All Actuaries Summit 2026
25 – 27 May 2026, Melbourne

Making all the right noise - Using diffusion deep learning models to perform multivariate prediction for simulation and reserving

Prepared by Hugh Miller, Justin Sik Kwok Wong and Callum Sleigh

Presented to the Actuaries Institute
2026 All-Actuaries Summit
25-27 May 2026

This paper has been prepared for the Actuaries Institute 2026 All-Actuaries Summit.

The Institute's Council wishes it to be understood that opinions put forward herein are not necessarily those of the Institute and the Council is not responsible for those opinions

This paper uses unit record data from Household, Income and Labour Dynamics in Australia Survey, HILDA. HILDA is conducted by the Australian Government Department of Social Services (DSS). The findings and views reported in this paper, however, are those of the author[s] and should not be attributed to the Australian Government, DSS, or any of DSS' contractors or partners. DOI: 10.26193/R4IN30.

© Hugh Miller, Justin Sik Kwok Wong, Callum Sleigh

Actuaries Institute
ABN 69 000 423 656
Level 2, 50 Carrington Street, Sydney NSW 2000
P +61 (0) 2 9239 6100 | actuaries.asn.au

Abstract

As AI adoption accelerates, there is continued interest in adapting ideas to new contexts to see how deep learning models can be applied. For actuaries, there is an ongoing opportunity for new models to handle multivariate prediction problems where there are important dependencies between outputs. Examples include: claims reserving (estimating a suite of accident and development period combinations), pricing (where different risks are estimated and combined into a risk price), microsimulation (where many different characteristics are simulated over time in a consistent way) and DFA models (where different parameters will evolve over time to determine capital positions, with strong correlations across measures).

Diffusion models are a common deep learning approach, particularly in image generation. Starting with random variation, a trained model will successively 'de-noise' a dataset until it resembles something coherent. In many cases they outperform other model structures, including in a small number of research papers testing applicability to tabular data.

Our paper applies diffusion models to two key actuarial problems. First, we apply diffusion to microsimulation, where interrelated outcomes must be jointly predicted. The diffusion setup significantly outperforms the GAN and VAE model approaches explored in our 2025 summit paper (Gains ratio on a discriminator model of 28%, compared to about 70% for prior models), at the expense of speed. Second, we reformulate reserving using actuarial triangles as a deep learning image completion problem and obtain promising results on a database of historical data. Results are reasonable, although some tail misfit is evident – comparable with results seen elsewhere in the literature.

The models represent an important contribution, with the potential to substantially simplify the training of complex systems, as well as augmenting standard actuarial practice. To the authors' knowledge, this is the first time such diffusion models have been applied in an actuarial context.

Keywords: Deep learning, AI, multivariate prediction, diffusion models, actuarial triangles

1 Introduction

1.1 Background and aims of the paper

The time and resources devoted to AI technologies continues to grow. While the cutting edge of large foundational multimodal models is now the domain of large tech firms, there remain substantial opportunities to build and use tailored models in more specialised contexts. Ironically, structured datasets of the type actuaries and data scientists often work with (tabular data with large numbers of variables of different types and scales) receive less attention than the more 'human' contexts such as language, image, video and speech. For structured datasets, generative techniques offer substantial opportunities beyond existing predictive machine learning (ML) models, particularly when distributional understanding is useful, or when dataset complexity requires the use of simulation.

This paper builds on recent work by the authors (Miller et al., 2025). That paper focused on microsimulation applications using two other generative AI models – the variational autoencoder (VAE) and generative adversarial network (GAN). The paper found good performance, with the VAE being the clear winner with more stable training performance and better generation in some contexts.

Our current paper explores the performance of a completely different deep learning model – diffusion models. We apply diffusion models to two contexts:

- **Microsimulation models** – following the setup of our previous paper, we explore whether diffusion models offer superior performance (or other advantages)
- **Traditional actuarial (triangle) reserving** – We reformulate reserving as an image completion problem ('inpainting') and use diffusion models to 'complete the rectangle', using a model trained on learned patterns from completed triangles. We build the model in a flexible way so that it can handle a variety of shapes, lines of business and development patterns.

These contexts are introduced further in sections 1.3 and 1.4.

The implications flowing from success in these areas are profound. In the case of microsimulation models, the opportunity is to replace complex structures that chain together dozens (or hundreds) of individual submodels with a single model that can simulate all variables in a single model generation step. This provides order-of-magnitude improvements in time and effort required to building such models. This means more complex models can be built in more contexts, realising the promise of simulated system models that allow managers to understand system trends and potential interventions.

In the case of reserving, such technology might drastically change how actuaries spend their time:

- Much of actuaries' time is applying 'judgement' to triangle completion models. For instance, deciding how much weight to put on later diagonals, or whether to temper chain-ladder factors based on experience with similar lines of business. Generation, of the type proposed here, means that some of this judgement could be embedded into a model, which can learn and apply such experience (and identify when things are particularly uncertain). Some forms of judgment (such as how an individual insurance portfolio differs from others with similar histories) would remain.
- Valuation risk margins have long been difficult to produce, often relying on heuristics and market practice. However, these are produced by default in a generative context – the reserve is set by looking at hundreds of alternative triangle completions. A model that produces distribution for 'free' is attractive. For example, a model that inherently generates a full predictive distribution during its sampling process, eliminating the need for computationally expensive secondary techniques, such as bootstrapping or manual sensitivity testing.
- Time can then be spent on other value-adding activity (beyond checking model outputs). For instance, there will be specific risks (known issues with claims processing, superimposed inflation risks) that are not embedded in the model, that can motivate adaptions to model output.
- Easy generation also gives greater opportunity for additional analysis. For instance, scenarios of what an insurer's liabilities will be in a year's time are often quite time consuming. But in cases where we 'complete the rectangle', it is straightforward to pick a few representative cases of the next diagonal, fix these, and then generatively complete these conditional rectangles. Detailed scenario work therefore becomes straightforward.

As with other professions, we expect the impact of AI will significantly change how work is done, but not change the need for oversight, governance and informed decision-making.

The remainder of the paper is organised as follows:

- The remainder of Section 1 covers related literature, and introductions to our problems and diffusion models
- Section 2 focuses on the microsimulation problem, divided into methodology, data, results and discussion, including computation considerations
- Section 3 focuses on the triangle completion problem, structured similarly to section 2
- Section 4 provides some final overarching comments.

1.2 Related literature in generative modelling

Borisov et al. (2022) provides a relatively recent survey on deep learning for tabular data including many references, including a range of open challenges – we do not attempt to replicate this broad perspective of existing work here. It is noteworthy that the survey noted no adaptations of diffusion models to tabular data, although there are some instances since 2022. Our approaches to microsimulation problem perhaps most closely align with approaches to variable imputation, in that we effectively recast generative prediction as a type of vector completion. Relevant references include Yoon et al. (2018), Nazabal et al. (2020), Yoon et al. (2020), Telyatnikov and Scardapane (2023) and Sun et al. (2023).

Diffusion models rely on ideas introduced by Sohl-Dickstein et al. (2015), with the key reference for practical diffusion models being Ho et al. (2020). We use denoising diffusion implicit models (DDIM) methods to speed up generation based on Song et al. (2020). Yang et al. (2023) provides a relatively recent survey on diffusion models. Readers seeking a gentle introduction (beyond that provided in this paper) may enjoy Lil'Log's articles.[1]

One important and recent reference for our work is the paper by Yandex researchers on diffusion models for tabular data (Kotelnikov et al., 2023). The code for this work is public[2] as TabDDPM and we have adopted many elements as part of our work, including treatment of categorical variables, continuous variable normalisation and setting the network model size & shape. Our work remains distinctive in the importance of conditional generation, and the very different contexts (longitudinal microsimulation and actuarial triangles). We also note that two more recent papers claim even stronger results for diffusion-based tabular data (Zhang et al., 2023, and Shi et al., 2024).

Deep learning remains a large and rapidly evolving space. As an example, Milligan (2025) is an example of a new R package designed to apply VAEs to tabular data within R, closely related to our previous paper.

1.3 Background to microsimulation models

Microsimulations are models that project a population or system at a very granular level – typically at the level of an individual person. Actuaries have been heavily involved in microsimulation work across Australia and New Zealand, with models spanning welfare, veterans, housing, healthcare, child protection, and other social sector domains. Similar simulations are sometimes called 'agent-based models' or 'digital twins' (see for example, Rasheed et al., 2020); in these contexts models also span tax, disease transmission and traffic. In financial services, microsimulation models are often employed to estimate customer lifetime value, simulating how a customer might remain or churn, and whether they increase their bundle of products held.

Our previous paper (Miller et al., 2025) provides further background, including links to specific microsimulation projects.

Microsimulation models, given their depth (many variables are modelled) and granularity (can define arbitrary cohorts from the individual-level setup) are incredibly flexible tools for understanding systems, monitoring, identifying key cohorts and designing interventions. However, they do carry drawbacks. They are expensive to build, maintain and troubleshoot, in large part due to the large number of component models that need to interact cohesively. Bespoke problems mean transferability is limited. And computing limitations mean that projections can be time consuming. These limitations are what our research attempts to overcome.

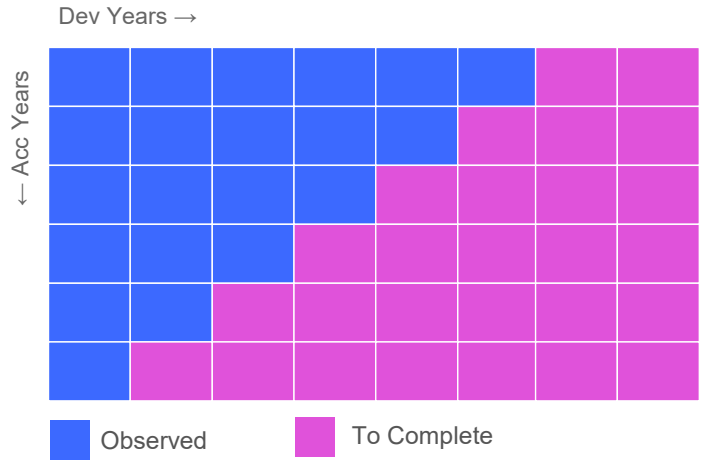
1.4 Background to triangle-based reserving

The concept of reserving triangles is a fundamental part of actuarial work. They occur by arranging aggregated payments (or other measure, such as incurred claims or claim numbers) in a grid with rows representing accident (or premium) year and columns for development year, each diagonal represents a time step. We will 'know' at a given timepoint an upper triangle of data as shown in Figure 1. 'Completion' of the triangle means estimation of the lower portion of the triangle. Typically the resulting shape is a square, but not always – the example shows a situation where additional development years are required for estimation creating a rectangle.

[1] https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

[2] https://github.com/yandex-research/tab-ddpm

Figure 1 – Basic triangle layout for reserving exercises



There are a range of standard techniques for estimation, with deterministic and stochastic methods a key distinction.

Classic papers in reserving include Mack (1993) on the stochastic chain ladder, Bornhuetter & Ferguson (1972) on their eponymous method, England & Verrall (1992) on their stochastic conceptualisation of reserving models. Key texts included Taylor (2000) and Hindley (2017).

Meyers (2015) is noteworthy for its Bayesian treatment of stochastic reserving, as well as the database of completed claims triangles that was made available – we use this database in our work for this paper.

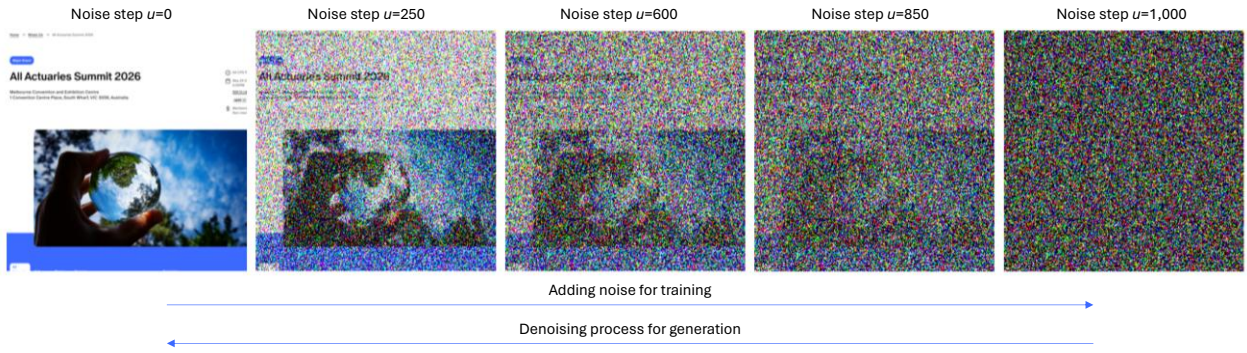
1.5 Intuition for diffusion models

While specific models are introduced in Sections 2.2 and 3.2 for our applications, this section provides some intuition and motivation for diffusion models.

Diffusion models are a class of generative models inspired by ideas from physics. In particular, the physics underlying how substances mix and spread over time. For example, a drop of blue ink placed in a glass of water will slowly diffuse through the water until the entire glass is a uniform blue, or heat in a metal plate will slowly diffuse through the entire plate. These are all physical processes where a complicated, structured state is slowly transformed into a random, unstructured state; whatever the initial distribution of ink in the glass, the final distribution of ink in the glass is uniform.

In the context of images (a common use-case for diffusion models), the process of adding noise involves steadily distorting an image until it's a random mass of pixels, as shown in the forward process of Figure 2.

Figure 2 – Noise being added to an image, to help learn the denoising process of a diffusion model



The connection to generative modelling is that it is possible to learn the reverse process, where a random and unstructured distribution (like a uniform or Gaussian distribution) gets transformed into a

structured distribution. In the case of the figure, the initial pixel configuration is random colours, and steps are applied to resolve this into something meaningful in the denoising process.

To make this work requires more than just standard likelihood estimation, the main insights needed are:

1. Breaking down the forward/reverse diffusion process into a series of small discrete time-steps. The forward process consists of adding a small amount of noise at each time-step. In this setup we can think of a path through time emanating from any data point and ending up somewhere described by a Gaussian.
2. We train the model to look at these forward trajectories and find those which match typical trajectories *in reverse*.

