Astin Bulletin (2024), **54**, pp. 239–262
doi:10.1017/asb.2024.4

International Actuarial Association
Association Actuarielle Internationale

RESEARCH ARTICLE

Telematics combined actuarial neural networks for cross-sectional and longitudinal claim count data

Francis Duval, Jean-Philippe Boucher and Mathieu Pigeon

Département de mathématiques, Université du Québec à Montréal, Montréal, H2X 3Y7, Québec, Canada
Corresponding author: Francis Duval; Email: francisduval37@gmail.com

Received: 2 August 2023; **Revised:** 13 January 2024; **Accepted:** 15 January 2024; **First published online:** 14 February 2024

Keywords: Automobile insurance; combined actuarial neural network; deep learning; claim count data; multivariate negative binomial

Abstract

We present novel cross-sectional and longitudinal claim count models for vehicle insurance built upon the combinedd actuarial neural network (CANN) framework proposed by Wüthrich and Merz. The CANN approach combines a classical actuarial model, such as a generalized linear model, with a neural network. This blending of models results in a two-component model comprising a classical regression model and a neural network part. The CANN model leverages the strengths of both components, providing a solid foundation and interpretability from the classical model while harnessing the flexibility and capacity to capture intricate relationships and interactions offered by the neural network. In our proposed models, we use well-known log-linear claim count regression models for the classical regression part and a multilayer perceptron (MLP) for the neural network part. The MLP part is used to process telematics car driving data given as a vector characterizing the driving behavior of each insured driver. In addition to the Poisson and negative binomial distributions for cross-sectional data, we propose a procedure for training our CANN model with a multivariate negative binomial specification. By doing so, we introduce a longitudinal model that accounts for the dependence between contracts from the same insured. Our results reveal that the CANN models exhibit superior performance compared to log-linear models that rely on manually engineered telematics features.

1. Introduction and motivations

Vehicle insurance products have traditionally been priced based on self-reported attributes provided by insureds. These attributes commonly include various risk factors, including gender, age, vehicle usage, and claim history. Insurers rely on this information to assess the level of risk associated with each insurance contract and determine appropriate premium rates. With the introduction of telematics technology, insurers can now collect a wide range of driving data through devices installed in the vehicles of policyholders or through mobile applications. This includes information such as vehicle speed, acceleration and braking behavior, mileage, location data, and factors like the time of day or types of roads frequently traveled. By leveraging this wealth of data, insurers can gain a more accurate and objective understanding of each individual's driving habits and style, enabling them to customize insurance offerings and pricing based on their actual driving behavior. This emerging paradigm, known as usage-based insurance (UBI), revolutionizes the insurance landscape in various ways. For insurers, telematics data means more accurate risk assessment algorithms, which can often translate into a competitive advantage. For insureds, it means fairer premium rates that align more closely with their actual risk profiles rather than being computed based on broad demographic categories. It also means that they are priced based on risk indicators over which they have control. From a societal perspective, UBI also offers many advantages. One of the key benefits is the potential to improve road safety. By giving incentives for safe driving behavior and reduced mileage, UBI not only helps reduce the frequency and severity of accidents, ultimately saving

lives and reducing the economic burden associated with road accidents, but also contributes to reducing greenhouse gas emissions. Additionally, telematics provide insurers with viable alternatives to sensitive risk factors, thereby helping to prevent unfair discrimination. For a more extensive overview of the benefits of UBI, we refer to the works of Litman (2007), Bordoff and Noel (2008), and Ziakopoulos *et al*. (2022).

One of the most prominent questions related to UBI is how to make the most out of the collected driving data. A significant subset of the literature has focused on incorporating mileage into pricing models due to its acknowledged importance as a risk factor in assessing risk and determining premium rates (see, for instance, Lemaire *et al*., 2015; Boucher *et al*., 2017, and Turcotte and Boucher, 2023). However, mileage alone fails to provide the whole story about an insured individual's driving behavior, prompting researchers to consider additional telematics information. One prevalent approach involves drawing upon domain knowledge to craft telematics features from raw data. By applying their expertise in the field, researchers can engineer features that capture critical aspects of driving behavior, specifically driving characteristics that are thought to be correlated with the risk of accident. Common examples of such features include harsh braking/acceleration events, cornering events, speeding, distracted driving, the fraction of driving during both different time slots (e.g., rush hour, late-night hours, weekdays), and on different road types (e.g., urban roads, highways), as well as the fraction of driving in different speed slots. This approach proves effective, providing benefits in terms of its simplicity and the ease of interpretation of the engineered features. However, it relies heavily on human judgment, with its inherent flaws and biases. With countless possible telematics features that can be engineered from raw telematics data, selecting the optimal ones for pricing is not straightforward. Furthermore, this process necessitates the setting of thresholds. For example, how should night driving or harsh braking be precisely defined?

The limitations of the aforementioned approach have motivated researchers to explore a new set of methods that rely more on data and decrease the need for human judgment. These could serve as alternatives or complements to the domain knowledge approach. As highlighted in a recent study by Embrechts and Wüthrich (2022), the increasing amount of data available presents a challenge in manually designing features, leading actuaries to increasingly depend on tools like neural networks to learn and extract meaningful representations from the data. Blier-Wong *et al*. (2021) underline the importance of learning valuable representations from emerging data sources such as text, image, and sensor data. These sources, which include telematics car driving data, can enrich traditional data and offer improved insights for predicting future losses in insurance contracts. Neural networks are regarded as the most effective means for automatically extracting valuable features from raw data, which validates their practical application. In recent years, researchers have successfully applied the toolbox of deep learning, namely neural networks architectures with a large number of hidden layers, to handle telematics data and other types of unstructured data. In their work, Wüthrich (2017) introduces the speed-acceleration heatmap, a matrix representation that characterizes the driving style of an insured driver, which is well-suited for processing by deep learning algorithms. Subsequent studies (Gao and Wüthrich, 2018, 2019; Gao *et al*., 2019, 2022) have effectively leveraged these heatmaps by employing neural networks to learn representations from them. In Meng *et al*. (2022), the authors propose a supervised driving risk scoring convolutional neural network (CNN) model that uses telematics car driving data to improve automobile insurance claims frequency prediction. Blier-Wong *et al*. (2020) propose a convolutional regional autoencoder model for generating geographical risk encodings using CNNs. The resulting encodings, which aim to replace the traditional territory variable, proved beneficial for risk-related regression tasks. So *et al*. (2021) develop a dataset generation algorithm based on feed-forward neural networks. Using a synthetic dataset generated with So *et al*. (2021), Jeong (2022) uses categorical embedding and principal component analysis to handle the high dimensionality of telematics data.

In this paper, we present novel claim count models based on the combined actuarial neural network (CANN) approach, initially proposed by Wüthrich and Merz (2019). The CANN approach involves embedding a classical regression model, such as a generalized linear model (GLM; see Nelder and Wedderburn, 1972 and Dionne and Vanasse, 1989), into a neural network, achieved by blending the

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

regression functions of both models. Consequently, the resulting model comprises the two following components: the classical regression (or actuarial) model and the neural network. This blending process can be interpreted as a form of neural network boosting for the actuarial model, combining the strengths of both approaches. The calibration of the CANN neural network is performed using the classical actuarial model as the initial value in the gradient descent algorithm, with the negative log-likelihood of the specified distribution used as the loss function. One of the key benefits of this specific architecture is the solid foundation offered by the classical model, complemented by the network component's flexibility and pattern recognition capabilities. Neural networks excel in approximating highly nonlinear functions and possess the ability to compute valuable interactions between input variables automatically. Consequently, the CANN approach combines the best of both worlds, leveraging the interpretability and reliability of the classical model while capitalizing on the power of neural networks to capture complex relationships and patterns in the data. A few studies have successfully leveraged this approach: Schelldorfer and Wuthrich (2019) present a case study where a Poisson GLM for predicting claims frequencies is initially used, then enhanced through generalized additive models (GAMs) with natural cubic splines, and finally combined with a neural network, resulting in a CANN approach. The study also explores the use of embedding layers for more efficient treatment of categorical variables; Gabrielli *et al*. (2020) boost an overdispersed Poisson model with a multilayer perceptron (MLP) to improve individual loss reserving; Tzougas and Kutzkov (2023) use the CANN approach to enhance binary classification; Laporta *et al*. (2023) apply the CANN architecture in the context of quantile regression.

Our models employ a log-linear model for the actuarial model part and a MLP for the network part. Telematics information is incorporated into the MLP as a telematics vector, which is given as input to represent the driving behavior of each insured driver. The MLP part is given traditional risk factors as inputs, allowing interactions between traditional and telematics inputs. Meanwhile, the log-linear part, constrained in estimating complex functions, is only given traditional risk factors. We explore three distinct distribution specifications for the claim count: Poisson and negative binomial for cross-sectional analysis, and multivariate negative binomial (MVNB; see (MVNB; see Hausman *et al*., 1984 and Boucher *et al*., 2008), also known as *negative multinomial*, for longitudinal analysis. The MVNB distribution is a popular choice for modeling longitudinal claim count data, as it captures the dependence between contracts from the same insured. However, to our knowledge, this specification has never been adapted to a neural network model for claim count regression. In this study, we extend the application of the MVNB distribution by incorporating it into the neural network framework, specifically the CANN architecture, for modeling longitudinal claim count data. This adaptation allows us to leverage the strengths of both the MVNB distribution and the neural network architecture. Our findings indicate that the CANN models perform better than their log-linear counterparts that rely on manually engineered telematics features. Furthermore, the CANN model using the MVNB specification exhibits a significant improvement compared to the two cross-sectional specifications.

