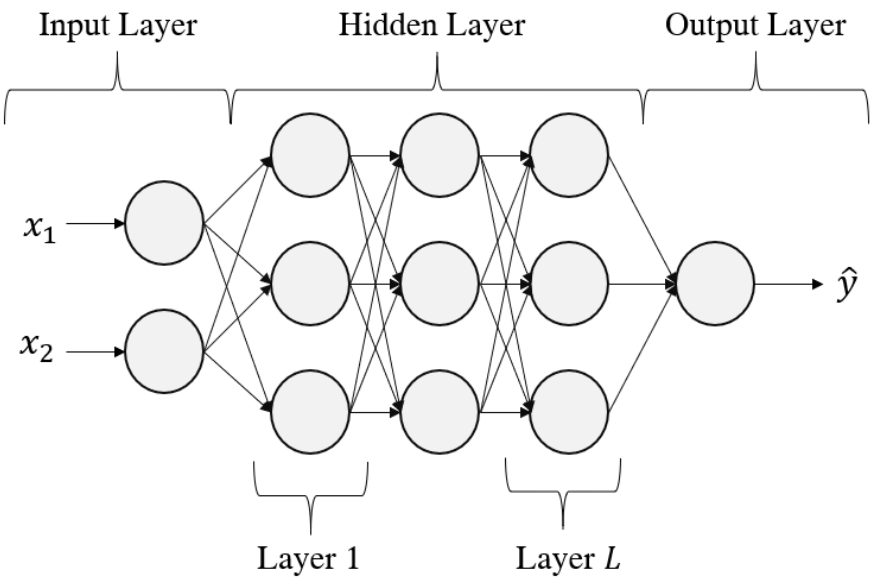


Figure 2.2: Structure of a Feedforward Neural Networks, composed of the Input, Hidden and Output layers.

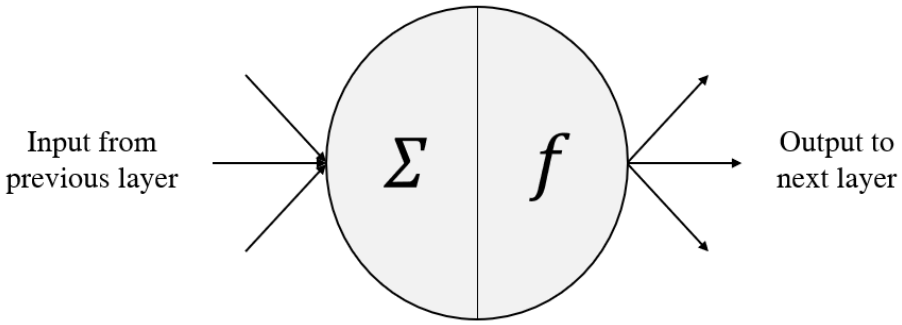


Figure 2.3: Structure of Neuron. A weighted sum of the previous layer's input is passed through an activation function, then passed to the next layer

Suppose there are n data points, labelled x_i for $i = 1, 2, 3..., n$, with each point containing p covariates, labelled $x_{i,j}$, for $j \in \{1, p\}$. Suppose the network has L hidden layers, with d neurons in each layer. During a forward propagation of the network, the input layer will consist of p neurons, with the output of each input neurons being a covariate of a data point, $x_{i,j}$, for $j \in \{1, p\}$. The first hidden layer takes a weighted sum of neurons in the input layer and passes it through an activation function f as shown below. Let $h^l_{k,i}$ denote the output of the kth neuron in the lth hidden layer when data point i is the input, and f_l be the activation function for layer l. $h^1_{k,i}$, the output of nodes in the first hidden layer, are calculated as such:

$$h^1_{k,i} = f_1\left(\sum_{j=1}^{p} w_{j,k,1}x_{i,j} + b^0_k\right)$$

Where b^0_k is a bias term added to each neuron in each hidden and output layer. As

linear transformations are made through the weighted summation, the activation functions should be non-linear to capture non-linearities in the data. Given L hidden layers, each hidden layer will similarly take a weighted sum of output from the previous layer and pass it through an activation function, producing the following output:

$$h_{k,i}^{l} = f_l \left(\sum_{j=1}^{d} w_{j,k,l} h_{i,j}^{l-1} + b_k^{l-1} \right)$$

The output layer takes a weighted sum of the Lth hidden layer. Suppose the target response is m dimensional, denoted by $\hat{y}_i$ for $i \in \{1, m\}$. The output is produced as such:

$$\hat{y}_i = f_L \left(\sum_{j=1}^{d} w_{j,k,L} h_{i,j}^{L} + b_k^{L} \right)$$

Different network configurations involving the choice of hidden layers, neurons in each layer and activation functions can yield a different fit. Below are some common activation functions:

- Sigmoid: $f(z) = \frac{e^z}{1+e^z}$
- Tanh: $f(z) = \frac{e^{2z}-1}{e^{2z}+1}$
- Rectified Linear Unit (ReLU): $f(z) = max(0, z)$

The NN model parameters are the weights carrying the output from each layer to the next. The weights are initialised arbitrarily, as an optimisation algorithm is used to adjust the weights iteratively to improve the goodness-of-fit. Stochastic gradient descent (SGD) is a commonly used optimisation algorithm. The fit is measured based on a pre-defined loss function, applied on the output layer and real responses. For producing central estimates in a regression setting, the Mean Squared Error is a common loss function:

$$MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

2.2.2 Training the Network

Fitting the model involves splitting the available data into training, validation and testing sets. A training/testing split of 90% to 10% is common, demonstrated by Noll et al. (2020). A training/validation split of 90% to 10% is also common, demonstrated by Wüthrich (2018b). Given the number of parameters in even a simple Neural Network, running the

optimisation algorithm for too many iterations can overfit the model to the data, reducing predictive power (Gabrielli et al., 2020). Under the above-mentioned data split, the model is fit on the training data. At each iteration, the loss value is calculated using output that was produced by running the validation data through the updated model, called the validation loss.

During most training scenarios, the training loss; the loss value calculated using output generated by running the training data through the model, will steadily decrease, which is to be expected as gradient descent continues to search for a minimum (Gabrielli, 2019). The validation loss will initially decrease after each iteration, then steadily increase. Given that the validation data isn't used in training, it is a good indicator of the model's performance on unseen data. Hence, the steady rise of validation loss indicates over-fitting, as the network has been optimised too closely to the training data. The number of iterations used to train the model is then chosen to minimise the validation error.

After optimising the validation error, the model is run on the testing data, with the fit evaluated using a suitable goodness-of-fit measure, usually the MSE or cross-entropy. The network design is adjusted by changing the Network's hyper-parameters, such as the number of hidden layers, number of neurons, regularisation penalties, etc. The hyper-parameters producing the best fit on the testing data are chosen to construct the final model.

Neural Networks are flexible in their architecture and the data they can process; being able to process numerical data, categorical data, images, text, time series (Poon, 2019) and more. The feedforward network is a basic model relative to other complex network architectures in use, such as LSTMs (Kuo, 2020), Convolutional and Encoder-Decoder networks.

2.3 Neural Network Applications to Loss Reserving

The application of Neural Networks to Loss Reserving has been recent, with Mulquiney (2006) being the earliest as of my knowledge. Their application has been focused on yielding point estimates or boosting existent GLM models (Gabrielli et al., 2020). To my knowledge, only Kuo (2020) has focused on providing non GLM based probabilistic forecasting of losses. NNs have been applied to both aggregate Loss Triangles and Individual Claims, with most documenting the superiority of their fit over basic traditional Loss Reserving models.

2.3.1 Parametric Models

Assuming aggregate claims follow a user-specified distribution, such as Poisson, Gamma and Gaussian, the Neural Network output layer could be designed to produce estimates of the distributional parameters. The model is then calibrated using a Deviance loss function corresponding to the chosen distribution, such as Poisson Deviance. For example, Wüthrich (2018b) assumes a Mack model (Mack, 1993), and optimises the loss development factors $f_{j,\mathbf{x}}$, where $\mathbf{x}$ is a feature space consisting of factors such as Line of Business (LoB), labor sector, age, etc. The ability to optimise these development factors for each set of variables in a single Network shows the predictive power and flexibility of NNs. A standard NN design uses the MSE metric to directly optimise the claims paid in the output layer. This architecture was applied successfully by Kuo (2019). Rossouw and Richman (2019) worked with granular data and assumed claims follow a Poisson distribution. Gabrielli et al. (2020) assumed the losses follow an ODP distribution, hence their Neural Network estimated the mean claims paid, $\lambda_{i,j}$, as such:

$$(i, j) \mapsto \lambda_{i,j}$$

These applications served to apply traditional GLM frameworks in a Machine learning setting, seeking to boost the GLM by capturing complex interactions within the data. Building on existing model frameworks like the Chain Ladder and GLMs produces more relatable and interpretable models. A downside is that the parametrisation will restrict the Network's output to the assumptions of traditional models and may not fully exploit its ability to model deep interactions between variables.

2.3.2 Big Data

Neural Networks have demonstrated the ability to work with large datasets featuring many covariates, modelling deep interactions and dependencies between them. The network uses an embedding layer to convert categorical variables into a numerical vector. Where claims would be separated by their Line of Business and modelled separately, Neural Networks can efficiently accommodate variables such as Line of Business, age, Labor Sector into the one model, increasing the data available, learning dependencies and improving the fit. This flexibility has been demonstrated by Gabrielli et al. (2020), who learned from different LoBs simultaneously, and Wüthrich (2018b) who used LoB, age, Labor sector and injred body part to learn claim development factors. Both articles reported an improvement in model accuracy from traditional models.

2.3.3 Residual Neural Networks - ResNet

Residual Neural Networks (ResNet) feature skip connections, which are weight parameters that connect non-adjacent hidden layers. The number and location of hidden layers connected by a skip connection is up to the modeller. A hidden layer l connected to layer $l-c$ in such a way will receive output from hidden layer $l-1$ and $l-c$, producing its output as such:

$$h_{k,i}^{l} = f_l\left(\sum_{j=1}^{d} w_{j,k,l} h_{i,j}^{l-1} + b_k^{l-1} + w_{j,k,l-c} h_{i,j}^{l-c} + b_k^{l-c}\right)$$

While a ResNet can improve the convergence of deep networks, it has been applied for boosting GLM models in Loss Reserving. Figure 2.4 provides a visualisation of the basic ResNet implementation in Loss Reserving.

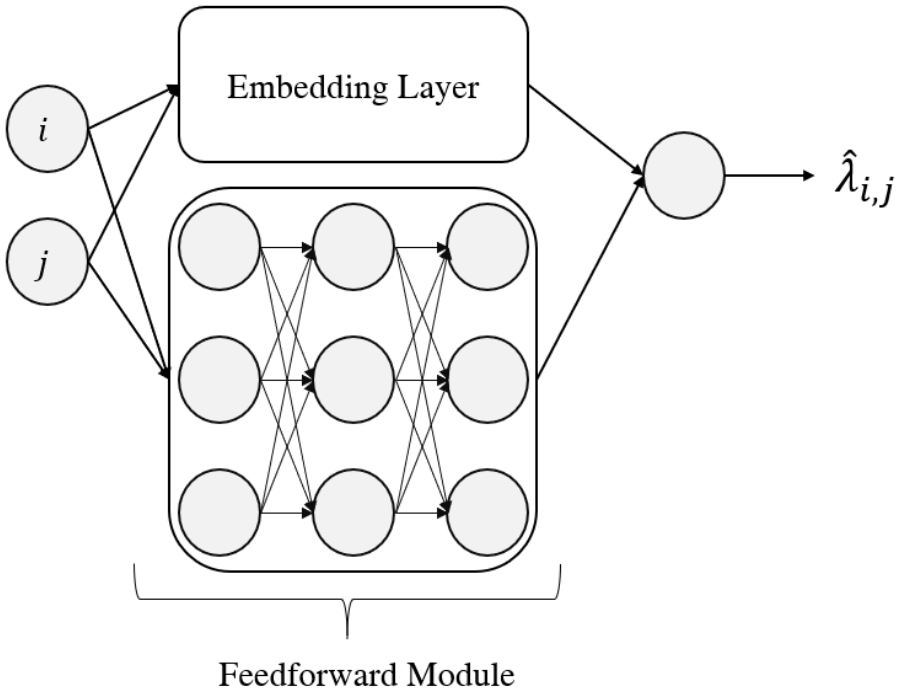


Figure 2.4: ResNet structure as applied by Gabrielli et al. (2020). The embedding layer provides the ccODP GLM parameters, while the feedforward module boosts the GLM by adding extra terms which improve the goodness-of-fit

Assume aggregate claims follow a cross-classified ODP distribution, introduced in Section 2.1.2. Consider a Neural Network with Accident Period i and Development Period j inputs, no hidden layer and a single output estimating the ODP mean parameter $\lambda_{i,j}$, then the network will produce output as such:

$$\hat{\lambda}_{i,j} = f(\hat{w}_1 i + \hat{w}_2 j),$$

which is a GLM. To initialise the Network with the output of a pre-fit GLM, for example, the ccODP model, an embedding layer would convert the Accident Year and Development Year inputs into A_i and B_j, the parameters of the ccODP model, and transfer these factors through a weight with a value of 1 to the output layer containing an exponential activation function, as shown below:

$$\textbf{Embedding Layer: } (i,j) \mapsto (ln(A_i), ln(B_j))$$

$$\textbf{Output Layer: } (ln(A_i), ln(B_j)) \mapsto e^{(ln(A_i)+ln(B_j))} = A_i B_j$$

Hence this neural network model has the ccODP fit as its initial output. Connecting a feedforward module would incorporate a non-linear equation into the output, as such:

$$\gamma_{i,j}^{NN} = \left(\sum_{j=1}^{d} w_{j,k,L} h_{i,j}^{L} + b_k^L \right), \text{ the output of the Feedforward module.}$$

The non-linear term is then added to the GLM fit as a boosting term as such:

$$(i,j) \mapsto e^{(ln(\alpha_i)+ln(\beta_j))+\gamma_{i,j}^{NN}} = \alpha_i \beta_j * e^{\gamma_{i,j}^{NN}}$$

With the model initialising at the ccODP fit, any further training on the feedforward module, represented by $\gamma_{i,j}^{NN}$ will improve the fit, as it captures deep interactions within the variables.

The Residual Network embedding of a GLM model was, to my knowledge, proposed by Wüthrich and Merz (2019), titled the Combined Actuarial Neural Network (CANN). The CANN concept has since achieved widespread popularity in the literature due to its ability to make use of black-box machine learning modelling without significantly compromising the interpretability observed with GLMs (Poon, 2019; Wüthrich, 2019; Schelldorfer and Wüthrich, 2019). This benefit aims to increase the acceptance of Neural Networks in the Actuarial community.

In the Loss Triangle framework, the RNN architecture was applied by Gabrielli et al. (2020), boosting a ccODP model, while Gabrielli (2019) simultaneously learned and boosted claim amount and claim count data. These articles all recorded an improvement in fit, showing

the effectiveness of boosting by using skip connections. Gabrielli et al. (2020) showed a reduction in claims forecasting bias by 10% to 30%.

Another improvement is the increased interpretability of the model due to having a GLM backbone to it. The final model obtained by Poon (2019) had a 71% and 54% weighting on the initialised GLM, allowing the driving factors of the forecast to be traced more effectively than a randomly initialised network. Furthermore, the GLM initialisation produces more stable output than a random initialisation, as was noted by Gabrielli et al. (2020). This is due to the initial fit being reasonable, requiring less iterations to optimise the parameters. The issue of variable output due to random initialisations has been noted by Kuo (2019) and Kuo (2020), which was solved by running the Network 10 times and averaging results.

A downside in the implementation of ResNets is that the GLM distributional assumption remains. The ODP structure used by Gabrielli et al. (2020) computes claim variance as a function of the mean, restricting quantile estimates of claims to the mean and dispersion parameters. Similarly for Poon (2019), the modelling power of NNs was focused on producing central estimates of outstanding claims, rather than probabilistic forecasting.

2.3.3.1 Individual Claims Modelling

Individual Claims Modelling (ICM) has gained popularity since the development of Machine Learning models (Delong et al., 2020), but has been introduced conceptually to the literature by Arjas (1989) and Norberg (1993). It focuses on using Individual Claims data to model the development of claim micro-processes, such as reporting time, payment, claim status, recoveries and settlement. Claims are commonly classified by a static feature space, such as Line of Business, Labour Sector, age, injured body part, etc. This form of modelling allows analysis of claim micro-processes on a more granular level, which should improve forecasting accuracy, however, the superiority of Individual Claims hasn't been demonstrated conclusively in the literature (Taylor, 2019). Understanding claims on an individual level will provide more accurate forecast in datasets where policyholder composition shifts, such as changes in the volume of short-tail/long-tail business within a company (Harej et al., 2017).

Individual Claims modelling has seen growing application and success in the literature (Gabrielli, 2020; Delong et al., 2020; Kuo, 2020). For example, Delong et al. (2020) uses six feedforward neural networks to model elements of claim development: claim status,

recovery payments and paid losses, allowing for dependencies within these variables.

Long Short-Term Memory (LSTM) networks specialise in dealing with sequential input data. They have seen wide usage, including speech recognition (Greff et al., 2016) and stock market modelling (Chen et al., 2015) . Individual claims develop in a sequential manner, hence an LSTM is effective in this setting. Kuo (2020) demonstrated this, modelling losses and recoveries based on all prior claim payment and status history. This methodology contrasts with Delong et al. (2020), who assumed claim status and payments follow the Markov property, thereby only using the previous month's claim information as input for forecasting future Individual claim development.

2.4 Review of Neural Network Applications to Loss Reserving

A growing field of literature have applied Neural Networks to Loss Reserving and yielded more accurate fits than basic GLM or Chain Ladder models. NNs have demonstrated their ability to model complex non-linear interactions within the data and model with numerical and categorical variables. While they work best with large datasets, they have proven effective with Loss Triangles in Gabrielli et al. (2020), although the model's stability was aided by a GLM initialisation.

No article to my knowledge has focused on attaining distributional estimates of future claims using Loss Triangles. Despite Delong et al. (2020) producing a stochastic individual model, the focus was attaining central estimates. Gabrielli et al. (2020) fit a boosted ccODP model, whose variance is restricted as a function of the mean. Furthermore, NNs are considered a black box due to their lack of interpretability. The main factors driving a forecast, while accurate, cannot be analysed as closely as traditional GLM models. Vaughan et al. (2018) proposes the "Explainable Neural Network", which assumes the output to follow a flexible additive model. Gabrielli (2019) and Poon (2019) used the ResNet to provide a GLM backbone, with the NN only boosting the GLM's residuals. Given the success of NNs has come from their complexity, a compromise in that regard may reduce this success. An alternative solution would be to maintain the NN complexity and develop diagnostic tools to better understand the drivers of a Neural Network's output, such as Partial Dependency Plots (PDP) and Individual Conditional Expectation (ICE) plots (Goldstein et al., 2015).

Building on the issue of interpretability, time-based trends are harder to decipher with NNs. Given reserving involves forecasting future claims, there is a risk of the network extrapolating inaccurately into the future (Rossouw and Richman, 2019). The NN Loss Reserving literature does not test the projection accuracy of their Machine Learning models within the Upper Triangle, hence no comprehensive model testing and selection framework has been implemented which works specifically within Neural Networks. In a Loss Triangle setting, only Balona and Richman (2020) partitioned the available data into training and testing sets, however, machine learning models were not tested in this article. It is recommended to perform sequential data splitting to explicitly assess the NNs time-based forecasting ability (Tashman, 2000).

2.5 Probabilistic Forecasting with Neural Networks

Several Neural Network architectures have been used for probabilistic forecasting. These models have seen common use in areas such as hydrology (Kasiviswanathan and Sudheer, 2017), electrical load forecasting (Vossen et al., 2018) and stock market modelling (Ormoneit and Neuneier, 1996). Applying these NN models for loss reserving would harness the predictive power of Neural Network towards probabilistic forecasting of Outstanding claims. Bayesian Neural Networks are the most common method of probabilistic forecasting; however, non-Bayesian ensemble methods have shown promise.

2.5.1 Bayesian Neural Networks

Bayesian NNs are the most commonly applied NN model used for probabilistic forecasting (Kasiviswanathan and Sudheer, 2017) . The model assumes the weight parameters and output to follow a prior distribution $p(w)$ and $p(y|x, w)$, where x, y, w are the input, output and weights respectively. Using Bayes rule, the posterior distribution of the weights is calculated as such:

$$p(w|X) = \frac{p(X|w)p(w)}{p(X)},$$

where X is the dataset provided. Samples of the posterior are taken to yield samples of the output, providing a distribution of output y. Calculating the posterior of the weights is analytically intractable, hence a wide class of posterior approximation algorithms have been developed, such as MCMC, BBB, PBP and variational inferences techniques (Laksh-

minarayanan et al., 2017) . Variational inference has allowed for reasonable and efficient approximations. A simple distribution $q(w)$ is taken as the student distribution, which could take any interpretable form, such as Gaussian. The parameters of q are optimised to be as close as possible to the real posterior p by minimising a divergence statistic, the Kullback Leibler statistic being common.

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(w) log\left(\frac{p(w)}{q(w)}\right) dw$$

BNNs are applicable to any Network design and can provide reasonable predictions with small datasets. However, even with variational inference, they are still computationally expensive (Barber and Bishop, 1998) . The success of BNNs is sensitive to the selection of the parameter prior distribution, and given the expert input needed in hyper-parameter specification, their ability to match non-Bayesian NNs in point forecasting accuracy isn't guaranteed (Jouko and Aki, 2001) .

2.5.2 Recent Developments in Literature

Due to the BNNs computational expense, Gal and Ghahramani (2016) proposed Monte Carlo Dropout, in which dropout (a training process that randomly sets parameters to 0 during training to reduce over-fitting) was maintained when predicting on the testing data. The stochasticity introduced by turning each parameter into a weighted Bernoulli distribution, as shown below, approximated the BNN inference under the assumption of a Gaussian weight posterior:

$$w_{i,j,k} \sim \hat{w}_{i,j,k} * Ber(dropout\ rate),$$

with the dropout rate being the probability of each parameter being set to zero in each iteration of training. MC Dropout outperformed the PBP and VI posterior approximation algorithms used in BNN and required lower computational effort.

Non-Bayesian probabilistic networks have also demonstrated potential in probabilistic forecasting. These models have focused on using a weighted mixture of distributions to quantify uncertainty in the output. Lakshminarayanan et al. (2017) suggested an ensemble model, which trained several independent neural networks with different weight initialisations. Each network assumed a Gaussian distribution of the response and had an output layer estimating the mean and variance of the Gaussian. Each independent model was averaged with an equal weight. While non-Bayesian models are simpler and more

computationally efficient, Bayesian networks remain more popular in practice.

2.5.3 Mixture Density Networks - MDNs

Bishop (1994) introduced Mixture Density Networks (MDNs). Assume the output data follows a mixture distribution, commonly a mixed Gaussian, as such:

$$f(y_i|x_i) = \sum_{k=1}^{K} \alpha_{i,k}\phi(\frac{y_i - \mu_{i,k}}{\sigma_{i,k}}),$$

where $\phi(x)$ is the density function of a Standard Gaussian distribution. The MDN estimates the $\alpha_{i,k}$, $\mu_{i,k}$ and $\sigma_{i,k}$ that best approximate the real output distribution, as Figure 2.5 illustrates. MDNs are not a separate network design, they only specify the output layer, which contains output nodes for each α_k and its corresponding μ_k and σ_k. An MDN can be applied to feedforward NNs (Zen and Senior, 2014), LSTMs (Kuo, 2020), convolutional NNs (Iso et al., 2017) and more. The negative log likelihood loss function is used to assess the goodness of fit, while a softmax layer is applied on the α_k estimates so they sum to 1.

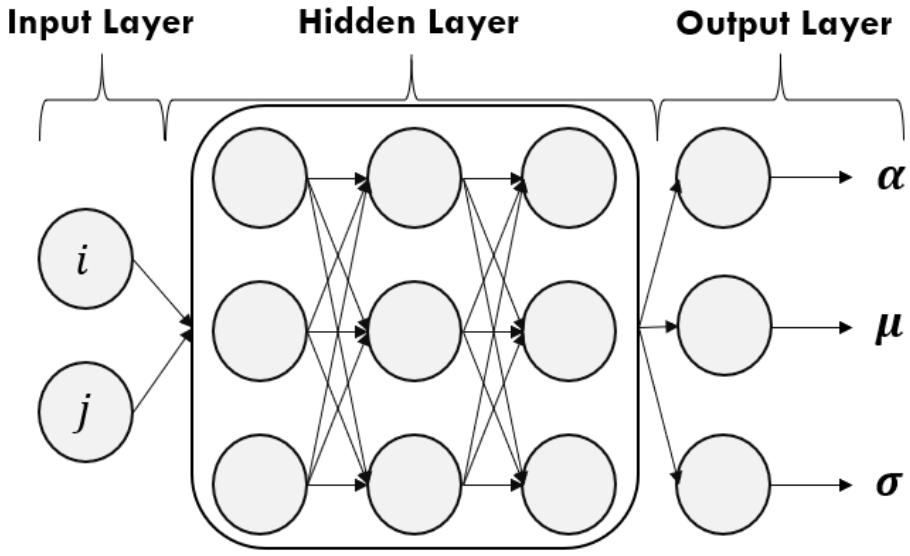


Figure 2.5: The basic structure of an MDN. It differs from standard NNs in the output layer, which approximate the parameters for a mixed distribution (commonly Gaussian)

MDNs are simple in implementation for a probabilistic forecasting network. As mentioned above, they are versatile in application and flexible to many network designs. Furthermore, the choice of mixed distribution is flexible; Poisson, Gamma, Log-Gaussian distributions (and even a variety of them in the one model) can be chosen. The network's output generally outperforms point forecasting Networks (Bishop, 1994), as a mean forecast can

be extracted from the mixed distribution as shown below. Furthermore, the MDNs assumption of heteroscedasticity (non-constant variance of the response) assists in producing stable point forecasts, as such:

$$E(\hat{y}_i|x_i) = \sum_{k=1}^{K} \alpha_{i,k}\mu_{i,k}$$

A downside of MDNs is the increased difficulty in convergence, as noted by Hjorth and Nabney (2000), who also expanded that singularities may exist in the loss function that could cause the model to diverge. Kuo (2020) notes the increased difficulty in converging a mixed Log-Gaussian distribution. Furthermore, despite mixed Gaussians being able to approximate any distribution (Nguyen and McLachlan, 2019), the model is technically a parametrisation of the output distribution and will hence be limited in its modelling. Assuming more Gaussians in the mixture can alleviate this shortcoming, but at increased complexity.

2.6 Model Validation Methodologies

2.6.1 Neural Network Loss Reserving Literature

The majority of Neural Network literature in the Loss Reserving field use the Upper Triangle only for training and validation. In the literature, there is no comprehensive partitioning of the Loss Triangle into training, validation and testing sets. Hence, a framework for testing and selecting different Neural Network designs within the Upper Triangle is lacking. The implementation of Machine Learning models in practice will require a framework that partitions the Upper Triangle and allows the out-of-sample testing of different designs.

Several notable model validation methodologies were employed. Wüthrich (2018b) performed a random 90/10 training/validation split on the upper triangle. This split risks unstable projections, as the model trained hasn't been tested on data in future calendar years. Rossouw and Richman (2019) and Kuo (2019) mitigate this risk by using the latest calendar years of the triangle for validation. Gabrielli et al. (2020) partitions Individual Claims data into training and validation sets before aggregation. Balona and Richman (2020) performs sequential partitioning of the Loss Triangle, however only applies this methodology to test Traditional Loss Reserving models.

Following the methodology of Gabrielli et al. (2020) requires Individual Claims data, which would limit the applicability of the proposed thesis. Otherwise, there is no comprehensive partition of the Loss Triangle described in the literature. Using testing data inside the Upper Triangle that is unseen during training will provide a more reliable measure of projection accuracy.

2.6.2 Time Series Model Validation

Given the focus of predicting future losses, the Loss Triangle can be viewed as a set of time series, one for each accident period. Studying time series fitting methodologies will assist in finding ways to test different Neural Network models.

Traditional Cross-Validation (Taylor and McGuire, 2016) approaches to model validation involve splitting the data into K sections. The model is fit on $K-1$ sections and evaluated on the test data (the remaining section). This process is repeated K times, using each section as the test data once. The residual error is averaged from each fit, to yield the Cross-Validation (CV) error. A model with the lowest CV error is preferred.