This process is fairly general, and so can be applied to different types of problems, albeit with different setups. Section 2 uses diffusion for structured tabular data, which involves a relatively classical neural network structure at its core. Section 3 treats claims rectangles as images (each cell corresponds to a pixel), so a structure suitable for image generation is used instead.

2 Microsimulation

2.1 Notation and problem definition

For convenience we will refer to our unit of modelling as a 'person'. Suppose our microsimulation has p 'dynamic' variables $\tilde{X}_t = (X_{1,t}, X_{2,t}, \dots, X_{p,t})$ that are evolving non-deterministically for each person that we observe at time t. Our task is to estimate these variables at the next timestep $t+1$. Assume also we have 'static' variables $\tilde{S} = S_1, S_2, \dots, S_q$ that are constant for an individual (e.g. sex) or evolve deterministically (e.g. age).

We can express our task as estimating the following distribution f:

$$f\left(\tilde{X}_{t+1} \mid \tilde{X}_t, \tilde{S}\right) = f\left(X_{1,t+1}, X_{2,t+1}, \dots, X_{p,t+1} \middle| X_{1,t}, X_{2,t}, \dots, X_{p,t},\ S_1, S_2, \dots, S_q\right)$$

The key complicating factor is that these variables are not independent; for example if a person enters hospital, that changes their probability of also receiving welfare benefits. A natural approach used in existing microsimulation models is to treat this as a chained conditional probability problem. First estimate the distribution for $X_{1,t+1}$:

$$f\left(X_{1,t+1} \middle| \tilde{X}_t, \tilde{S}\right)$$

Then estimate $X_{2,t+1}$ conditional on $X_{1,t+1}$ (as well as the other values at time t):

$$f\left(X_{2,t+1} \middle| \tilde{X}_t, \tilde{S}, X_{1,t+1}\right)$$

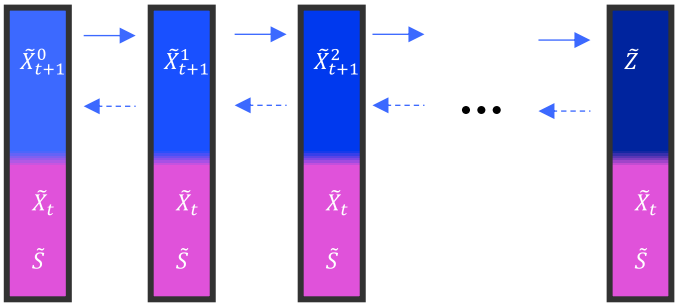
This can be continued, estimating each variable in sequence. The sequential approach creates the long list of sub-models described in the previous section.

A second complicating factor is that we are genuinely interested in **distributions**, not just expected values across the distribution. Since we are simulating, we need the projection to take plausible different values. This is more challenging than most machine learning approaches that focus on a particular metric (such as the mean) without regard for distribution.

2.2 Model structure

Diffusion models consider a process that takes the distribution of our input variables $\tilde{X}_{t+1}$ and sequentially adds noise, ending up with pure random noise $\tilde{Z}$, as per the figure below. The addition of noise is often indexed by a 'time' variable t, but we use u to distinguish from our main time index in our microsimulation. By studying how the variables change as noise is added, it can build a model to understand the reverse process (removing noise). Once trained, running the denoising model successively can take randomly generated noise $\tilde{Z}$ to a generated $\tilde{X}_{t+1}$. We need to condition on $\tilde{X}_t, \tilde{S}$ throughout.

Figure 3 – Process for adding and subtracting noise



More formally, if $\tilde{X}_{t+1} \sim f(\tilde{X}_{t+1} \mid \tilde{X}_t, \tilde{S}) \equiv q(\tilde{X}^0_{t+1})$ (dropping the conditioning for convenience) is the starting distribution we define the forward diffusion process as adding some Gaussian noise at each step according to a schedule β_u

$$q(\tilde{X}^u_{t+1} | \tilde{X}^{u-1}_{t+1}) = \mathcal{N}(\sqrt{1-\beta_u}\, \tilde{X}^{u-1}_{t+1}, \beta_u \boldsymbol{I})$$

The variance schedule β_u is a sequence of noise magnitudes $\{\beta_u\}_{u=1}^U$ where each $\beta_u \in (0,1)$, representing the amount of Gaussian noise added at each individual step u to incrementally obscure the data.

It is possible to express later noise stages in terms of the starting time $u = 0$,

$$q(\tilde{X}^{1:U}_{t+1} | \tilde{X}^0_{t+1}) = \prod_{u=1}^{U} q(\tilde{X}^u_{t+1} | \tilde{X}^{u-1}_{t+1}) = \mathcal{N}(\sqrt{\alpha_U}\, \tilde{X}^0_{t+1}, (1-\alpha_U) I)$$

where $\alpha_u = \prod_{i=1}^{u}(1-\beta_i)$. U signifies the total number of diffusion steps in the process.

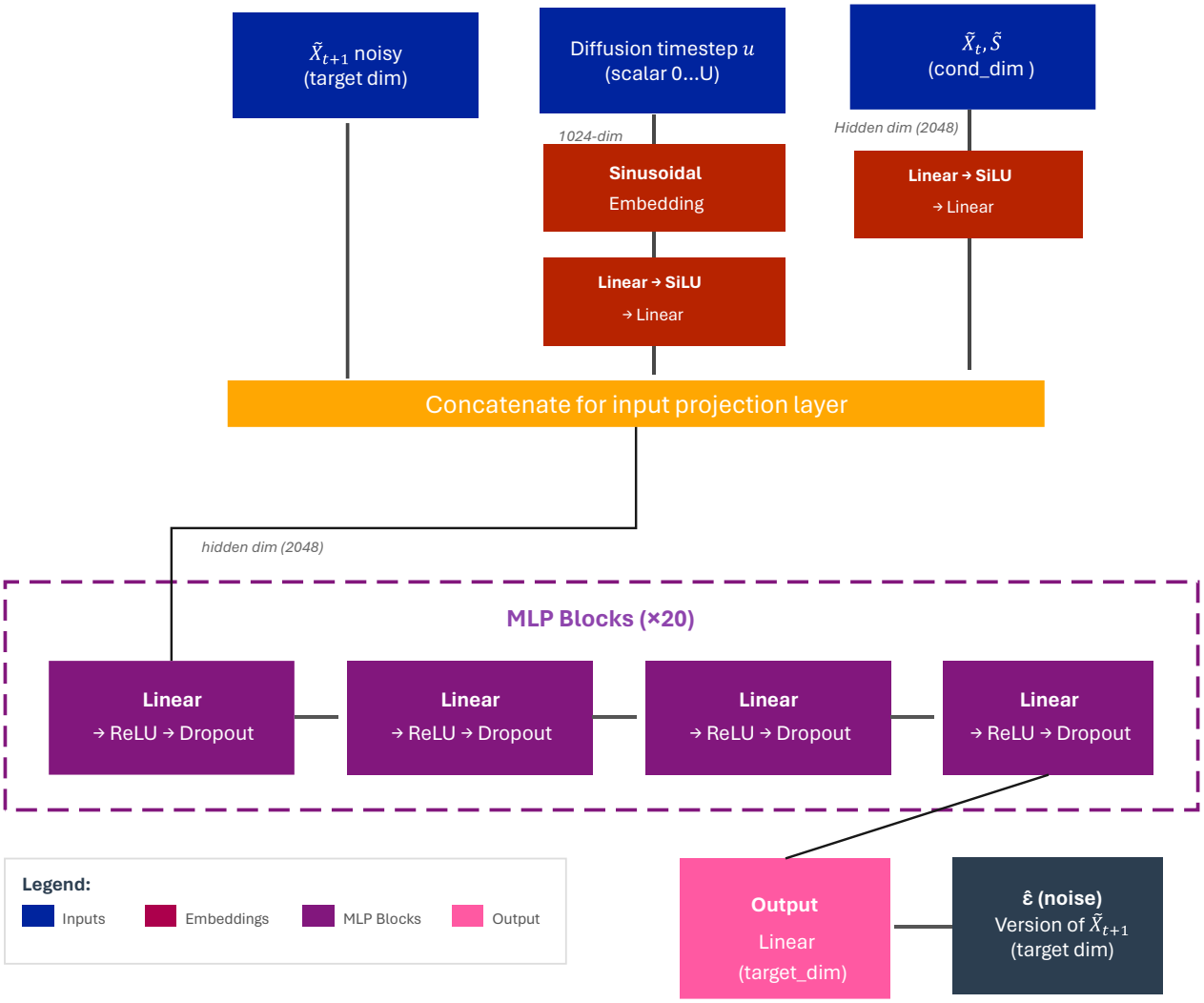
We are interested in the reverse process $p(\tilde{X}^{u-1}_{t+1} | \tilde{X}^u_{t+1})$, which can be approximated with a similar Gaussian process (but with learned parameters controlling the mean and correlations of the noise).

Training is subject to a loss function, based on the negative log-likelihood (and equivalent to a multivariate mean-squared error loss). Gradients of the loss function are used to update parameters of the neural network model for random points of the learning schedule $1, \ldots, u, \ldots U$.

We have adopted the structure for the underlying denoiser model as shown in Figure 4. It comprises:

- An initial stage where information about the current target variable (still carrying noise), conditional variables and diffusion timestep u information are combined. The transformations of u are relatively standard and give the model the ability to have differentiation on how to denoise at different stages of the process.
- The main stage of multilayer perceptrons that allow complex interactions between variables to be applied. It includes a linear combination followed by application of a rectified linear unit (ReLU) and 10% dropout.
- An output stage recovering the required dimension of the variable being estimated, and a noise estimate, which can be subtracted off the running estimate of $\tilde{X}_{t+1}$.

Figure 4 – Adopted structure for diffusion model structure for denoising, HILDA microsimulation



2.3 Implementation considerations

We note a few practical issues in the model design

Categorical variables

Deep learning models are typically designed for continuous inputs. It is important the model works for categorical variables as well as continuous variables, since these are common in modelling. We follow the treatment of Kotelnikov et al. (2023) for categorical variables. Categorical variables are one-hot encoded, and the noise process adds small amounts of uniform noise to the distribution (as implied by the one-hot encoding), so that by the end of the diffusion process (step U) the variable is uniformly distributed across the categories of the variable.

The loss function requires modification for the categorical variables, and is a combination of squared loss for continuous components plus Kullback–Leibler (KL) divergence for categorical variables.

Variable normalisation

Deep learning models generally train best when continuous variables are on similar scales. We transform continuous variables empirically to a normal distribution[3], which provides robustness against outliers and other potential oddities in the input data.

A common practical challenge arises when working with variables that exhibit mixed distributions. This occurs frequently in the HILDA data and is discussed in greater detail in Section 2.5.1. Such variables typically combine a point mass with a continuous component, making them poorly suited to standard distributional assumptions and difficult for machine learning models to learn effectively. In particular, heavily imbalanced point masses can lead to model collapse. To address this issue, we introduce and apply what we term the *gapped quantile transformer*.

This approach first introduces noise around the point mass, effectively smoothing it into a local continuous distribution, before applying a quantile transformation. A gap is then inserted around the point mass in the transformed space. The resulting representation converts a mixed distribution with a point mass into a bimodal distribution separated by a gap, which we hypothesise is easier for the model to learn.

This is a departure from our previous paper, where normalisation was done using mean and standard deviation scaling.

One key limitation of the quantile transformer (including our gapped quantile transformer) is that the model is not able to generate outputs that exceed the observed range. For example, if the training data only includes incomes up to $200,000, then upon inference, the model will not be able to produce outputs exceeding $200,000. In many cases this will be appropriate, but in others a variant of the transformation (setting a different arbitrary limit on the transformation) would be needed to better replicate feasible values.

Other parameters

In **model loss calculations**, there are effectively two loss functions – one for continuous variables and one for categorical. We have found through trial and error a weight w of 100 appears to balance the two components when combining into an overall loss function:

$$Loss_{overall} = Loss_{continuous} + w \times Loss_{categorical}$$

In our simulated example we use a learning rate of 0.001, 400 epochs and a batch size of 256. For our real-data example we use a lower learning rate 5e-05, 1,000 epochs and a batch size of 512.

Generation speed

We consider two approaches to reversing the diffusion process for data generation: Denoising Diffusion Probabilistic Models (DDPM) and Denoising Diffusion Implicit Models (DDIM).

DDPM operates as a stochastic Markov chain, reconstructing data through hundreds or thousands of probabilistic steps; while this "random walk" ensures high diversity and accurately captures the full distribution of the training data, it is computationally slow.

In contrast, DDIM reformulates the math as a non-Markovian, deterministic process that follows a fixed "probability flow" trajectory. By removing the noise injection at each step, DDIM allows the model to take much larger "strides", enabling sampling in as few as 20 to 50 steps.

[3] We use the `QuantileTransformer(output_distribution='normal')` function in the sklearn Python package

Consequently, DDPM is best suited for scenarios where statistical diversity is the top priority and time is not a constraint, whereas DDIM is the preferred choice for practical applications requiring rapid generation speeds or deterministic results from a specific noise starting point.

In our testing, we found that DDIM sampling performed well for triangle modelling; however, the additional complexity of the microsimulation models meant that DDPM produced significantly better samples. In particular, the DDIM appeared worse at coherently resolving categorical variables.

Code generation

We used an AI coding assistant (Claude) at various points of the project to speed development. The authors remain responsible for overall quality of the code, model testing and the results.

Diffusion schedule

We use a cosine schedule along the lines of Nichol & Dhariwal (2021). If α_u represents the cumulative amount of the original signal remaining with $\beta_u = 1 - \alpha_u/\alpha_{u-1}$, then we define $\alpha = \cos\left(\frac{u+0.008}{1.008} \cdot \frac{\pi}{2}\right)^2$. This is a smooth curve that adds less noise in early stages (relative to a linear schedule), which researchers suggest leads to a more uniform loss of information across the schedule, improving training.