In Section 2, we present the two datasets available to us: the contract dataset and the telematics dataset. Following that, in Section 3, we delve into the theory behind the CANN claim count models and also discuss the log-linear models that serve as benchmarks. Moving on to Section 4, we provide an explanation of how we apply the models on our specific dataset and show how we preprocess telematics data. In Section 5, we assess the performance of the models on a holdout sample and interpret the CANN models through permutation feature importance and partial dependence plots (PDPs). Lastly, we conclude in Section 6.

2. Data

We have access to data from a Canadian property and casualty insurance company, which comes in two distinct datasets: the contract and the telematics dataset.

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Table 1. *Variables of the contract dataset*

Variable name	Description	Type
`vin`	Unique vehicle identifier	ID
`annual_distance`	Annual distance declared by the insured	Numeric
`commute_distance`	Distance to the place of work declared by the insured	Numeric
`conv_count_3_yrs_minor`	Number of minor contraventions in the last three years	Numeric
`distance`	Real distance driven	Numeric
`expo`	Contract duration in years	Numeric
`gender`	Gender of the insured	Categorical
`marital_status`	Marital status of the insured	Categorical
`pmt_plan`	Payment plan chosen by the insured	Categorical
`veh_age`	Vehicle age	Numeric
`veh_use`	Use of the vehicle	Categorical
`years_licensed`	Number of years since obtaining driver's license	Numeric
`nb_claims`	Number of claims	Numeric

2.1. Contract dataset

In the contract dataset, each row represents a unique insurance contract. Contracts typically last for one year, but there are instances where their duration may be shorter or longer. Each vehicle is observed over one or more contracts; therefore, one vehicle can be represented by one or more rows in this dataset. Based on risk factors, a premium must be computed for each contract. When using a cross-sectional data model, contracts from the same vehicle are assumed to be independent of each other. On the other hand, a longitudinal data model assumes dependence between contracts, allowing it to use information from previous contracts (including traditional risk factors, telematics data, past claims, etc.) to compute the premium. The contract dataset includes attributes commonly used in vehicle insurance pricing models. These traditional risk factors, displayed in Table 1, are recorded for 117,268 insurance contracts initiated between December 15th, 2015 and December 31st, 2018.

In cases where multiple drivers are associated with a particular contract, attributes of the principal driver are used. Additionally, the dataset includes the vehicle identification number (VIN), allowing us to identify the insured vehicle accurately, alongside the reported claim count. As our goal is to perform claim count regression on contracts, the claim count variable will serve as the response for our supervised learning algorithms, namely the log-linear and the CANN models. The 117,268 contracts are associated with 49,671 distinct vehicles, resulting in an average of approximately 2.36 contracts per vehicle. The histogram of the number of contracts per vehicle is shown in Figure 1.

2.2. Telematics dataset

All 117,268 contracts have been logged using an on-board diagnostics device, capturing driving information. These data are stored as trip summaries in the telematics dataset, which comprises 117,566,259 trips. Each row in the dataset represents a specific trip, and every trip is described by four attributes: the departure and arrival date and time, the distance driven, and the maximum speed reached. Additionally, each trip is associated with a VIN, and with the date information, it is thus possible to link each trip with one of the 117,268 contracts. An extract from the telematics dataset is presented in Table 2.

2.3. Training, validation, and testing datasets

In supervised learning analysis, splitting the available data into training, validation, and testing sets is paramount to ensure the reliability and generalization ability of the learned model. The training set,

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

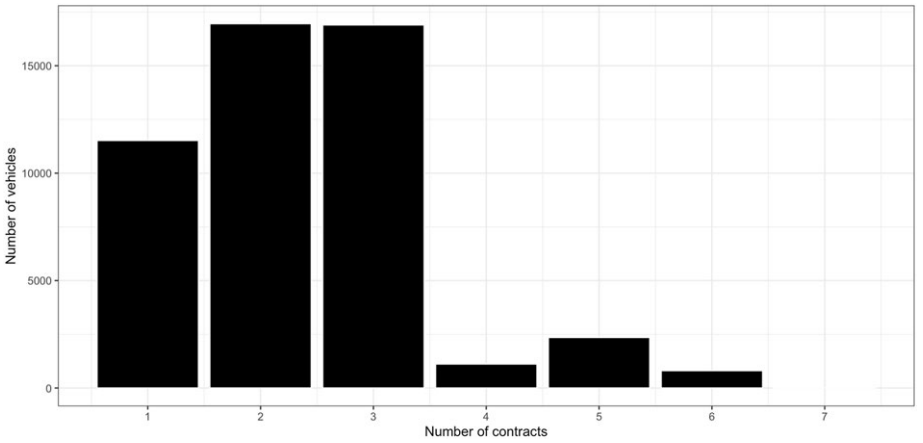

Figure 1. *Number of contracts per vehicle.*

which usually comprises the largest portion of the data, is used to train the model's parameters and optimize its performance. However, relying solely on the training set for performance assessment can lead to overfitting, particularly when the model has a high capacity. To address this, the validation set is used during the modeling process to assess the model's performance on unseen data. It plays an important role in tuning hyperparameters, selecting the optimal model architecture, and preventing overfitting. By assessing the model's performance on the validation set, one can obtain an estimate of its generalization performance and make necessary adjustments to improve its ability to generalize well to new, unseen examples. However, it is important to note that the back-and-forth process of evaluating the model on the validation set and adjusting its hyperparameters can introduce information leakage from the validation set into the training set. This can create an illusion of better performance than the model would exhibit in real-world scenarios. As a result, the testing set is reserved for the final evaluation of the learned model. It serves as an unbiased assessment of how well the model will perform on completely unseen data. This final evaluation provides an estimate of the model's true performance and helps determine its reliability in real-world scenarios. By keeping the testing set separate from the training and validation sets, we can ensure an unbiased evaluation and avoid any potential data leakage. We partition the data as outlined in Table 3 for our analysis. Approximately 60% of the vehicles are allocated for training, while approximately 20% is assigned to the validation and testing sets.

3. Count regression models

We consider a training dataset denoted as $\mathcal{T}_r$, which consists of $|\mathcal{T}_r|$ rows representing vehicle insurance contracts. Contracts are grouped by vehicle and each vehicle i is observed over T_i contracts. We define Y_{it} as a discrete random variable denoting the number of claims during the t^{th} contract of vehicle i. Furthermore, we have $\boldsymbol{x}_{it}$ a vector containing relevant predictor variables associated with the t^{th} contract of vehicle i. Importantly, we assume independence among all insured vehicles. In claim count regression, the ultimate goal is to estimate the probability mass function (PMF) of the number of claims, given all past and current information about the vehicle. Mathematically, we seek to estimate:

$$\mathbb{P}\left(Y_{it} = y_{it} | \boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,(1:t)}\right), \quad y_{it} \in \mathbb{N}, \tag{3.1}$$

where $\boldsymbol{y}_{i,(1:t-1)} = (y_{i1}, \ldots, y_{i,t-1})$ is the vector of past claims and $\boldsymbol{x}_{i,(1:t)} = \{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{it}\}$ is the set of past and current covariate vectors for vehicle i.

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Table 2. *Extract from the telematics dataset. Dates are displayed in the yyyy-mm-dd format. The actual VINs have been hidden for privacy purposes*

VIN	Trip ID	Departure datetime	Arrival datetime	Distance	Maximum speed
A	1	2017-05-02 19:04:15	2017-05-02 19:24:24	25.0	104
A	2	2017-05-02 21:31:29	2017-05-02 21:31:29	6.4	66
⋮	⋮	⋮	⋮	⋮	⋮
A	2320	2018-04-30 21:17:22	2018-04-30 21:18:44	0.2	27
B	1	2017-03-26 11:46:07	2017-03-26 11:53:29	1.5	76
B	2	2017-03-26 15:18:23	2017-03-26 15:51:46	35.1	119
⋮	⋮	⋮	⋮	⋮	⋮
B	1485	2018-03-23 20:07:08	2018-03-23 20:20:30	10.1	92
C	1	2017-11-20 08:14:34	2017-11-20 08:40:21	9.7	78
⋮	⋮	⋮	⋮	⋮	⋮

Table 3. *Data partitioning*

Set	Symbol	Number of vehicles	Number of contracts	Number of trips
Training	$\mathcal{T}_r$	30,000	70,451	71,416,560
Validation	$\mathcal{V}_a$	10,000	23,368	22,611,829
Testing	$\mathcal{T}_e$	9671	23,449	23,537,870
Total	–	49,671	117,268	117,566,259