Cross-Validation measures a model's predictive power on unseen data. Bergmeir and Benítez (2012) point out that the validity of this method will break down for time-series analysis, due to time dependencies within the data sequence and the lack of future forecasting involved. As argued by Tashman (2000), data splitting should be done in chronological order, ie, training data preceding validation data and validation data preceding testing data. Tashman (2000) analyses fixed and rolling origin model evaluation techniques, commonly used in time series analysis.

Fixed origin validation assumes a fixed training, validation and testing partition, done in chronological order as mentioned above. Rolling origin is similar to the cross validation methodology. It involves a chronological partition of data, similar to fixed origin validation. It differs in that multiple partitions are made; the initial partition uses only the earlier portion of data. In subsequent partitions, the training data is progressively expanded until all data is used. The testing loss is the sum of the loss in all testing data used throughout the fitting process. Balona and Richman (2020) just recently applied the Rolling Origin methodology to the Loss Triangle, showcasing its effectiveness in model selection. Figure 2.6 visualises these methodologies.

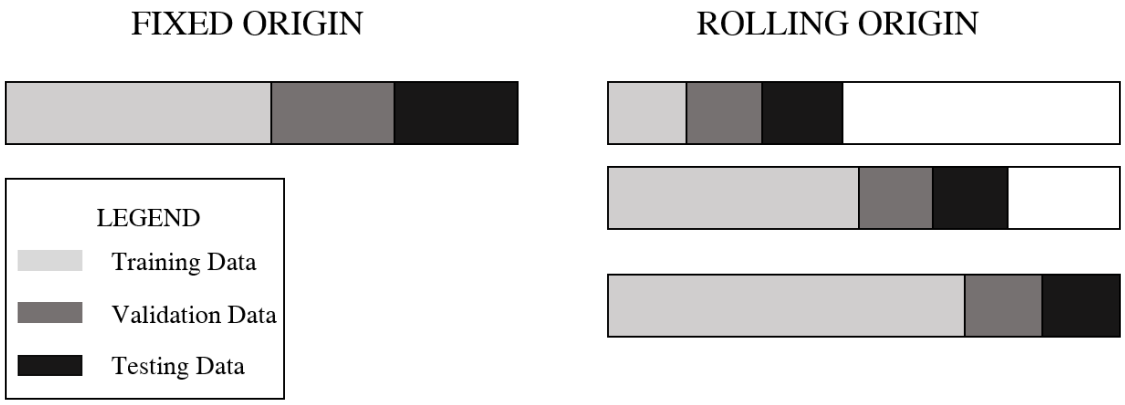


Figure 2.6: Fixed and Rolling Origin Validation

While fixed origin validation is simpler to implement, it will require a large forecasting period to be feasible, especially in the Loss reserving setting where the forecasting period is equal in size to the data available. A larger test set partition will reduce the training data used for fitting, leading to more volatile forecasts. Rolling origin maximises the use of training and testing data, with earlier partitions using more testing data to assess the projection accuracy of the model, while later partitions assess the model's ability to capture trends in the dataset.

2.7 Literature Summary

Loss Reserving is an issue central to actuarial science. Traditional Loss Reserving models, such as the Chain Ladder and GLMs, have been used reliably to model Outstanding Claims. Neural Networks have recently seen an increase in application to Loss Reserving and have demonstrated high versatility, flexibility and accuracy, out-performing basic benchmarks such as the Chain Ladder and ccODP models.

Current gaps which exist in the implementation of Neural Networks in the literature are hindering their potential, such as the lack of probabilistic forecasting, lack of interpretability and unstable extrapolation. The literature analysed outside the actuarial field regarding probabilistic forecasting with Neural Networks and Time Series Model Validation provided valuable models and methodologies, which have successfully tackled the gaps identified in Neural Network Loss Reserving.

Several Network architectures provide probabilistic forecasts, which are used in practice, such as Bayesian Networks and Mixture Density Networks. The works of Gabrielli et al. (2020) and Poon (2019) have already attempted to tackle the lack of interpretability

through the ResNet, which has shown promising results in achieving high performance and bridging the gap between GLMs and Neural Networks. Time series validation methodologies, such as the fixed and rolling origin methods, are specifically designed to deal with sequential data and assess models based on their projection accuracy, and have been used in Neural Network modelling successfully.

Chapter 3 combines the knowledge acquired from the literature of all fields and develops the modelling frameworks that applies probabilistic forecasting with Neural Networks to Loss Triangles.

CHAPTER 3

MODELLING FRAMEWORKS

This chapter builds the modelling frameworks which will help achieve the Research Aims set out in Section 1.3. Section 3.1 explains in detail the model designs which will be used to perform probabilistic forecasting and achieve interpretability (the MDN and ResMDN models). Section 3.2 explains how the model designs will be developed, in terms of fine-tuning the Network's hyper-parameters, performing the Rolling Origin data partition that allows assessment of a model's projection accuracy, to technical details surrounding the model's training and fitting.

Section 3.3 concludes with a model evaluation framework designed to assess the accuracy of the proposed Network models. The ccODP benchmark is set, along with qualitative and quantitative metrics that will be used to compare models to the ccODP. A set of objectives will be used as guidelines in determining the success of models.

3.0.1 Problem Formulation and Solution

It is important to note that this thesis will focus on Loss Triangle Reserving, hence any further analysis and modelling will be conducted from this framework. Gathering all information obtained from the Literature Review (Chapter 2), we find that several shortcomings exist in the implementations of Neural Networks in Loss Reserving, which are hindering their application in practice and the fulfilment of their potential. This

thesis aims to fix some of these shortcomings, helping to increase the acceptance of Neural Networks in actuarial practice. The shortcomings that will be addressed are:

1. The lack of focus on distributional forecasting with Neural Networks for Loss Triangle Reserving. While articles indirectly assume Incremental Claims to follow a distribution through using loss functions such as the MSE (homoscedastic Gaussian distribution) or the Poisson deviance (Poisson distribution), no Neural Network has been applied which focuses its modelling power on finding the right distribution.

2. No comprehensive model testing and selection framework has been developed for Loss Triangles, to my knowledge. Balona and Richman (2020) did implement such a method, but only to select between Traditional Reserving models, not Machine Learning models. Such a model selection framework needs to partition the Loss Triangle into training, validation and testing sets, in order to allow systematic comparison between different Network designs.

Probabilistic forecasting was conducted using the Mixture Density Network (Section 3.1.1), while the Rolling Origin Model Validation Method (Section 3.2.3) was used to partition the data and assess different models. The issue of interpretability was addressed, but not extended conceptually beyond the ResNet framework developed by Wüthrich and Merz (2019) and applied by Gabrielli et al. (2020); Gabrielli (2020); Poon (2019); Schelldorfer and Wüthrich (2019).

3.0.2 Notation

- Let $\Phi(x|\mu,\sigma) = P(Z \leq x)$, where $Z \sim N(\mu,\sigma)$
- Let $\phi(x|\mu,\sigma) = \frac{d\Phi(x|\mu,\sigma)}{dx}$, that is, $\phi(x|\mu,\sigma)$ is the probability density function of a Gaussian variable with mean μ and standard deviation σ.
- Let $X_{i,j}$ be the Incremental Claims paid in Accident period i and Development period j.
- Let $\hat{X}_{i,j}$ be a random variable which a model predicts to be the distribution of $X_{i,j}$.

3.1 Model Design

3.1.1 Probabilistic Forecasting - Mixture Density Networks

Mixture Density Networks (MDNs) were used to perform probabilistic forecasting of Outstanding Claims. This network assumes that output follows a Mixed Gaussian, as Equation (3.1) shows. The MDN's inputs are be the Accident and Development Quarters, (i, j), while the outputs are the parameters of the mixed distribution, $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, which are used as parameters for a Mixed Gaussian density. Figure 3.1 provides a visualisation MDN's design. A Negative Log Likelihood (NLL) loss function is used to measure the MDN's goodness-of-fit, shown in Equation (3.2).

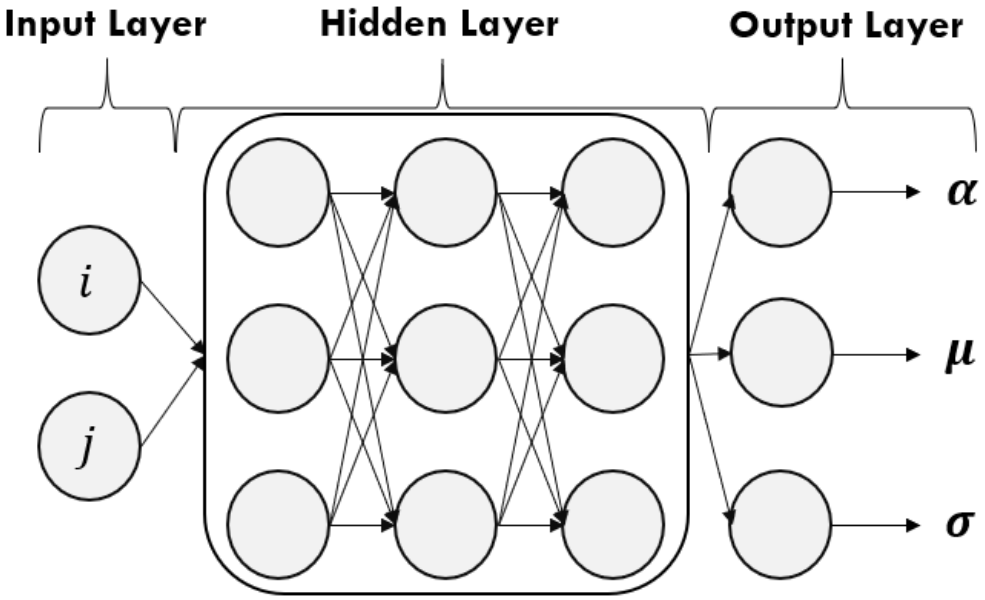


Figure 3.1: The basic design of the Mixture Density Network (MDN). The inputs (i, j) are the Accident and Development Quarters respectively. The outputs are the parameters of the Mixed Gaussian distribution, α, μ, σ

$$f_{\hat{X}_{i,j}}(x) = \sum_{k=1}^{K} \alpha_{k,i,j} \phi(x|\mu_{i,j,k}, \sigma_{i,j,k}) \tag{3.1}$$

$$NLLLoss(\mathbf{X}, \hat{\mathbf{X}}|\mathbf{w}) = -\frac{1}{|\mathbf{X}|} \sum_{i,j:X_{i,j} \in Train} ln(f_{\hat{X}_{i,j}}(X_{i,j}|\mathbf{w})) \tag{3.2}$$

3.1.2 Density of Individual Components

During modelling, both Mixed Gaussians and Mixed Log-Gaussians were fit to Incremental Claims, $X_{i,j}$. A Mixed Log-Gaussian can be fit to $X_{i,j}$ by fitting a Mixed Gaussian to

$ln(X_{i,j})$, as Lemma 3.1.1 outlines.

Lemma 3.1.1 *Let Y and $X = ln(Y)$ be random variables. Suppose that X follows a Mixed Gaussian distribution with parameters $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, such that:*

$$f_X(x) = \sum_{k=1}^{K} \alpha_k \phi(x|\mu_k, \sigma_k)$$

Then Y follows a Mixed Log-Gaussian distribution with parameters $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, such that:

$$f_Y(y) = \frac{1}{y} \sum_{k=1}^{K} \alpha_k \phi(ln(y)|\mu_k, \sigma_k)$$

Proof: See Appendix A.0.1

3.1.3 MDN Computations

Input:

The input variables of the MDN are i and j, the Accident and Development Quarters, respectively. Suppose there are N data points in the training set, (i_n, j_n), for $n = 1, 2, 3, ..N$. The input will be normalised, see the Appendix A.1 for more details.

Hidden Layers:

The input variables (i_n, j_n) are fed individually into the first hidden layer through weight parameters. Each node in that layer takes a weighted sum of i_n and j_n, adds an intercept term (known as the bias), before applying an activation function to that weighted sum. Let the output of the p^{th} node in the first hidden layer for data point n be denoted by $\mathbf{z}^1_{p,n}$. Furthermore, let $w_{1,p}$ and $w_{2,p}$ be the weight parameters connecting the input i and the input j to the p^{th} node, respectively. $\mathbf{z}^1_{p,n}$ is calculated following Equation (3.3):

$$\mathbf{z}^1_{p,n} = f_1(i_n w^1_{1,p} + j_n w^1_{2,p} + b^0_p), \tag{3.3}$$

where f_1 is the activation function of the first layer and b^0_p is the bias term applied to that node. Let there be L hidden layers in the MDN. Each node in a hidden layer takes a weighted sum of the output of all nodes in the preceding layer and an interpect term (bias), before passing it through an activation function. Let the output of the p^{th} node in the l^{th} hidden layer for data point n be denoted by $\mathbf{z}^l_{p,n}$. Let there be D nodes in the $l - 1^{th}$ layer. Let $w^l_{a,b}$ be the weight parameter connecting the a^{th} nodes in hidden layer l to the b^{th} nodes in hidden layer $l + 1$. $\mathbf{z}^l_{p,n}$ is calculated following Equation (3.4):

$$\mathbf{z}_{p,n}^{l} = f_l(\sum_{d=1}^{D} w_{d,p}^{l-1}\mathbf{z}_{d,n}^{l-1} + b_p^{l-1}) \tag{3.4}$$

For data point n, the output of node p in the final hidden layer, L, is similarly calculated following Equation (3.5):

$$\mathbf{z}_{p,n}^{L} = f_L(\sum_{d=1}^{D} w_{d,p}^{L-1}\mathbf{z}_{d,n}^{L-1} + b_p^{L-1}) \tag{3.5}$$

In this thesis, through initial modelling, the sigmoid activation function, shown in Equation (3.6), gave the best results, hence was used throughout this thesis for all hidden layers.

$$f_l(z) = \text{Sigmoid}(z) = \frac{1}{1+e^{-z}} \text{ for } l = 1, 2, 3, ...L-1 \tag{3.6}$$

Output Layer:

The output layer is split into three section, each with K nodes, K being the number of components in the Mixture Density. Let's call these section the alpha, mu and sigma sections. Similarly to the hidden layers, each node in the output layer takes a weighted sum of the output of all nodes in the last hidden layer, Layer L. The weighted sums are then passed through different activation functions for each section to yield the final output of the MDN, $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$. Denote $z_{k,n}^{\alpha}$ as the output of the k^{th} nodes of the alpha section, for data point n. Similarly define $z_{k,n}^{\mu}$ and $z_{k,n}^{\sigma}$ for nodes in the mu and sigma sections, respectively. Let $w_{d,k}^{L,\alpha}, w_{d,k}^{L,\mu}, w_{d,k}^{L,\sigma}$ be the weights connecting the output of node d in layer L to node k in the alpha, mu and sigma layers, respectively. These nodes in each of these 3 output layers are calculated following Equations (3.7, 3.9, 3.11), while the final output parameters, $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, are calculated following Equations (3.8, 3.10, 3.12) :

Alpha:

$$\mathbf{z}_{k,n}^{\alpha} = \sum_{d=1}^{D} w_{d,k}^{L,\alpha}\mathbf{z}_{k,n}^{L} + b_k^{L,\alpha}, \text{ for } k = 1, 2, ..K \tag{3.7}$$

$$\alpha_{k,n} = \frac{e^{\mathbf{z}_{k,n}^{\alpha}}}{\sum_{k=1}^{K} e^{\mathbf{z}_{k,n}^{\alpha}}} \tag{3.8}$$

Mu:

$$\mathbf{z}_{k,n}^{\mu} = \sum_{d=1}^{D} w_{d,k}^{L,\mu}\mathbf{z}_{k,n}^{L} + b_k^{L,\mu}, \text{ for } k = 1, 2, ..K \tag{3.9}$$

$$\mu_{k,n} = \mathbf{z}_{k,n}^{\mu} \tag{3.10}$$

Sigma:

$$\mathbf{z}_{k,n}^{\sigma} = \sum_{d=1}^{D} w_{d,k}^{L,\sigma} \mathbf{z}_{k,n}^{L} + b_{k}^{L,\sigma}, \text{ for } k = 1, 2, ..K \tag{3.11}$$

$$\sigma_{k,n} = e^{\mathbf{z}_{k,n}^{\sigma}} \tag{3.12}$$

As the formulas above showed, the output of the alpha layer, $\mathbf{z}_{k,n}^{\alpha}$, is passed through a Softmax activation function, which ensures that $\sum_{k=1}^{K} \alpha_{k,n} = 1$. The output of the mu layer has no activation function, hence $\mu_{k,n} = \mathbf{z}_{k,n}^{\mu}$. The sigma output is passed through an exponential function, following Hjorth and Nabney (2000), which ensures the standard deviation is always positive.

Hence, for each input cell (i, j), a unique combination of parameters,

$$(\alpha_{i,j,1}, \alpha_{i,j,2}.....\alpha_{i,j,K}, \mu_{i,j,1}, \mu_{i,j,2}.....\mu_{i,j,K}, \sigma_{i,j,1}, \sigma_{i,j,2}.....\sigma_{i,j,K}),$$

is produced in the output layer of the MDN, which then generates the probability distribution for $\hat{X}_{i,j}$ shown in Equation (3.13):

$$f_{\hat{X}_{i,j}}(x) = \sum_{k=1}^{K} \alpha_{i,j,k} \phi(x|\mu_{i,j,k}, \sigma_{i,j,k}) \tag{3.13}$$

3.1.4 Mean, Variance and Quantile Estimates

3.1.4.1 Mixed Gaussian

If a **Mixed Gaussian** is fit to the raw data, then

$$f_{\hat{X}_{i,j}}(x) = \sum_{k=1}^{K} \alpha_{i,j,k} \phi(x|\mu_{i,j,k}, \sigma_{i,j,k}), \tag{3.14}$$

for which the mean and variance can be calculated as following:

- $E[\hat{X}_{i,j}] = \sum_{k=1}^{K} \alpha_{i,j,k} \mu_{i,j,k}$
- $Var(\hat{X_{i,j}}) = \sum_{k=1}^{K} \alpha_{i,j,k} (\sigma_{i,j,k}^2 + \mu_{i,j,k}^2) \quad - \quad E[\hat{X}_{i,j}]^2$

Let $\hat{X}_{i,j,q}$ be the q^{th} quantile estimate of $X_{i,j}$. The quantile estimate is estimated such that $\hat{X}_{i,j,q} \approx F^{-1}_{\hat{X}_{i,j}}(q)$, where

$$F_{\hat{X}_{i,j}}(x) = \sum_{k=1}^{K} \alpha_{i,j,k} \Phi(x|\mu_{i,j,k}, \sigma_{i,j,k}) \tag{3.15}$$

3.1.4.2 Mixed Log-Gaussian

If a **Mixed Log-Gaussian** is fit, that is, a Mixed Gaussian is fit to the Log of **X**, then

$$f_{ln(\hat{X}_{i,j})}(x) = \sum_{k=1}^{K} \alpha_{i,j,k} \phi(\hat{X}_{i,j}|\mu_{i,j,k}, \sigma_{i,j,k}),$$

and the following properties can be calculated:

- $E[\hat{X}_{i,j}] = \sum_{k=1}^{K} \alpha_{i,j,k} e^{\mu_{i,j,k} + 0.5 * \sigma_{i,j,k}}$
- $Var(\hat{X}_{i,j}) = \sum_{k=1}^{K} \alpha_{i,j,k} e^{\sigma^2_{i,j,k}} e^{2\mu_{i,j,k} + \sigma^2_{i,j,k}} \quad - E[\hat{X_{i,j}}]^2$

Let $ln(\hat{X}_{i,j})_q$ be the q^{th} quantile estimate of $ln(X_{i,j})$. Since $ln(\hat{X}_{i,j})$ follows a Mixed Gaussian distribution, calculating $ln(\hat{X}_{i,j})_q$ follows the same process as Equation (3.9). Let $\hat{X}_{i,j,q}$ be the q^{th} quantile estimate of $X_{i,j}$, then

$$\hat{X}_{i,j,q} = e^{ln(\hat{X}_{i,j})_q} \tag{3.16}$$

3.1.5 Interpretability - The ResMDN

The Mixture Density Network (MDN) has even greater computational complexity than the standard feedforward Neural Network, hence its interpretability is even lower. To implement a more interpretable structure, this thesis adapts the ResNet design implemented successfully by Gabrielli et al. (2020), Gabrielli (2019) and Poon (2019), boosting GLM models while producing a more interpretable and stable model. See Section 2.3.3 for a detailed explanation of how the ResNet functions. In this thesis, the ResNet design was adapted to the MDN to create the ResMDN. The ResMDN uses a skip connection, applied in the form of an Embedding Layer, to connect the input layer directly to the output layer.

A GLM is approximated by a Mixed Gaussian, the parameters of which are generated by the Embedding Layer and passed to the output layer. This configuration initialises the

ResMDN with the results of a GLM approximation. A Neural Network module connected to the ResMDN is trained on the residuals of the GLM approximation, capturing structure which the GLM missed and feeding it to the output, hence boosting the GLM. Figure 3.2 illustrates the design and implementation of the ResMDN.

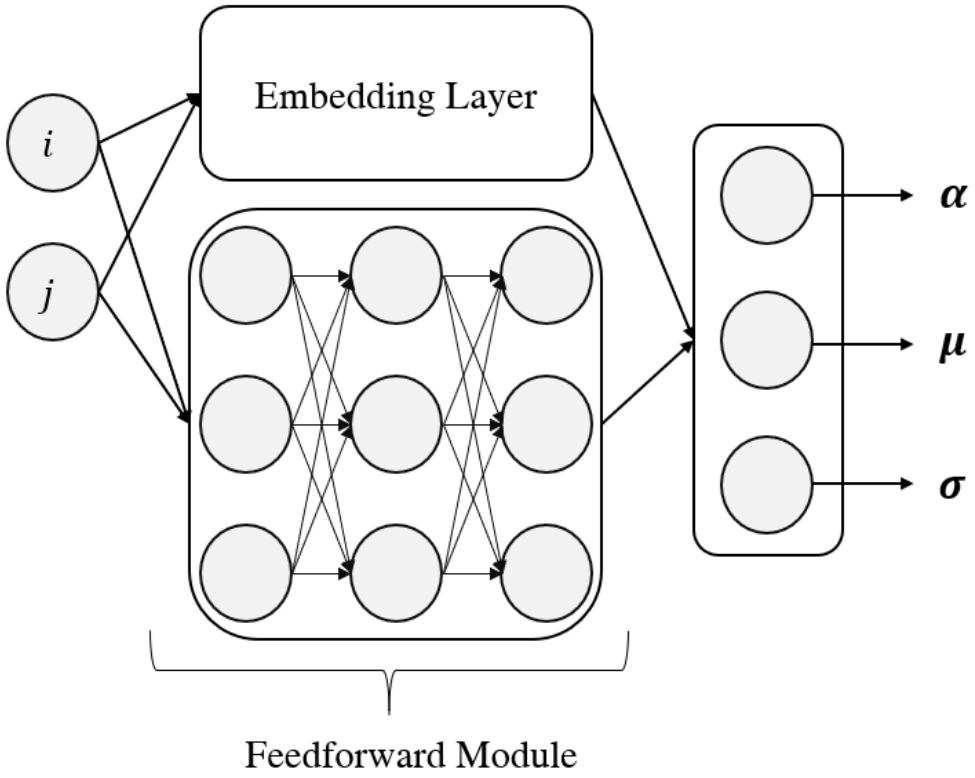


Figure 3.2: The ResMDN design with an MDN output. The Embedding Layer converts the input to Mixed Gaussian parameters approximating the GLM fit. The feedforward module boosts the GLM initialisation during training.

The implementation of the ResMDN closely followed the methodology demonstrated by Gabrielli et al. (2020). The ResMDN embeds an approximation of the Cross-Classified Over-Dispersed Poisson (ccODP) model (see Section 3.3.1 for more detail), which follows this distribution outlined in Equation (3.17):

$$\frac{\hat{X}_{i,j}}{\phi} \sim \text{Poi}\left(\frac{A_i B_j}{\phi}\right) \tag{3.17}$$

3.1.6 Approximating the ccODP Model through a Mixed Gaussian

Since the output of the ResMDN takes the form of parameters for a Mixed Gaussian distribution, the GLM embedding must also be in the form of Mixed Gaussian parameters.

Hence, the ccODP fit is approximated by a Mixed Gaussian, as shown in Equation (3.18):

$$f_{\hat{X}_{i,j}^{ccODP}}(x) \approx \sum_{k=1}^{K} \alpha_{i,j,k}^{GLM} \phi_{i,j,k}(x, \mu_{i,j,k}^{GLM}, \sigma_{i,j,k}^{GLM}), \tag{3.18}$$

where:

- $f_{\hat{X}_{i,j}^{ccODP}}(x)$ is the distribution of $X_{i,j}$ estimated by the ccODP model, that is, $\frac{\hat{X}_{i,j}^{ccODP}}{\phi} \sim \text{Poi}\left(\frac{A_i B_j}{\phi}\right)$
- $\alpha_{i,j,k}^{GLM} = \frac{1}{K}$
- $\mu_{i,j,k}^{GLM} = E[\hat{X}_{i,j}^{ccODP}] = A_i B_j$
- $\sigma_{i,j,k}^{GLM} = \sqrt{Var[\hat{X}_{i,j}^{ccODP}]} = \sqrt{\phi A_i B_j}$

Equation (3.18) approximates the ccODP distribution of a Gaussian distribution, spread evenly over K components.

3.1.7 ResMDN Computations

Input Layer:
The Input Layer of the ReMDN consists of the Accident and Development Quarters, (i, j), as well as an integer, $c_{i,j} = 40 * (i-1) + j$, which is fed as a categorical input to the network. Each cell (i, j) is assigned a unique $c_{i,j}$, which acts as a categorical identifier of that cell. An embedding layer takes the categorical input $c_{i,j}$ and produces as output $\left(ln(\boldsymbol{\alpha}_{\boldsymbol{i,j}}^{\boldsymbol{GLM}}), \boldsymbol{\mu}_{\boldsymbol{i,j}}^{\boldsymbol{GLM}}, ln(\boldsymbol{\sigma}_{\boldsymbol{i,j}}^{\boldsymbol{GLM}})\right)$. Those parameters will be concatenated with output from the last Neural Network hidden layer and transformed with the output layer activation functions to produce the final output Mixed Gaussian parameters $(\boldsymbol{\alpha}^{\boldsymbol{ResMDN}}, \boldsymbol{\mu}^{\boldsymbol{ResMDN}}, \boldsymbol{\sigma}^{\boldsymbol{ResMDN}})$.

Embedding Layer:
The Embedding Layer maps categorical input to an n-dimensional numerical vector. It is the driving factor behind the success of Kuo (2019) and Gabrielli (2019) in learning from multiple triangles simultaneously, since it converts categorical variables into numerical output, allowing them to be run in the same Network simultaneously. The Embedding Layer weights are pre-set to provide a mapping as shown in Equation (3.19):

$$\textbf{Embedding Layer: } c_{i,j} \mapsto \left(ln(\boldsymbol{\alpha}_{\boldsymbol{i,j}}^{\boldsymbol{GLM}}), \boldsymbol{\mu}_{\boldsymbol{i,j}}^{\boldsymbol{GLM}}, ln(\boldsymbol{\sigma}_{\boldsymbol{i,j}}^{\boldsymbol{GLM}}\right) \tag{3.19}$$

The Log of the $\boldsymbol{\alpha}$ and $\boldsymbol{\sigma}$ parameters are produced in the embedding layer, since the Softmax and exponential Activation functions will take the exponent in the output layer nodes.

Neural Network Module:
A feedforward Neural Network module also connects the input and output layers, capturing interactions which the GLM may have missed. The output generated by the module follows Equation (3.26). Note that the Softmax and exponential activation functions are excluded from this module, as the Embedding and Neural Network modules must be added before those activations are performed.