Sinusoidal time embeddings

The time embeddings allow the model to apply different approaches to denoising at different parts of the schedule, vaguely analogous to basis expansion in more traditional regression work. The embedding itself is a large number (64) of sine and cosine functions at different frequencies, ranging from 0.1 radians over the schedule through to 1000 radians. For i in 1, … , 64 the i^{th} pair of embeddings at timestep u are:

$$\sin\left(\frac{u}{10000^{\frac{i}{64}}}\right),\ \cos\left(\frac{u}{10000^{\frac{i}{64}}}\right)$$

Computation setup

As with most deep learning models, training and generation are significantly faster on a graphics processing unit (GPU) All models were trained in Python using the PyTorch package. Models were trained on a workstation with entry-level hardware, including:

- GPU: NVIDIA T1000 8GB
- CPU: Intel Core i7-13700
- RAM: 32GB

Towards the end of the project, we also investigated the performance of the model on a modern setup, with the following key specifications:

- GPU: NVIDIA RTX PRO 4000 Blackwell 24GB
- CPU: Intel Core Ultra 9 285K
- RAM: 64GB

Further discussion on our computational setup is included below in Section 2.5.3.

2.4 Simulation exercise

Before the full microsimulation setup (covered in section 2.5), we first apply the approach to a synthetic example. We remove the microsimulation time dimension, so the problem is one of conditional generation where we are provided with four continuous variables $(X_1, \dots, X_4)$ and two categoricals C_1, C_2,

and must generate two new continuous variables and one categorical (X_5, X_6, C_3). These are all generated in a way to give non-trivial inter-variable dependencies and nonlinearities.

The six continuous variables $(X_1, \dots, X_6)$ are initially drawn from a multivariate normal $\mathcal{N}(0, \Sigma)$ with $\Sigma_{ii} = 1$ on the diagonal and $\Sigma_{ij} = 0.5$ for $i \neq j$. We then apply the following transformations:

- $X_1 \leftarrow X_1 + 4$
- $X_2 \leftarrow (X_2 + 2)^2 + 4$
- $X_3 \leftarrow 4X_3$
- $X_4 \leftarrow 4X_4$
- $X_5 \leftarrow (X_1 - 5)^2 + X_2 - 2X_3 + (1.5X_5 + 1)^2$
- $X_6 \leftarrow 0.5X_4 + 2\sin\left(\frac{X_5}{4}\right) + 1 + 0.7X_6$.

For the categorical variables,

- C_1 has three categories with relative probabilities proportional to $\exp(-0.5\ (X_1 - \mu_k)^2)$ for $\mu_k \in \{3,4,5\}$
- C_2 has five categories with probabilities proportional to $\exp\big(-0.3(X_3 - \mu_k)\big), with\ \mu_k \in \{-8, -4, 0, 8\}$
- C_3 has four categories. The base probabilities are either $(0.4, 0.3, 0.2, 0.1)$, $(0.2, 0.3, 0.3, 0.2)$, $(0.1, 0.2, 0.3, 0.4)$, depending on the value of C_1. These probabilities are then permuted with a numpy.roll command with values [-1,0,1], based on the integer value of $\min\ (1, \max\ (-1, (X_4 - med(X_4))/10)\ ))$. Then one of the categories is increased by 0.1, depending on the value of C_2.

Training was performed on 10,000 observations and 400 Epochs, and took a few minutes to train, after which predictions were generated from the same training set. One-way statistics and distributions for generated variables are shown in Table 1, Table 2 and Figure 5. Selected two-way relationships are shown in Figure 6 and Figure 7.

Table 1 – Summary statistics for original and generated continuous variables

	X_5		X_6	
	Original	Generated	Original	Generated
Mean	14.27	14.25	0.96	0.98
Std Dev	8.48	8.51	2.88	2.90
Skewness	0.37	0.44	0.00	0.02

Table 2 – Category probabilities for original and generated values of C_3

Category	Original Prob	Generated Prob	Abs Diff
1	0.216	0.215	0.001
2	0.273	0.271	0.002
3	0.281	0.294	0.013
4	0.230	0.220	0.010

Figure 5 – Original and generated histograms for continuous variables, simulated example

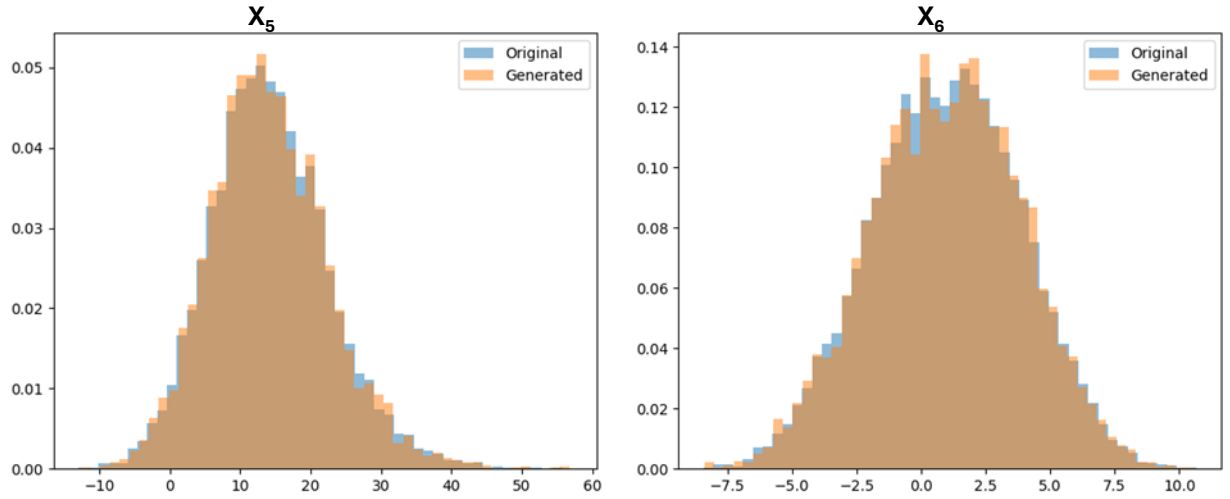


Figure 6 – Two-way plots, original and generated, for selected pairs of variables

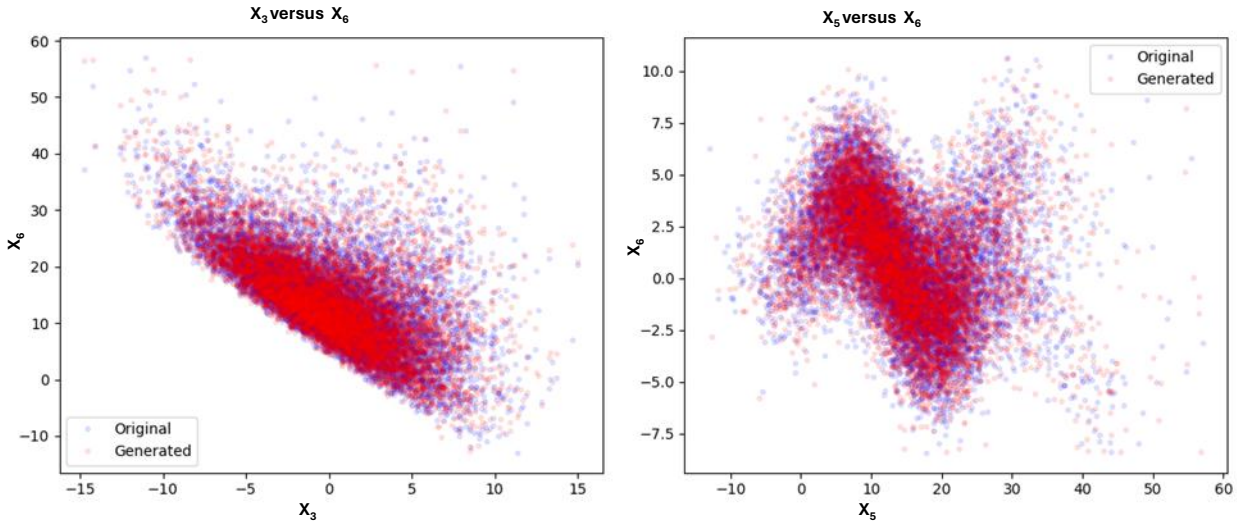
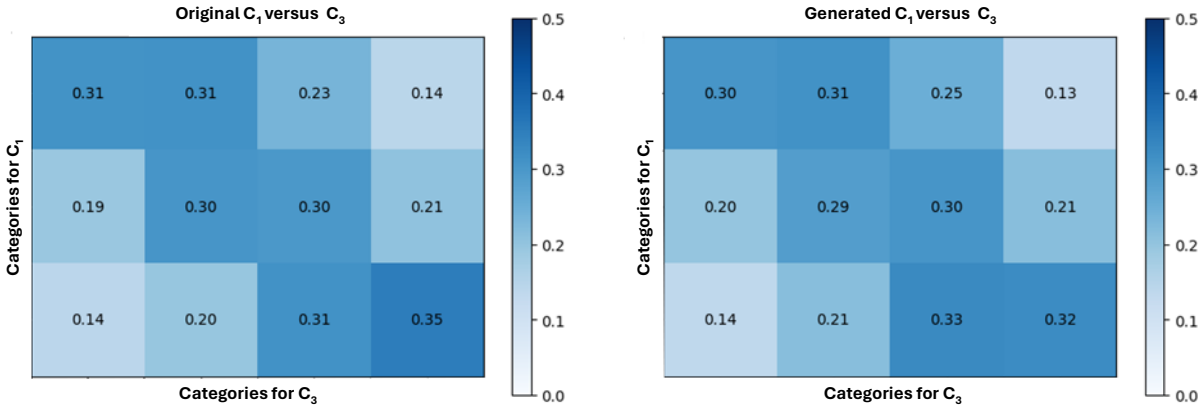


Figure 7 – Two-way categorical probability plots, original and generated, values of C_1 and C_3



We regard the results as very good – distributions and probabilities match well, and some complex structures (including nonlinear correlations) appear to be well captured by the model. Specifically we note:

- Close alignment between one-dimensional summary statistics for continuous and categorical variables
- Good alignment on the histograms (one-way distributions) for the generated continuous variables
- Good capture of complex two-way dependencies. The X_5 variable respects the slanted lower bound shape from the X_3 variable, and the Sine curve relationship between X_5 and X_6 is also well captured
- The strong dependence on C_1 by the generated categorical variable C_3 is reproduced well.

2.5 Real-world data

2.5.1 Introducing HILDA

The Household, Income and Labour Dynamics in Australia (HILDA) survey is a household-based study and is the leading longitudinal survey in the country. It captures data across a range of domains, including work, education and training, demographics, fertility and children, health and welfare usage. It is a longitudinal dataset, captured each year since 2001 by the Melbourne Institute, with Commonwealth funding.

HILDA release 24 (results up to 2024) was used for this project. We use survey responses from 2017 to 2023 for modelling (dates aligning with our previous paper for comparability). Given in a row of data we require both time t and $t+1$, this corresponds to five years of data and 76,400 rows of data (with another 18,200 rows of data withheld and used as a holdout dataset on our diagnostics). The survey data includes both *responding persons* (those members of a household who responded to the survey) and *enumerated persons* (members of a household who did not respond); we only use survey responses from *responding persons*.

Our choice of HILDA is threefold:

- It is longitudinal, meaning that we can setup the time t→t+1 generation and also test chaining generations over multiple time periods (as is done in microsimulation).
- It contains a broad range of variables (and types of variables), similar to those typically used in microsimulation modelling.
- The data is of good quality and has undergone an extensive cleaning and imputation process.[4]

We selected a subset of HILDA variables, consistent with our previous paper, that provide a challenging modelling task across variable types and interdependencies – see Table 3. Of the 24 variables, two were deterministic (age and sex), and the other 22 dynamic, of which six were continuous and the remainder categorical. After one-hot encoding this was 74 dynamic variables, and after attaching 74 variables relating to time $t+1$ the final full vector contained (2+74+74=) 150 variables.

[4] Further detail available at https://melbourneinstitute.unimelb.edu.au/hilda/for-data-users/user-manuals

Table 3 – Variables used from HILDA for modelling

HILDA Variable ID	Short descriptor	Description
hhiage	Age	Age last birthday at date of interview
hgsex	Sex	Sex of respondent
mrcurr	Marital status	Current marital status
hhs3add	SEIFA decile (location)	SEIFA relative socioeconomic advantage/disadvantage decile 2021
hgndi	NDIS flag	NDIS-agreed support package
hglth	Disability flag	Long-term health condition, disability or impairment
cccinhh	Children flag	Respondent has children aged under 14 in the household
ccftb	FTB flag	Respondent receives family tax benefit
edhigh1	Education level	Highest level of education attained
bncap	Age pension flag	Respondent currently receiving aged pension
bncapui	Benefits amount	Value of current weekly public transfers excluding family tax benefit, imputed
bnccrp	Carer flag	Respondent currently receiving carer payment
bncdsp	DSP flag	Respondent currently receiving disability support pension
bncdvaa	DSP amount	Value of latest disability support pension payments
bncnws	Jobseeker flag	Respondent currently receiving Jobseeker
bncpar	Parent Payment amount	Value of latest weekly parenting payment
tifefp	Total annual income	Total regular gross income in the last financial year, including transfers/welfare
wscei	Weekly wages	Current weekly gross wages and salary across all jobs, imputed
wscmtoj	Multiple job flag	Respondent is currently working in more than one job
es	Employment status	Respondent's current employment status
esbrd	Labour force status	Respondent's current labour force status
jbcasab	Casual flag	Casual worker
jbcmocc	Change in job flag	Occupation changed since last interview
rtyr	Year retired	Year retired
gh1	Health rating	Self-assessed health (Excellent / Very good / Good / Fair / Poor)
losat	Life satisfaction	Respondent's life satisfaction (varying from 0 to 10, where 0 is not satisfied and 10 is most satisfied)

2.5.2 Results

2.5.2.1 Diffusion results on HILDA – one-year time-step

As in our previous paper, our preferred overall diagnostic for model fit is based on what we term the *GBM discriminator* model. This involves concatenating generated and actual data, creating a new target indicating whether a line is generated, and training a tree-based gradient boosting machine (GBM) to predict whether a row is generated data. Under this framework, superior generative performance is indicated by poorer discriminator performance, as reflected in a lower gains ratio, since this implies that the model struggles to distinguish synthetic data from observed data.

We first assess model performance in a single time-step setting, focusing on the ability of the model to predict population outcomes one period into the future. For the single time-step model, the GBM

discriminator records a gains ratio[5] of 27.8%, representing a substantial improvement relative to both the GAN and VAE models, which exhibited gains ratios close to 70% (Miller et al., 2025).