3.1. Cross-sectional models

In addition to assuming independence between vehicles, cross-sectional models also assume independence between contracts from the same vehicle. Consequently, these models do not use the history of a vehicle to estimate its future risk. The PMF of the number of claims can thus be written as:

$$\mathbb{P}\left(Y_{it} = y_{it} | \boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,(1:t)}\right) = \mathbb{P}\left(Y_{it} = y_{it} | \boldsymbol{x}_{it}\right), \quad y_{it} \in \mathbb{N}. \tag{3.2}$$

3.1.1. Poisson regression

The Poisson distribution is widely used in supervised learning analysis for claim count data due to its good properties and simplicity. Under the Poisson specification, the PMF of the claim count for the t^{th} contract of vehicle i, denoted by Y_{it}, given its predictor vector, denoted by $\boldsymbol{x}_{it}$, is defined by

$$\mathbb{P}(Y_{it} = y_{it} | \boldsymbol{x}_{it}) = \frac{e^{-\mu(\boldsymbol{x}_{it})} \mu(\boldsymbol{x}_{it})^{y_{it}}}{y_{it}!}, \quad \text{for} \quad y_{it} \in \mathbb{N}, \tag{3.3}$$

with $\mathbb{E}[Y_{it} | \boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \text{Var}[Y_{it} | \boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \mu(\boldsymbol{x}_{it})$. The mean parameter $\mu(\boldsymbol{x}_{it})$ denotes the conditional expectation (and conditional variance) of Y_{it}. The regression function $\mu(\cdot)$ captures the relationship between the predictors $\boldsymbol{x}_{it}$ and the mean parameter in the Poisson distribution, indicating how the conditional expected count is influenced by the predictors. Choosing a functional form for $\mu(\cdot)$ defines a hypothesis function space $\mathcal{H}$ that includes all the candidate functions for modeling $\mu(\cdot)$. Typically, one aims at minimizing the Poisson deviance loss[1] (see subsection 4.1.2 and Table 4.1 of Wüthrich and Merz, 2023) over the training set:

[1]The criterion in Equation (3.4) is, in fact, the deviance loss scaled by a factor of 2.

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

$$\widehat{\mu} = \text{argmin}_{\mu \in \mathcal{H}} \left\{ -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t) \in \mathcal{T}_r} y_{it} \ln \left[\mu(\boldsymbol{x}_{it}) \right] - \mu(\boldsymbol{x}_{it}) - \ln (y_{it}!) \right\}. \tag{3.4}$$

Note that this is equivalent to maximizing the likelihood function. For some specifications of $\mu(\cdot)$, notably the log-linear specification, the criterion in Equation (3.4) is convex, which enables various convex optimization techniques to be applied.

Log-linear Poisson regression

In the Poisson regression context, one notable specification for the regression function is the log-linear form, where the mean parameter is expressed as the exponential of a linear function of the predictors:

$$\mu^{\text{LL}}(\boldsymbol{x}; \boldsymbol{\beta}) = \exp \{ \langle \boldsymbol{x}, \boldsymbol{\beta} \rangle \}, \tag{3.5}$$

where $\boldsymbol{\beta}$ denotes a vector of parameters, and $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ stands for the inner product between the predictor vector $\boldsymbol{x}$ and the coefficient vector $\boldsymbol{\beta}$. Log-linear Poisson regression has favorable properties, notably its interpretability stemming from the quasi-linearity of the link function $\mu(\cdot)$. Moreover, when maximum likelihood is used for parameter estimation, this regression model falls within the framework of GLMs. GLMs provide valuable properties, such as the asymptotic Gaussian distribution of the parameters $\boldsymbol{\beta}$, allowing for the estimation of standard errors, hypothesis testing, and construction of confidence intervals.

However, log-linear regression does have a significant drawback – its regression function, being linear, lacks flexibility. To address this limitation, various techniques can be employed. In fact, any supervised learning technique could be used for the specification of $\mu(\cdot)$, such as GAMs and tree-based models. Alternatively, one could incorporate non-linearity by computing polynomial terms (e.g., quadratic or cubic terms) of the predictors.

CANN Poisson regression

In some cases, the supervised learning problem may require even more flexibility, and neural networks are particularly useful in such scenarios. Neural networks are formidable function approximation machines, well-known for their ability to estimate a wide range of highly non-linear multivariate functions. One of the key advantages of neural networks is their ability to handle raw and unstructured data effectively. Because we deal with detailed telematics data, this capability forms the basis for adopting the CANN approach of Wüthrich and Merz (2019), which embeds a classical actuarial model into a neural network architecture. A CANN model consists of two distinct components: the classical regression model component and the neural network component. This architecture offers great flexibility, allowing for seamless integration of any classical model whose regression function is compatible with a neural network architecture. Likewise, the neural network component can employ various types of supervised architectures, such as CNNs, recurrent neural networks (RNNs), and other architectures tailored to the specific problem at hand. The classical regression model provides good initial estimations and serves as a guide for the neural network component. It offers a starting point for the network's optimization process, enabling faster convergence. The neural network component, in turn, refines the initial estimations, capturing additional signals and uncovering patterns that may have been missed by the classical model alone.

In our specific case, we use log-linear count regression as the classical model and a MLP as the neural network component in the CANN model. As a result, we have the following specification for the regression function:

$$\mu^{\text{CANN}}(\boldsymbol{x}; \boldsymbol{\beta}, \boldsymbol{\theta}) = \mu^{\text{LL}}(\boldsymbol{x}; \boldsymbol{\beta}) \times \mu^{\text{MLP}}(\boldsymbol{x}; \boldsymbol{\theta}), \tag{3.6}$$

where $\mu^{\text{MLP}}(\cdot)$ is the regression function of a MLP parametrized with $\boldsymbol{\theta}$. In a nutshell, an MLP consists of interconnected layers, including an input layer, hidden layers, and an output layer. Each layer applies an affine transformation to the inputs it receives, followed by a non-linear activation function ϕ. This

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

combination of linear transformations and non-linear activations allows MLPs to model complex non-linear relationships in the data. To delve into the mathematical description of an MLP, we can break down its structure starting from the input layer and progressing toward the output layer:

1. **Input layer** ($l = 0$): The input layer consists of n_0 nodes representing the input variables $\boldsymbol{x} = [x_1, x_2, \ldots, x_{n_0}] = \boldsymbol{z}^{(0)}$.
2. **Hidden layers** ($l = 1, 2, \ldots, L-2$): Each hidden layer l consists of n_l nodes, connected to the nodes from the previous layer $(l-1)$ and the nodes in the following layer $(l+1)$. The computations in the hidden layers involve an affine transformation of the inputs $\boldsymbol{z}^{(l-1)}$ followed by the application of the non-linear activation function ϕ. Let us denote the weight matrix between layers $l-1$ and l as $\boldsymbol{W}^{(l)}$ with dimensions (n_l, n_{l-1}) and the bias vector as $\boldsymbol{b}^{(l)}$ with dimensions $(n_l, 1)$. The computations in each hidden layer l can then be expressed as:
$$\boldsymbol{a}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l)}, \quad \boldsymbol{z}^{(l)} = \phi\left(\boldsymbol{a}^{(l)}\right), \tag{3.7}$$
where $\boldsymbol{a}^{(l)}$ represents the preactivation values in the l^{th} hidden layer, and $\boldsymbol{z}^{(l)}$ represents the post-activation values.
3. **Output layer** ($l = L-1$): The output layer consists of n_{L-1} nodes, representing the final output(s) of the MLP. The computations in the output layer consist of an affine transformation followed by an activation function g. We denote the weight matrix between layers $L-2$ (last hidden layer) and $L-1$ as $\boldsymbol{W}^{(L-1)}$ with dimensions (n_{L-1}, n_{L-2}) and the bias vector as $\boldsymbol{b}^{(L-1)}$ with dimensions $(n_{L-1}, 1)$. The computations within the output layer can be expressed as:
$$\boldsymbol{a}^{(L-1)} = \boldsymbol{W}^{(L-1)}\boldsymbol{z}^{(L-2)} + \boldsymbol{b}^{(L-1)}, \quad \boldsymbol{z}^{(L-1)} = g\left(\boldsymbol{a}^{(L-1)}\right). \tag{3.8}$$

Note that the number of output neurons n_{L-1} should match the number of modeled distribution parameters. In the context of Poisson regression, where we are modeling a single parameter μ, only one output neuron is necessary. The choice of the output activation function $g(\cdot)$ is important and should be aligned with the specific problem being tackled since it determines the range and properties of the output values. In our case, we need to ensure that the parameter μ, which represents the expected count, is always positive. While the exponential function is a natural choice to enforce positivity, it can sometimes lead to numerical instability, especially for large input values. As a better alternative, we choose to use the softplus function as the activation function for the output layer, defined as $\zeta(x) = \log(1 + \exp(x))$. The softplus function is well-behaved even for large input values, mitigating the issue of numerical instability that can arise with the exponential function. The parameters of the generic MLP described above, consisting of weight matrices and bias vectors, can be denoted as:
$$\boldsymbol{\theta} = \left\{\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}, \ldots, \boldsymbol{W}^{(L-1)}, \boldsymbol{b}^{(L-1)}\right\}. \tag{3.9}$$

Naturally, these parameters must be estimated. However, the criterion in Equation (3.4) is typically not convex, making it challenging to find the global minimum of the empirical risk. In practice, the goal is to find a "good enough" local minimum that yields satisfactory performance on the task. Gradient descent algorithms, such as stochastic gradient descent and its variants, are commonly employed to update iteratively the parameters $\boldsymbol{\theta}$ in the direction of the steepest descent. The backpropagation algorithm efficiently computes the gradients and propagates them through the network, enabling parameter updates.