The computations performed in the Neural Network layer are identical to the calculations performed in Equations (3.3-3.5, 3.7,3.9,3.11) for the MDN. Let $w^L_{d,k}$ and b^L_k be defined similarly to Equations (3.7,3.9,3.11). Let $\mathbf{z}^{\alpha}_{i,j,k}$ be the weighted sum of output from the L^{th} hidden layer of the Neural Network module with input (i,j), which forms the k^{th} $\boldsymbol{\alpha}$ component. Similarly define $\mathbf{z}^{\mu}_{i,j,k}$ and $\mathbf{z}^{\sigma}_{i,j,k}$. Let there be D nodes in the L^{th} hidden layer. The Neural Network module produces output following Equations (3.20-3.22):

$$\mathbf{z}^{\alpha}_{i,j,k} = \left(\sum_{d=1}^{D} w^{L,\alpha}_{d,k} \mathbf{z}^{L}_{i,j} + b^{L,\alpha}_{k} \right), \text{ for k = 1,2,..K} \tag{3.20}$$

$$\mathbf{z}^{\mu}_{i,j,k} = \left(\sum_{d=1}^{D} w^{L,\mu}_{d,k} \mathbf{z}^{L}_{i,j} + b^{L,\mu}_{k} \right), \text{ for k = 1,2,..K} \tag{3.21}$$

$$\mathbf{z}^{\sigma}_{i,j,k} = \left(\sum_{d=1}^{D} w^{L,\sigma}_{d,k} \mathbf{z}^{L}_{i,j} + b^{L,\sigma}_{k} \right), \text{ for k = 1,2,..K} \tag{3.22}$$

For simplicity, we let:

$$ln(\alpha^{NN}_{i,j,k}) = \mathbf{z}^{\alpha}_{i,j,k} \tag{3.23}$$

$$\mu^{NN}_{i,j,k} = \mathbf{z}^{\mu}_{i,j,k} \tag{3.24}$$

$$ln(\sigma^{NN}_{i,j,k}) = \mathbf{z}^{\sigma}_{i,j,k} \tag{3.25}$$

Hence the Neural Network module performs the process as described in Equation (3.26):

$$\textbf{Neural Network Module: } (i,j) \mapsto \left(ln(\boldsymbol{\alpha}^{\boldsymbol{NN}}_{\boldsymbol{i,j}}), \boldsymbol{\mu}^{\boldsymbol{NN}}_{\boldsymbol{i,j}}, ln(\boldsymbol{\sigma}^{\boldsymbol{NN}}_{\boldsymbol{i,j}}) \right) \tag{3.26}$$

Addition:
The output from the Embedding and Neural Network Layers are added together element-wise, as described in Equations (3.27-3.29):

$$\left(ln(\alpha_{i,j,k}^{GLM}), \mu_{i,j,k}^{GLM}, ln(\sigma_{i,j,k}^{GLM}), ln(\alpha_{i,j,k}^{NN}), \mu_{i,j,k}^{NN}, ln(\sigma_{i,j,k}^{NN})\right) \tag{3.27}$$

$$\mapsto \left(ln(\alpha_{i,j,k}^{GLM}) + ln(\alpha_{i,j,k}^{NN}), \mu_{i,j,k}^{GLM} + \mu_{i,j,k}^{NN}, ln(\sigma_{i,j,k}^{GLM}) + ln(\sigma_{i,j,k}^{NN})\right) \tag{3.28}$$

$$= \left(ln(\alpha_{i,j,k}^{ResMDN}), \mu_{i,j,k}^{ResMDN}, ln(\sigma_{i,j,k}^{ResMDN})\right) \tag{3.29}$$

Final Activations:
The added variables are then passed through the activation functions of the final output layer. The α parameters are passed through a Softmax, while the σ parameters are passed through an exponential activation, as shown in Equations (3.30,3.31).

$$\left(ln(\alpha_{i,j,k}^{ResMDN}), \mu_{i,j,k}^{ResMDN}, ln(\sigma_{i,j,k}^{ResMDN})\right) \mapsto \left(\frac{e^{ln(\alpha_{i,j,k}^{ResMDN})}}{\sum_{k=1}^{K} e^{ln(\alpha_{i,j,k}^{ResMDN})}}, \mu_{i,j,k}^{ResMDN}, e^{ln(\sigma_{i,j,k}^{ResMDN})}\right) \tag{3.30}$$

$$= \left(\alpha_{i,j,k}^{ResMDN}, \mu_{i,j,k}^{ResMDN}, \sigma_{i,j,k}^{ResMDN}\right) \tag{3.31}$$

Hence the Mixed Gaussian parameters are produced in the Output Layer.

Initialisation:
At the beginning of training, the parameters $(\mathbf{w}_L, \mathbf{b}_L)$, used in Equations (3.20-3.22), are initialised at 0, such that:

- $ln(\boldsymbol{\alpha}^{\boldsymbol{NN}}), \boldsymbol{\mu}^{\boldsymbol{NN}}, \boldsymbol{\sigma}^{\boldsymbol{NN}} = 0$
- $\alpha_{i,j,k}^{ResMDN} = \alpha_{i,j,k}^{GLM}$
- $\mu_{i,j,k}^{ResMDN} = \mu_{i,j,k}^{GLM}$
- $\sigma_{i,j,k}^{ResMDN} = \sigma_{i,j,k}^{GLM}$

Hence producing the GLM approximation in the Output Layer at the initialisation of the ResMDN. During training, the Embedding Layer maintains constant output, while the Neural Network module adjust its weights to capture non-linearities which the GLM has missed. The ResMDN's output at the termination of training is shown in Equations (3.32,

3.33):

$$(i,j,c_{i,j}) \mapsto \left(\frac{\alpha_{i,j,k}^{GLM}\alpha_{i,j,k}^{NN}}{\sum_{k=1}^{K}\alpha_{i,j,k}^{GLM}\alpha_{i,j,k}^{NN}}, \mu_{i,j,k}^{GLM} + \mu_{i,j,k}^{NN}, \sigma_{i,j,k}^{GLM}\sigma_{i,j,k}^{NN} \right), \text{ for } k = 1,2,...,K \quad (3.32)$$

$$= \left(\alpha_{i,j,k}^{ResMDN}, \mu_{i,j,k}^{ResMDN}, \sigma_{i,j,k}^{ResMDN}\right), \text{ for } k = 1,2,...,K \quad (3.33)$$

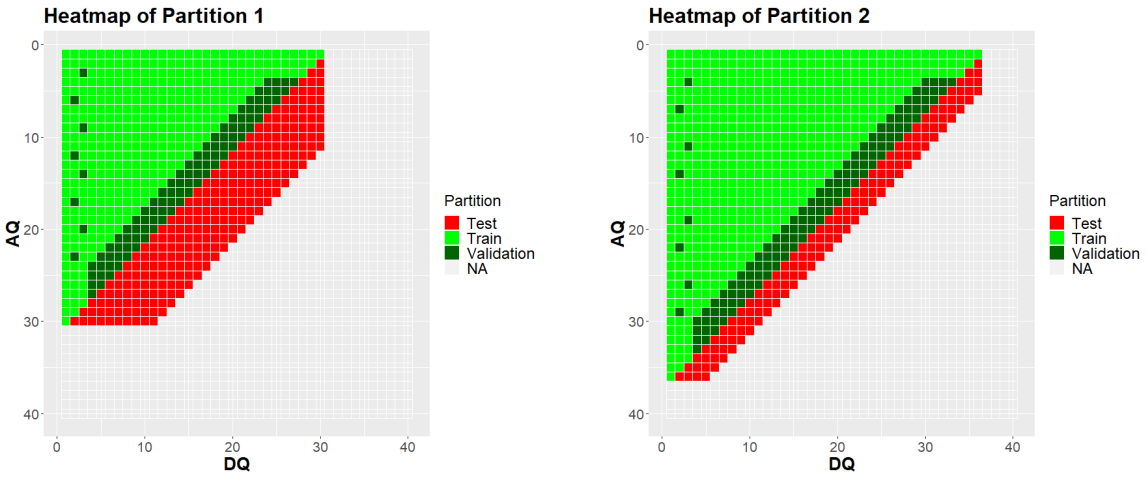
The NN boosting terms are relatively easy to analyse in relation to the GLM fit, especially the mean and volatility terms. Furthermore, the black box Neural Network modelling is only applied to the residuals, meaning the lack of interpretability is restricted to that domain only. Hence the ResMDN improves the interpretability of the model. As mentioned before, the GLM backbone in this model increases the stability of model convergence and output.

3.2 Model Development

3.2.1 Assessing Projection Accuracy - Rolling Origin Model Validation

The Neural Network Loss Reserving literature doesn't outline an explicit training/testing partition of the Loss Triangle, meaning that no framework has been provided for testing and selecting different Network designs. The accuracy of the Neural Network depends heavily on its hyper-parameters, such as the number of hidden layers, number of neurons, weight regularisation penalty, etc. Therefore, to assume that one model design will work well in all environments will lead to sub-optimal performance. This thesis partitions the Loss Triangle into training,validation and testing sets, allowing for a systematic hyper-parameter fine-tuning algorithm to be implemented (See Section 3.2.2 and 3.2.3).

The Loss Triangle has the characteristics of a time series, with the claims paid generally decaying over successive development periods. Where the objective of modelling is to improve interpolation accuracy, randomly splitting the data into training, validation and testing sets is common and sufficient. With Loss Triangles, the objective is extrapolation, hence the testing set needs to focus on assessing the model's projection accuracy. This is done by assigning the latest calendar periods of the triangle to the testing set and the earliest to training. Similarly, the Validation set is chosen to be the latest calendar periods which aren't assigned for testing. That way, when combined with Early Stopping (see Section 2.2.2), the MDN stops training when short term projection accuracy is maximised.



(a) Partition 1

(b) Partition 2

Figure 3.3: The 2-stage partition of the triangle into training, validation and testing sets. The first partition focuses on assessing projection, while the second assesses the model's ability to fit the trend of the data.

Hence it is important to sequentially split the data into training, validation and testing sets to more effectively assess the model's accuracy when extrapolating. The Rolling Origin Validation Method was used (Tashman, 2000) in two stages:

- In the first stage, the testing data is comprised of a larger number of the later calendar periods, at the expense of training periods. This partition will focus on assessing the modelâs accuracy when projecting into the Lower Triangle. A downside is reduced training data, especially since the later calendar periods contain valuable insight into future claim trends.

- The second stage will provide more calendar periods for the training set at the expense of the testing set. This partition will assess the modelâs accuracy when using almost the entirety of the triangle and how well it captures trends in the whole dataset.

The validation data included the 4 latest non-testing calendar periods, excluding the first 3 Accident and Development periods. This exclusion was done to provide the MDN more training data for the latest Accident and Development periods. Excluding DQ2 and DQ3 points from the validation set reduced the MDN's attention towards accurately projecting claims in those development periods. Hence, points from DQ2 and 3 in earlier AQs were added to the validation sets, helping to maintain as much training data for the later AQs as possible. Figure 3.3 visualises the data partitions.

3.2.2 Direct Projection Constraints

This thesis implemented a mechanism for directly constraining central estimates of cells in the Lower Triangle, thereby directly controlling projections made by the MDN. The user can place upper and lower bounds on the central estimates of any desired set of cells (i, j) in the Lower Triangle and penalising the MDN if its central estimates fell outside those boundaries.

Let $\mathbf{C}$ be the set of cells (i, j) in the Lower Triangle, which have had constraints placed on their projections. Let $C_{i,j}^{Lower}$ and $C_{i,j}^{Upper}$ be the lower and upper constraints of the central estimates for cell (i, j). The loss function during training becomes as such:

$$NLLLoss(\mathbf{X}, \hat{\mathbf{X}}|\mathbf{w}) = -\frac{1}{|\mathbf{X}|} \sum_{i,j:X_{i,j} \in Train} ln(f_{\hat{X}_{i,j}}(X_{i,j}|\mathbf{w})) + \lambda_C \sum_{i,j:(i,j) \in \mathbf{C}} [max(0, \hat{\mu}_{i,j} - C_{i,j}^{Upper})]^2 + [max(0, C_{i,j}^{Lower} - \hat{\mu}_{i,j})]^2$$

Where λ_C is a constraint violation penalty coefficient. The constraints apply a square distance penalty to the loss function if the central estimate of constrained cells in the Lower Triangle violate the constraints. With a sufficiently high penalty coefficient, the MDN's projection will satisfy the constraints specified, providing projections that are more reasonable.

3.2.3 Optimising Network Hyper-parameters

As mentioned earlier, Neural Networks have demonstrated their flexibility in design and implementation. Different model designs will yield a different accuracy of fit, hence different model architectures will be tested, with the one leading to the lowest testing error being chosen. Section 3.5 will explain in detail how the testing error is calculated for each model. The following hyper-parameters will be tested in finding the best performing architecture.

1. Weight Regularisation

L2 regularisation is commonly imposed on the Network's weights to reduce the model's flexibility, which reduces over-fitting. Large weights can cause large fluctuations in the output under slightly differing inputs (Reed and Marks, 1999), which will cause projections to become unstable. Having a small dataset also encourages over-fitting, since a wider range of functions can be fit with a deceptively low test error, which in reality project inaccurately on unseen data. Despite the Loss triangle containing 820 data points, it is still considered a small dataset in the Neural Network setting, which further necessitates

the need for regularisation.

Datasets with a simpler trend will generally require more regularisation in order to penalise unnecessary complexity. L2 penalty coefficients were tested on a logarithmic scale in order to efficiently test a wide parameter space, hence in this thesis the coefficients tested were $[0, 0.0001, 0.001, 0.01, 0.1]$.

2. Sigma Activity Regularisation

A local optimum in the loss function may involve unreasonably high variance, which produces unrealistic results (Hjorth and Nabney, 2000). Activity regularisation penalises the output, σ, rather than the weights preceding the output node. Datasets with a more complex trend will generally require higher sigma activity regularisation, to encourage the MDN to capture the trend rather than assign a high sigma to the region. Similarly with weight regularisation, the penalty coefficients tested were $[0, 0.0001, 0.001, 0.01, 0.1, 1]$.

Let $\mathbf{w}$ be a vector of all weights in the Network, excluding bias weights. Let λ_w and λ_σ be the L2 weight and sigma penalty coefficients, respectively. With the penalties applied above, the Loss function during training follows Equation (3.34).

$$Loss(\mathbf{X}_{Train}, \hat{\mathbf{X}}_{Train}|\mathbf{w}, \lambda_w, \lambda_\sigma) = -\frac{1}{|\mathbf{X}_{Train}|} \sum_{i,j:X_{i,j} \in Train} ln(f_{\hat{X}_{i,j}}(X_{i,j}|\mathbf{w})) \tag{3.34}$$

$$+ \lambda_w \mathbf{w} \cdot \mathbf{w} \tag{3.35}$$

$$+ \lambda_\sigma \sum_{i,j:X_{i,j} \in Train} \sum_{k=1}^{K} \sigma_{i,j,k}^2 \tag{3.36}$$

3. Dropout Rate

Dropout is a common regularisation technique used in Neural Network to minimise over-fitting. In the Loss Reserving literature, dropout has frequently been successfully applied (Gabrielli et al., 2020; Gabrielli, 2019; Kuo, 2019; Poon, 2019) for that purpose. Let Dropout be applied to a specific hidden layer. During training, at each epoch, the MDN will set the output of individual nodes in that layer to 0, with a rate p. This stops the Network from over-training neurons by assigning too much weight on them relative to the rest of the Network. When such over-trained Neurons have their output set to 0 in one epoch, the Network will be forced to place importance on other neurons. This reduces over-fitting and increases the stability of the MDN's output (Srivastava et al., 2014).

When the MDN has been trained and is producing projections, dropout is not applied. However, the output of each neurons will be multiplied by $1 - p$, which gives a similar

effect to ensembling models. This increases the accuracy and stability of output.

Despite this theoretical backing, early modelling showed that having no dropout, or low rates of dropout, commonly outperformed. Dropout rates tested were $[0, 0.1, 0.2]$. Since the MDN's modelling capacity is reduced, the number of neurons was always increased by $\frac{1}{1-p}$ when a Dropout rate of p was applied to the Network.

4. Number of Components

Increasing the number of components will increase the ability of the mixture to approximate the true distribution, at the cost of an increased risk of over-fitting and over-parametrisation. Poisson, Gamma and Log-Normal distributions, commonly used in GLM fitting, are unimodal, hence less components are required to achieve a satisfactory fit. A range of 1-4 components was found to be satisfactory in approximating the distribution of Incremental Claims.

5. Number of Hidden Layers

Between $1 - 4$ Hidden layers were tested and found to be sufficient in achieving a satisfactory fit. This range is concurrent with the Actuarial literature (Wüthrich, 2018b; Gabrielli et al., 2020; Delong et al., 2020) and literature applying MDNs (Bishop, 1994; Zen and Senior, 2014; Ormoneit and Tresp, 1996). According to the universal approximation theory, as explained by Cybenko (1989), one hidden layer is enough to approximate any continuous function. Despite this theory, it may take many nodes to model an accurate function, hence using more than one layer can more effectively capture deep interactions between variables. An excessive number of hidden layers could lead to the vanishing gradient problem, which will slow down training (Ioffe and Szegedy, 2015).

6. Number of Neurons

Similarly, to hidden layers, a low number of neurons will reduce the flexibility of the model, potentially causing a less accurate fit. Too many neurons can cause over-fitting. A range of 10-30 neurons for each layer was concurrent with the Actuarial literature on Loss Reserving. However, the literature focused on central estimate modelling, which have simpler output. In this thesis, a range of 20-100 neurons yielded the most accurate results. To streamline the process of hyper-parameter selection, all hidden layers were set to contain the same number of neurons.

7. Activation Functions

Activation functions must be non-linear, as a linear activation function is equivalent to

the weighted summation which already occurs between each layer. Furthermore, these functions will help capture non-linearities between variables. The tanh and sigmoid (see Section 2.2.1) functions are simple, common and efficient in implementation. Those two functions may suffer from gradient vanishing with deeper networks (Ioffe and Szegedy, 2015), which slows down training, hence the ReLU activation function is also common and has been demonstrated by Kuo (2019) and Rossouw and Richman (2019).

To streamline the Model Architecture selection process, early testing found that the **Sigmoid** activation function performed best, hence all further modelling used this activation.

3.2.4 Network Hyper-parameter Selection Algorithm

The MDN's architecture was selected using a manual process, which is considered the simplest in the Neural Network model selection algorithm literature (Abreu, 2019). In this thesis, Grid Search and Random Search were used to select hyper-parameters.

Grid Search: Let $\boldsymbol{\theta}$ be the set of values to test for a hyper-parameter. Grid Search tests each element in $\boldsymbol{\theta}$. Grid Search is exhaustive, thus increasing the chance of finding the best value for that hyper-parameter. However, it is time consuming (Yang and Shami, 2020).

Random Search: Let $\boldsymbol{\theta}$ be the set of values to test for a hyper-parameter. Random Search selects a random subset of $\boldsymbol{\theta}$ and tests values only in this subset. Random Search is more efficient, allowing $\boldsymbol{\theta}$ to encompass a wider range of values (Bergstra and Bengio, 2012).

Before the algorithm is run, certain aspects of the MDN's design are set constant:

- The sigmoid activation function is used for all hidden layers
- The number of neurons is equal for all hidden layers

The only hyper-parameters left to fine-tune are:

1. λ_w, the L2 weight penalty
2. λ_σ, the sigma activity penalty
3. p, the dropout rate

4. n, the number of neurons

5. h, the number of hidden layers

6. K, the number of components in the mixture density

Denote $\boldsymbol{\theta} = \{\lambda_w, \lambda_\sigma, p, n, h, K\}$ as the set of hyper-parameters to fine-tune. The hyper-parameter selection algorithm was conducted as such:

1. Start off with $\boldsymbol{\theta}^{initial} = \{0, 0, 0, n^{initial}, h^{initial}, K^{initial}\}$, a set of initial hyper-parameters deemed suitable through judgement. Setting $\boldsymbol{\theta}^{initial} = \{0, 0, 0, 60, 2, 2\}$ is recommended, as allowing the algorithm to explore unregularised models vastly improved the fit in some instances.

2. Using $\boldsymbol{\theta}^{initial}$ and keeping all other hyper-parameters fixed, use **Grid Search** to test all desired values of λ_w, the weight penalty coefficient. Select the coefficient with the lowest test error, $\hat{\lambda}_w$, and update $\boldsymbol{\theta}^1 = \{\hat{\lambda}_w, 0, 0, n^{initial}, h^{initial}, K^{initial}\}$

3. Using $\boldsymbol{\theta}^1$ and keeping all other hyper-parameters fixed, use **Grid Search** to test all desired values of λ_σ, the sigma activity penalty coefficient. Select the coefficient with the lowest test error, $\hat{\lambda}_\sigma$, and update $\boldsymbol{\theta}^2 = \{\hat{\lambda}_w, \hat{\lambda}_\sigma, 0, n^{initial}, h^{initial}, K^{initial}\}$

4. Using $\boldsymbol{\theta}^2$ and keeping all other hyper-parameters fixed, use **Grid Search** to test all desired values of p, the dropout rate. Select the rate with the lowest test error, $\hat{p}$, and update $\boldsymbol{\theta}^3 = \{\hat{\lambda}_w, \hat{\lambda}_\sigma, \hat{p}, n^{initial}, h^{initial}, K^{initial}\}$

5. Using $\boldsymbol{\theta}^3$, let $N = \{n_1, n_2, ..n_{p_1}\}$, $H = \{h_1, h_2, ..h_{p_2}\}$ and $C = \{K_1, K_2, ..K_{p_3}\}$ be the set of neurons, hidden layers and components desired for testing. Create the set $NHK = \{(n_i, h_j, K_k) : n_i \in N, h_j \in H, K_k \in C\}$. Use **Random Search** to select a subset of 24 elements from NHK. Test all 24 combinations of neurons/hidden layers/components and select the combination with the lowest test error, $(\hat{n}, \hat{h}, \hat{K})$. Update $\boldsymbol{\theta}^4 = \{\hat{\lambda}_w, \hat{\lambda}_\sigma, \hat{p}, \hat{n}, \hat{h}, \hat{K}\}$

Following the algorithm, select $\boldsymbol{\theta}^4$ as the final set of hyper-parameters and run the final model.

3.2.5 Training the Network

The SynthETIC Simulator produces Individual Claims, which are aggregated into a 40x40 triangle. It is a common procedure in Neural Network modelling to standardise the input variables, in order to stabilise the training. Through early experimentation, normalising

the output as well ($X_{i,j}$) was crucial in achieving convergence during training.

3.2.5.1 Training and Validation Error

For each hyparameter combination tested, $\boldsymbol{\theta} = \{\lambda_w, \lambda_\sigma, p, n, h, K\}$, an MDN with such hyperparameters is trained on the training set of each partition, then projected on to the testing set. For MDNs, the loss function is the Negative Log-Likelihood. During training, at each iteration, or epoch, the MDN adjusts its weights in the aim of reducing the loss function on the training set, as computed in Equation (3.37):

$$Loss(\mathbf{X}_{Train}, \hat{\mathbf{X}}_{Train}|\mathbf{w}, \lambda_w, \lambda_\sigma) = -\frac{1}{|\mathbf{X}_{Train}|} \sum_{i,j:X_{i,j}\in\mathbf{X}_{Train}} ln(f_{\hat{X_{i,j}}}(X_{i,j}|\mathbf{w})) \tag{3.37}$$

$$+ \lambda_w \mathbf{w}\cdot\mathbf{w} + \lambda_\sigma \sum_{i,j:X_{i,j}\in\mathbf{X}_{Train}} \sum_{k=1}^{K} \sigma^2_{i,j,k} \tag{3.38}$$

Through early experimentation, the Adam (ADAptive Momentum) optimiser with a learning rate of 0.001 provided the most stable training and accurate results. The MDN calculates the validation loss at each epoch, as shown in Equation (3.39).

$$Loss(\mathbf{X}_{Validation}, \hat{\mathbf{X}}_{Validation}|\mathbf{w}, \lambda_w, \lambda_\sigma) = -\frac{1}{|\mathbf{X}|_{Validation}} \sum_{i,j:X_{i,j}\in\mathbf{X}_{Validation}} ln(f_{\hat{X_{i,j}}}(X_{i,j}|\mathbf{w})) \tag{3.39}$$

$$+ \lambda_w \mathbf{w}\cdot\mathbf{w} + \lambda_\sigma \sum_{i,j:X_{i,j}\in\mathbf{X}_{Validation}} \sum_{k=1}^{K} \sigma^2_{i,j,k} \tag{3.40}$$

The validation error will tend to decrease in early iterations as the model is better fit, however, will increase later due to overfitting. The MDN cannot adjust its weights to directly minimise the validation loss, however, overfitting can be minimise by Early Stopping (Gabrielli et al., 2020), which involves stopping the model's training as soon as the validation error is minimised. It was common during training to see the validation error rise up for hundreds of epochs before going down again. Therefore, a patience of 1000 epochs was set when using Early Stopping, that is, training only stops if the Validation error hasn't reached its lowest level in the last 1000 epochs. This gives the Network room to rebound when it hits a high loss area in the parameter space during training.

3.2.5.2 Test Error

Denote $\boldsymbol{\theta}$ as the hyper-parameter values of the MDN being run. Let $\mathbf{w}_1$ and $\mathbf{w}_2$ be the final weights of the MDN when trained on the training data of the first and second partitions, respectively. In addition, let $f_{\hat{X}_{i,j}}(x|\mathbf{w}, \theta)$ be the density of $\hat{X}_{i,j}$ projected by an MDN with hyper-parameters $\boldsymbol{\theta}$ and weights $\mathbf{w}$.

Let $T1$ and $T2$ be the set of cells (i, j) in the testing set of the first and second partitions, respectively. A separate MDN is trained T times in each partition, let $\mathbf{w}_{i,t}$ be the weights of the t^{th} model trained on the i^{th} partition. The test error of the MDN with hyper-parameters $\boldsymbol{\theta}$ is calculated from Equations (3.41 - 3.43).

$$\text{TestError}(\boldsymbol{\theta}, \text{Partition 1}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i,j:(i,j)\in T1} ln(f_{\hat{X}_{i,j}}(X_{i,j}|\mathbf{w}_{1,t}, \boldsymbol{\theta})) \tag{3.41}$$

$$\text{TestError}(\boldsymbol{\theta}, \text{Partition 2}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i,j:(i,j)\in T1} ln(f_{\hat{X}_{i,j}}(X_{i,j}|\mathbf{w}_{2,t}, \boldsymbol{\theta})) \tag{3.42}$$

$$\text{TestError}(\boldsymbol{\theta}) = \frac{|T1| * \text{TestError}(\boldsymbol{\theta}, \text{Partition 1}) + |T2| * \text{TestError}(\boldsymbol{\theta}, \text{Partition 2})}{|T1| + |T2|} \tag{3.43}$$

Hence, the MDN is trained $2T$ times for each set of hyper-parameters θ, as each run will have a different weight initialisation and hence a different fit. Averaging the error of these runs reduces the impact of random weight initialisation on the performance of the hyper-parameter set $\boldsymbol{\theta}$.

3.2.6 Fitting the Final Model

Once all desired hyper-parameter combinations are tested, the combination with the lowest test error, $\boldsymbol{\theta}^{min}$ (see Section 3.2.3 for details), is set as the model architecture of choice. To produce distributional forecasts of claims in the Lower Triangle, the chosen MDN is run on the entire Upper Triangle. Only a training/validation split is needed, since the testing set was only used to compare different hyper-parameters. The training/validation partition of the Upper Triangle was done sequentially, with similar considerations for DQ2 and 3. The third partition is visualised in Figure 3.4.