Figure 8 – gains chart on holdout data for diffusion model

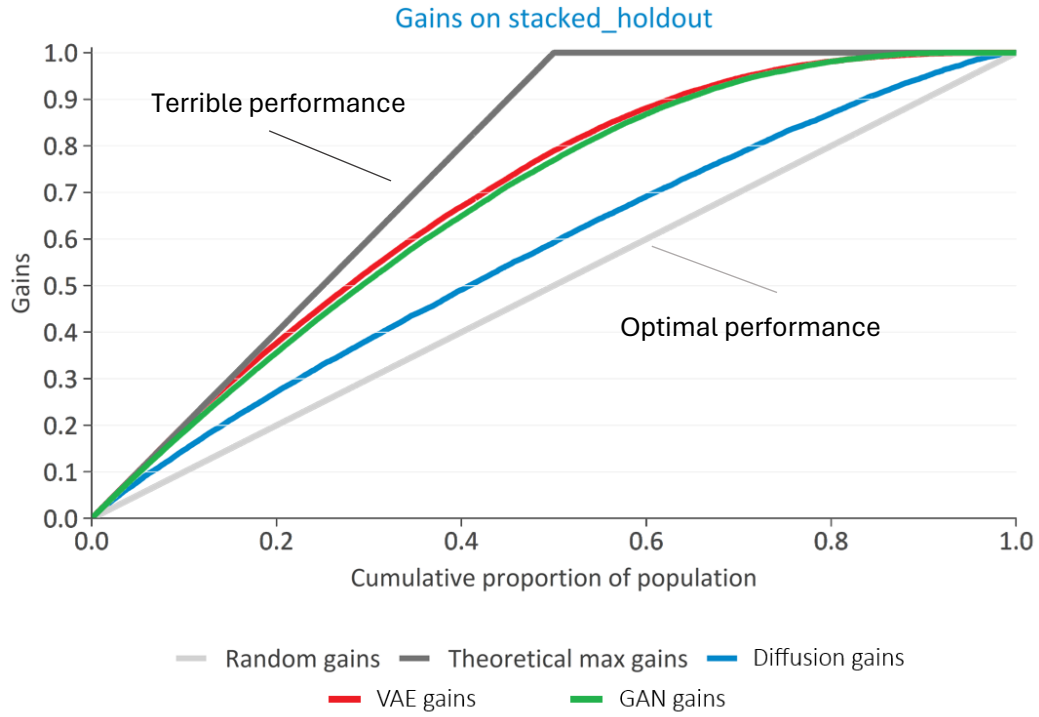


Table 4 – comparison of GBM discriminator gains ratio by model

Model	Gains ratio (lower is better)
GAN	0.711
VAE	0.756
Diffusion	0.278

In addition to overall performance, the GBM discriminator provides measures of relative variable importance, indicating which variables contribute most strongly to distinguishing between real and generated observations. Consistent with the results for the GAN and VAE models, the most influential predictors are continuous variables, indicating there may be some misfit that allows the model to distinguish real and simulated records.

In addition to the discriminator-based evaluation, we examine one-way fits of the generated data against the holdout set. For categorical variables, model fit is assessed using histograms and the total variation distance statistic (TVD), with higher values (closer to one) indicating better agreement between real and generated distributions.

$$TVD = 1 - \frac{1}{2}\sum_{n=1}^{N} |P_n - Q_n|$$

[5] In the context of model evaluation, the gains ratio is a metric to assess how effectively a model's predicted probabilities or scores differentiate between outcomes compared to a random baseline. It is often visualized through a gains chart, where the ratio represents the proportion of the total positive outcomes captured by the model at a specific population threshold relative to the total possible outcomes. Further details available online.

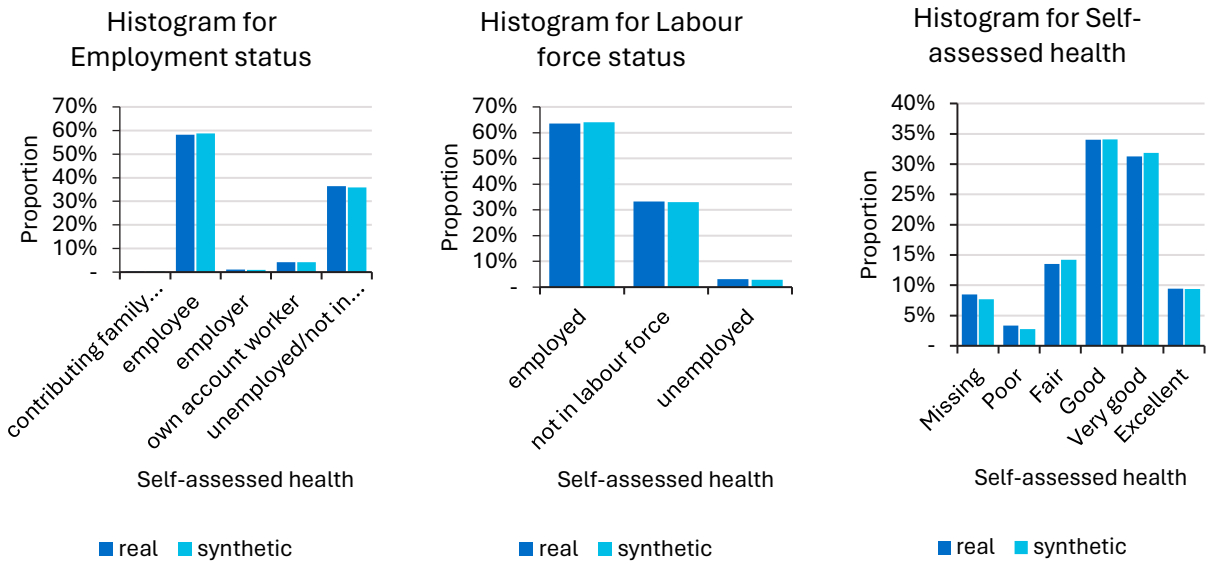
In this context, P_n and Q_n represent the proportions of records belonging to the nth category in the holdout dataset and the generated synthetic dataset, respectively.

Table 5 – Total variation distance goodness of fit statistic for categorical variables, compared across diffusion, GAN and VAE models

Categorical variable	Diffusion	GAN	VAE
Children flag	1.000	0.987	0.992
FTB flag	0.997	0.998	0.999
Education level	0.996	0.994	0.987
Age pension flag	0.999	0.993	0.997
Employment status	0.994	0.961	0.986
Labour force status	0.995	0.968	0.988
SEIFA decile	0.992	0.978	0.996
Carer flag	0.998	0.999	0.993
DSP flag	0.999	0.999	0.999
Jobseeker flag	0.997	0.989	0.994
Multiple job flag	0.997	0.981	0.996
Casual work flag	0.995	0.996	0.990
Change job since flag	0.995	0.998	0.978
Retired flag	0.998	0.999	0.996
Marital status	0.995	0.972	0.992
NDIS flag	0.999	0.998	0.996
Disability flag	0.992	0.888	0.997
Health rating	0.986	0.925	0.959

Table 5 shows the TVD statistic calculated on holdout data, and compares this against the GAN and VAE models. Cells highlighted in green show the model with the best TVD for each variable. Like the GAN and VAE, the diffusion model shows strong one-way performance, and is superior or performs similarly to the GAN and VAE for all variables. In particular, the diffusion model shows a marked improvement over the previous models for employment status, labour force status and the self-assessed health rating. We show histograms for these variables below.

Figure 9 – histograms for selected categorical variables, real and generated (synthetic)



For continuous variables, we assess fit using frequency polygons, quantile–quantile plots, and the Kolmogorov–Smirnov statistic (KS), where values closer to one similarly indicate a better fit.

$$KS = 1 - \sup_x |F(x) - G(x)|$$

The KS statistic is a relatively tough statistic, focusing on the 'worst case' mismatch in distributions.

Table 6 – Kolmogorov-Smirnov goodness of fit statistic for continuous variables, compared across diffusion, GAN and VAE models

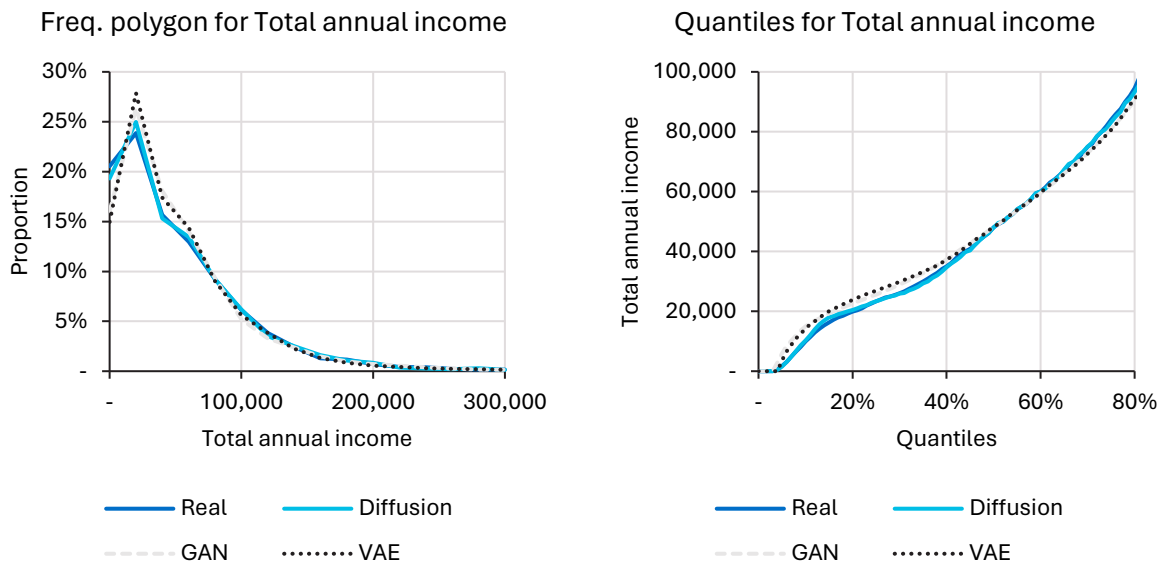
Continuous variable	Diffusion	GAN	VAE
Total annual income	0.980	0.957	0.940
Weekly wages[(b)]	0.782	0.591	0.982
Benefits amount[(b)]	0.619	0.272	0.274
DSP amount[(b)]	0.497	0.004	0.999
Parenting amount[(b)]	0.493	0.983	0.019
Life satisfaction	0.982	0.862	0.897

(a) 1 represents perfectly matching distributions.

(b) KS scores strongly affected by the handling of the mixed/skewed distribution

The weekly wages, benefits, DSP and parenting payment amounts variables have distributions which are strongly skewed by the point mass (in each case, at 0). That is, the majority of persons in the dataset have 0 payments for these variables. While the GAN and VAE appear to outperform the diffusion model (as specified by the KS statistic), this is to be expected because of the structure used to specify these variables. In the GAN and VAE models, zero quantities are modelled explicitly using a flag. In the diffusion model, heavily zero-inflated variables are transformed into bimodal distributions using the gapped quantile transformer, as discussed in Section 2.3. The diffusion model thus implicitly learns the shape of the distribution, as opposed to the point mass being explicitly stated in the GAN and VAE models. For variables which are not heavily zero-inflated (total annual income and life satisfaction), the diffusion model achieves superior performance.

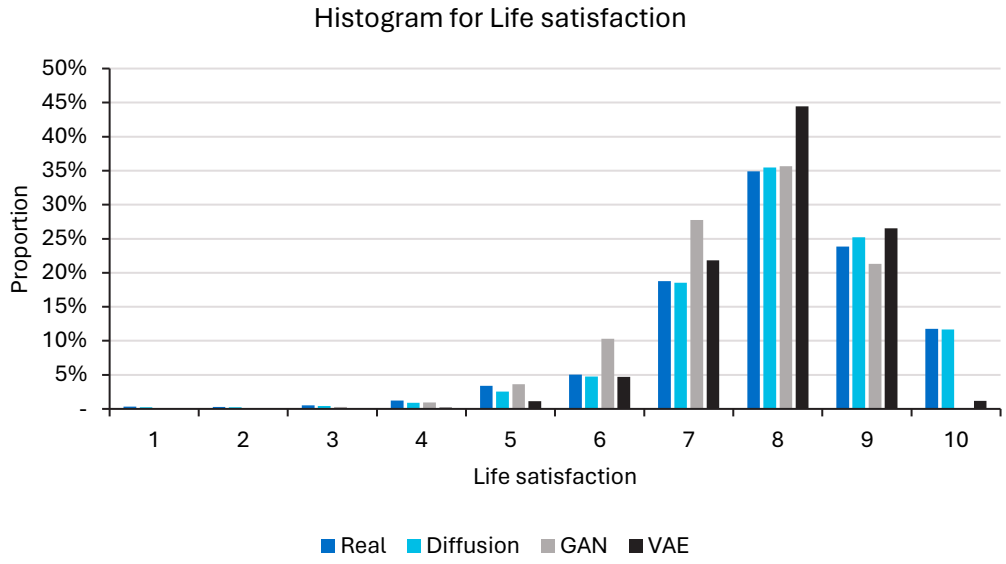
Figure 10 – Frequency polygon and quantile plots for Total annual income



Note: Plots truncated at $300,000 / 80th percentile respectively for better visibility of important parts of the distribution.

In our previous work, we observed that the GAN and VAE models failed to model the top of the distribution for the life satisfaction variable (rarely generating a score of 10). Figure 11 shows the diffusion model's superior performance in modelling this variable.

Figure 11 – Distribution of life satisfaction score for diffusion model



We also consider the interactions between variables, using a number of heat plots. We observe that the diffusion model captures two-way distributions well.

In the heat plots, we show real distributions on the left, synthetic distributions on the right, and the legend represents the number of observations. Ideally, the synthetic distribution will mirror the real distribution accurately by capturing interactions between the two variables shown.