With the introduced notation for the MLP, we can now express the specification in (3.6) as
$$\mu^{\text{CANN}}(\boldsymbol{x}; \boldsymbol{\beta}, \boldsymbol{\theta}) = \zeta\left\{\langle \boldsymbol{x}, \boldsymbol{\beta}\rangle + \boldsymbol{a}^{(L-1)}(\boldsymbol{x}; \boldsymbol{\theta})\right\}. \tag{3.10}$$
The corresponding computational graph for $L = 5$ (i.e., 3 hidden layers) is shown in Figure 2. Like a standard MLP, the network parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ can be estimated using gradient descent. A simplified pseudo-algorithm for the training of the Poisson CANN model is provided in Algorithm 3.1.1. The learning rate η is a hyperparameter that determines the step size taken every time a gradient descent step is performed. In other words, it controls how quickly or slowly the network parameters are updated

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

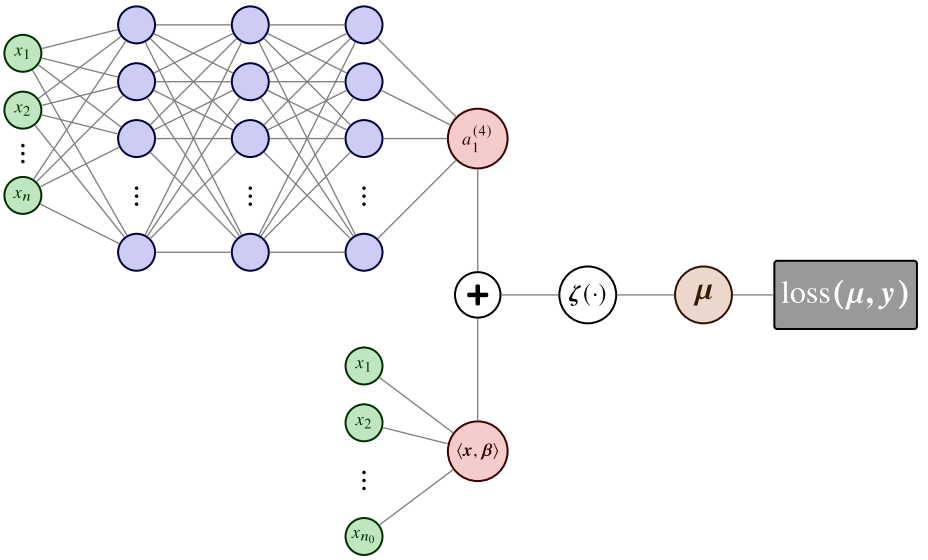

Figure 2. *CANN architecture for the Poisson specification. The MLP's preactivation output value $a_1^{(4)}$ is added to the log-linear model's preactivation output value $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ before being transformed with the softplus function $\zeta(\cdot)$. The resulting μ value is compared to the ground truth y using Poisson deviance loss. The architecture shown employs a 3-hidden-layer MLP but can be customized with any number of layers.*

during training. A higher learning rate allows for larger steps, which can lead to faster convergence. However, an excessively high learning rate may cause the optimization process to overshoot or oscillate around the minimum, hindering convergence. Conversely, a very low learning rate might result in slow convergence, requiring more iterations to reach an acceptable solution. Finding the right learning rate is important and is typically an empirical process that requires experimentation and tuning. In practice, mini-batch gradient descent is commonly used for training neural networks. It works by dividing the training data into smaller subsets, called mini-batches, and computing the gradients and parameter updates based on these mini-batches. This approach offers computational efficiency and improved generalization compared to regular gradient descent, making it a preferred choice in practice. For a comprehensive understanding of neural networks, we refer to the excellent book (Goodfellow *et al.*, 2016).

3.1.2. Negative binomial regression

One issue with the Poisson distribution is its equidispersion assumption. Indeed, we have that $\mu(\boldsymbol{x}) = \mathbb{E}[Y_{it}|\boldsymbol{X} = \boldsymbol{x}] = \text{Var}[Y_{it}|\boldsymbol{X} = \boldsymbol{x}]$. In practice, claim count data often exhibit overdispersion, where the observed variance of the claim count is greater than the mean. To address this limitation, alternative distributions allowing for overdispersion can be used. Among them, the negative binomial distribution (see, for instance, Denuit *et al.*, 2007 and Cameron and Trivedi, 2013) stands out as a common choice. Under the negative binomial specification, the PMF of the claim count for the i^{th} contract of vehicle i (Y_{it}), given its predictor vector ($\boldsymbol{x}_{it}$), can be written as

$$\mathbb{P}(Y_{it} = y_{it}|\boldsymbol{x}_{it}) = \frac{\Gamma(y_{it} + \phi)}{y_{it}!\Gamma(\phi)} \left(\frac{\phi}{\phi + \mu(\boldsymbol{x}_{it})}\right)^{\phi} \left(\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it}) + \phi}\right)^{y_{it}}, \quad \text{for} \quad y_{it} \in \mathbb{N}, \tag{3.11}$$

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Algorithm 1: Parameter estimation procedure – Poisson CANN model

Input: Training dataset $\{(\boldsymbol{x}_{it}, y_{it})\}_{(i,t)\in\mathcal{T}_r}$, learning rate η, number of epochs E
Output: Trained model parameters $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\beta}}$

Initialize model parameters $\hat{\boldsymbol{\theta}} = \mathbf{0}$ and $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^{\text{MLE}}$.

for *epoch* $\leftarrow$ *1* **to** E **do**

1. For each contract (i,t), apply the CANN regression function with current network parameters to compute the current estimated mean parameter $\hat{\mu}_{it}$:
$$\hat{\mu}_{it} = \mu^{\text{CANN}}(\boldsymbol{x}_{it}; \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}).$$
2. Compute the empirical risk over the training dataset:
$$\mathcal{R} = -\frac{1}{|\mathcal{T}_r|}\sum_{(i,t)\in\mathcal{T}_r} y_{it}\ln[\hat{\mu}_{it}] - \hat{\mu}_{it} - \ln(y_{it}!).$$
3. Perform backpropagation to compute the gradients of $\mathcal{R}$ with respect to the network parameters:
$$\nabla_{\boldsymbol{\beta}}\mathcal{R} \quad \text{and} \quad \nabla_{\boldsymbol{\theta}}\mathcal{R}.$$
4. Perform gradient descent using the learning rate:
$$\hat{\boldsymbol{\beta}} \leftarrow \hat{\boldsymbol{\beta}} - \eta\nabla_{\hat{\boldsymbol{\beta}}}\mathcal{R},$$
$$\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \eta\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{R}.$$

end

where $\phi > 0$ is a dispersion parameter. This can be seen as a generalization of the Poisson distribution. Indeed, the Poisson distribution is recovered when $\frac{1}{\phi} \to 0$. The first two centered moments are given by:

$$\mathbb{E}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \mu(\boldsymbol{x}_{it}) \quad \text{and} \quad \text{Var}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \mu(\boldsymbol{x}_{it}) + \frac{\mu(\boldsymbol{x}_{it})^2}{\phi}. \tag{3.12}$$

As can be seen, the negative binomial specification assumes overdispersion since $\text{Var}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] > \mathbb{E}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}]$. One can estimate the parameters of the regression function $\mu(\cdot)$ along with the dispersion parameter ϕ by minimizing the negative binomial deviance loss over the training set:

$$\{\widehat{\mu}, \widehat{\phi}\} = \text{argmin}_{\mu\in\mathcal{H},\phi>0}\left\{-\frac{1}{|\mathcal{T}_r|}\sum_{(i,t)\in\mathcal{T}_r}\ln\left[\frac{\Gamma(y_{it}+\phi)}{y_{it}!\Gamma(\phi)}\right] + \phi\ln\left[\frac{\phi}{\phi+\mu(\boldsymbol{x}_{it})}\right] + y_{it}\ln\left[\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it})+\phi}\right]\right\}. \tag{3.13}$$

Log-linear negative binomial regression

As in the Poisson case, a common specification for $\mu(\cdot)$ is the log-linear form, defined in Equation (3.5). In this case, the criterion in (3.13) is convex, and convex optimization can be used to estimate $\boldsymbol{\beta}$ and ϕ.