An MDN with hyper-parameters $\boldsymbol{\theta}^{min}$ is fit 5 times on the training data of Partition 3, under different weight initialisations. Let w_z be the set of the MDN's final weights in the

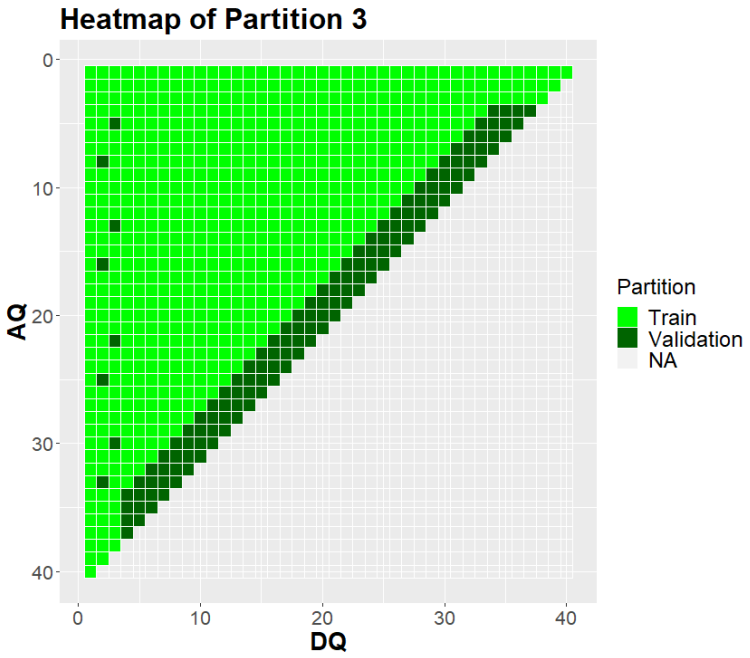


Figure 3.4: The training/validation partition of the Upper Triangle. The chosen MDN design is fit on the training data and used to project claims in the Lower Triangle

z^{th} run. Once trained, the Upper and Lower triangles are fed an input into the MDN, which produces the (α, μ, σ) output needed to form a distributional estimate, as shown in Equation (3.44).

$$MDN_\theta(w_z) : (i,j) \overset{w_z}{\longmapsto} (\alpha_{i,j}, \mu_{i,j}, \sigma_{i,j}), \tag{3.44}$$

which forms the distribution for $\hat{X}_{i,j}$, as shown in Equation (3.45):

$$f_{\hat{X}_{i,j}}(x|w_z) = \sum_{k=1}^{K} \alpha_{i,j,k}^{w_z} \phi(x|\mu_{i,j,k}^{w_z}, \sigma_{i,j,k}^{w_z}) \tag{3.45}$$

Running the MDN 5 times with different weight initialisations yields an ensemble of models, which are then averaged to produce the distribution of incremental claims shown in Equation (3.46).

$$f_{\hat{X}_{i,j}}(x) = \frac{1}{5} \sum_{z=1}^{5} \sum_{k=1}^{K} \alpha_{i,j,k}^{w_z} \phi(x|\mu_{i,j,k}^{w_z}, \sigma_{i,j,k}^{w_z}) \tag{3.46}$$

Mean and variance estimates follow from the equations in Section 3.1.4, except with 5 times the number of distributions.

3.3 Model Evaluation

The final MDN model is fit on the Lower Triangle and compared to the ccODP model. The results of two main variables were analysed:

1. Individual cells, $X_{i,j}$
2. Total reserves, $R = \sum_{i,j:i+j>41} X_{i,j}$

The measure of total reserves, R, is used directly in reserving. Capital standards set by APRA and Solvency II require reserve allocations to meet total reserves in the Lower Triangle with a 75% and 99.5% probability of sufficiency, respectively. However, it is important for a model to achieve accurate total reserves by correctly modelling the individual cells, $X_{i,j}$. A model can over-estimate claims for part of the Lower Triangle, but under-estimate for others, yielding accurate total reserves and misleading the user as to the model's predictive accuracy. Such a model may not be so fortunate in other environments, hence it is important for a model to excel primarily in accurate forecasting of individual cells.

Results were analysed qualitatively and quantitatively. Sections 3.3.2 and 3.3.3 provide further detail. The use of qualitative analysis, primarily graphical plotting, provide a detailed picture of the model's ability to capture the main trends of the data and produce reasonable forecasts, serving as a sanity test of the model. Quantitative tests, such as the Root Mean Squared Error (RMSE) and Log Score, allow for more objective evaluation of different models.

3.3.1 Benchmark - Cross-Classified Over-Dispersed Poisson Model

The benchmark model was the Cross-Classified Over-Dispersed Poisson model, or the ccODP, as it's a popular model in Loss Reserving. The ccODP is simplistic, **providing the same central estimates as the Chain Ladder**. Furthermore, a vast amount of the Neural Network Loss Reserving literature uses the Chain Ladder or ccODP as its benchmark (Kuo, 2019; Gabrielli et al., 2020; Gabrielli, 2019; Delong et al., 2020; Kuo, 2020; Wüthrich, 2018b).

The ccODP model assumes that Incremental Claims, $X_{i,j}$, follow a Cross-Classified

Over-Dispersed Poisson distribution, as Equation (3.47) shows.

$$\frac{\hat{X}_{i,j}}{\phi} \sim \text{Poi}\left(\frac{A_i B_j}{\phi}\right) \tag{3.47}$$

The ccODP model produces the following mean and volatility estimates:

- $E[\hat{X}_{i,j}] = A_i B_j$
- $Var(\hat{X}_{i,j}) = \phi A_i B_j$

The Cross-Classified structure of the model, as well as the assumption of a constant ϕ, leads it to produce mean estimates that are identical to the Chain Ladder. Hence, the ccODP's shortcomings in central estimate accuracy follow from the Chain Ladder's shortcomings:

- The ccODP assumes that claim development is homogeneous across Accident periods. Hence, it will not be able to capture Calendar period based changes, such as an inflation shock or a change in legislation with affects claim processing speed.
- The ccODP assumes the Over-Dispersed Poisson distribution for claims, which is a popular assumption. However, the volatility estimates are a function of the central estimates, meaning that where central estimates are inaccurate, it is likely that volatility estimates will be as well.

See the Appendix A.2 for details on how the ccODP model is fit.

3.3.2 Qualitative Analysis

Central Estimates: Plots of the MDN and ccODP's central estimates $\hat{\mu}_{i,j}$ will be compared to actual losses of the dataset $X_{i,j}$, as well the empirical mean calculated from hundreds of simulations of the same dataset. To be successful, the MDN's central estimates should be closer than the ccODP's estimates to actual losses and the empirical mean of Incremental Claims.

Risk Margins: Plots of the MDN and ccODP's mean-centred risk margins (at the 25%, 75% and 95% level) will be compared to empirical risk margins. The model with closer margins to the empirical results will have captured the distribution shape of Incremental Claims more accurately.

Quantile Estimates: Plots of the MDN and ccODP's quantile estimates (at the 25%,

75% and 99.5% level) will be compared to empirical quantiles. The model with closer quantile estimates, $\hat{X}_{i,j,q}$ for $q = 0.25, 0.75, 0.995$, to the empirical results will have captured them more accurately.

Total Reserves: The distributions of total reserves estimated by the MDN and ccODP, $\hat{R}$, will be plotted alongside the empirical distribution of total reserves. The bias, dispersion and quantile estimates will be analysed visually. The MDN, to be successful, should have a lower bias than the ccODP, as well as dispersion estimates that is closer to the empirical distribution.

3.3.3 Quantitative Analysis

When analysing individual cells, $X_{i,j}$, several statistics will be calculated to each Loss Triangle involved in the modelling. For total reserves, R, the MDN and ccODP will be fit on **10 triangles for each of the four data environments**, to generate reserve estimates for each triangle, $\hat{R}_i$ for $i = 1, 2, 3..10$. Let $\mathbf{X} = \{X_{i,j} : i + j > 41\}$, $\mathbf{R} = \{R_i, i = 1, 2, 3, ..10\}$ and $f_{\hat{\mathbf{X}}} = \{f_{\hat{X}_{i,j}} : X_{i,j} \in \mathbf{X}\}$. The quantitative metrics used are calculated as follows:

1. Distributional forecast accuracy, using the Log score metric (Equation 3.48):

$$LogScore(\mathbf{X}, f_{\hat{\mathbf{X}}}) = \frac{\sum_{(i,j):X_{i,j}\in\mathbf{X}} ln(f_{\hat{X}_{i,j}}(X_{i,j}))}{|\mathbf{X}|} \tag{3.48}$$

This formula is equivalent to the Likelihood function; a higher Log Score is desirable as it indicates a more accurate distributional fit for the Lower Triangle. The Log Score wasn't calculated when analysing total reserves, as the fitted distributions usually fell completely outside the simulated empirical distribution, setting the Likelihood to 0.

2. Central estimate forecast accuracy, using the MSE metric (Equation 3.49, 3.50):

$$RMSE(\mathbf{X}, \hat{\mathbf{X}}) = \sqrt{\frac{\sum_{(i,j):X_{i,j}\in\mathbf{X}} (X_{i,j} - \hat{X_{i,j}})^2}{|\mathbf{X}|}} \tag{3.49}$$

$$RMSE(\mathbf{R}, \hat{\mathbf{R}}) = \sqrt{\frac{\sum_{i=1}^{10} (R_i - \hat{R}_i)^2}{10}} \tag{3.50}$$

A lower RMSE indicates more accurate central estimates for the Lower Triangle and total reserves.

3. Quantile forecast accuracy, using Quantile Scores (Equation 3.51, 3.52): The 75% and 99.5% quantiles are analysed quantitatively, using the **quantile scoring function**. Let $\hat{X_{i,j,q}}$ be the q^{th} quantile estimate of $X_{i,j}$ and $\hat{R}_{i,q}$ be the q^{th} quantile estimate for total reserves R_i. The Quantile Score is calculated as follows:

$$QS(\hat{\boldsymbol{X}}_{\boldsymbol{q}}, \mathbf{X}) = \frac{\sum_{(i,j):X_{i,j}\in\mathbf{X}}(\mathbf{1}(X_{i,j} < \hat{X}_{i,j,q}) - q)(X_{i,j} - \hat{X}_{i,j,q})}{|\mathbf{X}|} \tag{3.51}$$

$$QS(\hat{\boldsymbol{R}}_{\boldsymbol{q}}, \mathbf{R}) = \frac{\sum_{i=1}^{10}(\mathbf{1}(X_{i,j} < \hat{X}_{i,j,q}) - q)(X_{i,j} - \hat{X}_{i,j,q})}{10} \tag{3.52}$$

A lower Quantile Score indicates more accurate quantile estimates, for individual cells and total reserves. The score is a weighted sum of residuals. The weights are $1 - q$ or q, depending on whether the actual loss is higher or lower than the quantile estimate.

3.3.4 Objectives

This thesis proposes the MDN as a Neural Network that specialises in distributional forecasting and achieves accurate results when applied to Loss Triangle reserving. The MDN's success depends on its performance relative to the ccODP, in terms of central estimates, distributional and quantile estimate accuracy.

Following the research aims for this thesis, if the MDN model achieves the following objectives in the Lower Triangle, when compared to the ccODP fit, it will be considered a success:

- Produce a higher log score, lower RMSE and lower quantile scores for the 75% and 99.5% levels, demonstrating more accurate distributional and central estimate forecasting, in **most** triangles for each of Datasets 1,2,3 and 4 (see Section 4.2).
- Produce stable claims forecasts, especially for the volatile Dataset 4 (see Section 4.2.4).
- For Datasets 2,3 and 4, the MDN produces total reserve estimates which have a lower RMSE and lower Quantile Scores. Furthermore, the bias and dispersion estimates of total reserves must be closer to the empirical distribution of R in the majority of triangles run for each claim environment. The ccODP is naturally suited to performing well on the simplistic Dataset 1, hence no requirements are set for the

MDN to outperform it in that setting.

These goals, which the MDN achieved, will increase the reliability and applicability of Neural Networks in reserving. The literature is at an early stage of applying Neural Networks in this field, however, with benchmarks only being set against relatively simplistic traditional models such as the Chain Ladder.

CHAPTER 4

DATA ANALYSIS

This chapter analyses the data being used for this thesis. Section 4.1 analyses the benefits and drawbacks of using real and simulated data. Section 4.2 goes through the data simulation methodology, with an explanation of the features of each dataset simulated and used in this thesis.

4.1 Simulated vs Real Data

Two main sources of data exist: real and simulated data. Individual Claim simulators have arisen out of the accommodation of large datasets by modern computers and the lack of publicly available Individual Claims data, which has hampered the development and testing of Individual Claims models (Gabrielli and Wüthrich, 2018). While this thesis will focus on Loss Triangle reserving, the existence of a granular simulator will allow more flexibility in generating ample data for better estimating the empirical mean and volatility of claims, as well as testing consistency and robustness through generating multiple triangle under different claim process environments.

A pitfall of simulating claims, as noted by Murray et al. (2011) is that complex interactions in real data may not be captured by the simulator, 'hindering model development'. Fitting a model on unrealistic data will reduce its validity.

4.1.1 Simulator: SynthETIC (2020)

This thesis uses simulated data for modelling, using the SynthETIC claims simulator developed by Avanzi et al. (2020). This simulator generates Individual Claims following a model which isn't calibrated to data but is determined explicitly by the user. The following variables are determined as such:

- Maximum development period, I, is of arbitrary scale and determined by the user. Given user-defined business exposure and claim frequency, E and λ, the number of claims for the Accident period is Poisson distributed with mean $E * \lambda$ and are assorted in the specified period according to a Uniform distribution.
- The total claim size of each Individual Claim, along with reporting and settlement lags, are sampled through a user-defined distribution. Reporting and settlement lag distribution inputs are dependent on claim occurrence and size.
- The number, size and distribution of partial payments are simulated from a user-specified distribution which is dependent on claim size.
- Claims are adjusted based on a superimposed inflation index.

Advantages

- Allows high aggregation, as claim intervals can be defined freely by the user.
- Highly flexible claim simulation conditions, as distributions are user-specified, which allows many different business scenarios to be simulated.
- Allows for dependencies between reporting lag, settlement lag, partial payments and claim size.

Disadvantages

- Simulator does not calibrate to a given dataset, thus requiring a strong understanding of claim dynamics to specify realistic distributions, or a separate modelling process which calibrates to an external claims dataset.

4.2 Simulated Datasets

The size of each triangle simulated in this thesis is 40x40. Four different claim environments were simulated, of various features and complexities. The environments, or

Datasets, were:

1. **Dataset 1:** Simple, short tail claims
2. **Dataset 2:** Gradual shift from long tail to short tail claims, that is, an increase in claims processing speed
3. **Dataset 3:** Inflation shock
4. **Dataset 4:** Highly volatile claims

Simulating different environments was done to test the MDN's versatility and ability to capture complex trends and produce accurate forecasts in a variety of challenging environments. **For each Dataset, 10 independent triangles were simulated**, meaning that the MDN is fit on 40 triangles in total. As a Loss Triangle is a random variable, it is important to run the MDN on a large sample of triangles to gain a better understanding of its accuracy and also test its ability to provide consistent results.

For each Dataset, the dynamics that were kept constant were the number, amount and time distribution of the partial payments of each individual claim. The variables altered were the claim size distribution, reporting delay mean and coefficient of variation, the settlement delay mean and coefficient of variation and the Superimposed inflation occurrence and payment factors. The reporting and settlement delays followed a Weibull distribution. See Appendix A.3 for a technical methodology of the simulation of each Dataset.

4.2.1 Dataset 1: Simple, Short Tail Claims

This dataset includes simple, short tail claims, homogeneous in composition for all Accident Quarters. The reporting and settlement delays have been approximately calibrated to show similar characteristics to the simulator developed by Gabrielli and Wüthrich (2018). Figures 4.1 and 4.2 plot the Incremental Claims for Dataset 1:

What complicated this dataset is the peak in claims observed in Development Quarter 2 (DQ2). The assumption of reporting and settlement delays following a Weibull distribution plays a part in this peak. Despite this complexity, datasets of this nature are common. Another feature in this Dataset is a constant inflation of 2% per annum. With Incremental claims having low noise and almost perfect homogeneity, the ccODP forecasted OSC almost perfectly in this Dataset. Hence, this Dataset is a preliminary test to the feasibility

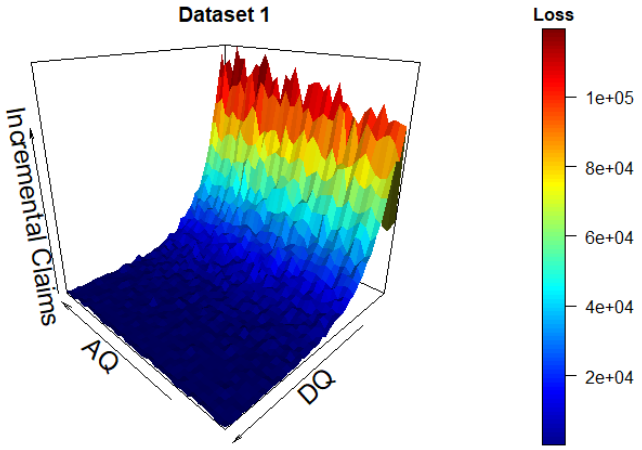


Figure 4.1: A colour-coded 3D plot of Incremental claims for Dataset 1

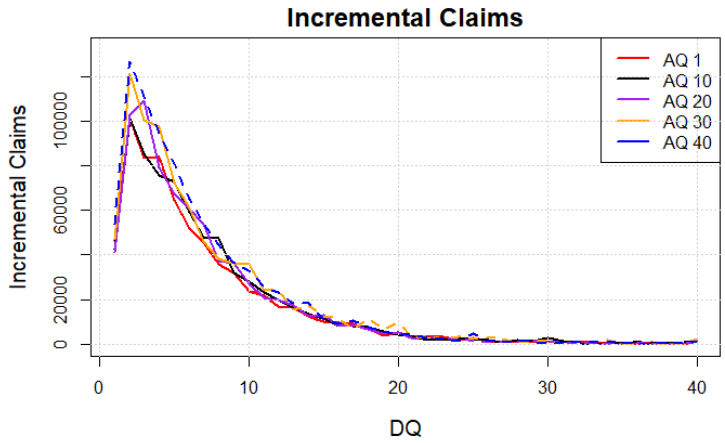


Figure 4.2: A plot of the Incremental Claims of Dataset 1, for selected Accident Quarters. Solid lines represents data in the Upper Triangle, while dashed lines represent data in the Lower Triangle

of MDNs in modelling 40x40 triangles and producing reasonable results.

4.2.2 Dataset 2: Shift from Long Tail to Short Tail Claims

This Dataset introduces complexities which the Chain Ladder should fail to capture. Initially, there are more long tail claims, however, the proportion of these claims decreases, while the proportion of short claims increases. From the Incremental Claim plot shown below, later AQs see higher losses early and less losses later on, due to the increasing proportion of short tail claims. Inflation is at a constant 2% per annum, while a 3% constant SI is compounded. Figures 4.3 and 4.4 plot the Incremental Claims:

In this Dataset, the claim characteristics gradually shift in nature. This systematic volatility is seen is two constrasting ways. Firstly, for DQ10 and less, claims increase

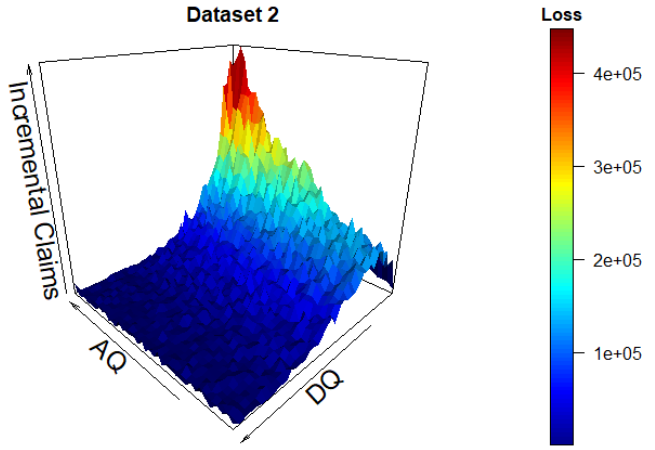


Figure 4.3: A colour-coded 3D plot of Incremental claims for Dataset 2

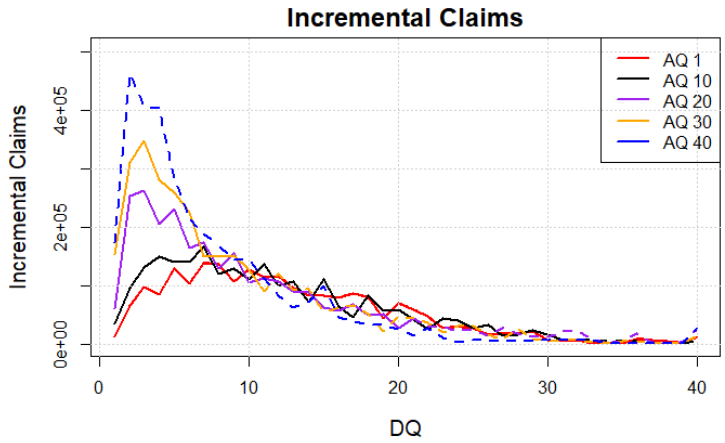


Figure 4.4: A plot of the Incremental Claims of Dataset 2, for selected Accident Quarters. Solid lines represents data in the Upper Triangle, while dashed lines represent data in the Lower Triangle

as the AQ increases. After DQ10, claims decrease as the AQ increases. The shortening of claims explains both these trends.The main question to be answered in testing this dataset is, given the systematic volatility in the claims data, can the MDN accurately distinguish between systematic and unsystematic volatility and capture the distribution of data points accurately? That is, will the MDN learn that claims are getting shorter, or will it attribute the trend to noise?

4.2.3 Dataset 3: Inflation Shock at Calendar Quarter 30

In this Dataset, Superimposed inflation is changed instantly from 0% to 8& per annum, starting at AQ30. The 8% inflation remains constant in the Lower Triangle. This Dataset tests the ability of the MDN to recognise changes in Calendar effects and adapt projections accordingly. Only the last 10 calendar quarters in the Upper Triangle contain information

regarding the inflation shock, which increases the difficulty for the MDN. Furthermore, the Rolling Origin validation method has lower training exposure to data in the latest Calendar Quarters, especially in the first partition. Hence, this Dataset tests whether the Rolling Origin method can still make use of the latest Quarters and capture the inflation shock effectively. Figures 4.5 and 4.6 plot the Incremental Claims:

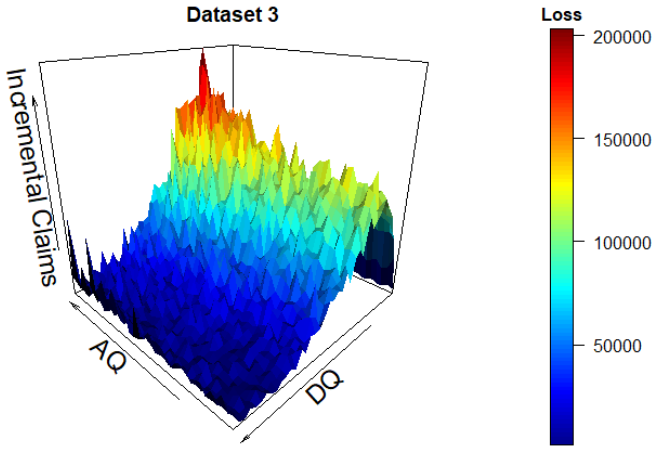


Figure 4.5: A colour-coded 3D plot of Incremental claims for Dataset 3

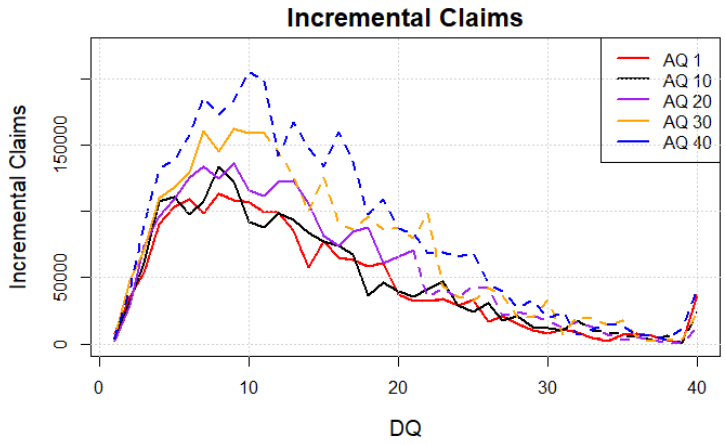


Figure 4.6: A plot of the Incremental Claims of Dataset 3, for selected Accident Quarters. Solid lines represents data in the Upper Triangle, while dashed lines represent data in the Lower Triangle

4.2.4 Dataset 4: High Volatility

This dataset is the default triangle generated by the SynthETIC simulator. The volatility of losses are incredibly high. Claims are very low in frequency, but follow a power-Normal distribution with power 0.2, which results in a very volatile severity. As claim size increases, reporting delay decreases, while settlement delay increases on average. Small claims settle slightly faster as they occur later, although that effect stops at AQ21. After

AQ20, claims are multiplied by a factor which is 0.6 for zero claims and rises to 1 as claim size increases. Superimposed inflation is at 30% per annum, however that rate is multipled by a factor which decreases as claim size increases. These additional trends exacerbate the volatility which was already present. It is normal for claims in one AQ to follow a completely different pattern (reporting, settlement, volume, development pattern) than claims in the adjacent AQ. One trend is prevalent in thi dataset; the slow development of claims and high superimposed inflation. Figures 4.7 and 4.8 plot the Incremental Claims:

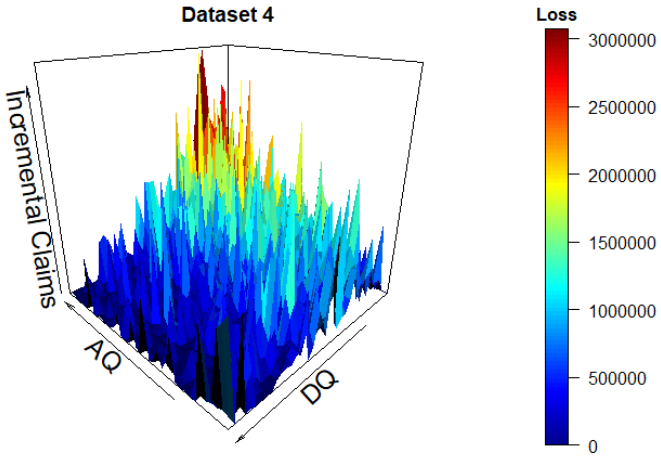


Figure 4.7: A colour-coded 3D plot of Incremental claims for Dataset 4

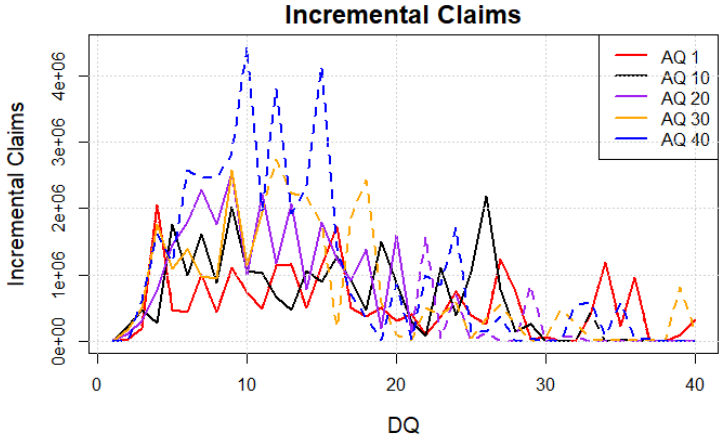


Figure 4.8: A plot of the Incremental Claims of Dataset 4, for selected Accident Quarters. Solid lines represents data in the Upper Triangle, while dashed lines represent data in the Lower Triangle

CHAPTER 5

RESULTS

This chapter outlines the results obtained from modelling with the MDN and ResMDN in this thesis, comparing results to the ccODP model. Section 5.2 analyses the impact that the Rolling Origin method had on delivering smooth, robust and stable forecasts with the MDN. Section 5.3 analyses the MDN's performance when it came to probabilistic forecasting, with a constant benchmarking to the ccODP. Section 5.4 analyses the ResMDN's performance, compared to the ccODP and MDN. The costs of interpretability are analysed based on the results obtained from modelling.

5.1 Stable Forecasts - Rolling Origin Model Validation

The Rolling Origin Model Validation method served two purposes in this thesis:

1. To partition the Upper Triangle into training, validation and testing sets. This allowed different models and hyper-parameter combination to be tested and selected based on their accuracy in the test set.

2. To specifically assess different models based on their projection accuracy, which increases the chance of selecting a model which projects accurately into the Lower Triangle. This is done through the sequential partitioning of data, such that the

testing data is comprised of the latest calendar quarters.

Generally, the set of hyper-parameters selected by the Rolling Origin method produced very reasonable and accurate central and distributional forecasts. All models were successful in predicting a decrease in the mean and volatility of claims in the later Development Quarters (DQs), which is a significant achievement given the low number of data available to the MDN in those periods. Figure 5.1 plots the MDN's mean and volatility estimates on Dataset 2, showing the accuracy of projections given its systematic complexity.