Figure 12 – heat plot of total annual income against wages

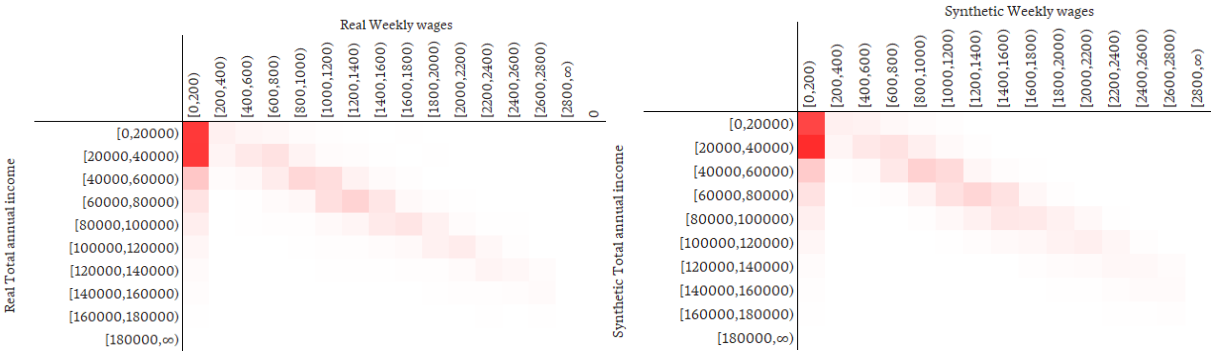


Figure 13 – receipt of age pension by age (deterministic)

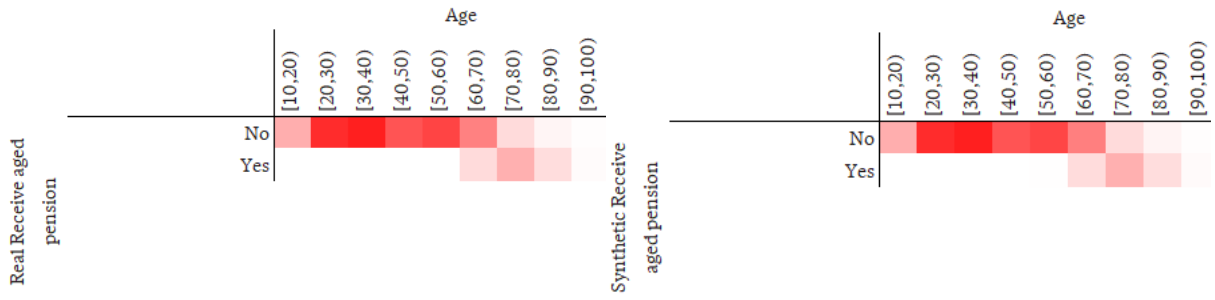


Figure 14 – weekly wages by labour force status

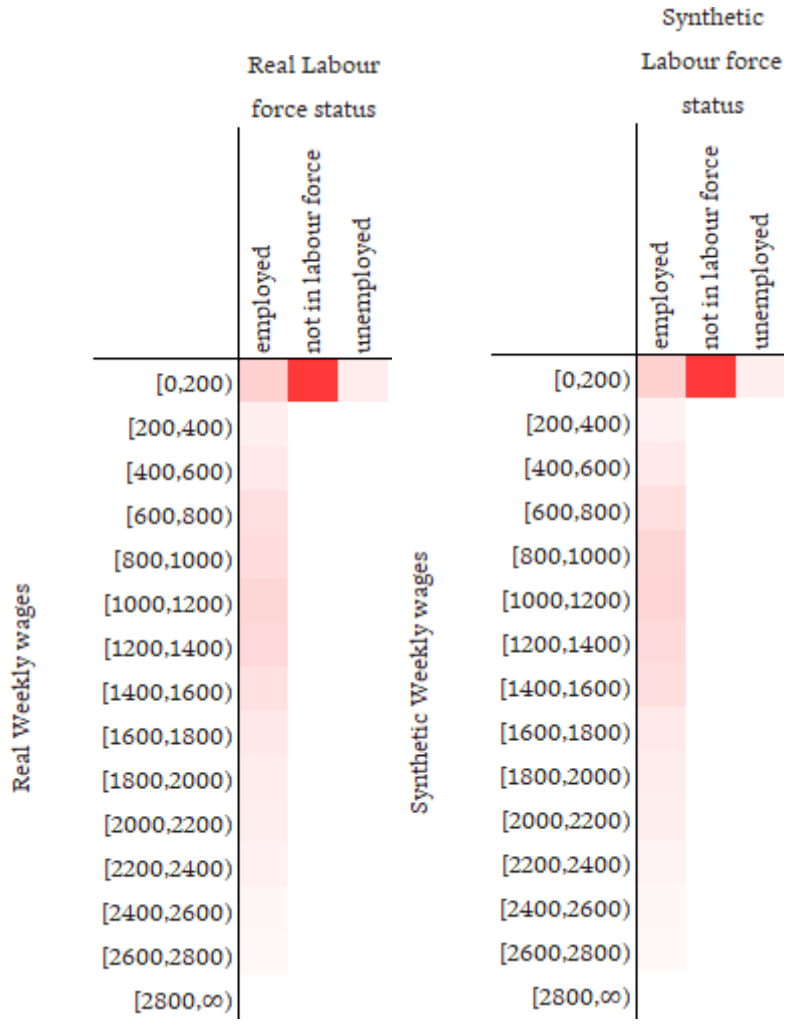
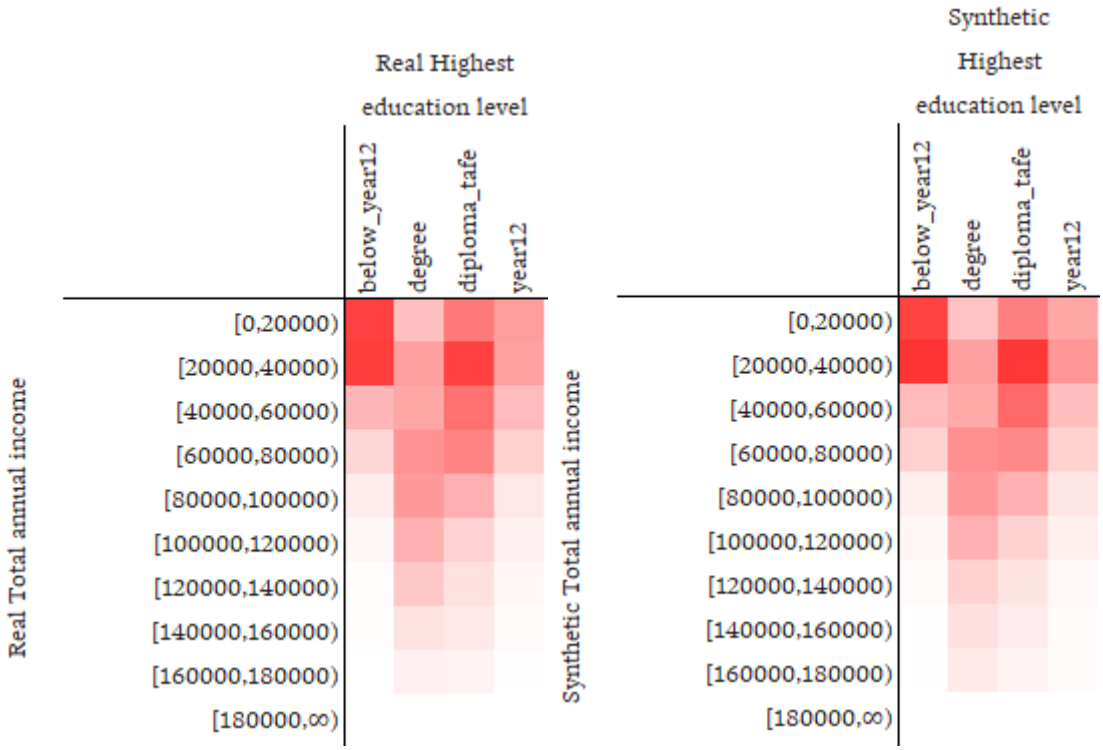


Figure 15 – total annual income by highest level of education



2.5.2.2 Diffusion results on HILDA – three-year time step

As per our previous work, we test the diffusion model on an equivalent dataset, constructed using HILDA data between 2017 and 2020 with 46,138 rows in the training data and 10,908 rows in the holdout. We then investigate the performance of a three-year 'chained' diffusion model (applying the generator three times successively), using the same model described in Section 2.5.2.1.

Table 7 – comparison of GBM discriminator gains ratio by model

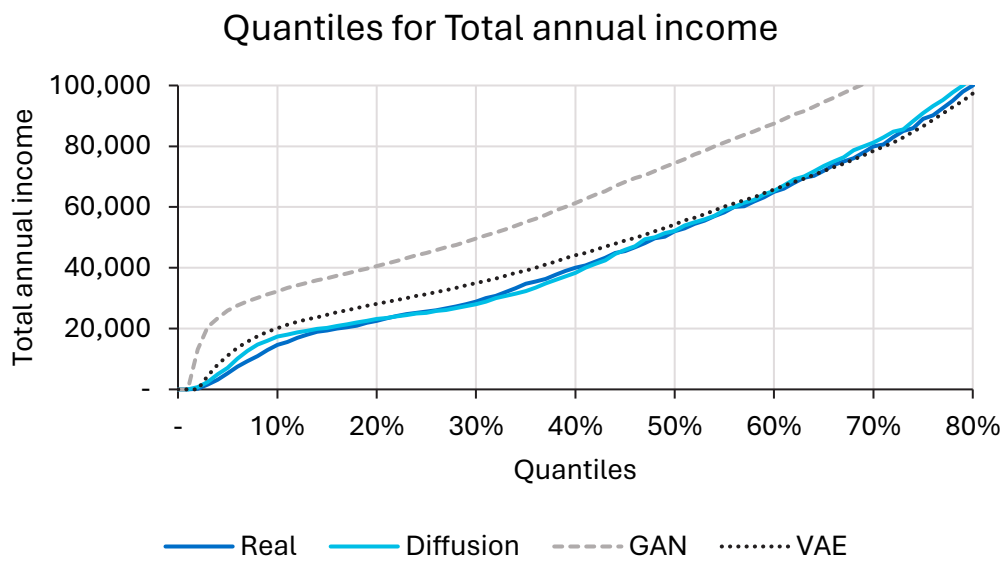
Model	Gains ratio (lower is better)
GAN	0.927
VAE	0.782
Diffusion	**0.420**

Table 7 shows the overall performance of the diffusion model in predicting three years ahead, as measured by the GBM discriminator. As with our previous work, there is a substantial deterioration in performance between the one- and three-year predictions, as the gains ratio has increased to 0.420. However, it is a very material improvement compared to the GAN and VAE, which showed gains ratios of 0.927 and 0.782 respectively when predicting three years ahead.

As with the one-year result, the strongest predictors from the GBM discriminator are continuous variables.

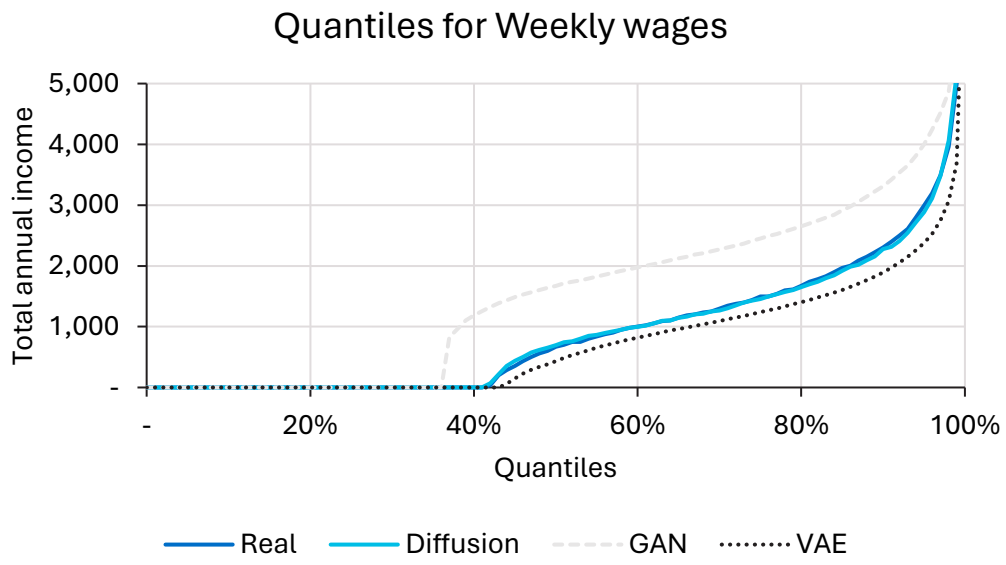
In our previous work, we observed significant drift when using the GAN and VAE to simulate three years ahead. This is obvious in continuous variables, such as total annual income (depicted in Figure 16). Both the GAN and VAE appear to overestimate the real distribution three years ahead (significantly so for the GAN), while the diffusion model captures the shape of the distribution accurately. Weekly wages (depicted in Figure 17 also show significant misfits for the GAN and VAE model, in opposite directions. The diffusion model outperforms both.

Figure 16 – quantiles for total annual income, from three-year diffusion



Note: Plots truncated at 80th percentile respectively for better visibility of important parts of the distribution.

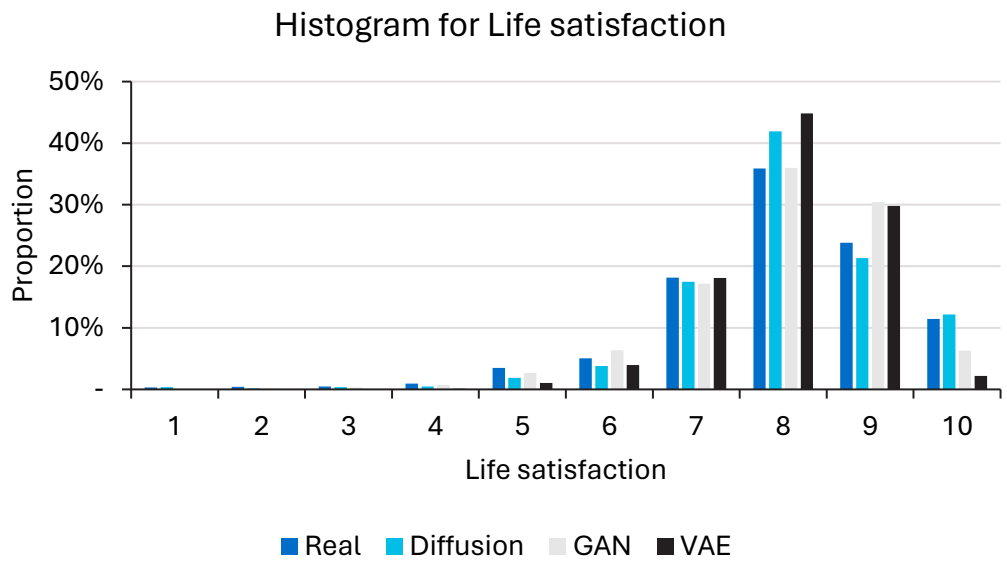
Figure 17 – quantiles for weekly wages, from three-year diffusion



Note: Plots truncated at $5,000 for better visibility of important parts of the distribution.