CANN negative binomial regression

Similar to the approach used for the Poisson case, a CANN architecture can be used to model the mean parameter in the negative binomial distribution. The specification for the regression function $\mu(\cdot)$

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

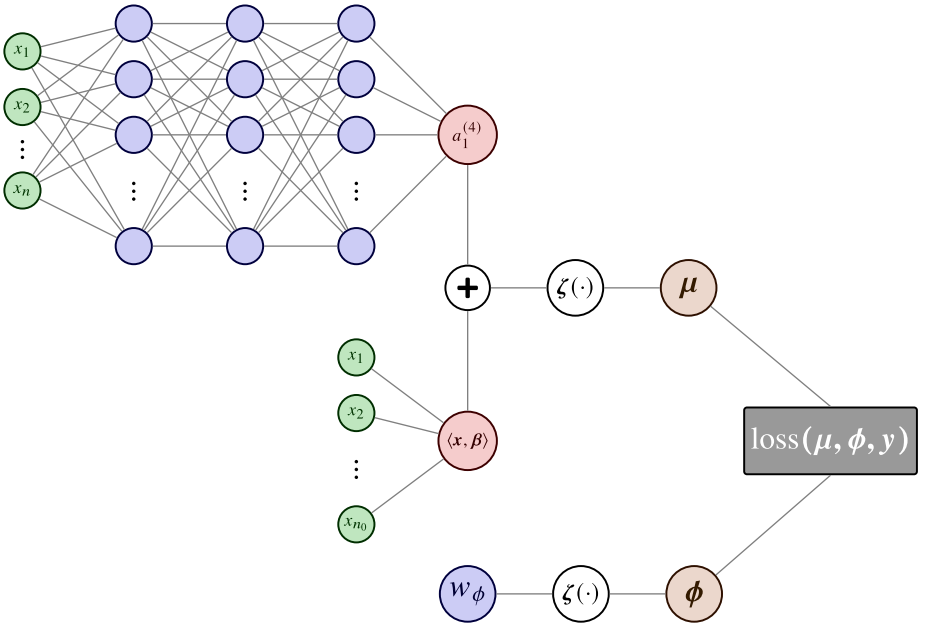

Figure 3. *CANN architecture for the negative binomial specification. The MLP's preactivation output value $a_1^{(4)}$ is added to the log-linear model's preactivation output value $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ before being transformed with the softplus function $\zeta(\cdot)$ to obtain the μ value of the negative binomial distribution. The ϕ value is obtained by transforming a real-valued parameter w_ϕ through the softplus function. The resulting parameters μ and ϕ are then compared to the ground truth y using negative binomial deviance loss. The architecture shown employs a 3-hidden-layer MLP but can be customized with any number of layers.*

remains identical to the Poisson case, as defined in Equation (3.10). In order to incorporate the extra distribution parameter ϕ, an additional output neuron is introduced in the network. This output neuron is connected to a neural network weight $w_\phi \in \mathbb{R}$ through the softplus function, ensuring that ϕ remains positive, that is, $\phi = \zeta(w_\phi)$. It is important to highlight that the distribution parameter ϕ is not directly connected to the input variables $\boldsymbol{x}$. As a result, no heterogeneity is incorporated into this parameter, and a common estimated value $\widehat{\phi}$ is used for all observations. The exact architecture for the negative binomial CANN model is depicted in Figure 3. The network parameters $\boldsymbol{\beta}$, $\boldsymbol{\theta}$, and w_ϕ can be learned by minimizing the criterion in Equation (3.13) using a procedure analogous to Algorithm 3.1.1.

3.2. Longitudinal models

Cross-sectional models assume independence between all contracts. However, in our case, the data exhibit clustering due to contracts being grouped by vehicle. While it is reasonable to assume independence between contracts from distinct vehicles, this assumption is less valid for contracts from the same vehicle. In reality, the claim counts of contracts within the same vehicle may be influenced by shared vehicle-specific characteristics, unobserved risk factors, or policy-level effects, resulting in dependence between observations within each vehicle cluster. To appropriately address this dependence, we transition from cross-sectional to longitudinal models, enabling the introduction of within-vehicle dependence. In the case of claim count data, a longitudinal model can efficiently leverage the history of the vehicles to refine the risk estimation for future contracts.

While various models are available to analyze longitudinal data, such as random effects models, fixed effects models, generalized estimating equations, and autoregressive models, among others, empirical

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

evidence in the context of claim count regression supports the effectiveness of random (or mixed) effects models (see Boucher *et al.*, 2008). In these models, a random effect, which is a random variable, is introduced in the specified distribution. For instance, in the case of count data, the specified distribution could be the Poisson distribution. The random effect is assumed to follow a certain distribution, such as a normal, gamma, or another appropriate distribution. The inclusion of the random effect allows for capturing the unobserved heterogeneity or individual-specific effects that cannot be accounted for by the observed covariates. It introduces additional variability into the model and accounts for the dependence within clusters. In longitudinal analysis, we need, for each vehicle i, to model the random vector of claim counts $\boldsymbol{Y}_{i,(1:T_i)} = (Y_{i1}, \ldots, Y_{i,T_i})$. The joint PMF can be expressed with

$$\mathbb{P}\left(\boldsymbol{Y}_{i,(1:T_i)} = \boldsymbol{y}_{i,(1:T_i)} | \boldsymbol{x}_{i,(1:T_i)}\right) = \int_{-\infty}^{\infty} \left(\prod_{t=1}^{T_i} \mathbb{P}(Y_{it} = y_{it} | \boldsymbol{x}_{i,(1:T_i)}, \theta_i) \right) f(\theta_i) d\theta_i, \tag{3.14}$$

where $f(\theta_i)$ is the PDF of the random effect.

3.2.1. MVNB regression

A MVNB regression model is obtained by introducing a gamma-distributed random effect in the mean parameter of the Poisson distribution. Specifically, we assume that the conditional distribution of Y_{it}, given $\Psi_i = \psi_i$, follows a Poisson distribution with mean $\mu(\boldsymbol{x}_{it})\psi_i$, where Ψ_i is a gamma-distributed random variable with mean 1 and variance $1/\phi$. The density of Ψ_i is given by

$$f_{\Psi_i}(\psi_i) = \frac{\phi^{\phi}}{\Gamma(\phi)} \psi_i^{\phi-1} e^{-\phi\psi_i}, \quad \psi_i > 0. \tag{3.15}$$

By using Equation (3.14), one can derive the joint distribution for the vector of claim counts:

$$\mathbb{P}\left(\boldsymbol{Y}_{i,(1:T_i)} = \boldsymbol{y}_{i,(1:T_i)} | \boldsymbol{x}_{i,(1:T_i)}\right) = \prod_{t=1}^{T_i} \left(\frac{\mu(\boldsymbol{x}_{it})^{y_{it}}}{y_{it!}} \right) \frac{\Gamma(y_{i\bullet} + \phi)}{\Gamma(\phi)} \left(\frac{\phi}{\mu_{i\bullet} + \phi} \right)^{\phi} \left(\frac{1}{\mu_{i\bullet} + \phi} \right)^{y_{it}}, \tag{3.16}$$

where $\mu_{i\bullet} = \sum_{t=1}^{T_i} \mu_{it}$ and $y_{i\bullet} = \sum_{t=1}^{T_i} y_{it}$. This joint distribution is commonly referred to as MVNB or negative multinomial distribution. Note that the Poisson distribution is retrieved when $\frac{1}{\phi} \to 0$. Furthermore, given the past claim history denoted as $\boldsymbol{y}_{i,(1:t-1)} = (y_{i1}, \ldots, y_{i,t-1})$ as well as current and past covariate vectors denoted as $\boldsymbol{x}_{i,(1:t)} = (\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{it})$, one can show that the number of claims at time (or contract) t follows a negative binomial distribution. The probability of observing y_{it} claims at time t, given the past claim history as well as past and current covariate vectors, is thus expressed with

$$\mathbb{P}(Y_{it} = y_{it} | \boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,1:t}) = \frac{\Gamma(y_{it} + \alpha_{it})}{y_{it}!\Gamma(\alpha_{it})} \left(\frac{\gamma_{it}}{\gamma_{it} + \mu(\boldsymbol{x}_{it})} \right)^{\alpha_{it}} \left(\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it}) + \gamma_{it}} \right)^{y_{it}}, \quad t = 1, 2, \ldots, T_i, \tag{3.17}$$

where $\alpha_{it} = \phi + \Sigma_{it}^{(y)}$ and $\gamma_{it} = \phi + \Sigma_{it}^{(\mu)}$. $\Sigma_{it}^{(y)} = \sum_{t'=1}^{t-1} y_{it'}$ and $\Sigma_{it}^{(\mu)} = \sum_{t'=1}^{t-1} \mu(\boldsymbol{x}_{it'})$ represent the number of past claims and the sum of past μ values for contract (i, t), respectively. In the special case when $t = 1$, there is no past history and we set $\Sigma_{it}^{(y)} = \Sigma_{it}^{(\mu)} = 0$, which yields $\alpha_{i1} = \gamma_{i1} = \phi$. The expected claim count, given the past history, is given by:

$$\mathbb{E}\left[Y_{it} | \boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,1:t}\right] = \mu(\boldsymbol{x}_{it}) \left(\frac{\phi + \Sigma_{it}^{(y)}}{\phi + \Sigma_{it}^{(\mu)}} \right) \tag{3.18}$$

$$= \mu(\boldsymbol{x}_{it}) \left(\frac{\alpha_{it}}{\gamma_{it}} \right). \tag{3.19}$$

Fitting an MVNB model, therefore, amounts to fitting a negative binomial model, where the parameters α_{it} and γ_{it} depend on the vehicle's history. The parameter ϕ and the regression function $\mu(\cdot)$ can be estimated by minimizing the deviance loss over the training set. This can be achieved through the following optimization problem:

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

$$\{\widehat{\mu}, \widehat{\phi}\} = \text{argmin}_{\mu\in\mathcal{H},\phi>0} \left\{ -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t)\in\mathcal{T}_r} \ln\left[\frac{\Gamma(y_{it}+\alpha_{it})}{y_{it}!\Gamma(\alpha_{it})}\right] + \alpha_{it} \ln\left[\frac{\gamma_{it}}{\gamma_{it}+\mu(\boldsymbol{x}_{it})}\right] + y_{it} \ln\left[\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it})+\gamma_{it}}\right] \right\}. \tag{3.20}$$

Log-linear MVNB regression

If the specification for $\mu(\cdot)$ is the log-linear form, defined in Equation (3.5), the criterion in (3.20) is convex, and convex optimization can be used to estimate $\boldsymbol{\beta}$ and ϕ.