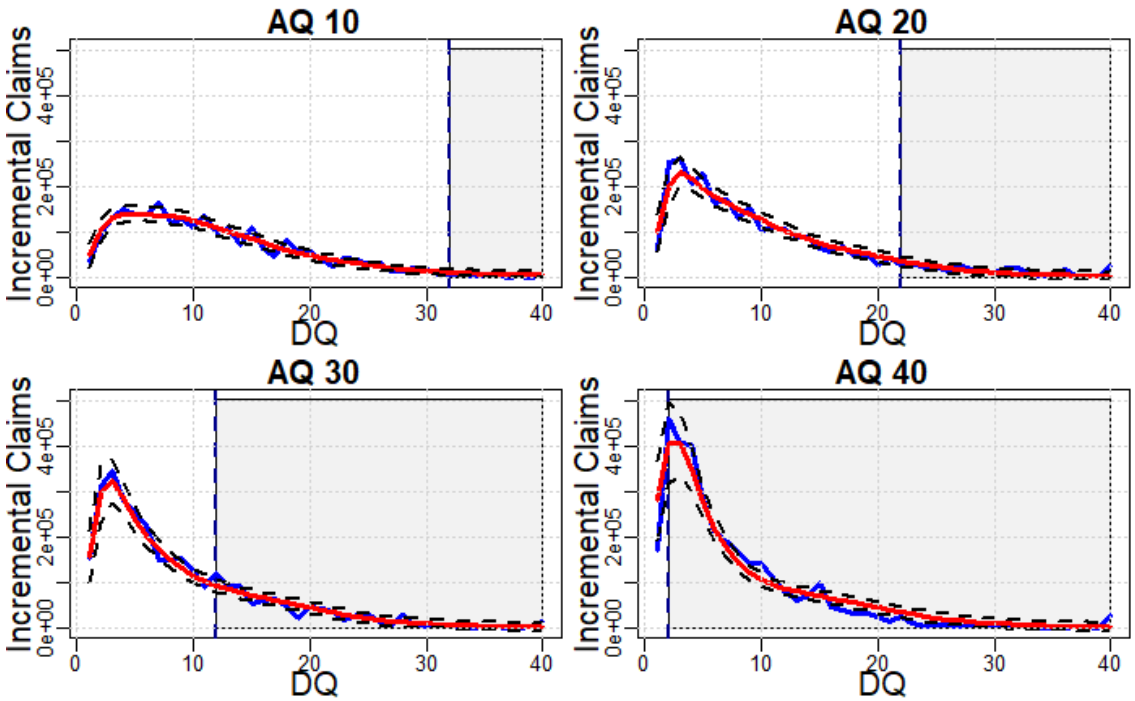


Figure 5.1: Dataset 2: Plots of the overall fit of the MDN. Blue represents actual losses, red is the MDN's central estimate, with the black dashes representing the MDN's one standard deviation margin. The grey area represents the Lower Triangle, the forecasting region.

The MDN also produced robust predictions. This can especially be seen in Figure 5.2, which plots the MDN's and ccODP's fit to the highly volatile Dataset 4. As the cell at $(40, 1)$ is equal to 0, the ccODP predicted 0 claims for the rest of AQ40, while the MDN maintained mean and volatility estimates consistent with other AQs. This shows that where the ccODP's fit can be overly dependent on select data points, the MDN produces a more holistic fit, as the data is input numerically, not categorically, into the model.

The MDN's robustness can be partially attributed to its smooth mean and volatility fit in both the Upper and Lower Triangles. The Rolling Origin method, in the third partition, uses the latest Calendar Quarters for validation. A model that overfits the

data won't project accurately, hence the MDN is encouraged to produce smoother and more robust fits. The smoothness can be visualised in Figure 5.2 as well, where the MDN produces a significantly smooth fit despite the huge volatility present in the dataset.

The Rolling Origin Model Validation method proved successful at partitioning a 40x40

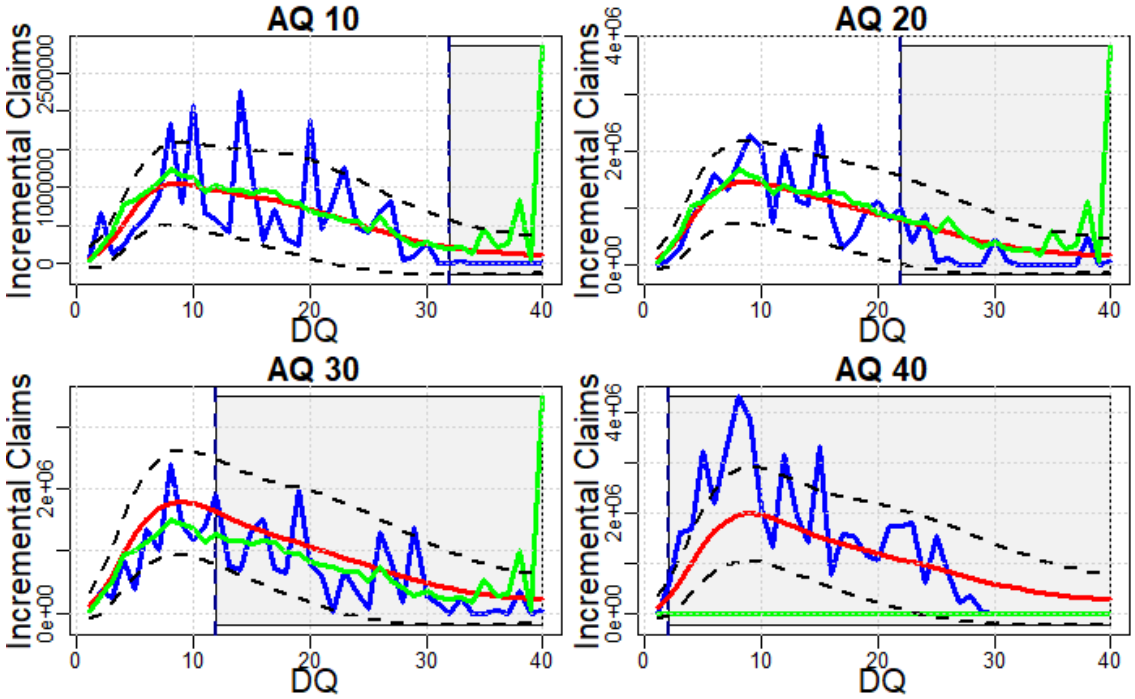


Figure 5.2: Dataset 4: Plots of the overall fits of the MDN and ccODP models. Blue represents actual losses, red is the MDN's central estimate, with the black dashes representing the MDN's one standard deviation margin. The green line is the ccODP's central estimate. The grey area represents the Lower Triangle, the forecasting region.

triangle. The scarcity of data relative to the large datasets Neural Networks are used to would discourage the use of the Rolling Origin method. However, both the MDN and Rolling Origin partition performed excellently on 820 points, making the MDN more appropriate in a practical setting.

5.2 Probabilistic Forecasting with the Mixture Density Network

The Mixture Density Network (MDN), overall, outperformed the ccODP for all Datasets in all qualitative and quantitative metrics. When analysing both Individual Cells and total reserves, the MDN outperformed the ccODP, often in a decisive manner.

5.2.1 Central Estimate Analysis

The MDN produced more accurate central estimate projections than the ccODP in all Datasets. Where the Dataset had more structural heterogeneity, the MDN decisively outperformed in all metrics. The MDN's central estimates were smooth and always positive, which is a significant achievement given that the input and output data was normalised. This is further evidence towards the accuracy of its projections.

In addition to its smooth fit, the MDN showed great modelling power and flexibility, capturing all trends simulated in the four different claim environments. In Dataset 2, the MDN successfully learned that claims processing speed is increasing, predicting a sharper spike in claim payments in the later AQs. Figure 5.3 plots the results for this dataset. The ccODP, assuming homogeneity in claim development, ended up approximating claims to be middle-tailed, leading to a clear over-estimation of claims in later AQs.

For Dataset 3, the MDN accurately captured the inflation shock at Calendar Quarter 30

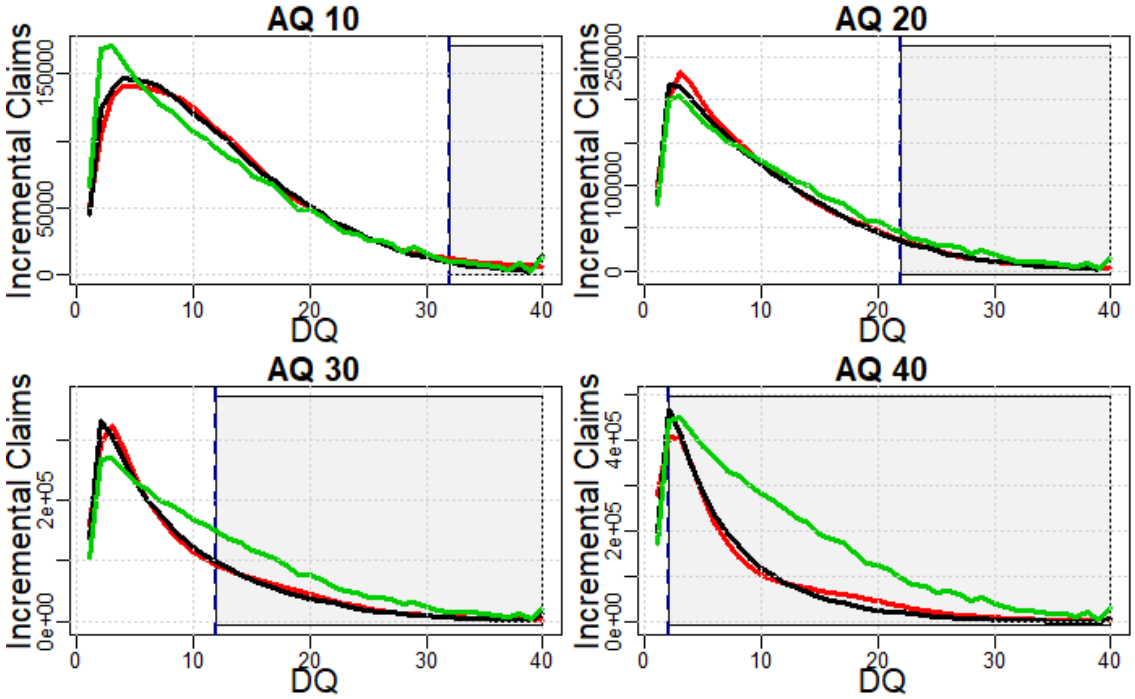


Figure 5.3: Dataset 2: Plots comparing the mean estimates of the MDN and ccODP models to the empirical mean claims based on 250 simulations. The red line represents the MDN's central estimate, the green line represents the ccODP's central estimate, while the black line represents the empirical mean of claims. The grey area represent the Lower Triangle.

(CQ30) onwards. Figure 5.4 plots the results. The ccODP did not keep up with the increased inflation due to its inability to handle heterogeneity, leading to it under-estimating

claims from CQ30 onwards.

Generally, the MDN's out-performance relative to the ccODP is due to its high flexibility,

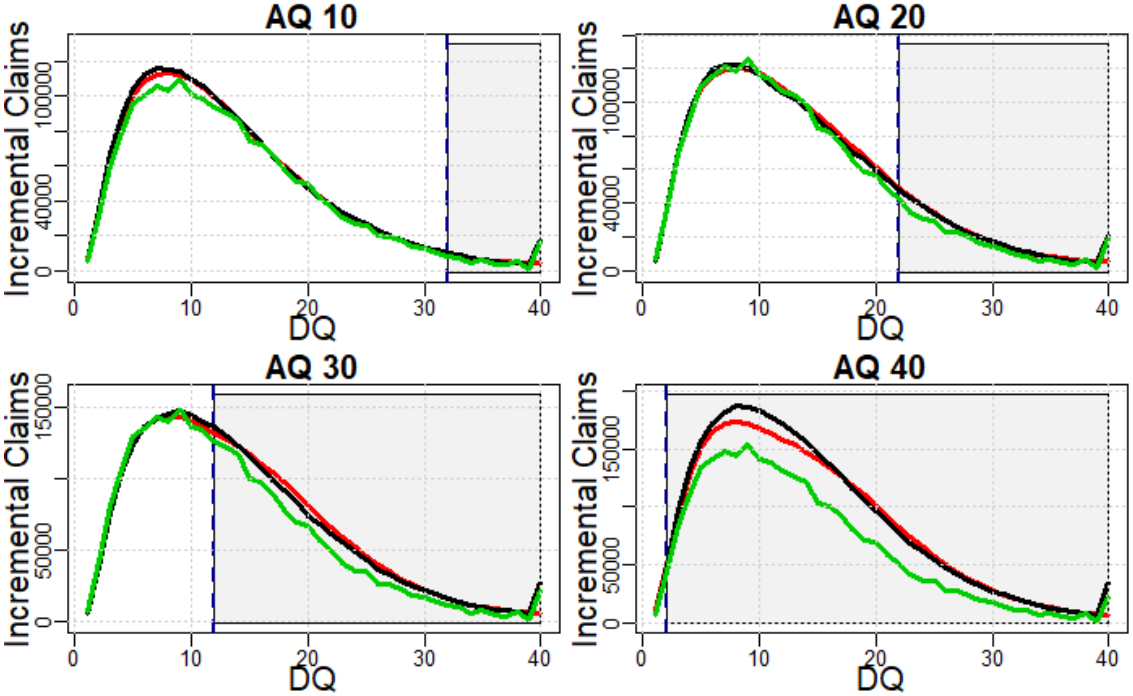


Figure 5.4: Dataset 3: Plots comparing the mean estimates of the MDN and ccODP models to the empirical mean claims based on 500 simulations. The red line represents the MDN's central estimate, the green line represents the ccODP's central estimate, while the black line represents the empirical mean of claims. The grey area represent the Lower Triangle.

being able to fit a highly flexible function to the data and capture non-linearities. This flexibility can easily lead to over-fitting, but the Rolling Origin Model Validation method ensured that the MDN's flexibility was enough to capture the relevant trends in the data while minimising over-fitting. Figure 5.5 provides boxplots of the MDN's % reduction of the RMSE relative to the ccODP for 10 triangles in each of Datasets 1,2,3 and 4. The boxplots show that the MDN achieved a positive reduction in the RMSE (lower RMSE) relative to the ccODP for the majority of triangles in each Dataset.

Despite the MDN's success, it showed weaknesses in several areas, which were fixed:

- The MDN can struggle to capture systematic spikes in claims. Such a spike was seen in AQ40, DQ2 for Dataset 2, in which the increased speed in claims processing led to an unprecedented spike in claims early on in later AQs. The MDN, not being given enough data to decipher that specific trend, failed to accurately capture the peak, instead under-estimating claims and over-estimating volatility. This reveals

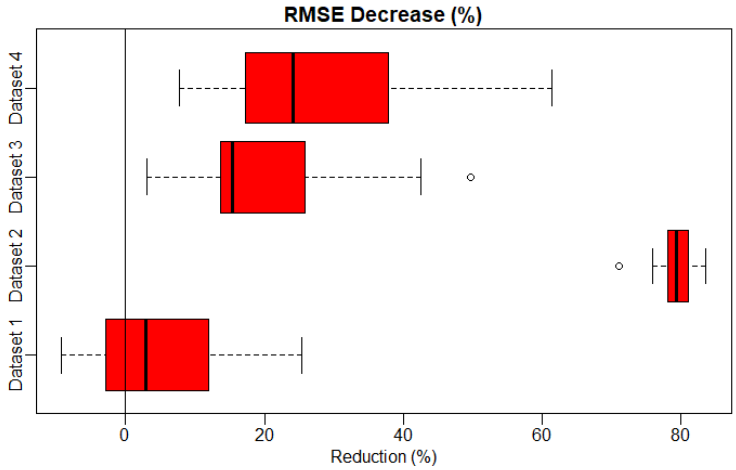


Figure 5.5: Boxplots displaying the MDN's % reduction in the RMSE relative to the ccODP for each of the 10 triangles run for Dataset 1,2,3 and 4. A positive Reduction indicates the MDN had a lower RMSE than the ccODP for a specific triangle.

a dynamic between central and volatility estimates. Where the central estimate is off, the MDN will raise volatility estimates to increase the likelihood. This dynamic exists when using the NLLLoss. This issue was solved by adding an MSE term to the Loss to encourage more accurate central estimates. It was successful in helping the MDN capture sharper spikes in claims. This shows that, using just the NLLLoss, the MDN can struggle to capture accurate central estimate for sharp systematic fluctuations in Incremental Claims.

- The Rolling Origin Model Validation method did not accurately capture trends embedded in the later Calendar Quarters. In Dataset 3, the MDN failed to accurately capture the inflation shock, as it wasn't trained on much data where the shock took effect. The first partition, especially, is barely trained or validated on the later Calendar Quarters, which contain crucial information for projection. To mitigate this issue, the data partitioning was performed differently. While the validation and testing data were given later quarters as a priority, training points were also taken in that period in a random sample to ensure the MDN was made aware of the inflation trend. See the Appendix B.3 for details on the data partition. This new partition achieved more accurate projections, capturing the inflation shock much more accurately.

- Fitting a Mixed Log-Normal on volatile data with many 0 cells will lead to unreasonably high mean and volatility estimates. This phenomenon was observed with Dataset 4, which features high volatility and many 0 cells in the later DQs. As a result, the MDN estimated a very high volatility rate for the later DQs, leading to high mean and volatility estimates when the data was converted to its raw form.

This issue was solved by smoothing the log claims from DQ30 onwards, reducing its volatility and stabilising projections. This shows that when dealing with highly volatile log data, smoothing results is necessary to achieve reasonable results.

The MDN also showed small weaknesses which are worth addressing for completeness of their analysis:

- The MDN didn't attempt to capture the spike in claims at DQ40. Removing the (1,40) cell didn't visibly impact results.
- Generally, the MDN slightly over-estimated claims in the later DQs, due to low amounts of data in that region. Where there is low data, the MDN is less swayed by it, hence the central estimates in the later DQs is still influenced by the higher central estimates from earlier DQs.
- The MDN struggled to capture high superimposed inflation. This is seen in dataset 4, which has approximately 30% SI, where the MDN continually underestimated claims in later AQs and earlier DQs. Inflation effects really show in the later AQs, but the MDN doesn't capture them as the data in that region is in small numbers.

5.2.2 Volatility Estimate Analysis

The Mixture Density Network produced very smooth and accurate volatility estimates. Where noise in the data was low, the MDN projected low volatility, and vice versa. Overall, it outperformed the ccODP qualitatively and quantitatively when it came to estimating the volatility of Individual Cells.

The MDN's risk margin estimates at the 25th and 75th percentile were more accurate overall than the ccODP's margins in almost all environments tested, an indicator that it captured volatility more accurately. The only exception was Dataset 1, which due to its simplicity, the ccODP performed almost perfectly in all regards. The ccODP's variance is a function of its mean, hence it failed where the central estimates failed. For example, in Dataset 2, the ccODP over-estimates claims in later AQs, which led to it over-estimating margins in that same period. In Dataset 3, the ccODP under-estimated claims in later Calendar Quarters (CQs), as it didn't effectively capture the inflation shock. This led to volatility estimates also being too low in those periods, as Figure 5.6 illustrates.

While the Neural Network Loss Reserving literature does fit stochastic models, the MDN's accuracy in estimating volatility justified the usage of a Neural Network that specif-

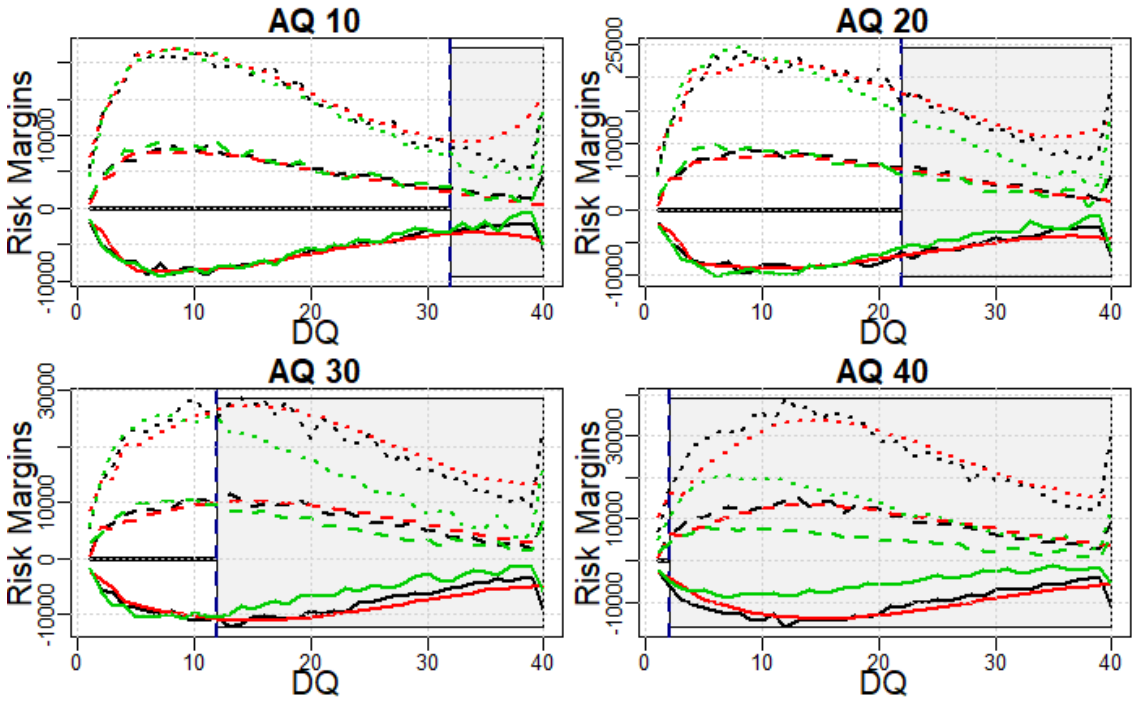


Figure 5.6: Dataset 3 (Inflation Shock): Plots comparing the 25%, 75% and 95% risk margin estimates of the MDN and ccODP models to the empirical margins based on 500 simulations. The red line represents the MDN's margin estimates, the green line represents the ccODP's margin estimate, while the black line represents the empirical margins. The solid lines, dashed and dotted lines represent the 25%, 75% and 95% margins, respectively. The grey area represent the Lower Triangle.

ically focuses on distributional forecasting. The MDN's out-performance in estimating volatility compared to the ccODP was due to its higher flexibility in modelling trends in the data, as well as more flexibility in the parametrisation of the Mixed Gaussian. While some correlation was found in the results between central and volatility estimates, the Mixed Gaussian places no dependence on these values, allowing the MDN to fit a wider range of distributions than the ccODP. Figure 5.7 provides boxplots of the MDN's increase in the Log Score relative to the ccODP for 10 triangles in each of Datasets 1,2,3 and 4. The boxplots show that the MDN achieved a higher Log Score relative to the ccODP for the majority of triangles in each Dataset, indicating a more accurate probabilistic forecast.

With the MDN's success, there are some weaknesses which must be addressed:

- The MDN still showed signs of attributing noise to systematic trends. In Dataset 2, even though the DQ2 spike was fixed, the volatility was still too high for AQ30 and AQ40. This is also because of the DQ1 'trough' not being captured accurately, leading to high volatility estimates which have influenced estimates in DQ2. Figure 5.8 plots the risk margin estimates for Dataset 2, where the claim processing speed

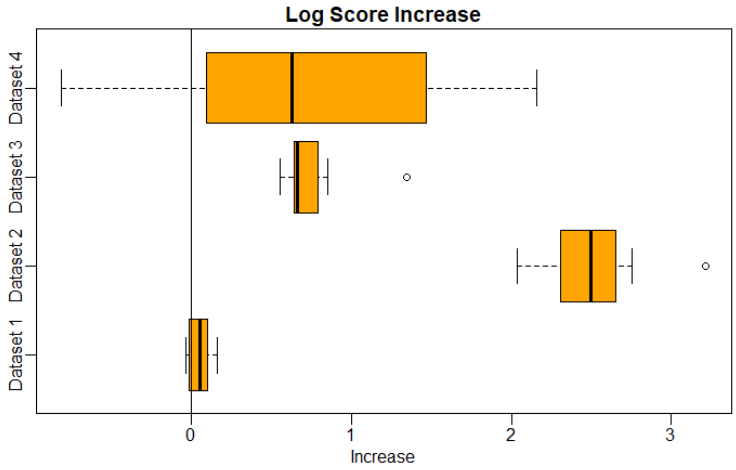


Figure 5.7: Boxplots displaying the MDN's increase in the Log Score relative to the ccODP for each of the 10 triangles run for Dataset 1,2,3 and 4.

gradually increases.

- Similar to central estimates, the MDN often over-estimates volatility in later DQs. Due to a lack of data in that region, the volatility estimates are influenced by estimated of earlier DQs, which are higher. This wasn't fixed by constraining mean projections for Dataset 2.

5.2.3 Quantile Estimate Analysis

The MDN provided more accurate 75% and 99.5% quantiles for all Datasets in the majority of triangles run for each Dataset. These results follow from the MDN's ability to provide more accurate central estimates and volatility estimates. The quantile analysis was mainly quantitative, using the Quantile Scores. Figure 5.9 plots the 99.5th quantile estimates for the MDN and ccODP models for individual cells, using the empirical 99.5th quantiles as a measure of the real value. The figure shows that the MDN yields 99.5% quantile estimates that are much closer to the empirical quantiles than the ccODP, confirming that the MDN has modelled the mean and distribution of Incremental Claims more accurately.

The ccODP often gave low mean and volatility results in later DQs, leading to quantile estimates that were lower than actual losses for that cell. This penalised the quantile loss heavily. The MDN's robustness and smoothness helped it avoid forecasting such under-dispersed distributions. Figure 5.10 provides a boxplot of the MDN's percentage reduction in the 75% and 99.5% quantile scores relative to the ccODP for each triangle in each Dataset. The plot confirms that in all Datasets, the MDN reduces the 75% and 99.5% Quantile scores for the majority of triangles, indicating more accurate quantile estimates at the 75% and 99.5% levels.

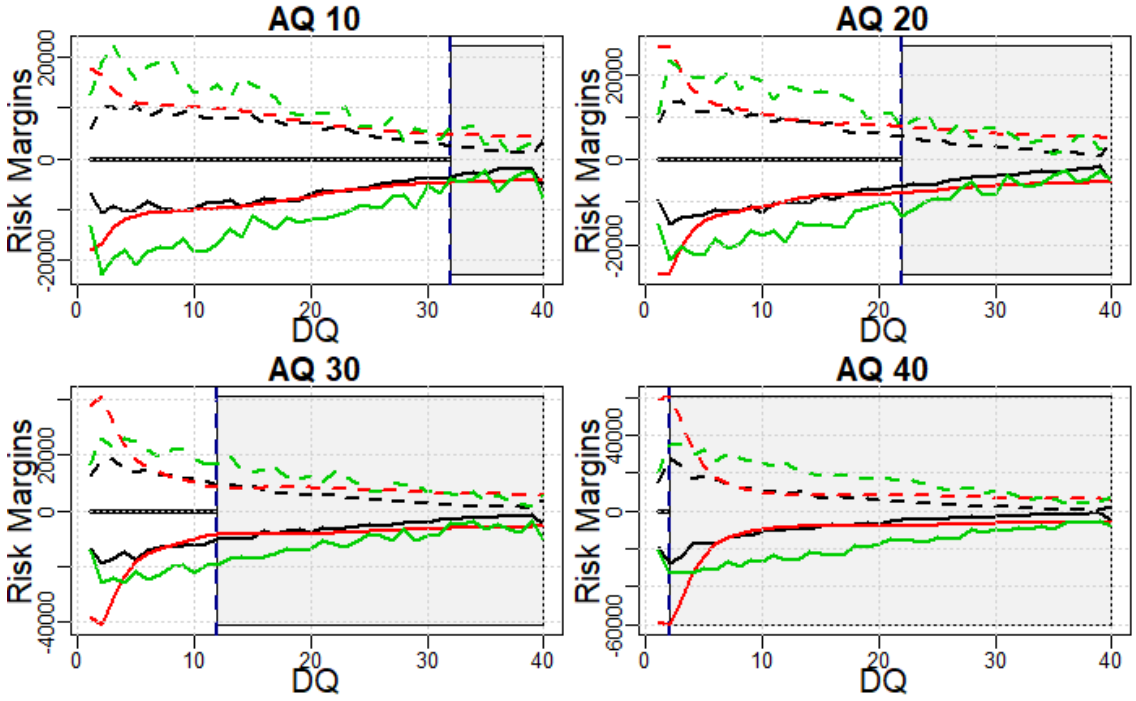


Figure 5.8: Dataset 2: Plots comparing the 25% and 75% risk margin estimates of the MDN and ccODP models to the empirical margins based on 250 simulations. The red line represents the MDN's margin estimates, the green line represents the ccODP's margin estimate, while the black line represents the empirical margins. The solid lines represent the 25% margins, while the dashed lines represent the 75% margin. The grey area represent the Lower Triangle.