Figure 18 shows the performance of the diffusion model when predicting life satisfaction three years ahead, compared to the GAN and VAE. Like with the one-year diffusion, the diffusion model does a better job predicting the top end of the distribution than the GAN and VAE (life satisfaction of 10), though shows some misfits in other parts of the distribution.

Figure 18 – histogram for life satisfaction, from three-year diffusion



Two-way relationships also appear to be preserved in the three-year diffusion results.

Figure 19 – heat plot of total annual income against highest education level, for three-year diffusion

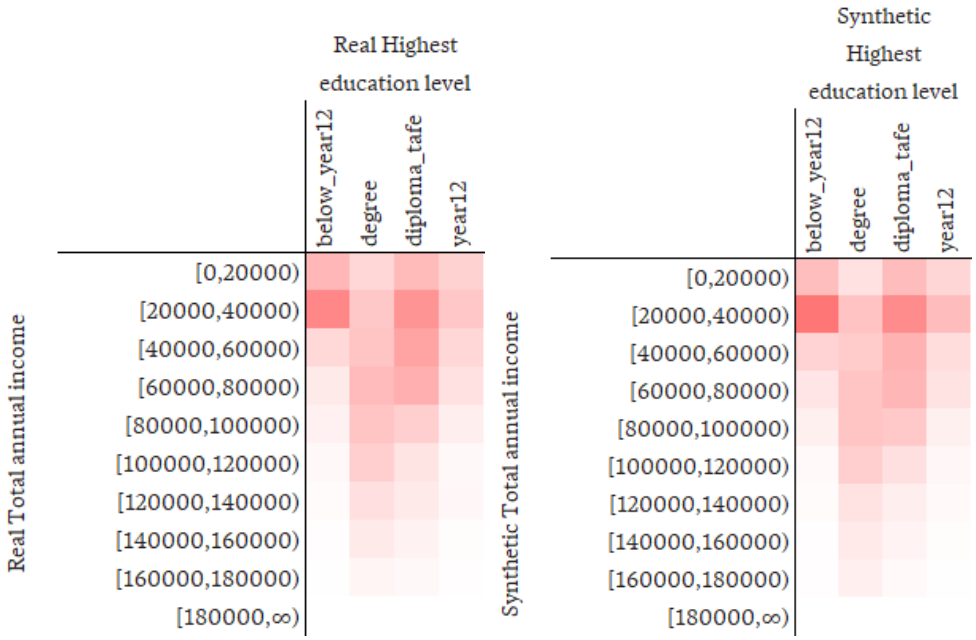
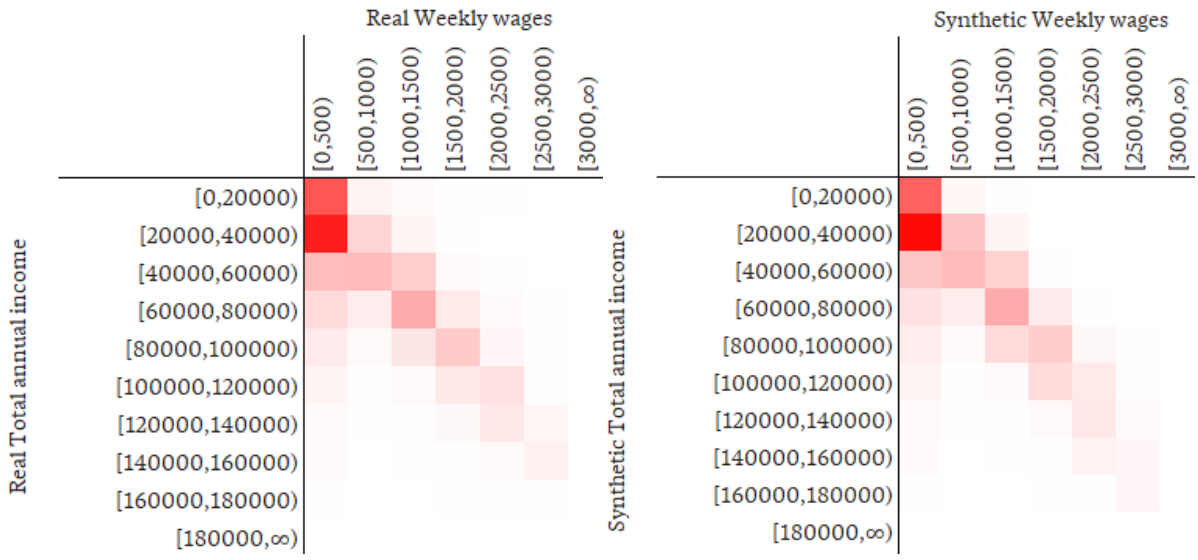


Figure 20 – heat plot of total annual income against weekly wages, for three-year diffusion



2.5.3 Computational considerations

While our diffusion model demonstrates superior performance by producing high-fidelity samples that are often indistinguishable from real data – marking a significant quality improvement over previously explored GAN and VAE architectures – this fidelity comes at the cost of substantially increased computational intensity. This section evaluates the computational trade-offs associated with both the training and inference phases of the diffusion process.

To ensure a direct comparison with earlier benchmarks, the diffusion model was initially evaluated using the same entry-level hardware utilized for the GAN and VAE models in the previous study. Furthermore, we extended our investigation to assess how the model scales on modern hardware.

Our results confirm that our diffusion model setup offers a significant increase in performance over traditional GAN and VAE architectures, at the cost of significantly greater computational demand.

The primary driver of this overhead is the substantial increase in model capacity; our final implementation utilized a deep 20-layer architecture with 183 million parameters, a 2048-unit hidden dimension, and a 1024-unit time embedding. While the previous GAN and VAE models were trained within two hours, the high capacity diffusion model required just over 8 hours on entry level hardware to complete 1000 epochs. The GAN/VAE models each contained fewer than 50,000 parameters, explaining much of this difference.

Using modern hardware, we reduced the training time of our full 183 million parameter model to just 50 minutes, demonstrating that the computational burden of diffusion can be effectively mitigated with advanced parallel processing or multi-GPU setups.

Table 8 – Training time comparisons

Model	Parameters	Training time, NVIDIA T1000 GPU	Training time, NVIDIA RTX PRO 4000 Blackwell GPU
GAN/VAE	<50,000	~2 hours	N/A
Diffusion	183 million	8 hours	50 minutes

The relative slowness of the diffusion was even more pronounced for inference (generation). For the diffusion model, our main inference results use Denoising Diffusion Probabilistic Models (DDPM), which simulate the whole denoising chain.

In Table 9 we show the time taken to score the final diffusion model on a dataset of 10,000 rows (using entry-level hardware), and present a comparison against the projection speed for the GAN and VAE models.

Table 9 – Time taken to simulate 10,000 HILDA records

Model	Inference time (seconds) NVIDIA T1000 GPU	Inference time (seconds) NVIDIA RTX PRO 4000 Blackwell GPU
GAN	0.2	N/A
VAE	0.02	N/A
Diffusion (DDPM)	2,870	330

Using DDPM for inference, the diffusion model took approximately 45 minutes to score 10,000 records. This is several orders of magnitude slower than the previous GAN / VAE attempts. Improved hardware can help – the same generation was reduced to less than six minutes using a more powerful GPU.

A common approach to speeding up diffusion model inference is to use Denoising Diffusion Implicit Models (DDIM), which is an alternative algorithm (that can be used on the same diffusion model) that significantly reduces the number of denoising timesteps required. The standard implementation produces a 20x speed-up (50 steps rather than 1,000) but we obtained very poor generation fidelity – there may be opportunities to improve this in the future.

2.5.4 Practical issues

We note some practical issues arising during the training process, which was iterative.

- **Under-coverage –** we did sometimes see significant divergence in the learning trajectories of categorical and continuous features. While categorical loss functions saw steady improvement, continuous error would sometimes improve steeply and then plateau; this usually corresponded to a collapse in the continuous variable so that it would only generate a single point (such as zero). The mixed distribution variables may have contributed to this behaviour.
- **Hyperparameter sensitivity –** Fit performance showed sensitivity to hyperparameter configurations, which made finding a good model with a lower parameter count difficult. Our final model had a large parameter count as a result.
- **Generation failure –** Even in cases where the model appeared to train well, there was still failure at the generation stage; a consistent percentage of generated records resulted in infinite or NaN (Not a Number) values for some variables. While a heuristic approach of re-simulating these records was largely effective (resolving approximately 90% of failures within ten attempts), a small subset (~0.1%) of records remained persistently unstable. This suggested some remaining ‘dead zones’ or singularities in the model's learned vector field.

  These "dead zones" or singularities are regions in the model's learned probability space where the mathematical gradients become infinite, causing the sampling trajectory to diverge. They may occur if the model’s approximation of the data distribution is non-smooth, creating a point that a small subset of noise configurations cannot successfully navigate.

2.5.5 Discussion

We find that overall, our diffusion setup for conditional generation in microsimulation contexts produces synthetic data of equal or superior quality to the GAN and VAE models from our previous work. This produces superior fits to continuous and mixed distribution variables, and better results for multiyear chained projection (with significantly less drift).

While promising, the current setup comes with a large trade-off, in the form of significantly increased computational burden. The diffusion is orders of magnitude more complex than the GAN and VAE

models, and as a result it takes significantly longer to perform both training and inference. Some of this can be defrayed by using more powerful GPUs, but the computational burden could prove to be problematic when developing and running a bigger microsimulation model, where millions of person-time steps are common.

There are opportunities for future research in alternative setups for diffusion modelling (including those Zhang et al., 2023, and Shi et al., 2024) that could maintain high predictive performance but deliver speed improvements. Of particular interest is whether forms of DDIM sampling, which are much faster, could be successfully adapted to our tabular data.

3 Triangle reserving

3.1 Data

To train and test our triangle reserving model, we use a publicly available dataset of claims data for a range of personal and commercial lines of property-casualty insurers in the United States – this dataset was originally compiled by Glenn G. Meyers and Peng Shi[6]. This data was originally sourced from the National Association of Insurance Commissioners (NAIC) Schedule P disclosures, which contains insurer-level run-off triangles of losses, aggregated by line of insurance. We use 176 triangles of claims data (a subset of the full database) from six lines of business:

1. Private passenger auto liability / medical
2. Commercial auto/truck liability / medical
3. Workers' compensation
4. Medical malpractice – claims made
5. Other liability – occurrence
6. Product liability – occurrence.

Each triangle corresponds to claims from accident year 1988-1997 with 10 years development lag. Data was extracted from the following 10 years of filings, and thus includes the actual observed outcomes for the lower half of the triangle. As described in Figure 1, the prediction task is to use the observed claims to predict unseen (future) claims in each development year – essentially completing the triangle to a rectangle.

The triangles have information on paid and incurred claims, as well as claim numbers. We focus on incremental paid claims only, except where otherwise discussed. Table 10 shows an example triangle.

Table 10 – Example triangle from Meyers' NAIC dataset (Product Liability #1252)

		Development year									
		0	1	2	3	4	5	6	7	8	9
Accident year	1988	179	213	272	255	228	200	196	194	194	193
	1989	71	227	228	233	215	224	141	128	128	128
	1990	147	279	204	176	191	191	195	192	192	190
	1991	271	181	320	268	304	423	415	355	353	353
	1992	200	307	305	313	313	312	302	292	292	292
	1993	186	240	442	338	343	336	338	338	338	338
	1994	394	395	338	379	343	343	343	343	343	343
	1995	232	309	229	230	371	367	367	367	367	367
	1996	270	403	408	415	410	416	377	374	374	374

[6] Available in, among other places, the reservetestr R package at https://github.com/problemofpoints/reservetestr

1997	320	265	257	260	260	262	262	262	262	262

3.2 Model structure

Our diffusion model for triangles is very different to the setup in Section 2, since we have spatial structure across the reserving rectangle. We use the intuition that a claims triangle is similar to a very crude image with few pixels, where (grayscale) pixel value is analogous to the number of claims. Our problem becomes an 'inpainting' or masked generation problem, where we can see part of the picture (the upper triangle) but the lower triangle is masked. For the generation stage, this lower part of the triangle starts as random noise and then iteratively resolves into a plausible completion of the known information. One generation corresponds to estimating the entire lower half of the claims rectangle, and will draw on all observable information in the upper half.