CANN MVNB regression

In the MVNB case, the CANN architecture, as defined in Equation (3.10), can also be used as a specification for the regression function $\mu(\cdot)$. To incorporate the additional distribution parameters α_{it} and γ_{it}, two additional output neurons are introduced in the network, as depicted in Figure 4. The distribution parameters α_{it} and γ_{it} stem from a common parameter $\phi > 0$, and for the t^{th} contract of vehicle i, we have $\alpha_{it} = \phi + \Sigma_{it}^{(y)}$ and $\gamma_{it} = \phi + \Sigma_{it}^{(\mu)}$. The neuron representing ϕ is connected to a network weight $w_\phi \in \mathbb{R}$ through the softplus function, i.e., $\phi = \zeta(w_\phi)$. An MVNB CANN model can be trained with backpropagation and gradient descent, as outlined in Algorithm 3.2.1. Notice that for a vehicle i, the parameter γ_{it} depends on the μ parameter values for its past contracts. As the training procedure of the CANN model is iterative, the estimated μ values change at each iteration. Hence, it is crucial to update $\Sigma_{it}^{(\mu)}$ for each contract (i, t) at every iteration. This updating procedure is carried out in step 2 of Algorithm 3.2.1.

4. Practical application with telematics data

In this section, we explain how our CANN regression models are applied to our dataset. Additionally, we describe the application of the log-linear models, which serve as benchmark models in our analysis.

4.1. Log-linear models

The Poisson, negative binomial, and MVNB log-linear models are benchmarks for the Poisson, negative binomial, and MVNB CANN models. These models incorporate all 11 traditional risk factors from Table 1, including the real distance driven (although not strictly classified as a traditional risk factor). It is important to note that the `expo` variable is used as a covariate rather than as an offset. For each contract (i, t), these traditional risk factors are denoted by the vector $\boldsymbol{x}_{it}^{(\text{trad})}$. Notice that among the 11 traditional risk factors, 4 are categorical: `gender`, `marital_status`, `pmt_plan`, and `veh_use`. For these risk factors, the approach involves initially grouping all rare categories, defined as those representing 5% or less of the total number of observations, and labeling them as "others." We then encode them numerically using dummy encoding. All the resulting traditional covariates are then centered and scaled. Moreover, `commute_distance` contains missing values, which we fill in using median imputation.

Unlike neural networks, log-linear models have a hard time learning features directly from raw data. To put them on an equal footing with the CANN models, we manually engineer features from the telematics data used by these models. These 13 telematics features, described in Table 4, were specifically engineered from the telematics dataset as risk factors potentially correlated with the claiming risk. For each contract (i, t), these numerical handcrafted telematics features are denoted by the vector $\boldsymbol{x}_{it}^{(\text{hand})}$. Note that these handcrafted telematics features are also centered and scaled prior to being input into the log-linear models. The regression function for the μ parameter can thus be written as

$$\mu\left(\boldsymbol{x}^{(\text{trad, hand})}; \boldsymbol{\beta}\right) = \exp\left(\langle \boldsymbol{x}^{(\text{trad, hand})}, \boldsymbol{\beta}\rangle\right), \tag{4.1}$$

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

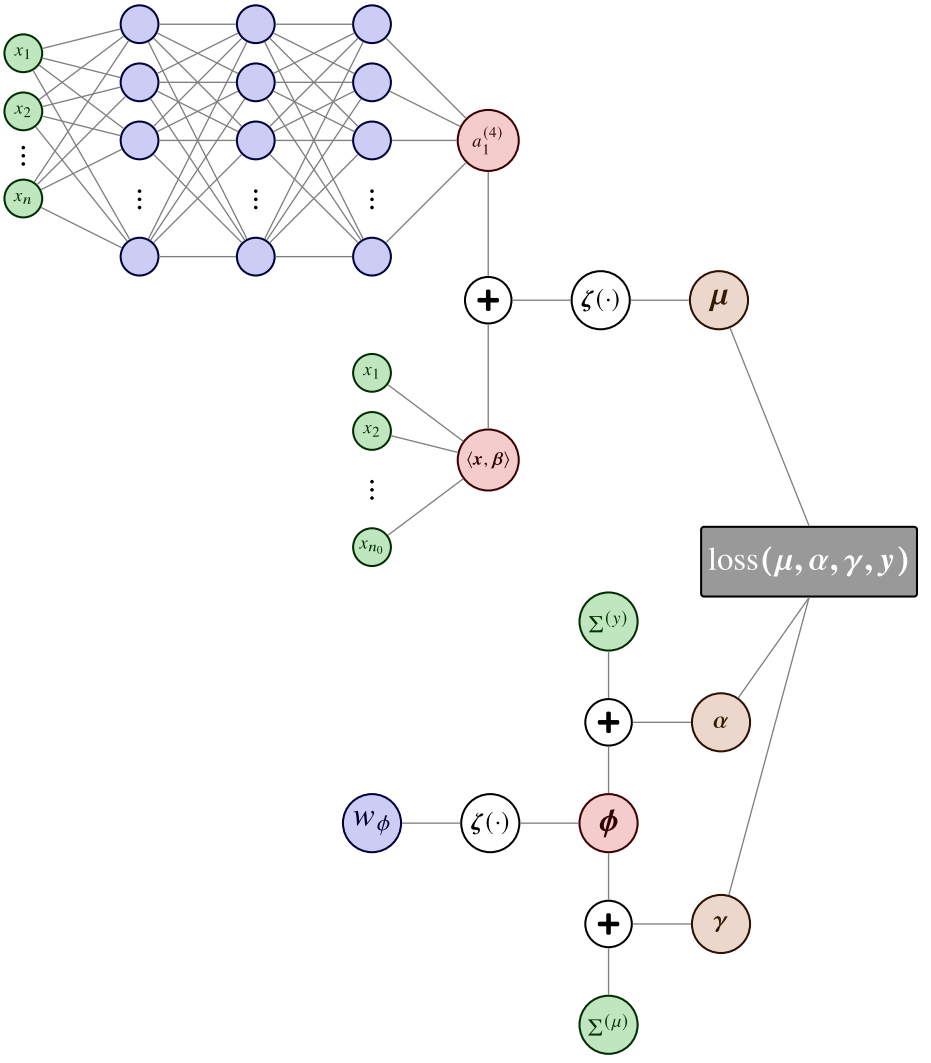

Figure 4. *CANN architecture for the MVNB specification. The MLP's preactivation output value $a_1^{(4)}$ is added to the log-linear model's preactivation output value $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ before being transformed with the softplus function $\zeta(\cdot)$ to obtain the μ value of the negative binomial distribution of Equation (3.17). The ϕ value is obtained by transforming a real-valued parameter w_ϕ through the softplus function. To obtain α, the sum of past claims $\Sigma^{(y)}$ is added to the ϕ parameter, while for γ, the sum of past μ values $\Sigma^{(\mu)}$ is added to the same ϕ parameter. The resulting distribution parameters μ, α, and γ are then compared to the ground truth y using negative binomial deviance loss. The architecture shown employs a 3-hidden-layer MLP but can be customized with any number of layers.*

where $\boldsymbol{x}^{(\text{trad, hand})}$ is the concatenation of $\boldsymbol{x}^{(\text{trad})}$ and $\boldsymbol{x}^{(\text{hand})}$. The parameters estimated on the training set are shown in a table that can be found on the project's GitHub repository (see (see Duval, 2023). Notably, when using telematics information, the estimated ϕ parameter in the MVNB log-linear model is higher. A higher ϕ value brings the correcting factor in Equation (3.19) closer to one, indicating reduced importance on past experience when telematics features are used. This underscores the relevance of the engineered telematics features.