The MDN and ccODP models were run on ten triangles of each of Datasets 1,2,3 and 4. The quantitative metrics are calculated for the 10 triangles and averaged, with results between the MDN and ccODP models compared in Table 5.1 As the table shows, the MDN, on average, had a lower RMSE and Quantiles Scores and had a higher Log Score for each Dataset, which is a significant out-performance by the MDN. Table 5.2 further reinforces these results by showing the percentage of triangles in which the MDN outperformed the ccODP for each quantitative metric. In each Dataset, the MDN outperforms the ccODP in each metric in the majority of triangles, except Dataset 1.

These quantitative results confirm the MDN's out-performance of the ccODP, in terms of central, volatility and quantile estimate, in all Datasets. This is due to the MDN's flexibility, which allows it to capture complex heterogeneous trends in the data, producing more accurate forecasts as a result.

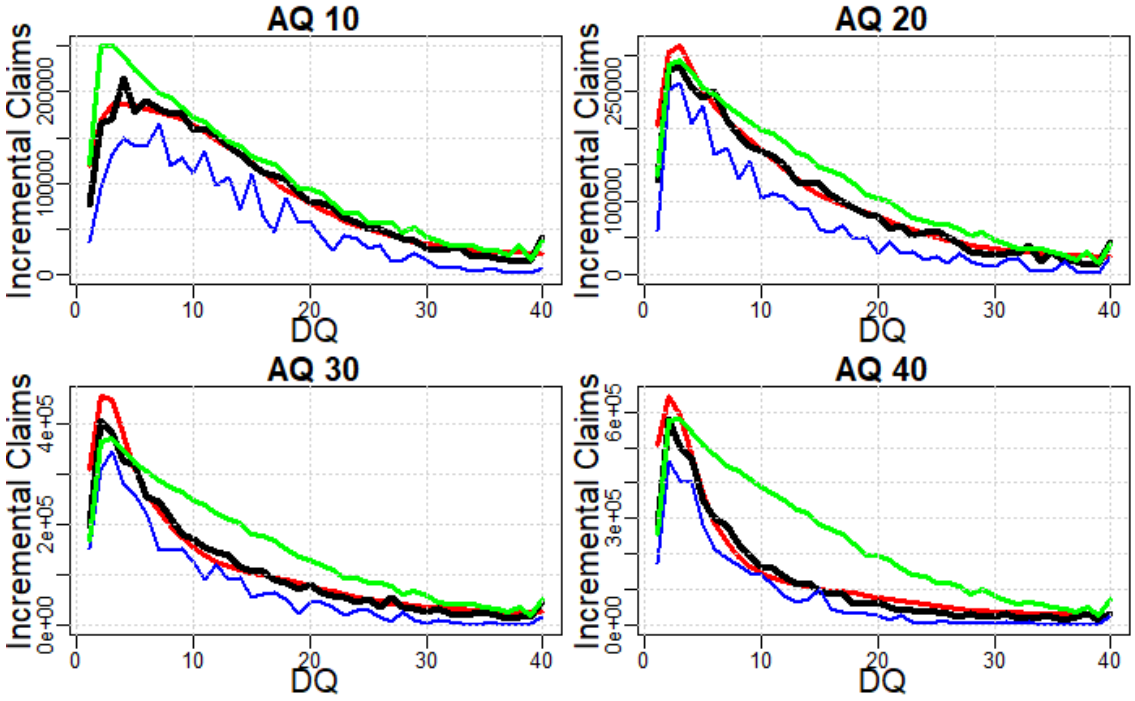


Figure 5.9: Dataset 2: Plots comparing the 99.5th quantile estimates of the MDN and ccODP models to the empirical 99.5th quantile claims based on 250 simulations. The red, green and black lines represents the MDN's estimate, the ccODP's estimate and the empirical quantile respectively. The blue line represents actual losses, while the grey area represent the Lower Triangle.

Dataset	Model	Mean RMSE	Mean LS	Mean QS (75%)	Mean QS (99.5%)
1	**MDN**	**1499.0**	**-8.05**	**374.1**	**25.3**
1	ccODP	1621.9	-8.09	380.8	34.0
2	**MDN**	**11,031.1**	**-10.4**	**3,033.6**	**201.9**
2	ccODP	52,423.9	-12.9	10,181.3	365.0
3	**MDN**	**12,887.2**	**-10.7**	**4,087.3**	**341.7**
3	ccODP	16,985.2	-11.4	5,441.9	1,473.9
4	**MDN**	**623,301.0**	**-14.5**	**193,688.8**	**31,408.5**
4	ccODP	928,707.7	-15.2	252,213.2	40,589.7

Table 5.1: The average score, over 10 triangles, of each quantitative metric; the RMSE, Log Score (LS) and Quantile Scores (QS) for the 75% and 99.5% levels. The MDN outperformed the ccODP in all Datasets and metrics when the average is taken.

5.2.4 Total Reserves

The MDN, in all Datasets except Dataset 1, showed a lower bias and more accurate dispersion of total reserves compared to the ccODP estimate. Figure 5.11 plots these results. This confirms the results observed when analysing the models' fits on Individual Cells. The MDN, having more accurate central estimates, had a lower bias to the empirical distribution of total reserves. Having estimated volatility and quantiles more accurately,

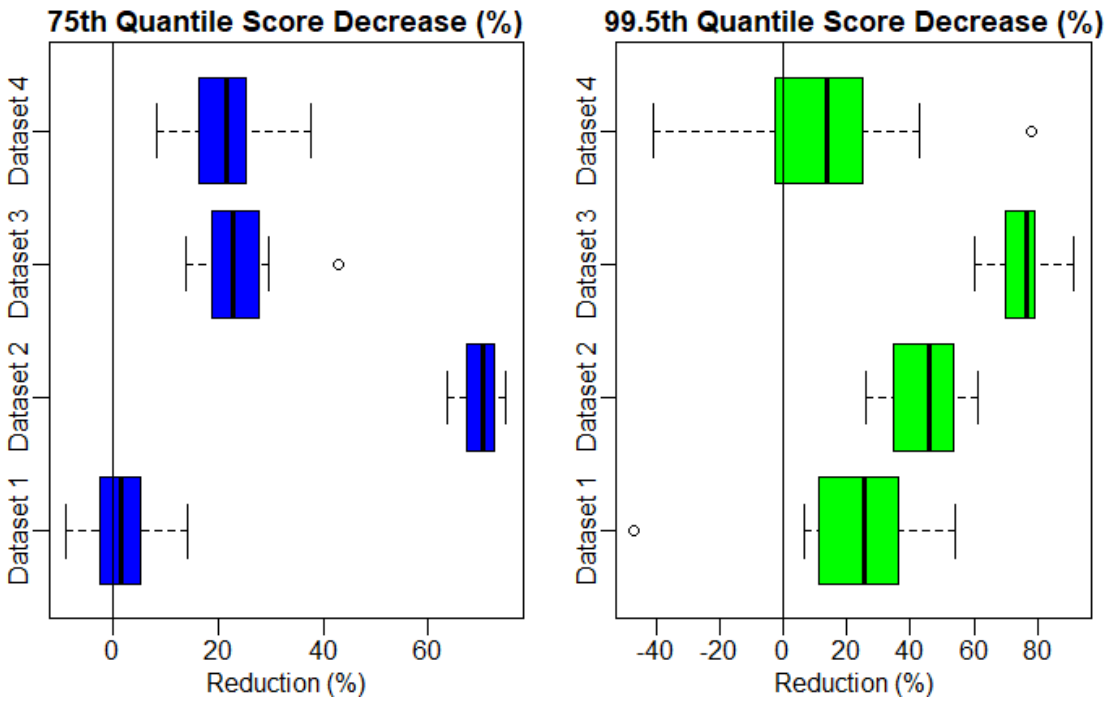


Figure 5.10: Boxplots displaying the MDN's (%) reduction in the 75% and 99.5% Quantile Scores relative to the ccODP for each of the 10 triangles run for Dataset 1,2,3 and 4.

Triangles Outperformed by MDN (%)				
Dataset	RMSE	Log Score	Quantile Score (75%)	Quantile Score (99.5%)
1	70	60	50	80
2	100	100	100	100
3	100	100	100	100
4	100	80	100	70

Table 5.2: The percentage of triangles in which the MDN outperformed the ccODP in that specific metric.

the MDN estimated a more accurate dispersion of total reserves than the ccODP. Similarly, given more accurate central and volatility estimates achieved by the MDN, it had more accurate 75% and 99.5% quantiles of total reserves compared to the ccODP. Table 5.3 calculated the quantitative metrics for both models for total reserve estimates, $\hat{R}$. The table confirms Figure 5.11, with Dataset 1 scoring a lower RMSE and Quantile Scores. The MDN outperforms for other Datasets, except it scores a high 99.5th Quantile Score for Dataset 4. This is due to the MDN occasionally under-estimating claims, which penalised the Quantile Score heavily. Otherwise, this analysis shows overall that the MDN significantly outperformed the ccODP when it came to measuring the location and shape of total reserves.

As mentioned earlier, the MDN's more accurate distributional fit is due to its flexibility in modelling complex trends in the data, as well as its ability to fit a more flexible distribution.

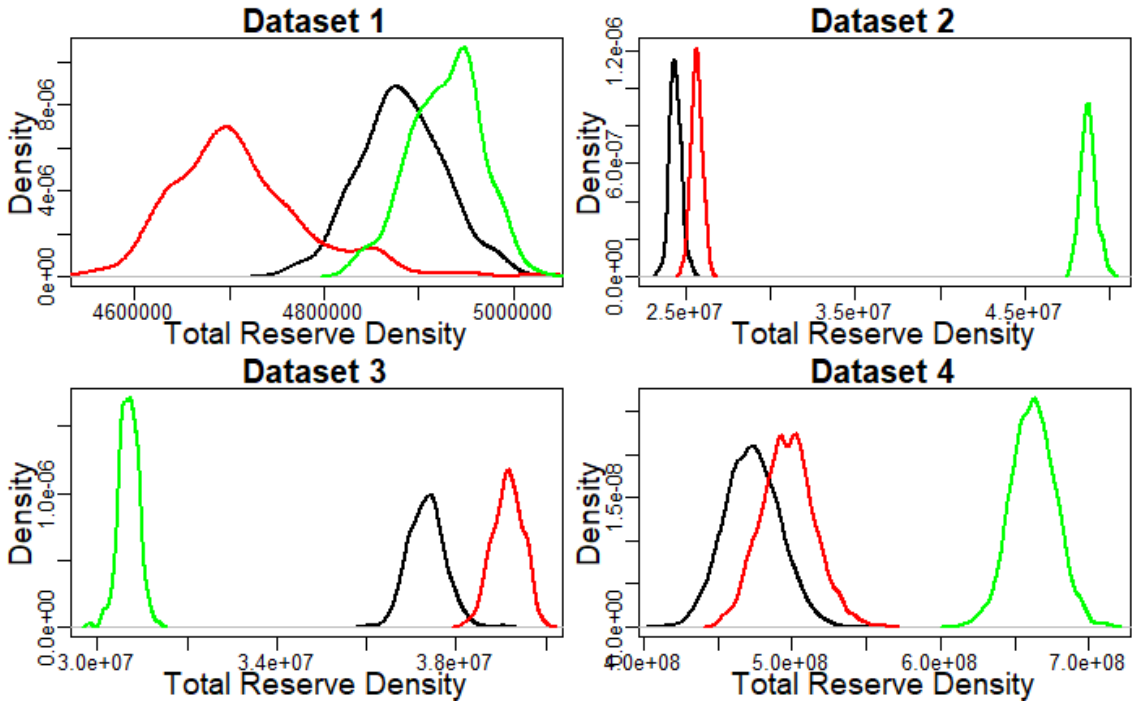


Figure 5.11: A plot of the total reserve density estimates for all Datasets, $\hat{R}$, with red and green being the MDN's and ccODP's estimated densities, respectively. For each Dataset, only one triangle is analysed for each plot. The black curve is the empirical density of total reserves. The MDN provides more accurate results, except for Dataset 1

Dataset	Model	RMSE	Quantile Score (75%)	Quantile Score (99.5%)
1	**MDN**	**113,766**	**43,617**	**7,752**
1	ccODP	96,556	28,399	21,877
2	**MDN**	**1,376,898**	**304,331**	**84,298**
2	ccODP	23,965,862	6,054,221	1,210,844
3	**MDN**	**2,449,403**	**1,286,303**	**984,280**
3	ccODP	5,481,193	3,899,808	5,173,746
4	**MDN**	**93,835,186**	**30,124,845**	**21,311,041**
4	ccODP	232,944,193	56,595,285	11,319,057

Table 5.3: The RMSE, Log Score (LS) and Quantile Scores (QS) at the 75% and 99.5% levels, calculated for total reserve estimates, $\hat{R}$. The ccODP outperforms for Dataset 1, but the MDN outperforms otherwise.

5.3 Interpretability: ResMDN

5.3.1 Overall Performance

Overall, the ResMDN successfully boosted the ccODP, capturing structure in the GLM's residuals, correcting them and improving the forecasting accuracy as a result. In situations

where the ccODP's residuals had a visible structure, the ResMDN found that structure, corrected it to an extent and projected these corrections to the Lower Triangle, improving the forecasting accuracy compared to the embedded ccODP. In situations where the ccODP's residuals had no visible structure, the ResMDN did not boost, with training ending very early. Projections for those datasets were relatively unchanged compared to the ccODP. These results are positive, as it indicates the ResMDN doesn't over-fit to the ccODP's residuals; it's effectively able to understand where the ccODP needs boosting and where it doesn't.

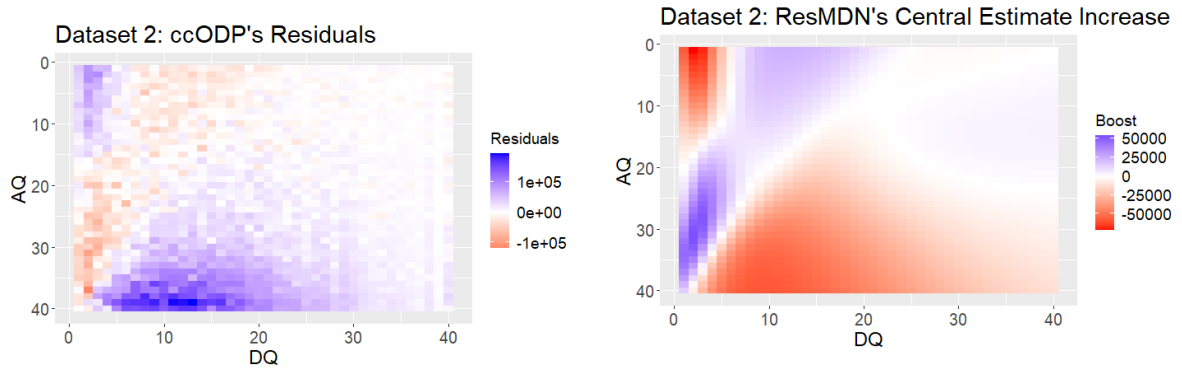
The ccODP's residual for Datasets 2 and 3 had clear structure:

- The ResMDN demonstrated the ability to recognise errors in the ccODP's central estimates and correct them. In Dataset 2, where the claim processing speed gradually increases, the ccODP assumes that claims are middle-tailed, as it assumes homogeneity in claim development. Hence, the ccODP under-estimated claims in early AQs and over-estimated claims later on. Figures 5.12(a) and 5.12(b) show heatmaps of the ccODP's residuals for Dataset 2, alongside the ResMDN's boosting for that Dataset. As can be seen, the ResMDN successfully understood the ccODP's shortcomings just mentioned, and counteracted them to an extent (visualised by the contrasting colours). Figure 5.13 plots the central estimates of the ResMDN and ccODP models, showing the ResMDN's corrections producing a more accurate forecast.

- The ResMDN also demonstrated the ability to recognise errors in the ccODP's volatility estimates and correct them. This is best demonstrated in Dataset 3, where an inflation shock occurs and persists from CQ30 onwards, the ccODP visibly under-estimates the volatility of claims beyond CQ30. The ResMDN successfully learns this shortcoming and increases volatility estimates accordingly for that period and in the Lower Triangle, producing more accurate mean and volatility estimates. Figure 5.14 visualises the ResMDN's boosting for Dataset 3. While the volatility is over-estimated for later DQs, the dominant trend visible is the ResMDN partially correcting volatility estimates after approximately CQ40, which is a significant achievement.

Overall, the ResMDN outperformed or provided similar results to the ccODP in all Datasets for the majority of triangles. Given these results, several practical considerations arose during modelling that are worth mentioning:

- The ResMDN projected the ccODP's residuals unreasonably in some instances. For



(a) A heatmap of the ccODP's residuals

(b) A heatmap of the boosting effects of the ResMDN

Figure 5.12: Heatmaps showing the ccODP's initial residuals in (a), calculated as $\mu_{i,j}^{ccODP} - X_{i,j}$. The ResMDN's boosting effects, calculated as $\mu_{i,j}^{ccODP} - \mu_{i,j}^{ResMDN}$, are shown in (b).

Dataset 2, it learned that the ccODP over-estimates claims in later AQs, so it reduces claims, and continues that trend incorrectly in the later DQs. The ResMDN was trained on information that indicated that the ccODP is fairly accurate in later DQs, however, it decided to give more importance to the training data in the later AQs indicating the the ccODP over-estimates. This issue was fixed by constraining projections, however that shortcoming indicates that the ResMDN may project the ccODP's residuals unreasonably.

5.3.2 Comparison to the MDN

While the ResMDN improved or didn't significantly change the central and distributional fit of claims relative to the ccODP, it failed to outperform the non-embedded MDN for any Dataset. Figure 5.13 shows the MDN providing more accurate central estimates for Dataset 2, while Figure 5.15 produces boxplots of the MDN and ResMDN's quantitative out-performance relative to the ccODP for Dataset 2. The results seen in Figure 5.15 are similar for Dataset 3. Both the qualitative and quantitative results confirm that the ResMDN under-performed the MDN. This under-performance is attributed to:

- The relevant structure in the ccODP's residuals is embedded in fewer training points, making projection more unstable. This was the case for Dataset 3
- The structure of residuals is harder to project; this was the case for Dataset 2. The ccODP increasingly over-estimated claims in the later DQs, but the ResMDN didn't have enough data to learn the extent that the ccODP over-estimated claims. The Rolling Origin method is a likely cause too for the ResMDN's shortcomings in this aspect, as some important information regarding the ccODP's residuals was in the

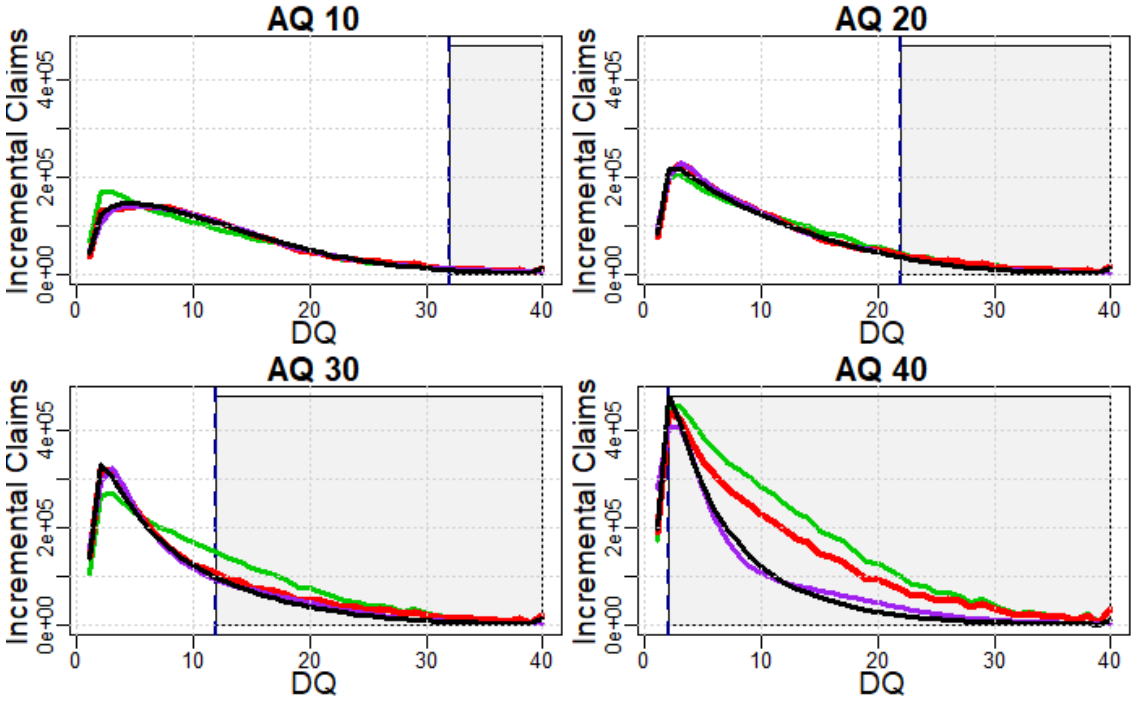


Figure 5.13: Dataset 2: Plots comparing the mean estimates of the ResMDN, MDN and ccODP models to the empirical mean claims based on 250 simulations. The red line represents the ResMDN's central estimate, the green line represents the ccODP's central estimate, the purple line is the MDN's estimate, while the black line represents the empirical mean of claims. The grey area represent the Lower Triangle.

latest CQs, which the ResMDN didn't train on.

The ResMDN's training time was much faster than the MDN. The MDN would usually exceed 5,000 epochs during training, while the MDN usually stopped training at 2,500 epochs. This observation confirms the findings of Gabrielli et al. (2020), who also noted faster training times.

5.3.3 Interpretability of Results

Unlike Individual Claims Modelling or Mortality Modelling, the issue of interpretability is less important in Loss Triangle Reserving, since results are only two-dimensional (Accident and Development Periods) and can be visualised relatively easily in a graph or heatmap. However, despite the simplicity of the Loss Triangle, the ResMDN's results are significantly influenced by the embedded GLM, the ccODP. Given that the MDN's black box modelling only occures on the ccODP's residuals, the ccODP forms the backbone of the ResMDN's fit. Hence, the ResMDN still provides results which are more interpretable and justifiable to stakeholders than the MDN.

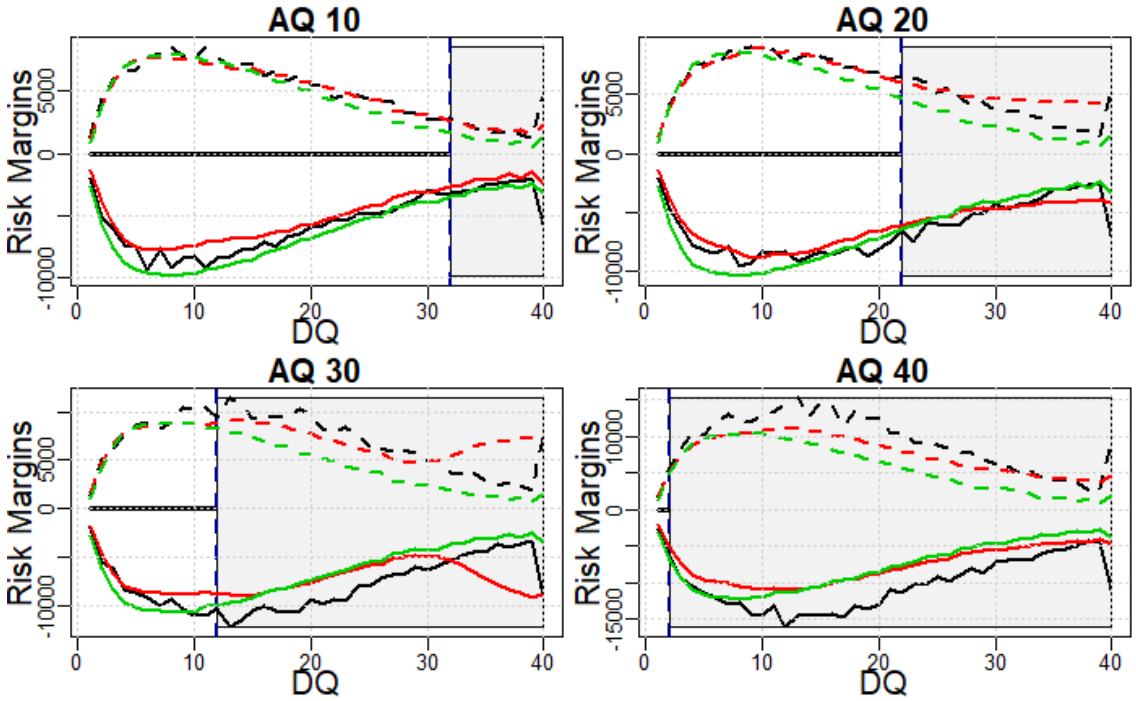


Figure 5.14: Dataset 3: Plots comparing the 25% and 75% risk margin estimates of the ResMDN and ccODP models to the empirical margins based on 250 simulations. The red line represents the ResMDN's risk margin estimates, the green line represents the ccODP's margin estimate, while the black line represents the empirical margins. The solid lines represent the 25% margins, while the dashed lines represent the 75% margin. The grey area represent the Lower Triangle.

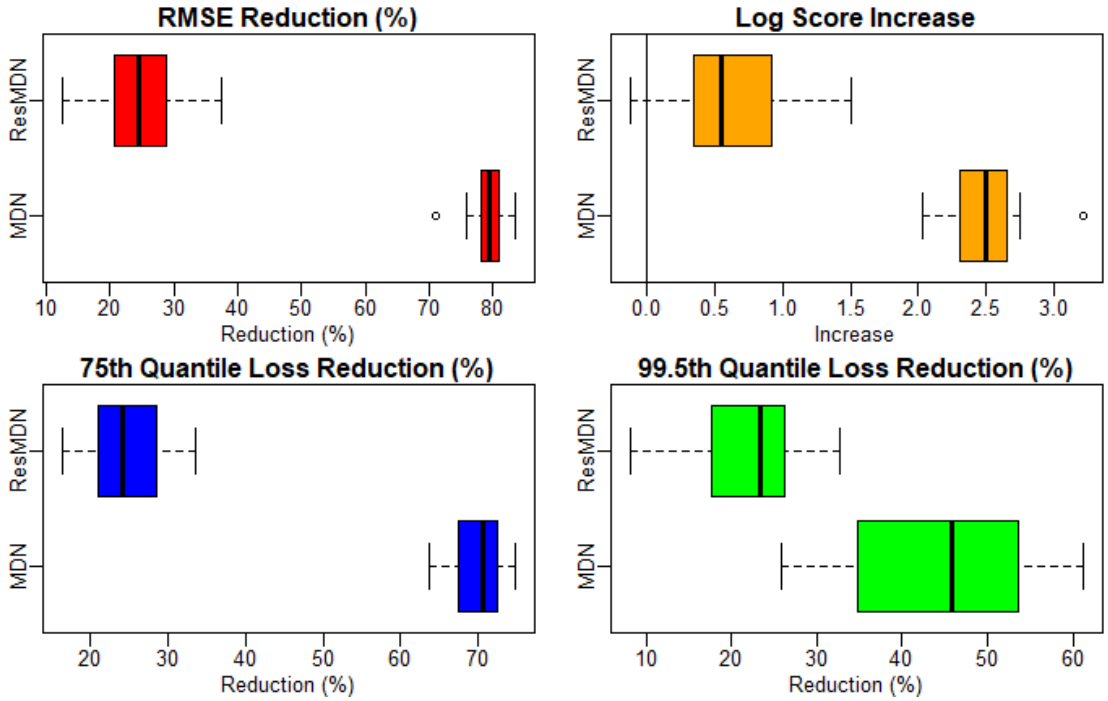


Figure 5.15: Boxplots displaying the ResMDN's and MDN's (%) reduction in the RMSE and Quantile Scores relative to the ccODP over the 10 triangles run for Dataset 2. The top right boxplot displays the MDN's increase in the Log Score relative to the ccODP.