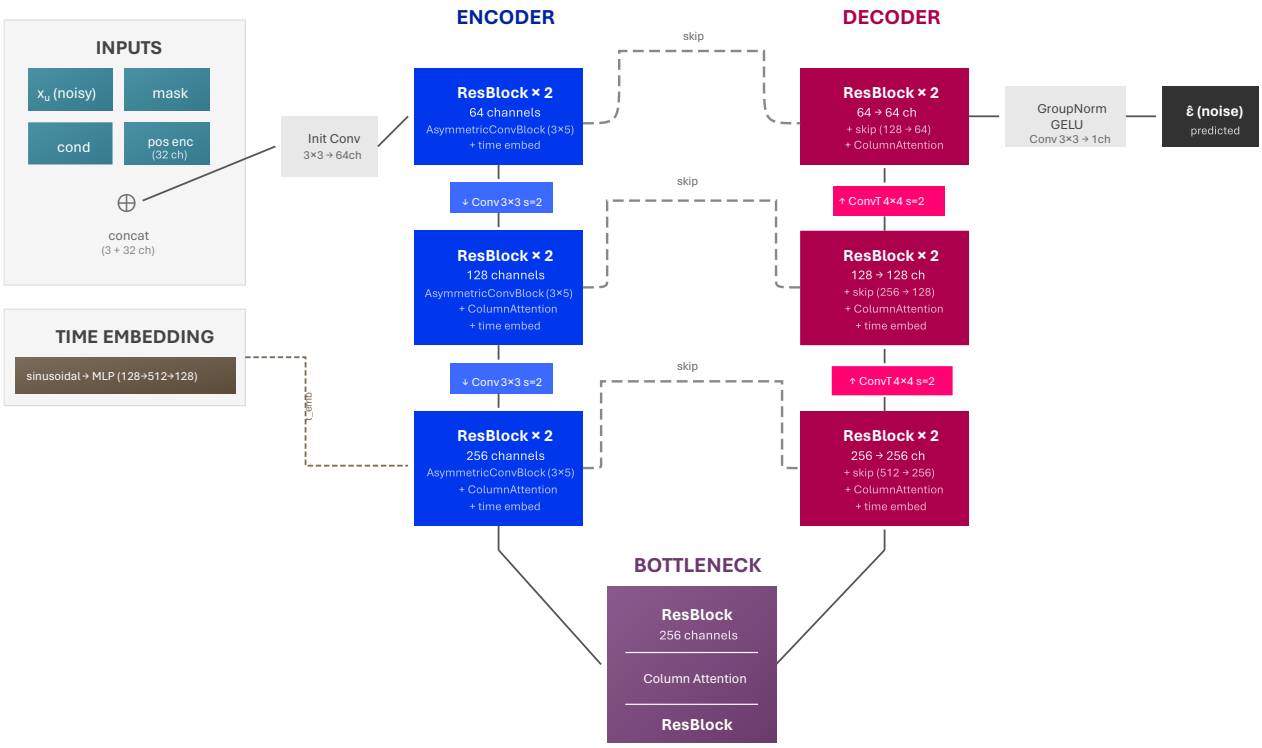
We therefore use a diffusion model architecture including a U-Net convolution setup, commonly used for image generation. This structure is shown in Figure 21 and includes:

- **Pre-normalisation** – Our preferred normaliser was a robust capped normaliser, which each cell was first transformed by $f(x) = \text{sign}(x)\log(|x| + 1)$, followed by division by the largest transformed absolute value. This is effectively a log transformation plus scaling, adapted to include the possibility of negative transformations.
- **Inputs –** Several inputs are needed for the generative denoising process:
  - The noisy triangle – This will be a rectangle of numbers, where the upper triangle is normalised actual values and the lower half the noisy values that are slowly resolving. The noisy variables start as $\mathcal{N}(0,1)$ sampled values and are updated at each step.
  - The binary mask, which is a rectangle of 0s and 1s indicating which parts of the rectangle are unknown and require generation (the lower triangle).
  - The positional encodings, which are a set of transformations of the row and column number; these attempt to generalise the learning so it is applicable to triangles of different sizes. It includes:
    - Scaled row i/I and column number j/J (so they are between 0 and 1), as well as the Sine values of these scaled values
    - Diagonal values $(j - i)$ / max dimension and $(i + j)$ / max(dimension) for row i and column j.
    - A decay shape by column $\exp(-2 \cdot j/J)$

    These encodings are passed through a connected multilayer perceptron to convert to 32 layers.
- **Time embedding –** similar to Section 2.2, transformations of the diffusion time u to recognise how denoising can vary at different stages of the time series.
- **Encoder layers** – Three levels with Residual Block (ResBlock) structure using a 5x3 size convolution. Column Attention functions were added at deeper levels, with the aim of capturing consistent development patterns across accident years. The levels are joined by convolutional downsampling.
- **A bottleneck –** ResBlock-Attention-ResBlock structure at 256 channels
- **Decoder layers –** Three levels that symmetrically match with the skip connections from the encoder, and transposed convolution upsampling
- **Output** – Group Normalisation → GELU → final convolution producing the predicted noise for that timestep $\hat{\varepsilon}$
- **Generation** – We used the faster DDIM approach for generating triangles.

The "U" refers to the shape made by the blue/purple/red boxes, where the image is reduced (in terms of the size of the grid), passed through the bottleneck and then expanded back to something of the required size.

Figure 21 – Adopted U-Net structure for triangle diffusion model, denoising step



Readers interested in learning further in U-Net design and architecture decisions are referred to Ho et al. (2020) and Dhariwal et al. (2021).

Testing different model architectures and practical adjustments

While final choices are covered in the model description above, we investigated a range of variations in the model architecture:

- Variation in convolution dimensions (rows and columns). This affects how far the convolution 'looks' around the current cell to infer behaviour. We found 'tall' blocks of 5x3 worked well.
- Enforcing the convolution to have no left padding – Convolutional steps typically need padding (additional pseudo-data) created at the edges to extend the convolution to the edges of the true shape. One idea is to only pad on the right, where claim payments are expected to be smaller, so reduce the potential for model bias. This tended to diminish performance.
- Altering the training parameters, such as the learning rate for the gradient descent and the number of epochs.
- Increasing/decreasing the number of 'time steps' in the diffusion process.
- Alternative normalisers, such as factor decompositions and median based transformations.

Our final model still showed some generation instability – occasionally the absolute value of a cell would be orders of magnitude too high. We applied a default 2% clipping rule as a simple adjustment, whereby the largest 2% and the smallest 2% reserve estimates would be deleted for the purposes of modelling the liability. This provides some robustness for average liability, but has some obvious drawbacks for modelling of tail risks, which we have not explored in this paper.

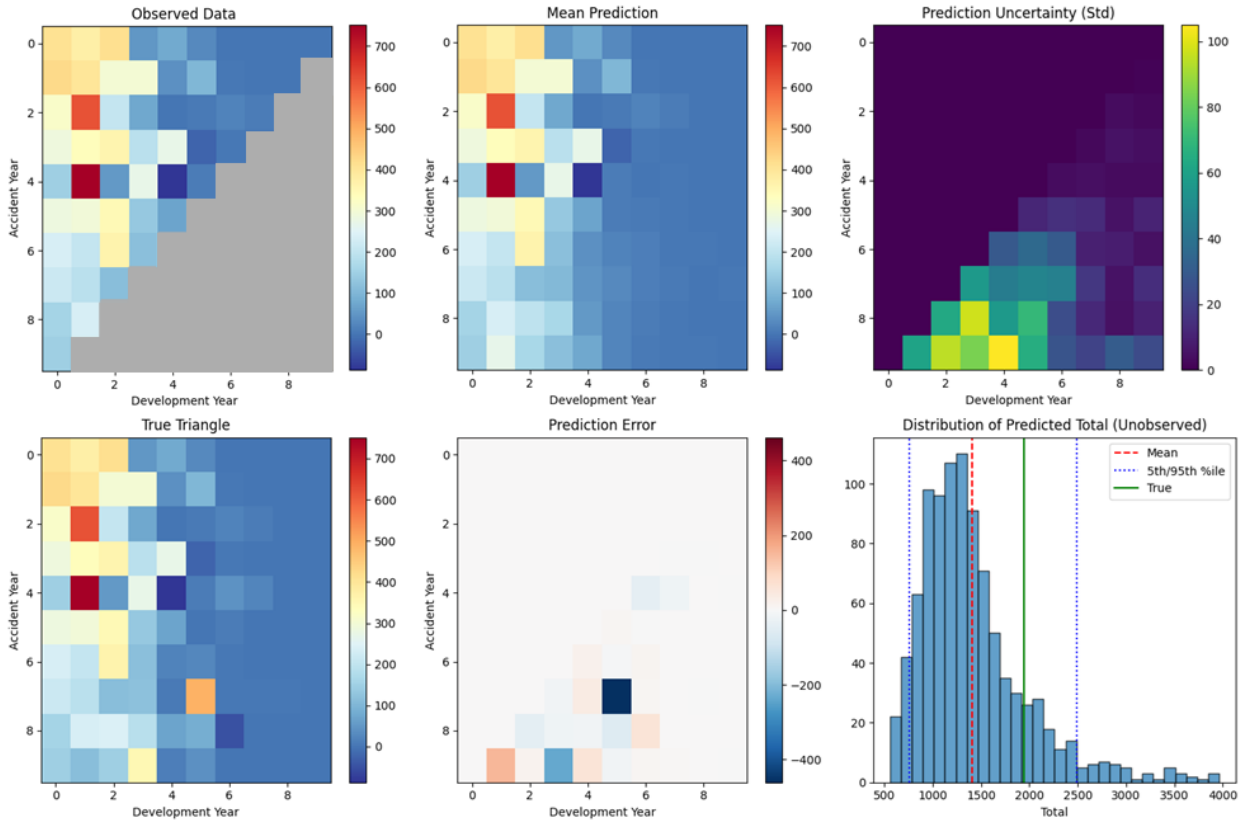
3.3 Results

Illustrative examples

It is useful to look at a few illustrative examples of individual triangle completion chosen from the holdout set. For example

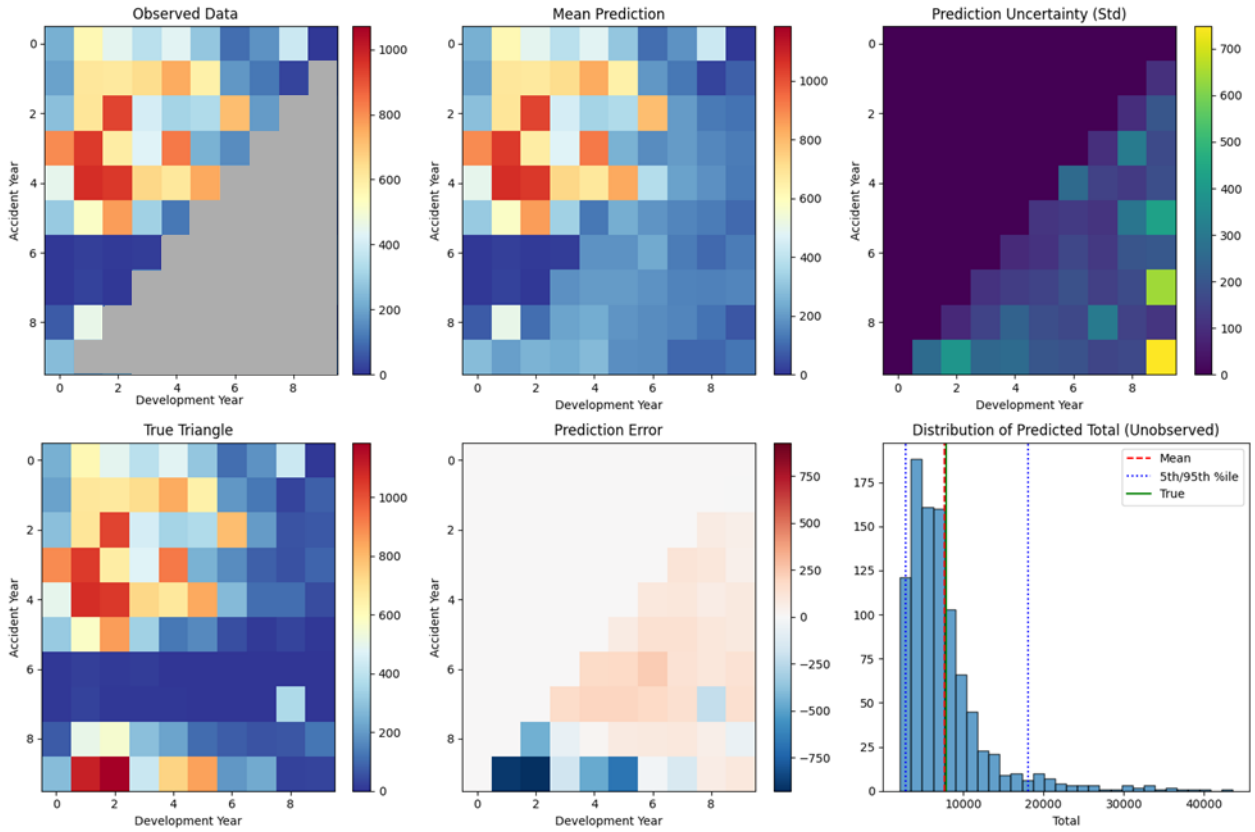
- Figure 22 shows generally good agreement, including the falling trend in later accident years, but cannot predict some anomalous one-off larger claims in the 5th development year. The lower right histogram in the figure shows the distribution of the liability (sum of lower triangle) across 100 simulations. The true value lies above the mean, but safely within the plausible range of generated values. The upper right shows the uncertainty of prediction, which correctly puts most of the uncertainty in the first few development periods in the last two accident years.
- Figure 23 is an example of poorer completion, where the very low experience in accident periods 6 and 7 are not correctly picked up, so these years are biased high. The high claims experience in the last two accident periods are harder to anticipate though. Despite this, the actual claims payments are relatively close to the mean prediction of the distribution.

Figure 22 - Individual triangle completion (1)



In Figure 23, the model has failed to extrapolate the pattern of 0 claims for all of in accident year 6 and 7.

Figure 23 - Individual triangle completion (2)



Overall performance

We use a pre-determined 70% training, 20% validation, 10% holdout split of the 176 non-simulated triangles in the Meyers dataset. To evaluate different model choices, we relied on several metrics, the most important being the overall p-p plot.

Since the data is real, we cannot simply assume the model is correct if it closely matches on the lower part of the triangle. Instead, we think of the observed values as percentiles from our generated distribution, and see whether these realisations match the spread that we would expect.

For each triangle we generated 1000 completions and then determined the percentile that the observed liability (sum of the lower triangle) within this distribution. Applying across our triangles, we can construct a percentile-percentile plot. As shown in Figure 24 and Figure 25. A 'good' model will see the empirical percentiles following the diagonally closely.

We observe:

- On the (smallish) holdout set, percentiles are within the expected range of variation, but the shape (left points below the diagonal, right above it) are consistent with the actual values being too heavy-tailed – more extreme percentiles than expected.
- This pattern is confirmed on the training data, where the same shape is seen, with too many actual values close to the extreme ends of the generated distributions.

Despite the misfit, we regard the results as promising; Meyers (2015) includes similar analysis using other stochastic approaches, and also saw significant misfits on models of paid amounts, often with a similar shape. We therefore regard the outcomes as comparable with the existing literature in its ability to generate distributions without excessive bias.