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Algorithm 2: Parameter estimation procedure – MVNB CANN model

Input: Training dataset $\{(\boldsymbol{x}_{it}, y_{it})\}_{(i,t)\in\mathcal{T}_r}$, learning rate η, number of epochs E
Output: Trained model parameters $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\beta}}$ and $\hat{w}_\phi$

Initialize model parameters $\hat{\boldsymbol{\theta}} = \mathbf{0}$, $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^{\text{MLE}}$ and $\hat{w}_\phi$.
Compute the number of past claims for each contract (i,t): $\Sigma_{it}^{(y)} = \sum_{t'=1}^{t-1} y_{it'}$.

for *epoch* $\leftarrow$ *1* **to** E **do**

1. For each contract (i,t), apply the CANN regression function with current network parameters to compute the current estimated mean parameter $\hat{\mu}_{it}$:
$$\hat{\mu}_{it} = \mu^{\text{CANN}}(\boldsymbol{x}_{it}; \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}).$$
2. Initialize or update the sum of past μ values for each contract (i,t): $\hat{\Sigma}_{it}^{(\mu)} = \sum_{t'=1}^{t-1} \hat{\mu}_{it}$.
3. Compute the current estimated parameter $\hat{\phi}$:
$$\hat{\phi} = \zeta(\hat{w}_\phi).$$
4. Compute the current estimated parameter $\hat{\alpha}_{it}$ and $\hat{\gamma}_{it}$:
$$\hat{\alpha}_{it} = \hat{\phi} + \Sigma_{it}^{(y)},$$
$$\hat{\gamma}_{it} = \hat{\phi} + \hat{\Sigma}_{it}^{(\mu)}.$$
5. Compute the empirical risk over the training dataset:
$$\mathcal{R} = -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t)\in\mathcal{T}_r} \ln\left[\frac{\Gamma(y_{it}+\hat{\alpha}_{it})}{y_{it}!\Gamma(\hat{\alpha}_{it})}\right] + \hat{\alpha}_{it} \ln\left[\frac{\hat{\gamma}_{it}}{\hat{\gamma}_{it}+\hat{\mu}_{it}}\right] + y_{it} \ln\left[\frac{\hat{\mu}_{it}}{\hat{\mu}_{it}+\hat{\gamma}_{it}}\right].$$
6. Perform backpropagation to compute the gradients of $\mathcal{R}$ with respect to the network parameters:
$$\nabla_{\boldsymbol{\beta}}\mathcal{R}, \quad \nabla_{\boldsymbol{\theta}}\mathcal{R} \quad \text{and} \quad \nabla_{w_\phi}\mathcal{R}.$$
7. Perform gradient descent using the learning rate:
$$\hat{\boldsymbol{\beta}} \leftarrow \hat{\boldsymbol{\beta}} - \eta\nabla_{\hat{\boldsymbol{\beta}}}\mathcal{R},$$
$$\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \eta\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{R},$$
$$\hat{w}_\phi \leftarrow \hat{w}_\phi - \eta\nabla_{\hat{w}_\phi}\mathcal{R}.$$

end

4.2. CANN models

For the CANN regression models, we extract low-level descriptor vectors that are specifically designed to accurately describe the driving patterns within a particular contract, at least with the dataset we have. We expect the MLP component within the CANN models to learn meaningful high-level features from these low-level vectors. The hope is that the learned features in the hidden layers will be more relevant than the handcrafted features of Table 4. Each contract (i, t) is described by the following descriptor vectors, which provide a summary of its telematics information:

$$\boldsymbol{x}_{it}^{(h)} = \left(x_{it,1}^{(h)}, \ldots, x_{it,24}^{(h)}\right) \in \mathbb{R}^{24},$$
$$\boldsymbol{x}_{it}^{(d)} = \left(x_{it,1}^{(d)}, \ldots, x_{it,7}^{(d)}\right) \in \mathbb{R}^{7},$$

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Table 4. *Handcrafted telematics features extracted from the telematics dataset.*

Feature name	Description
`avg_daily_nb_trips`	Average daily number of trips
`frac_expo_evening`	Fraction of evening driving[1]
`frac_expo_fri_sat`	Fraction of driving on Friday and Saturday
`frac_expo_mon_to_thu`	Fraction of driving on Monday to Thursday
`frac_expo_night`	Fraction of night driving[2]
`frac_expo_noon`	Fraction of midday driving[3]
`frac_expo_peak_evening`	Fraction of evening rush hour driving[4]
`frac_expo_peak_morning`	Fraction of morning rush hour driving[5]
`max_trip_max_speed`	Maximum of the maximum speed of the trips
`med_trip_avg_speed`	Median of the average speeds of the trips
`med_trip_distance`	Median of the distances of the trips
`med_trip_max_speed`	Median of the maximum speeds of the trips
`prop_long_trip`	Proportion of long trips (>100km)

[1] 20h–0h.
[2] 0h–6h.
[3] 11h–14h.
[4] 17h–20h Monday to Friday.
[5] 7h–9h Monday to Friday.

$$\begin{aligned}
\boldsymbol{x}_{it}^{(a)} &= \left(x_{it,1}^{(a)}, \ldots, x_{it,14}^{(a)}\right) \in \mathbb{R}^{14}, \\
\boldsymbol{x}_{it}^{(m)} &= \left(x_{it,1}^{(m)}, \ldots, x_{it,16}^{(m)}\right) \in \mathbb{R}^{16}, \\
\boldsymbol{x}_{it}^{(k)} &= \left(x_{it,1}^{(k)}, \ldots, x_{it,10}^{(k)}\right) \in \mathbb{R}^{10}.
\end{aligned}$$

- The elements in vector $\boldsymbol{x}_{it}^{(h)}$ represent the fraction of driving during each of the 24 h of the day. Therefore, $x_{it,j}^{(h)}$ is the fraction of driving during the j^{th} h of the day for contract (i, t).
- The elements in vector $\boldsymbol{x}_{it}^{(d)}$ represent the fraction of driving during each of the 7 days of the week. Therefore, $x_{it,j}^{(d)}$ is the fraction of driving during the j^{th} day of the week for contract (i, t). Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday are denoted by $j = 1, 2, 3, 4, 5, 6, 7$, respectively.
- The elements in vector $\boldsymbol{x}_{it}^{(a)}$ represent the fraction of trips made in different average speed slots. For instance, $x_{it,j}^{(a)}$ denotes the fraction of trips made at an average speed between $10(j-1)$ and $10j$ kilometers per hour.
- The elements in vector $\boldsymbol{x}_{it}^{(m)}$ represent the fraction of trips made in different maximum speed slots. For instance, $x_{it,j}^{(m)}$ denotes the fraction of trips made where the maximum speed reached falls between $10(j-1)$ and $10j$ kilometers per hour.
- The elements in vector $\boldsymbol{x}_{it}^{(k)}$ represent the fraction of trips made in different distance slots. For instance, $x_{it,j}^{(k)}$ denotes the fraction of trips between $5(j-1)$ and $5j$ kilometers.

These descriptor vectors capture specific aspects of the driving patterns, such as hourly, weekly, average speed, and maximum speed distribution, providing valuable information for the MLPs. Since MLPs can only accept vectors as input, we concatenate these five vectors into a global telematics vector:

$$\boldsymbol{x}_{it}^{(tele)} = \left(\boldsymbol{x}_{it}^{(h)}, \boldsymbol{x}_{it}^{(d)}, \boldsymbol{x}_{it}^{(a)}, \boldsymbol{x}_{it}^{(m)}, \boldsymbol{x}_{it}^{(k)}\right). \tag{4.2}$$

We incorporate this telematics vector into the MLP component of the CANN models, together with the traditional risk factors $\boldsymbol{x}^{(trad)}$, enabling interactions between telematics and traditional inputs. In contrast,

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

the log-linear part of the CANN models only includes the traditional risk factors due to the difficulty of processing low-level information. The regression function for the μ parameter can thus be written as

$$\mu^{\text{CANN}}\left(\boldsymbol{x}^{(\text{trad, tele})}; \boldsymbol{\beta}, \boldsymbol{\theta}\right) = \zeta\left\{\langle \boldsymbol{x}^{\text{trad}}, \boldsymbol{\beta}\rangle + \boldsymbol{a}^{(L-1)}(\boldsymbol{x}^{\text{trad,tele}}; \boldsymbol{\theta})\right\}. \tag{4.3}$$

where $\boldsymbol{x}^{(\text{trad, tele})}$ is the concatenation of $\boldsymbol{x}^{(\text{trad})}$ and $\boldsymbol{x}^{(\text{tele})}$.

The CANN models are trained using the `torch` library in the `R` programming language, using mini-batch gradient descent with 256 observations per batch. The optimizer we use to perform gradient descent is the Adam optimizer, which is a fairly popular choice for training neural networks. Additionally, we use the reduce-on-plateau learning rate scheduler, which dynamically adjusts the learning rate based on the model's performance, automatically reducing it when the improvement plateaus, allowing for better optimization and convergence during training. For the MLP component of our CANN models, we opt for 3 hidden layers ($L = 5$) with 128, 64, and 32 hidden units, respectively ($n_1 = 128, n_2 = 64, n_3 = 32$). Please note that the number of hidden layers and units per layer are hyperparameters that typically require tuning. However, due to our computational constraints, we limit our investigation to this specific architecture, which we believe is a reasonable choice given our input vector's size. We choose the rectified linear unit as the activation function $\phi(\cdot)$ used in the hidden layers as it is a widely accepted standard choice. Additionally, we add batch normalization and dropout layers in between fully connected layers, which contribute to network stabilization and regularization. Note that a batch normalization layer is present directly after the input layer, which means that the inputs are normalized before entering the network. Batch normalization applies a normalization transformation to the input of a layer by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. By maintaining a stable mean and variance throughout the network, it can mitigate the vanishing or exploding gradients problem, enabling more effective and efficient training. The dropout layers, on the other hand, serve as a regularization technique that helps prevent overfitting. During training, dropout randomly sets a fraction of the hidden units of a given hidden layer to zero at each iteration, which forces the network to learn redundant representations and reduces the reliance on specific features. This regularization technique improves the model's ability to generalize well to unseen data.