CHAPTER 6

CONCLUSION

6.1 Research Summary and Contributions

Several obstacles hindering the popularisation of Neural Networks in Loss Triangle Reserving have been identified, addressed, and mitigated in this thesis, to a significant extent. The lack of distributional forecasting in Loss Reserving has been tackled successfully using the Mixture Density Network. The Rolling Origin Model Validation method provides a model testing and selection framework that assists the implementation of MDN modelling in practice.

Successfully tackling the issues above, this thesis hopes to further popularise Neural Networks in Loss Reserving. This thesis does not aim to replace GLMs, nor does it propose that Neural Networks are better for Loss Reserving than all GLMs. The MDN's predictive power, as well as the Neural Networks implemented in the Loss Reserving Literature, are only as good as the benchmark model used, which is the basic Chain Ladder model in the majority of the literature.

By enhancing the applicability of Neural Networks in Loss Reserving, this thesis hopes to advance Neural Networks as an option for reserving in practice, which will enhance the field of Loss Reserving. Furthermore, tackling these issues facing the implementation of Neural Networks in practice, this thesis hopes to act as a stepping stone for further

work in raising Neural Networks to higher levels in the Loss Reserving field and further unlocking their modelling potential.

6.1.1 Probabilistic forecasting of Loss Reserves using Mixture Density Networks

There has been little focus, in the literature, on using Neural Network for Probabilistic forecasting of Loss Reserves. This thesis developed the probabilistic forecasting power of Neural Networks in a Loss Triangle reserving setting. This was done through implementing the Mixture Density Network, which fits a Mixed Gaussian to the Loss data. The flexibility of the Mixed Gaussian distribution allowed the MDN to focus its modelling power on achieving accurate distributional forecasts of Outstanding Claims.

The MDN significantly outperformed the ccODP in distributional forecasting accuracy of OSC. These results occurred in four different claim environments, with varying structural complexities. The MDN successfully captured a shift in claim processing speed, an inflation shock, and provided stable and accurate distributional forecasts with a highly volatile dataset. These results show that the MDN model can be successfully applied to a wide variety of Loss Triangles of different Lines of Business and company related complexities.

6.1.2 Smooth, Robust and Accurate Loss Projections

The Actuarial Neural Network literature has not focused on methodologies of splitting the Loss Triangle into training and testing sets, which is crucial for effective model assessment and selection. This thesis effectively partitioned the Loss Triangle using the Rolling Origin method, which provided a comprehensive Machine Learning model testing framework, and allowed different MDN designs to be selected based on their projection power. As a result, the MDN applied in this thesis consistently produced smooth, robust, and accurate Outstanding Claims projections.

6.1.3 MDN Interpretability through a ResNet Adaptation

This thesis adapts the MDN to a more practically interpretable and justifiable model, the ResMDN. This model is also a contribution to the literature, given that MDNs haven't been applied to Loss Triangle Reserving in the literature. The ResMDN successfully boosted the embedded GLM, the ccODP, yielding equal to or superior distributional forecasts in all claim environments tested, while maintaining the ccODP backbone in its forecasts.

6.2 Limitations

- This thesis assumes that Incremental Claims, $X_{i,j}$ are independent. In reality, Incremental Claims are dependent, which will distort Total Reserve estimates.
- Only simulated data was used for modelling. While simulated data has its benefits of more data insight, it can sometimes miss the complexities of a real insurance setting.
- Parameter error is not quantified, which is a component in quantifying prediction error.

6.3 Further Work

- This thesis performs modelling on the Loss Triangle only. Individual Claims modelling has been gaining momentum since Neural Networks have become popularised. Individual Claims modelling analyses claims at a more granular level, allowing the Neural Network's modelling power to be used more effectively.
- This thesis used Mixed Gaussians only when fitting the MDN. Despite fitting Mixed Log-Gaussians indirectly, allowing the MDN to use a wider variety of component densities, such as Exponential, Gamma or Pareto, will increase its versatility.
- To quantify parameter error in Traditional Modelling, bootstrapping is commonly used. Bootstrapping has been applied to Neural Networks successfully, meaning it's a viable approach to quantifying parameter error for Neural Networks. However, other methods such as taking an ensemble of models run under different weight initialisations (Lakshminarayanan et al., 2017) have found success.

APPENDIX A

COMPUTATIONS

A.0.1 Proof of Lemma 3.1.1

Lemma A.0.1 *Let Y and $X = ln(Y)$ be random variables. Suppose that X follows a Mixed Gaussian distribution with parameters $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, such that:*

$$f_X(x) = \sum_{k=1}^{K} \alpha_k \phi(x|\mu_k, \sigma_k)$$

Then Y follows a Mixed Log-Gaussian distribution with parameters $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, such that:

$$f_Y(y) = \frac{1}{y} \sum_{k=1}^{K} \alpha_k \phi(ln(y)|\mu_k, \sigma_k)$$

Proof:
Given that $X = ln(Y)$, and $f_X(x)$ being calculated following Equation (A.1):

$$f_X(x) = \sum_{k=1}^{K} \alpha_k \phi(x|\mu_k, \sigma_k) \tag{A.1}$$

The following statements can be made for Y:

$$P(Y \leq y) = P(ln(Y) \leq ln(y)) \tag{A.2}$$
$$= P(X \leq ln(y)) \tag{A.3}$$
$$= \sum_{k=1}^{K} \alpha_k \Phi(ln(y)|\mu_k, \sigma_k) \tag{A.4}$$

Taking the differential of Equation (A.4):

$$\frac{dP(Y \leq y)}{dy} = \frac{d\left(\sum_{k=1}^{K} \alpha_k \Phi(ln(y)|\mu_k, \sigma_k)\right)}{dy} \tag{A.5}$$
$$f_Y(y) = \frac{1}{y} \sum_{k=1}^{K} \alpha_k \phi(ln(y)|\mu_k, \sigma_k) \tag{A.6}$$

Hence, Y follows a Mixed Log-Gaussian distribution with parameters $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma})$. This completes the proof.

A.1 Data Processing

A.1.1 Input Data Processing

Let $\mathbf{X}$ be the set $\{X_{i,j} : i + j \leq 41\}$, and let $\mu_{i,\mathbf{X}}$, $\mu_{j,\mathbf{X}}$, $\mu_{Loss,\mathbf{X}}$, $\sigma_{i,\mathbf{X}}$, $\sigma_{j,\mathbf{X}}$ and $\sigma_{Loss,\mathbf{X}}$ be the means and standard deviations of the Accident and Development quarters and the Incremental Losses of elements in $\mathbf{X}$. The Normalisation of the input variables $i, j, X_{i,j}$ is as follows:

$$i_{Norm} = \frac{i - \mu_{i,\mathbf{X}}}{\sigma_{i,\mathbf{X}}}$$
$$j_{Norm} = \frac{j - \mu_{j,\mathbf{X}}}{\sigma_{j,\mathbf{X}}}$$
$$X_{i,j,Norm} = \frac{X_{i,j} - \mu_{Loss,\mathbf{X}}}{\sigma_{Loss,\mathbf{X}}}$$

A.1.2 Processing the MDN Output

The variables i_{Norm} and j_{Norm} are fed into the Network, which once trained, produces the output:

$$\{\boldsymbol{\alpha}_{i,j}^{Norm}, \boldsymbol{\mu}_{i,j}^{Norm}, \boldsymbol{\sigma}_{i,j}^{Norm}\} = \{\alpha_{i,j,1}^{Norm}, \alpha_{i,j,2}^{Norm} ... \alpha_{i,j,K}^{Norm}, \mu_{i,j,1}^{Norm}, \mu_{i,j,2}^{Norm} ... \mu_{i,j,K}^{Norm}, \sigma_{i,j,1}^{Norm}, \sigma_{i,j,2}^{Norm} ... \sigma_{i,j,K}^{Norm}\} \tag{A.7}$$

These output values are used to form the distributional fit:

$$f_{\hat{X}_{i,j,Norm}}(x) = \sum_{k=1}^{K} \alpha_{i,j,k}^{Norm} \phi(x|\mu_{i,j,k}^{Norm}, \sigma_{i,j,k}^{Norm})$$

The MDN output parameters are scaled and shifted to produce the distributional fit on the raw data, $X_{i,j}$, as such:

$$f_{\hat{X}_{i,j}}(x) = \sum_{k=1}^{K} \alpha_{i,j,k} \phi(x|\mu_{i,j,k}, \sigma_{i,j,k}),$$

where

$$\alpha_{i,j,k} = \alpha_{i,j,k}^{Norm}, \tag{A.8}$$

$$\mu_{i,j,k} = \mu_{i,j,k}^{Norm} * \sigma_{Loss,\mathbf{X}} + \mu_{Loss,\mathbf{X}}, \tag{A.9}$$

$$\sigma_{i,j,k} = \sigma_{i,j,k}^{Norm} * \sigma_{Loss,\mathbf{X}} \tag{A.10}$$

When fitting a Mixed Log-Gaussian, normalisation and re-scaling the output is done after the Log of $X_{i,j}$ is taken. To deal with 0 cells, that is instances where $X_{i,j} = 0$, $X_{i,j}$ is transformed to e^Z, such that $ln(X_{i,j}) = Z$. By default, $Z = 0$, however in many instances where the volatility caused by this transformation detriments the fit significantly, Z is set to higher numbers.

A.2 ccODP Modelling

A.2.1 Fitting the ccODP

The ccODP fit is as follows:

$$\frac{\hat{X}_{i,j}}{\phi} \sim Poi(\frac{A_i B_j}{\phi})$$

Hence $\hat{\mu}_{i,j}^{ccODP} = A_i B_j = e^{ln(A_i)+ln(B_j)}$. A GLM with the "quasipoisson" family is fit, without an intercept. The dispersion factor, ϕ, is constant for all $X_{i,j}$. Of the 80 parameters fit, one will be free (Taylor and McGuire, 2016), hence $\mathbf{B} = \{B_i, i = 1, 2, 3, ...40\}$ is scaled to have a sum of 1, such that:

- $A_i \mapsto A_i \sum_{j=1}^{40} B_j$
- $B_j \mapsto \frac{B_j}{\sum_{j=1}^{40} B_j}$

A.2.2 Calculating Quantitative Metrics

To calculate the quantitative metrics for the ccODP fit:

Central Estimates/Volatility:

The central and standard deviation estimates of the ccODP model are calculated as such:

- $E[\hat{X}_{i,j}] = \hat{\mu}_{i,j}^{ccODP} = A_i B_j$
- $Var(\hat{X}_{i,j}) = \phi \hat{\mu}_{i,j}^{ccODP} = \phi A_i B_j$

Log Score:

Take $\hat{Y}_{i,j} = \lfloor (\frac{\hat{X}_{i,j}}{\phi}) \rfloor$ and $\lambda_{i,j} = \frac{A_i B_j}{\phi}$. The the standard ccODP distributional assumption is that:

$$\hat{Y}_{i,j} \sim Poi(\frac{A_i B_j}{\phi}) \text{ for } \hat{Y}_{i,j} = 0, 1, 2, 3...$$

However, this assumption only defines $\hat{X}_{i,j}$ for the values $0, \phi, 2\phi....$ Hence, an additional step assumes $\hat{X}_{i,j}$ is uniform between the ϕ bins. That is, $\hat{X}_{i,j} | ((k-1)\phi < \hat{X_{i,j}} < k\phi) \sim U[(k-1)\phi, k\phi]$.

If we let $y = \lfloor (\frac{x}{\phi}) \rfloor$, then the adjusted density function for $\hat{X}_{i,j}$ follows Equation (A.11):

$$f_{\hat{X}_{i,j}}(x) = \frac{e^{-\lambda_{i,j}} \lambda_{i,j}^{y}}{\phi * y!} \tag{A.11}$$

And the Log Score follows, as such:

$$LogScore(\hat{\mathbf{X}}, \mathbf{X}) = \sum_{i,j:i+j>41} ln(f(\hat{X}_{i,j}))$$

Quantile Estimates:

Following the notation introduced earlier, $\hat{Y}_{i,j} \sim Poi(\frac{A_i B_j}{\phi})$. Let $\hat{Y}_{i,j,q}$ and $\hat{X}_{i,j,q}$ be the q^{th} quantile estimates for $\hat{Y}_{i,j}$ and $\hat{X}_{i,j}$ respectively. Then the quantile estimate is calculated following Equation (A.12):

$$\hat{X}_{i,j,q} = \phi * \hat{Y}_{i,j,q} \tag{A.12}$$

A.3 Data Simulation

This section provides technical details regarding the simulation of the four Datasets. The simulator used was SYnthETIC, develop by Avanzi et al. (2020). Unless otherwise stated, the default parameters of the simulator were used.

A.3.1 Notation

We define the following terms:

N_i : The number of claims in Accident Quarter i

$S_{i,r}$: The total size of claim r, incurred in Accident Quarter i

μ_R = The mean reporting delay of each claim

σ_R = The standard deviation of reporting delay of each claim

μ_S = The mean settlement delay of each claim

σ_S = The standard deviation of settlement delay of each claim

SI_i^{OCC} : The quarterly occurrence based Superimposed Inflation for Accident Quarter i

SI_c^{PAY}: The quarterly Superimposed Inflation for Calendar Quarter c.

A.3.2 Dataset 1

The claims are simulated with these parameters:

$N_i \sim Poi(20,000)$

$S_{i,r}^{0.2} \sim N(9.5, 3)$

$\mu_R = 0.493$

$\frac{\sigma_R}{\mu_R} = 1.92$

$\mu_S = 6.58$

$\frac{\sigma_S}{\mu_S} = 1.02$

$SI_i^{OCC} = 1$

$SI_c^{PAY} = 1$

A.3.3 Dataset 2

Two separate datasets are simulated and added to make the final table, one with short tail claims and the other long tail claims.

Denote incremental claims from the short tail and long tail datasets as $\mathbf{X}^{SHORT}$ and $\mathbf{X}^{LONG}$, respectively. To simulate claims in $\mathbf{X}^{SHORT}$:

$N_i^{SHORT} \sim Poi(5,000 + 55,000\frac{i-1}{39})$

$S_{i,r}^{0.25} \sim N(9.5, 3)$

$\mu_R^{SHORT} = 0.493$

$\frac{\sigma_R^{SHORT}}{\mu_R^{SHORT}} = 1.92$

$\mu_S^{SHORT} = 6.58$

$\frac{\sigma_S^{SHORT}}{\mu_S^{SHORT}} = 1.02$
$SI_i^{OCC} = 1$
$SI_c^{PAY} = 1.03$

To simulate claims in X^{LONG}:
$N_i^{LONG} \sim Poi(60,000 - 55,000\frac{i-1}{39})$
$S_{i,r}^{0.25} \sim N(9.5, 3)$
$\mu_R^{LONG} = 2.47$
$\frac{\sigma_R^{LONG}}{\mu_R^{LONG}} = 1.54$
$\mu_S^{LONG} = 11.74$
$\frac{\sigma_S^{LONG}}{\mu_S^{LONG}} = 0.61$
$SI_i^{OCC} = 1$
$SI_c^{PAY} = 1.03$

The two datasets are added together, as such:

$$\mathbf{X} = \mathbf{X}^{SHORT} + \mathbf{X}^{LONG}$$

A.3.4 Dataset 3

The claims are simulated according to these parameters:
$N_i \sim Poi(60,000)$
$S_{i,r}^{0.25} \sim N(9.5, 3)$
$\mu_R = 2.47$
$\frac{\sigma_R}{\mu_R} = 1.54$
$\mu_S = 11.74$
$\frac{\sigma_S}{\mu_S} = 0.61$
$SI_i^{OCC} = 1$
$SI_c^{PAY} = 1 * (\mathbf{1}(c < 30)) + 1.08 * (\mathbf{1}(c \geq 30))$

A.3.5 Dataset 4

This Dataset was simulated entirely based on the default parameters and functions in the SynthETIC simulator.

APPENDIX B

RESULTS

This Section looks at other graphical results not mentioned in Section 5.

B.1 Dataset 1

This Section provides plots of the central estimates and risk margin estimates of the MDN and ccODP models for Dataset 1.

B.1.1 Central Estimates

The central estimates of the MDN and ccODP models for Dataset 1 were very similar and accurate, as Figure B.1 shows.

B.1.2 Risk Margins

Similarly to the centrla estimates, both models had accurate margins compared to the empirical margins. However, the MDN over-estimates volatility for DQ1, as shown in Figure B.2. This error does not directly hurt projection accuracy, as only DQ2 onwards is a projection.

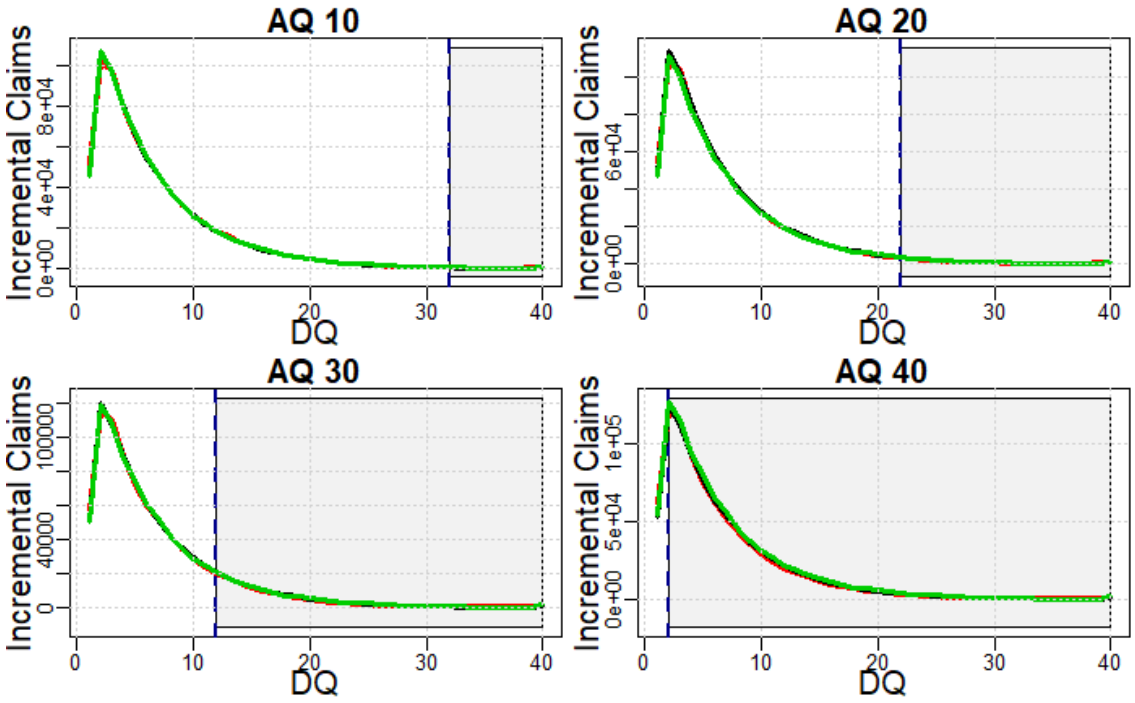


Figure B.1: Dataset 1 (simple claims): Plots comparing the mean estimates of the MDN and ccODP models to the empirical mean claims based on 250 simulations. The red line represents the MDN's central estimate, the green line represents the ccODP's central estimate, while the black line represents the empirical mean of claims. The grey area represent the Lower Triangle.

B.2 Dataset 2

This Section looks at how the MDN's fit was improved by adding an MSE term to the loss and directly constraining projections.

B.2.1 Adding the MSE Term

To increase the accuracy of central estimates, an MSE term was added to the Negative Log Likelihood Loss function. Figure B.3 shows the improvement in central estimates after this addition.

B.2.2 Constraining Projections

The MDN was over-estimating constraints in the later DQs. Hence, projections were constrained such that central estimates for DQ35-39 had to be between 0 and 5,000. The improvement in fit is shown in Figure B.4.

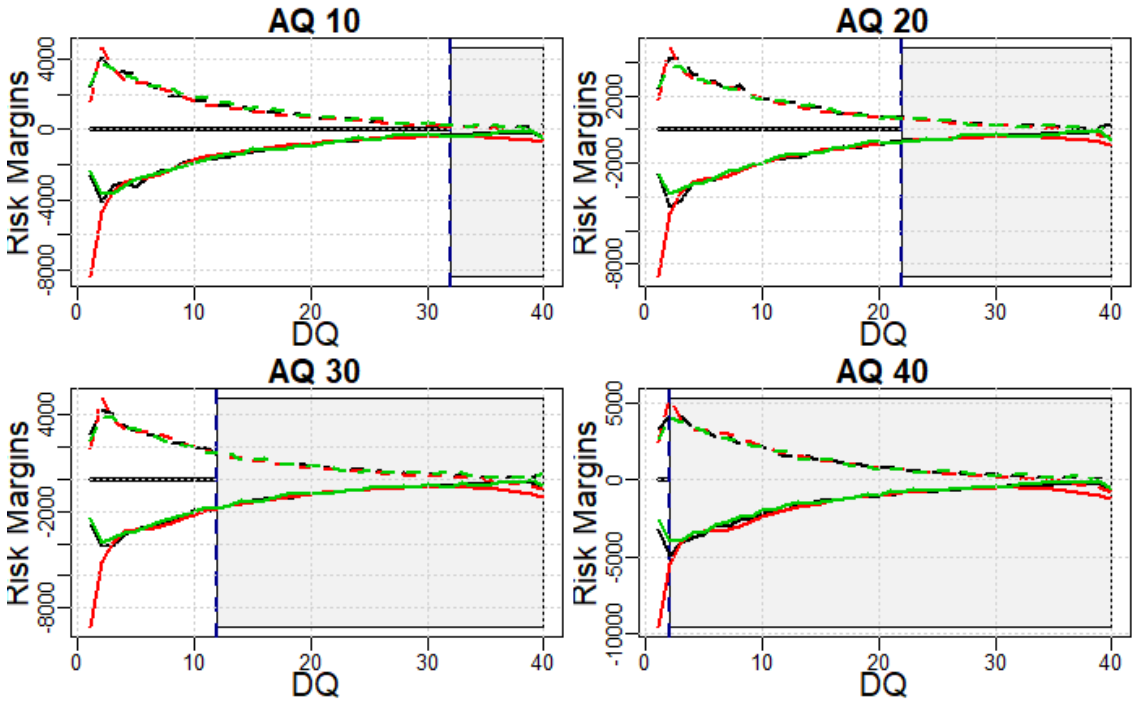


Figure B.2: Dataset 1(simple claims): Plots comparing the 25% and 75% risk margin estimates of the MDN and ccODP models to the empirical margins based on 250 simulations. The red line represents the MDN's margin estimates, the green line represents the ccODP's margin estimate, while the black line represents the empirical margins. The solid lines represent the 25% margins, while the dashed lines represent the 75% margin. The grey area represent the Lower Triangle.

B.3 Dataset 3

This Section goes through the data partition undertaken for Dataset 3 (Inflation Shock) in order to include more training data in the later Calendar Quarters.

B.3.1 Data Partition

After initial experimentation, the MDN failed to properly capture the inflation trend, which was predicted. Since the first partition in the Rolling Origin methodology doesn't include much data from CQ30 onwards, it was dropped for a cross-validation methodology, in which the validation and testing sets were chosen as follows:

- The Upper Triangle was split in half, with the first 29 CQs being in one, and the rest in the other half.
- The validation and testing splits for Partition 1 and 2 were both 10%. Both partitions used the entire upper triangle, as the Rolling Origin's first partition excludes any inflation shock data in the training and validation sets.

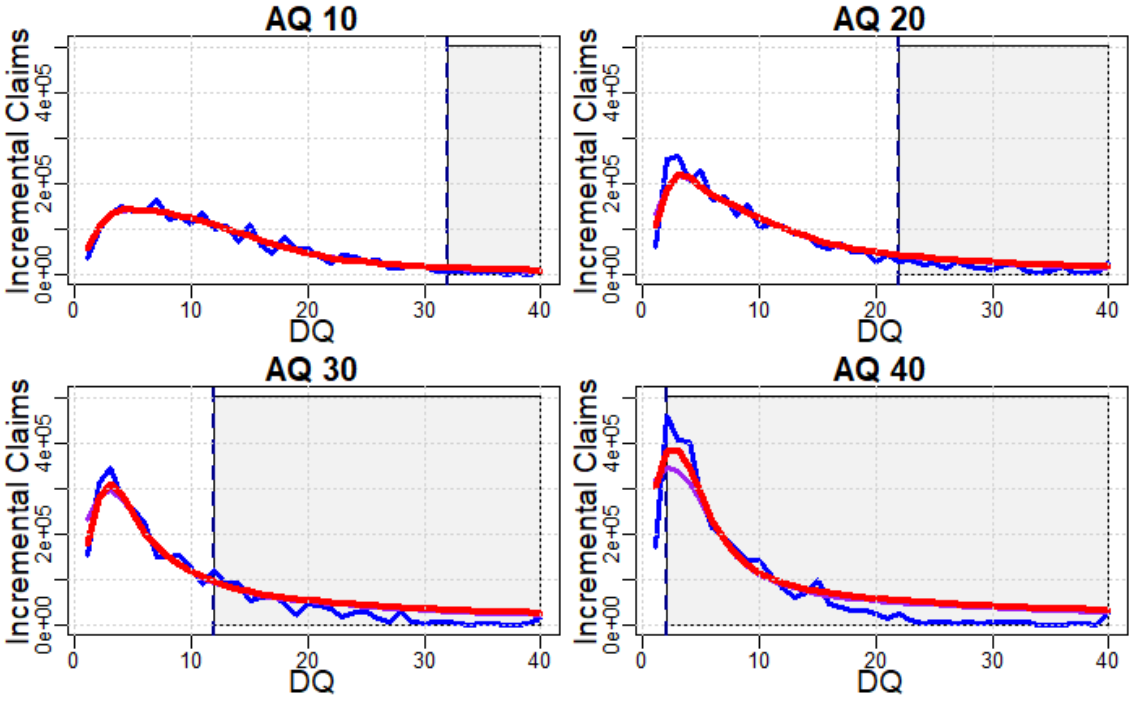


Figure B.3: Dataset 2 (Long to Short Claims): Plots comparing the central estimates of the MDN after an MSE term is added to the loss function. The purple and red lines represent the MDN trained on the NLL Loss and NLL + MSE Loss, respectively, while the blue line represents losses. The grey area represents the Lower Triangle.

- The testing set was allocated randomly in the later half of the triangle.
- Half of the validation set was allocated randomly in the later half.
- The remaining half of the validation set was allocated randomly across the whole Upper Triangle.
- For the second partition, all validation and testing data points in Partition 1 were set as training points. The allocation of testing and validation points was subsequently done as mentioned before, among non-training points.
- To fit the model on the whole dataset, the validation set was allocated similarly to Partitions 1 ans 2, with the remaining points allocated to the training set. The triangle was split in two different ways, to allow more coverage of the validation sets.

Figure B.5 provides a heatmap of the new partitions. As Figure B.6 shows, the new partition was much more effective at capturing the inflation shock, since the MDN had more training data in the shock period and therefore was better able to learn the trend.

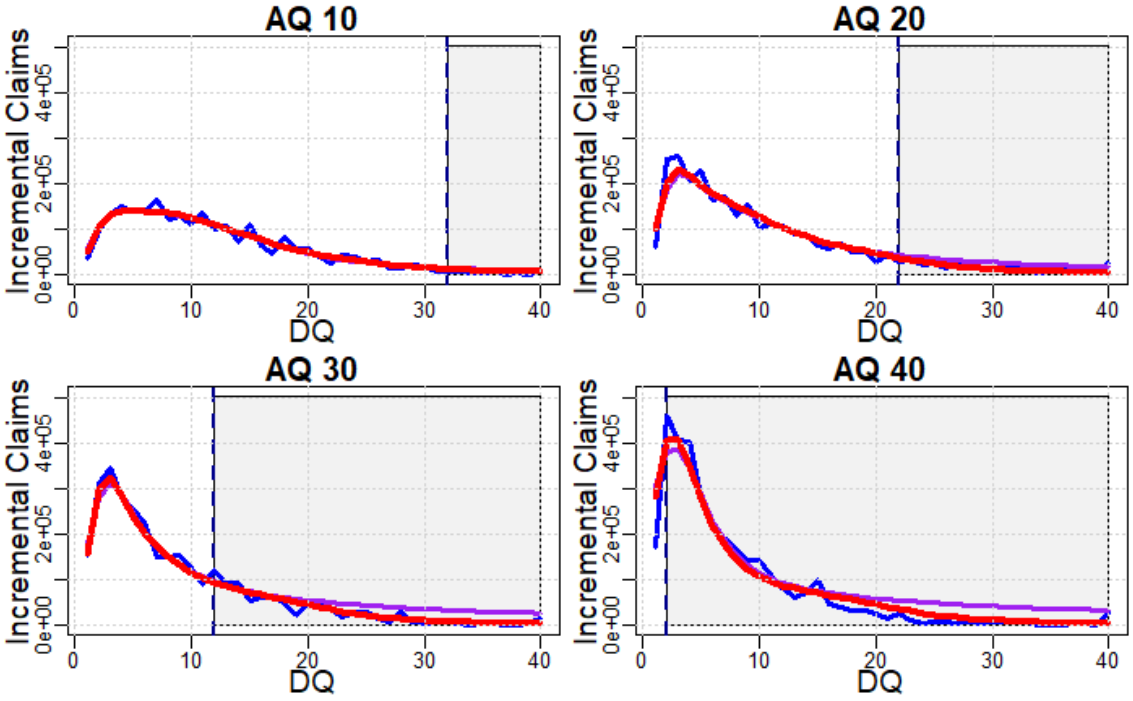


Figure B.4: Dataset 2 (Long to Short Claims): Plots comparing the central estimates of the MDN after projections are constrained. The purple and red lines represent the unconstrained and constrained MDN, respectively, while the blue line represents losses. The grey area represents the Lower Triangle.