Our ad hoc clipping adjustment (where the most extreme 2% of generations were discarded as a precaution, as some very extreme values occasionally appeared) contributes a small amount to the bias seen in the p-p plots (as can be seen by comparing the figures in Appendix A.1).

Figure 24 - P-P plot (holdout set)

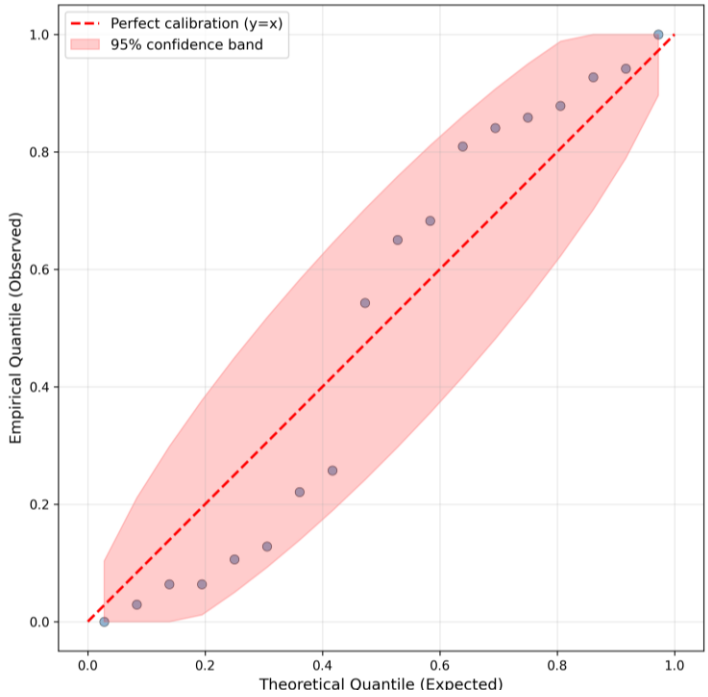
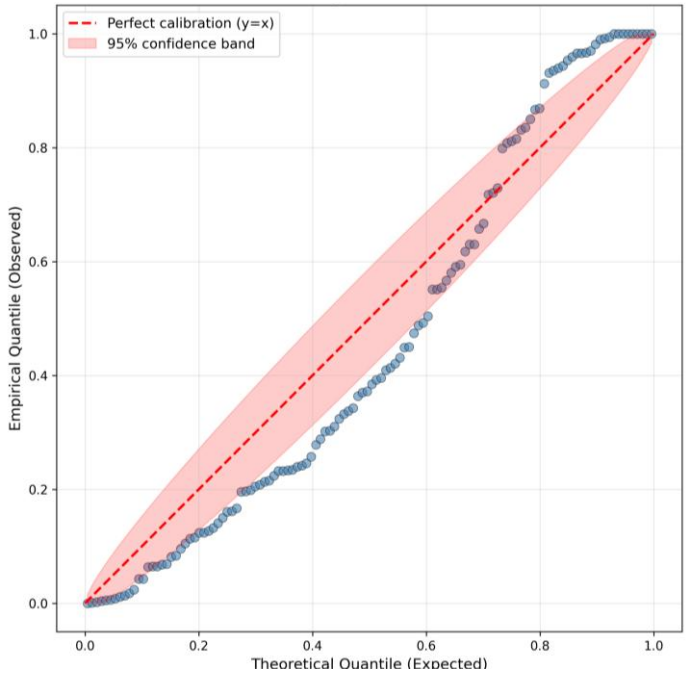


Figure 25 - P-P plot (training set)



3.4 Discussion

We observe:

- **The core methodology works –** we obtain highly plausible triangle completions from a deep learning model image generator. Basic patterns such as accident and development year effects are 'learnt' and applied. This means that broader experience on triangle completions can be brought to bear on specific cases, which is a new way of applying this experience to projections.
- **Distributional results are reasonable** – The generative approach means we get a range of potential reserving estimates (each a realisation of how the full lower half 'could' look), providing an (essentially free) estimate of reserving variability.
- **Some challenges remain** – While fitting and generation on the main dataset appears mostly robust, we note some issues observed
  - Outlier generations – Some (perhaps 1%) of generations included an outlier value orders of magnitude higher than expected (exacerbated by the inverse of our log-transform normalisation). These are easy to spot and discard, but speak to some residual instability
  - Scaling to larger sizes – Our model structure intends to adapt to different sizes, and also includes functionality to 'patch' the completion together using a successive set of completions on different parts of the image. However, our testing has been limited and results for significantly larger triangles unstable. Conversely, performance for smaller squares and rectangles have been good.
  - Training dataset requirements – We believe the model would be most aided by a range of different (completed) triangles, of various sizes and patterns. The Meyers database carries limitations, including significant numbers of zero rows in many of the triangles.

We also recognise there are **significant opportunities for further improvements**, with three important ideas:

- Expanding the training – Having a more diverse range of input datasets would likely improve model performance.
- Simultaneous estimation of other reserving components – Image generators simultaneously estimate a colour intensity value for three colour channels (RGB) to produce cohesive pictures. By direct analogy, there is no reason why a reserving model could not simultaneously generate paid claims, incurred claims and claim numbers at the same time. This could be expanded to even more channels (e.g. open claims). This would enable the tool to be even more general-purpose.
- Adding in contextual information to further guide generation – In addition to claim data in the upper triangles it would also be possible to input other variables into the training and generation. For example, a class of business field and a time scale would give the generator information on whether a particular triangle was likely to be long-tailed and have a better sense of the time scales for the rows and columns. Again by analogy, this is similar to how prompts for image generators guide the type of images created.

4 Conclusion

Performance of diffusion models

Diffusion models are popular in a range of deep learning applications, but our model results show they are useful in actuarial contexts too. Performance (in terms of plausible generation) is good, with best-in-class performance for complex microsimulations and good performance on triangle completion; both difficult problems for traditional machine learning. Speed is potentially an issue for contexts where large volumes of predictions are needed.

To our knowledge, this is the first time such diffusion models have been applied in an actuarial context.

Implication for predictive modelling

While we have focused on microsimulation applications (where the need for simultaneous generation of multiple outcome variables is clear), our approach to conditional generation could be applied to any predictive modelling context, whether or not there are multiple outcome variables. The ability to generate distributions is a key advantage.

For actuaries this includes a range of common problems that can be otherwise challenging. Examples include:

- Statistical case estimation – Often there is a need to predict payments for a claim across payment types (e.g. medical, legal, common law) and time (next year, subsequent year etc). These are hard to project in a way that maintains consistency across payment types and time, and distributions of outcomes are even harder. Diffusion models offer an opportunity to dramatically simplify this modelling.
- Time series forecast and economic scenarios – Dynamic Financial Analysis relies on scenarios that often require internally consistent scenarios across a range of different variables.
- Customer journey modelling – Predicting customer retention and changing circumstances (e.g. sum insured, responses to pricing changes) is typically a complex modelling task involving different output factors and multiple timesteps. Again, generative techniques have the potential to simplify such projection.

Implications for reserving

While there is a push towards reserving at a claim level (Wüthrich, 2018, Blier-Wong et al, 2020, Avanzi et al., 2023), we believe triangle-based reserving will remain popular in practice. Auto-completers that are smarter than traditional default models that can quickly generate both mean predictions and distributions have significant potential to augment existing practice.

References

- Avanzi, B., Taylor, G., & Wang, M. (2023). SPLICE: a synthetic paid loss and incurred cost experience simulator. *Annals of Actuarial Science*, *17*(1), 7-35.
- Blier-Wong, C., Cossette, H., Lamontagne, L., & Marceau, E. (2020). Machine learning in P&C insurance: A review for pricing and reserving. *Risks*, *9*(1), 4. Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2022). Deep neural networks and tabular data: A survey. IEEE transactions on neural networks and learning systems.
- Bornhuetter, R. L., & Ferguson, R. E. (1972, November). The actuary and IBNR. In *Proceedings of the casualty actuarial society* (Vol. 59, No. 112, pp. 181-195).
- Dhariwal, P., & Nichol, A. (2021). Diffusion models beat GANs on image synthesis. *Advances in neural information processing systems*, *34*, 8780-8794.
- England, P. D., & Verrall, R. J. (2002). Stochastic claims reserving in general insurance. *British Actuarial Journal*, *8*(3), 443-518.
- Hindley, D. (2017). *Claims Reserving in General Insurance*. Cambridge University Press.
- Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, 6840-6851.
- Kotelnikov, A., Baranchuk, D., Rubachev, I., & Babenko, A. (2023). TABDDPM: Modelling tabular data with diffusion models. In *International conference on machine learning* (pp. 17564-17579). PMLR.
- Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin: The Journal of the IAA*, *23*(2), 213-225.

- Miller H., Sik Kwok Wong, J., Sleigh, C. (2025). Yes we GAN: Adapting generative models for predictive modelling with multivariate response. Presented at the 2025 All Actuaries Summit. https://content.actuaries.asn.au/resources/resource-ce6yyqn64sx3-786882053-16103
- Milligan, S. (2025). autotab: Variational Autoencoders for Heterogeneous Tabular Data. R package version 0.1.1. doi:10.32614/CRAN.package.autotab.
- Meyers, G. (2015). Stochastic loss reserving using Bayesian MCMC models. Arlington, VA: Casualty Actuarial Society.
- Nazabal, A., Olmos, P. M., Ghahramani, Z., & Valera, I. (2020). Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, *107*, 107501.
- Nichol, A. Q., & Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *International conference on machine learning* (pp. 8162-8171). PMLR.
- Rasheed, A., San, O., & Kvamsdal, T. (2020). Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE access*, *8*, 21980-22012.
- Shi, J., Xu, M., Hua, H., Zhang, H., Ermon, S., & Leskovec, J. (2024). Tabdiff: a mixed-type diffusion model for tabular data generation. *arXiv preprint arXiv:2410.20626*.
- Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015, June). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning* (pp. 2256-2265). pmlr.
- Song, J., Meng, C., & Ermon, S. (2020). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.
- Sun, Y., Li, J., Xu, Y., Zhang, T., & Wang, X. (2023). Deep learning versus conventional methods for missing data imputation: A review and comparative study. *Expert Systems with Applications*, *227*, 120201.
- Taylor, G. (2000). *Loss reserving: an actuarial perspective* (Vol. 21). Springer Science & Business Media.
- Telyatnikov, L., & Scardapane, S. (2023, April). Egg-gae: scalable graph neural networks for tabular data imputation. In *International conference on artificial intelligence and statistics* (pp. 2661-2676). PMLR.
- Wüthrich, M. V. (2018). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, *2018*(6), 465-480.
- Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., ... & Yang, M. H. (2023). Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, *56*(4), 1-39.
- Yoon, J., Jordon, J., & Schaar, M. (2018, July). Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning* (pp. 5689-5698). PMLR.
- Yoon, J., Zhang, Y., Jordon, J., & Van der Schaar, M. (2020). Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in neural information processing systems*, *33*, 11033-11043.
- Zhang, H., Zhang, J., Srinivasan, B., Shen, Z., Qin, X., Faloutsos, C., ... & Karypis, G. (2023). Mixed-type tabular data synthesis with score-based diffusion in latent space. *arXiv preprint arXiv:2310.09656*.

Appendix A Additional results

A.1 P-P plots without 2% clipping of extreme results

Figure 26 - Holdout set P-P plot (no 2% clipping)

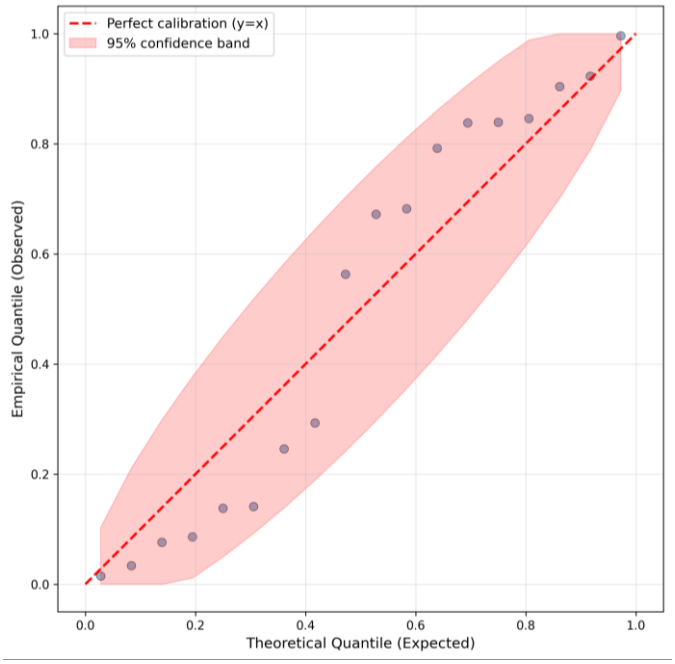
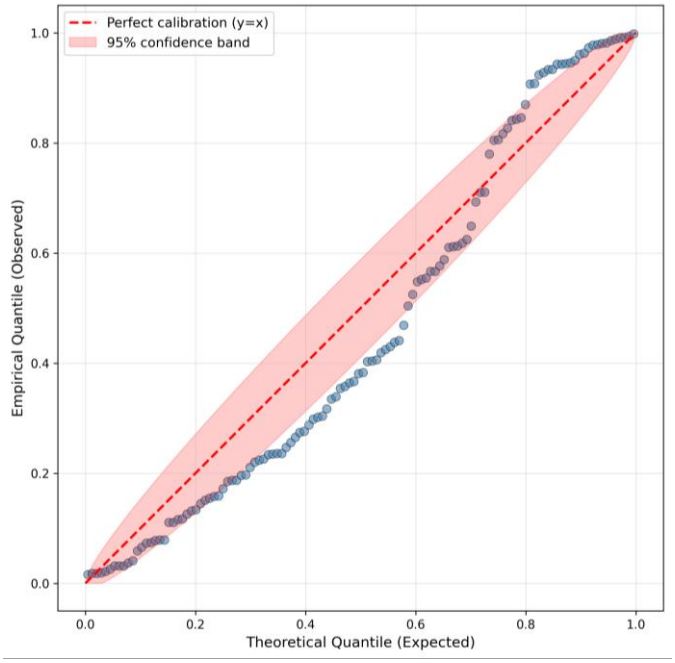



Figure 27 - Training set P-P plot (no 2% clipping)