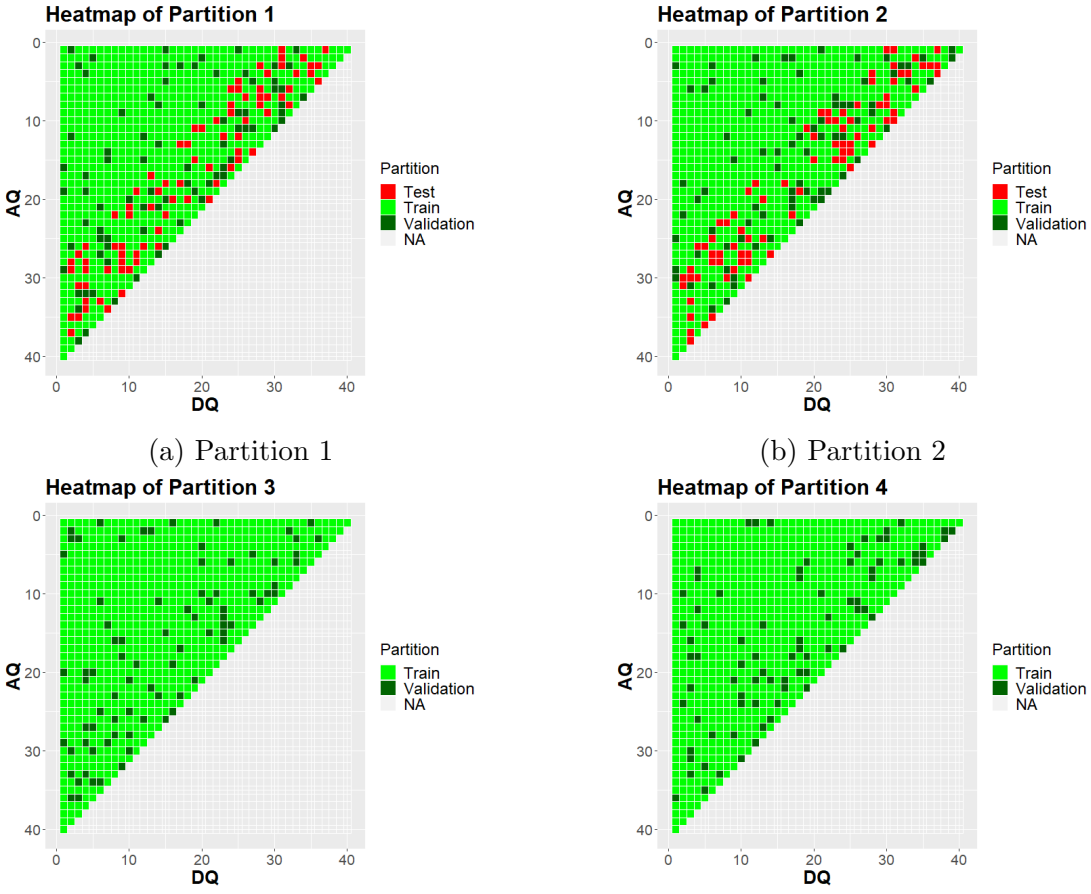
B.4 Dataset 4

This Section provides plots of the central estimates and risk margin estimates of the MDN and ccODP models for Dataset 4. It also outlines how the Log data was smoothed to help the MDN provide more reasonable projections.

B.4.1 Smoothing the Log Data

Having a high number of 0 cells in the Dataset is troubling for the MDN, as it will associate a high volatility to these areas. As Figure B.8 shows, the MDN predicted an extremely high volatility for later DQs on the Log Data, which was also translated into a high central estimate when the data was transformed back to its raw format. Furthermore, fitting a mixture of Gaussian distributions assumes the data follows a continuous density, however, having numerous 0 loss cells weakens that assumption. We tackled this problem here by modifying the dataset being fed to the Network as such:

- Any cell (i, j), where $X_{i,j} = 0$: $X_{i,j} \mapsto e^7$. The choice of e^7 will elevate the 0 cells and reduce volatility, while still allowing the MDN to recognise that these cells have a relatively low value. At a first glance, this transformation should increase mean



(a) Partition 1

(b) Partition 2

(c) Partition 3

(d) Partition 4

Figure B.5: Heatmaps of the 4 partitions involved in training and fitting Dataset 3 (Inflation Shock). Light Green, Dark Green and Red represent the training, validation and testing sets, respectively. The focus of this partition is to include training data in the latest Calendar Quarters. Partition 1 and 2 are used for model selection, while Partition 3 and 4 are used for training the final model.

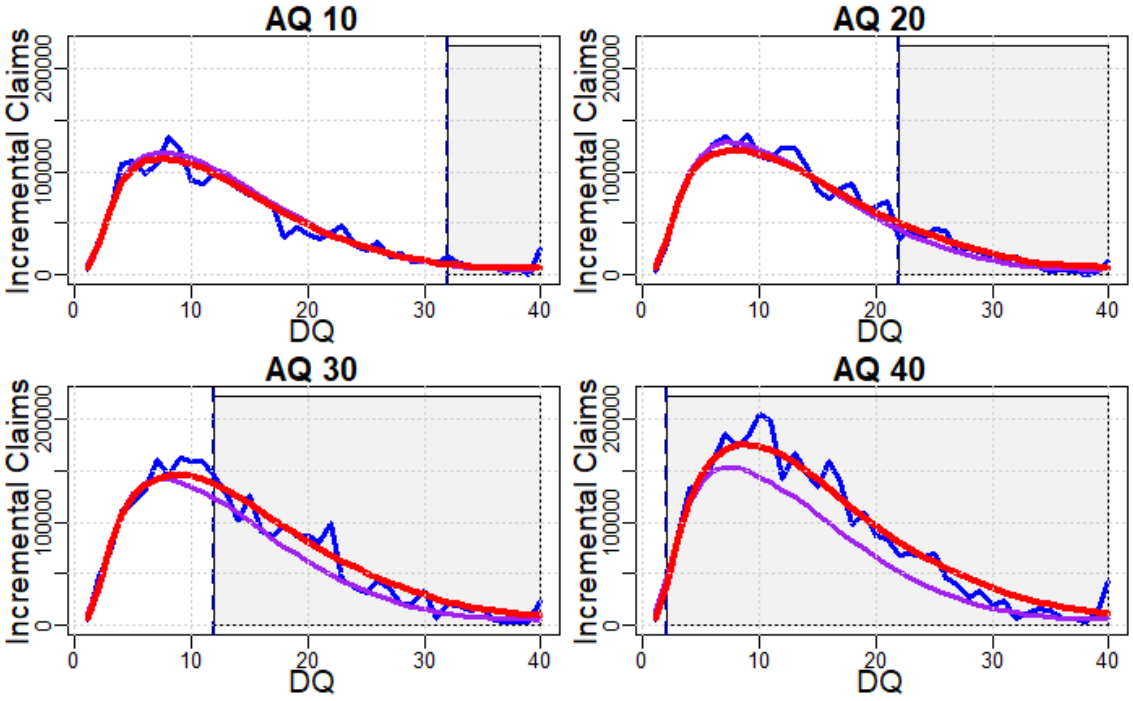


Figure B.6: Dataset 3 (Inflation Shock): Plots comparing the central estimates of the MDN under the new and old data partitions. The purple and red lines represent the MDN fit when trined under the old and new partitions, respectively, while the blue line represents losses. The grey area represents the Lower Triangle.

estimates of claims, however, the reduction in volatility brings the mean estimate down to more reasonable levels in later DQs.

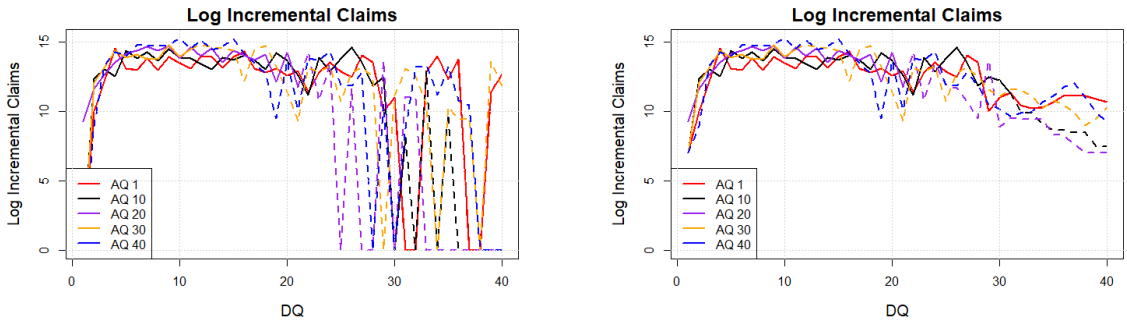
Then, in any cell (i, j), where $j \geq 30$:

- $ln(X_{i,j}) \mapsto \frac{\sum_{k=0}^{3} ln(X_{i,j-k})}{4}$. Taking the moving average as such smooths the curve in the later DQs, reducing mean and volatility estimates.

Then, any cell (i, j), where $4 \leq j < 30$ and $X_{i,j} = e^7$:

- $ln(X_{i,j}) \mapsto \frac{\sum_{k=0}^{3} ln(X_{i,j-k})}{4}$. In some datasets simulated, 0 cells were present as early as DQ20, which lead to excessively high volatility estimates in the corresponding region. A moving average of all data points there may distort the dataset too much, hence the transformation was only applied to 0 cells which occurred before DQ30.

Figure B.7 provides a visualisation of the Log Incremental Claims before and after smoothing was applied.



(a) Claims without smoothing the Log Data.

(b) Claims with smoothed Log Data.

Figure B.7: Dataset 4: The comparison of Log Incremental claims when not smoothed (a) and when smoothed (b).

B.4.2 Central Estimates

The MDN's central estimates were, on average, slightly more accurate than the ccODP's estimates. Figure B.9 compares the central estimates of the two models.

B.4.3 Risk Margins

Similarly to central estimates, the MDN's risk margin estimates were, on average, slightly more accurate than the ccODP's estimates. The ccODP's margin estimates were extremely volatile, similar to its central estimates. No structural shortcoming is visible beyond the noise. Figure B.10 compares the risk margin estimates of the two models.

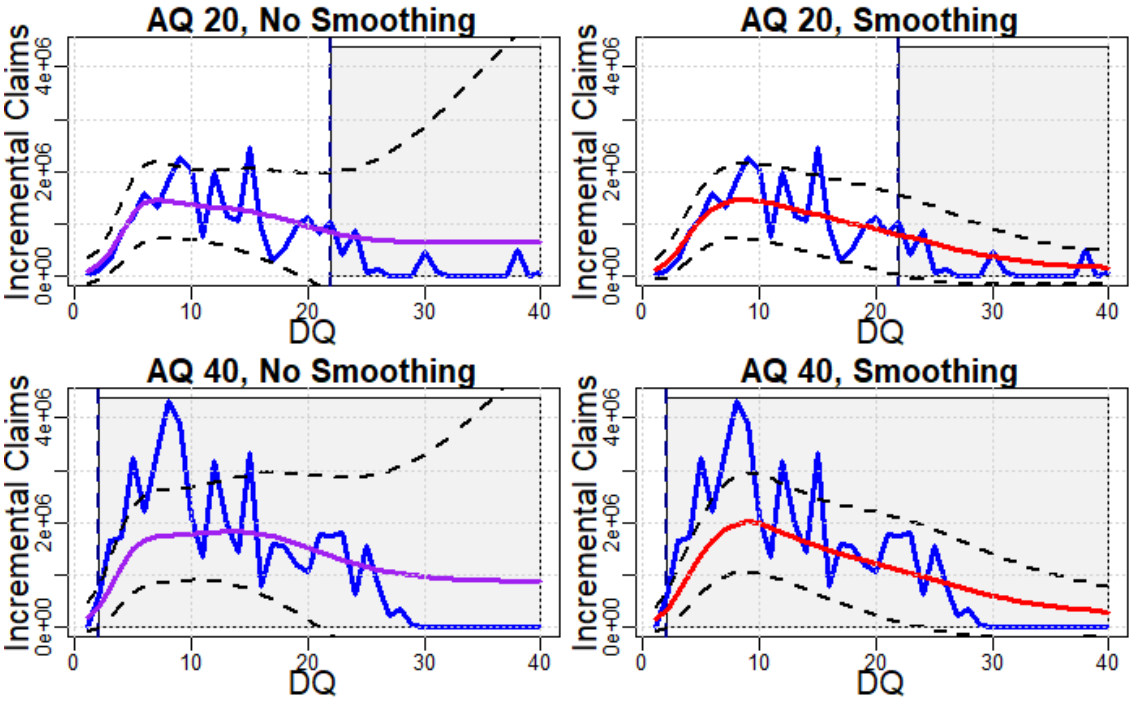


Figure B.8: Dataset 4: A comparison of the MDN fit with smoothed and non-smoothed Log Data. The purple and red lines represent the MDN fit on non-smoothed and smoothed data, respectively. The blue lines and black dashed lines represent the actual losses and 1 SD margin of the MDN, respectively. The grey area represents the Lower Triangle, the forecasting region.

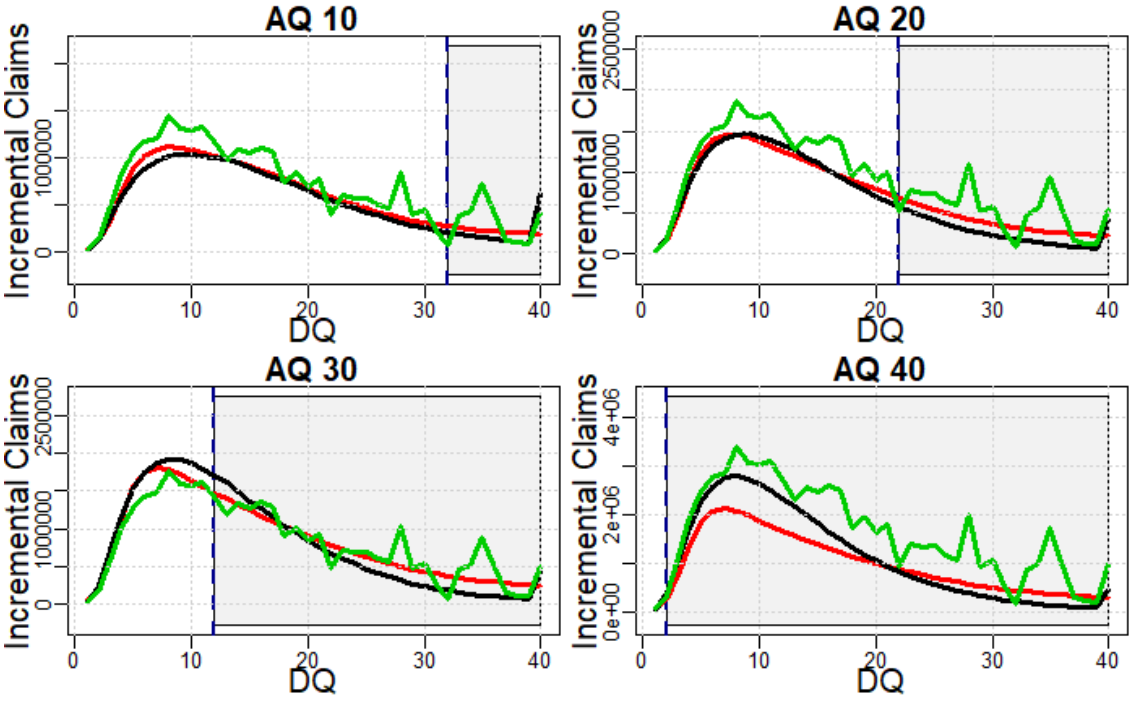


Figure B.9: Plots comparing the mean estimates of the MDN and ccODP models to the empirical mean claims based on 250 simulations. The red line represents the MDN's central estimate, the green line represents the ccODP's central estimate, while the black line represents the empirical mean of claims. The grey area represent the Lower Triangle.

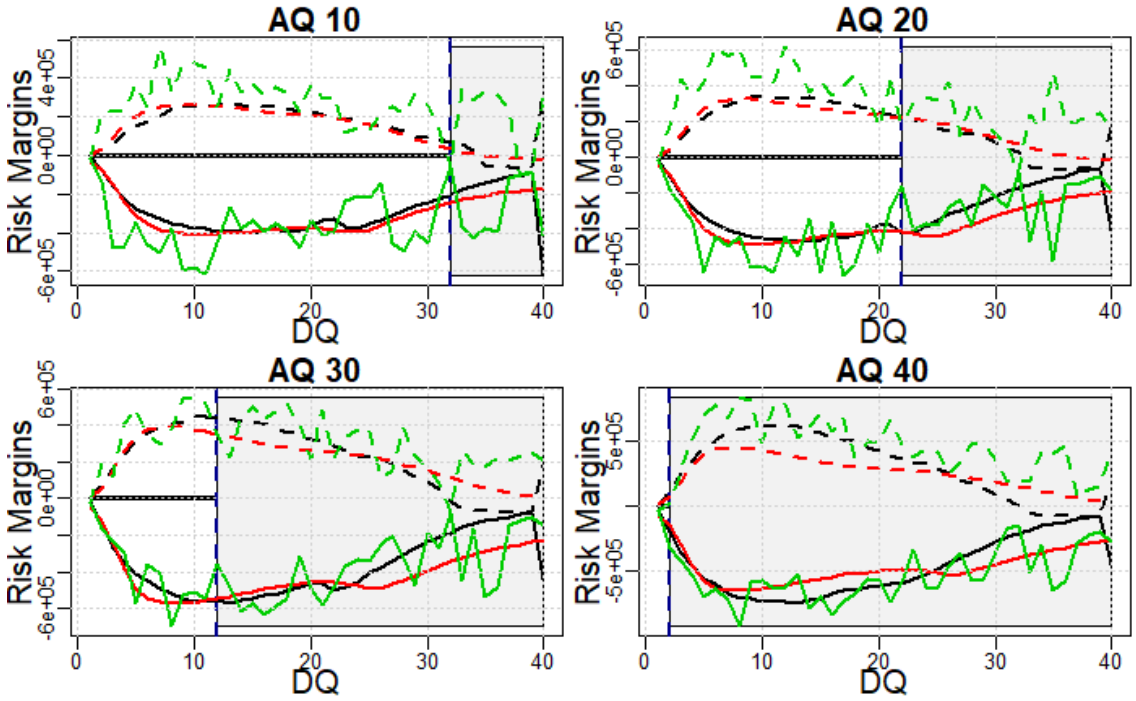


Figure B.10: Plots comparing the 25% and 75% risk margin estimates of the MDN and ccODP models to the empirical margins based on 10,000 simulations. The red line represents the MDN's margin estimates, the green line represents the ccODP's margin estimate, while the black line represents the empirical margins. The solid lines represent the 25% margins, while the dashed lines represent the 75% margin. The grey area represent the Lower Triangle.

BIBLIOGRAPHY

Abreu, S., 2019. Automated architecture design for deep neural networks. arXiv preprint arXiv:1908.10714 .

Arjas, E., 1989. The claims reserving problem in non-life insurance: Some structural ideas. ASTIN Bulletin: The Journal of the IAA 19, 139–152.

Avanzi, B., Taylor, G., Vu, P., Wong, B., 2016. Stochastic loss reserving with dependence: A flexible multivariate tweedie approach. Insurance Mathematics Economics 71, 63–78.

Avanzi, B., Taylor, G.C., Wang, M., Wong, B., 2020. Synthetic: an individual insurance claim simulator with feature control. arXiv preprint arXiv:2008.05693 .

Balona, C., Richman, R., 2020. The Actuary and IBNR Techniques: a Machine Learning Approach .

Barber, D., Bishop, C.M., 1998. Ensemble learning in bayesian neural networks. Nato ASI Series F Computer and Systems Sciences 168, 215–238.

Baudry, M., Robert, C.Y., 2019. A machine learning approach for individual claims reserving in insurance. Applied Stochastic Models in Business and Industry 35, 1127–1155.

Bergmeir, C., Benítez, J.M., 2012. On the use of cross-validation for time series predictor evaluation. Information Sciences 191, 192–213.

Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. The Journal of Machine Learning Research 13, 281–305.

Bishop, C.M., 1994. Mixture density networks .

Chen, K., Zhou, Y., Dai, F., 2015. A lstm-based method for stock returns prediction: A case study of china stock market, in: 2015 IEEE international conference on big data (big data), IEEE. pp. 2823–2824.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems 2, 303–314.

Delong, L., Lindholm, M., Wüthrich, M.V., 2020. Collective reserving using individual claims data. Available at SSRN .

England, P.D., Verrall, R.J., 2002. Stochastic claims reserving in general insurance. British Actuarial Journal , 443–544.

Gabrielli, A., 2019. A neural network boosted double overdispersed poisson claims reserving model. ASTIN Bulletin 50.

Gabrielli, A., 2020. An individual claims reserving model for reported claims. Available at SSRN 3612930 .

Gabrielli, A., Richman, R., Wüthrich, M.V., 2020. Neural network embedding of the over-dispersed poisson reserving model. Scandinavian Actuarial Journal 2020, 1–29. URL: `http://www.tandfonline.com/doi/abs/10.1080/03461238.2019.1633394`.

Gabrielli, A., Wüthrich, M., 2018. An individual claims history simulation machine. Risks 6, 29. Publisher: Multidisciplinary Digital Publishing Institute.

Gal, Y., Ghahramani, Z., 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, pp. 1050–1059.

Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E., 2015. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. Journal of Computational and Graphical Statistics 24, 44–65.

Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J., 2016. Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems 28, 2222–2232.

Harej, B., Gächter, R., Jamal, S., 2017. Individual claim development with machine learning. Report of the ASTIN Working Party of the International Actuarial Association. Available online: http://www. actuaries. org/ASTIN/Documents/ASTIN_ICDML_WP_Report_final. pdf (accessed on 19 July 2019) .

Hjorth, L.U., Nabney, I.T., 2000. Bayesian training of mixture density networks, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, IEEE. pp. 455–460.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 .

Iso, H., Wakamiya, S., Aramaki, E., 2017. Density estimation for geolocation via convolutional mixture density network. arXiv preprint arXiv:1705.02750 .

Jouko, L., Aki, V., 2001. Bayesian approach for neural networks–review and case studies. Neural networks 14, 257–274.

Kasiviswanathan, K., Sudheer, K., 2017. Methods used for quantifying the prediction uncertainty of artificial neural network based hydrologic models. Stochastic environmental research and risk assessment 31, 1659–1670.

Kuo, K., 2019. Deeptriangle: A deep learning approach to loss reserving. Risks 7, 97.

Kuo, K., 2020. Individual claims forecasting with bayesian mixture density networks. arXiv preprint arXiv:2003.02453 .

Lakshminarayanan, B., Pritzel, A., Blundell, C., 2017. Simple and scalable predictive uncertainty estimation using deep ensembles, in: Advances in neural information processing systems, pp. 6402–6413.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. nature 521, 436–444.

Mack, T., 1993. Distribution-free calculation of the standard error of chain ladder reserve estimates. ASTIN Bulletin: The Journal of the IAA 23, 213–225.

McGuire, G., Taylor, G., Miller, H., 2018. Self-assembling insurance claim models using regularized regression and machine learning. Available at SSRN 3241906 .

Mulquiney, P., 2006. Artificial neural networks in insurance loss reserving, in: 9th Joint International Conference on Information Sciences (JCIS-06), Atlantis Press.

Murray, R.E., Ryan, P.B., Reisinger, S.J., 2011. Design and validation of a data simulation model for longitudinal healthcare data, in: AMIA Annual Symposium Proceedings, American Medical Informatics Association. p. 1176.

Nguyen, H.D., McLachlan, G., 2019. On approximations via convolution-defined mixture models. Communications in Statistics-Theory and Methods 48, 3945–3955.

Noll, A., Salzmann, R., Wüthrich, M.V., 2020. Case study: French motor third-party liability claims. Available at SSRN 3164764 .

Norberg, R., 1993. Prediction of outstanding liabilities in non-life insurance 1. ASTIN Bulletin: The Journal of the IAA 23, 95–115.

Ormoneit, D., Neuneier, R., 1996. Experiments in predicting the german stock index dax with density estimating neural networks, in: IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr), IEEE. pp. 66–71.

Ormoneit, D., Tresp, V., 1996. Improved gaussian mixture density estimates using bayesian penalty terms and network averaging, in: Advances in neural information processing systems, pp. 542–548.

Poon, J.H., 2019. Penalising unexplainability in neural networks for predicting payments per claim incurred. Risks 7, 95.

Radtke, M., Schmidt, K.D., Schnaus, A., 2016. Handbook on loss reserving. Springer.

Reed, R., Marks, R.J., 1999. Neural smithing: supervised learning in feedforward artificial neural networks. Mit Press.

Richman, R., 2018. Ai in actuarial science. Available at SSRN 3218082 .

Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review 65, 386.

Rossouw, L., Richman, R., 2019. Using machine learning to model claims experience and reporting delays for pricing and reserving. Available at SSRN 3465424 .

Rumelhart, D.E., Hiton, G., Williams, R., 1986. Learning representations by backpropagating errors , 533–536.

Schelldorfer, J., Wüthrich, M.V., 2019. Nesting classical actuarial models into neural networks. Available at SSRN 3320525 .

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, 1929–1958.

Tashman, L.J., 2000. Out-of-sample tests of forecasting accuracy: an analysis and review. International journal of forecasting 16, 437–450.

Taylor, G., 2019. Loss reserving models: Granular and machine learning forms. Risks 7, 82.

Taylor, G., McGuire, G., 2004. Loss reserving with glms: a case study, in: Spring 2004 Meeting of the Casualty Actuarial Society, Colorado Springs, Colorado.

Taylor, G., McGuire, G., 2016. Stochastic loss reserving using Generalized Linear Models. volume 3 of *CAS Monograph Series*. Arlington, USA: Casualty Actuarial Society.

Taylor, G.C., 2000. Loss reserving : an actuarial perspective.

Vaughan, J., Sudjianto, A., Brahimi, E., Chen, J., Nair, V.N., 2018. Explainable neural networks based on additive index models. arXiv preprint arXiv:1806.01933 .

Vossen, J., Feron, B., Monti, A., 2018. Probabilistic forecasting of household electrical load using artificial neural networks, in: 2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS), IEEE. pp. 1–6.

Wüthrich, M.V., 2018a. Machine learning in individual claims reserving. Scandinavian Actuarial Journal 2018, 465–480.

Wüthrich, M.V., 2018b. Neural networks applied to chain–ladder reserving. European Actuarial Journal 8, 407–436.

Wüthrich, M.V., 2019. From generalized linear models to neural networks, and back. Available at SSRN 3491790 .

Wüthrich, M.V., Merz, M., 2008. Stochastic claims reserving methods in insurance. volume 435. John Wiley & Sons.

Wüthrich, M.V., Merz, M., 2019. Yes, we cann! ASTIN Bulletin: The Journal of the IAA 49, 1–3.

Yang, L., Shami, A., 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing 415, 295–316.

Zen, H., Senior, A., 2014. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis, in: 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE. pp. 3844–3848.

Zhou, J., Garrido, J., 2009. A loss reserving method based on generalized linear models. Society of Actuaries .