4.3. CANN hyperparameter tuning

To maximize the performance of our CANN models, we use grid search for hyperparameter tuning, with the average loss observed on the validation dataset $\mathcal{V}_a$ as our optimization criterion. Additionally, we incorporate a regularization technique known as "early stopping" to determine the best number of epochs. This approach allows us to prevent overfitting and select the optimal number of epochs based on the lowest average loss achieved during training. We focus on three key hyperparameters: `l_start`, denoting the initial learning rate used in the reduce-on-plateau learning rate scheduler; `p`, which represents the probability of dropout in the dropout layers; and `factor`, indicating the factor by which the learning rate is multiplied upon reaching a plateau. A plateau is the point where there is no observed improvement in the validation loss for two consecutive epochs. Overall, we compute the average validation loss for all 45 combinations derived from the following hyperparameter values:

- `l_start`: 0.00001, 0.00005, 0.0001, 0.0005, 0.001
- `p`: 0.2, 0.3, 0.4.
- `factor`: 0.3, 0.4, 0.5

Remember that the network parameters in the classical components of the CANN models are initialized with the maximum likelihood estimators of the corresponding log-linear model, which is why we use relatively small learning rates. At the initialization stage, the network already produces reasonable predictions, reducing the need for large gradient descent steps. Considering the relatively small size of our networks, which are less prone to overfitting, we use rather small dropout probabilities `p`, all below or equal to 0.4. For the `factor` hyperparameter, three rather standard values (0.3, 0.4, and 0.5) are

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Table 5. *Optimal CANN models' performance on the validation set*

Specification	Average validation loss	Number of epochs
Poisson	0.2352	35
Negative binomial	0.2351	35
MVNB	0.2349	35

tested. The validation loss for each of the 45 combinations and the three specifications is presented in Table S1. It is worth noting that each model is trained for 30 epochs, and as early stopping is employed, the displayed average validation loss is based on the optimal number of epochs, which can be less than 30.

As can be seen, for all learning rates higher than 0.00001, the minimum average validation loss is achieved after a very small number of epochs, indicating that the network learns too quickly. Although the negative binomial and MVNB models perform best at a learning rate of 0.0001, we believe that with more epochs, we could achieve a lower average loss with a learning rate of 0.00001. This is particularly true since the average losses are quite similar for `lr_start` $= 0.00001$ and `lr_start` $= 0.0001$. When examining the first nine rows of Table S1, it becomes apparent that the `factor` hyperparameter has a negligible effect on the validation loss. On the other hand, the `p` hyperparameter only seems to have an impact on the validation loss for the Poisson model, performing best when $\mathtt{p} = 0.4$. Although the dropout rate does not significantly affect the performance for both the negative binomial and MVNB models, we also choose $\mathtt{p} = 0.4$ for these two models since the best performance is achieved at a high number of epochs (29 and 30 epochs, respectively). This suggests that with more epochs, there is potential for further performance improvement. Therefore, we select `lr_start` $= 0.00001$, `factor` $= 0.3$, and $\mathtt{p} = 0.4$ as the hyperparameters for all three specifications. We train the models again on the training set, this time for 100 epochs. The performance of the three models on the validation set is displayed in Table 5. As can be seen, all 3 specifications require 35 epochs to minimize the average validation loss.

5. Analysis

5.1. Performance assessment on the testing set

After carefully tuning the hyperparameters of our CANN models, we have at hand promising claim count models that are now nearing implementation. The next crucial step is to estimate their generalization capabilities accurately. We assess the models' generalization performance using the testing set $\mathcal{T}_e$, which has remained untouched until now. Using this independent dataset, we can estimate the models' predictive performance on unseen data points and determine their suitability for real-world applications.

We perform a comparative analysis between the CANN and the log-linear benchmark models, which allows us to evaluate our CANN models' relative performance and effectiveness against established approaches. In addition to log-linear models using telematics handcrafted features, we also assess the performance of log-linear models using telematics information in the form of the global telematics vector of Equation (4.2). These models have access to exactly the same information as the CANN models and therefore allow an alternative insightful comparison.

In order to fully grasp the value of telematics data, we also evaluate the performance of all 6 models (Poisson, negative binomial and MVNB log-linear and CANN models) without telematics, using only the 11 traditional risk factors as inputs. In the CANN models, the MLP component therefore only comprises the 11 traditional risk factors. This analysis helps us understand the contribution of telematics information in improving the predictive power of the models.

Finally, we also explore a variant of the training procedure for CANN models in which the parameters in the log-linear component $\boldsymbol{\beta}$ are fixed, meaning they are not trainable. While this approach may result in reduced performance, it offers the benefit of preserving interpretability while still capturing signals that may have been overlooked by the log-linear component. For these three models (Poisson, negative

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Table 6. *Performance of the baseline model on the testing set*

Scoring rule	Baseline model
Poisson deviance	0.3682
Logarithmic score	0.2470
Squared error	0.0697

binomial, and MVNB), we choose to initialize the parameters $\boldsymbol{\beta}$ with coefficients from the MVNB log-linear model, as these are estimated considering time dependence between contracts.

All 18 models are trained on the learning set, and their performance is evaluated on the testing set. To assess the performance, we employ three different scoring rules, namely the Poisson deviance, the logarithmic score, and the squared error. For each scoring rule, we compute the average value on the testing set. To assess the magnitude of the achieved performance, we begin by calculating the average scoring rule values for a baseline model. This baseline model is defined as a homogeneous Poisson log-linear model, where the estimation of the mean (and variance) parameter μ is estimated by the average number of claims per contract observed in the learning set:

$$\hat{\mu} = \frac{1}{|\{\mathcal{T}_r, \mathcal{V}_a\}|} \sum_{(i,t)\in\{\mathcal{T}_r,\mathcal{V}_a\}} y_{it}. \tag{5.1}$$

The average scoring rule values for this baseline model on the testing set are reported in Table 6. We can then evaluate the performance of each of the six models in terms of percentage improvement over the baseline model, as shown in Table 7. As can be seen, our CANN models consistently outperform their respective log-linear benchmark models across all scoring rules, indicating that the MLP component of the CANN architecture effectively captures signals that have been missed by the log-linear part. The improvement in prediction performance offered by CANN models obviously comes with computational costs. For the Poisson specification, the log-linear model using handcrafted telematics features had a training duration of 20 s, while the CANN model required about 24 min to complete 35 epochs. Furthermore, longitudinal models, including the MVNB CANN model, provide a substantial improvement over cross-sectional models, indicating that traditional and telematics data alone cannot fully capture the extent of the dependence between observations. This supports the use of longitudinal specifications, as they can capture dependence induced by unobserved variables. CANN models trained with fixed $\boldsymbol{\beta}$ parameters perform similarly to their trainable counterparts. Because CANN models using fixed $\boldsymbol{\beta}$ values are more interpretable and faster to train, we recommend using them over their counterpart with trainable parameters in the log-linear component. Lastly, it is worth noting that log-linear models using the global telematics vector are worse than those using handcrafted telematics features, indicating that they have a hard time learning from such detailed low-level data.

As highlighted in the works of Wüthrich (2022) and (2020), one limitation of neural networks is their lack of the balance property. This property, which GLMs satisfy (provided that the canonical link function is used), ensures that the sum of predicted values matches the sum of actual values. This is indeed highly desirable in pricing models, as it ensures that premiums adequately cover the expected loss costs. Table 8 displays the sum of predicted and actual values on both learning (in-sample) and testing (out-of-sample) sets for CANN models using telematics. As can be seen, the models overshoot the premiums by about 7% for in-sample predictions and about 3.5% for out-of-sample predictions. This is an issue for pricing because premiums that are too high may lead to adverse selection. While our focus is primarily on comparing models based on the quality of individual predictions, it is essential for pricing models that are actually used in practice to meet the balance property. Fortunately, several techniques (see Wüthrich, 2020, 2022 and Denuit *et al*., 2021) are available to address this balance issue in neural networks, including in CANN models.

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

Table 7. *Performance comparison of the CANN models and their corresponding log-linear benchmark model on the testing set. The percentages denote the improvement in the scoring rule compared to the baseline model*

	No telematics		With telematics			
Scoring rule	Log-linear model	CANN model	Log-linear model (handcrafted features)	Log-linear model (global telematics vector)	CANN model (trainable $\boldsymbol{\beta}$)	CANN model (fixed $\boldsymbol{\beta}$)
			Poisson			
Poisson deviance	5.23%	5.53%	5.68%	5.35%	5.78%	5.77%
Logarithmic score	3.90%	4.12%	4.23%	3.99%	4.31%	4.30%
Squared error	2.10%	2.26%	2.30%	2.07 %	2.38 %	2.38%
			Negative binomial			
Poisson deviance	5.24%	5.58%	5.68%	5.36%	5.81%	5.81%
Logarithmic score	3.99%	4.24%	4.31%	4.08%	4.41%	4.41%
Squared error	2.10%	2.27%	2.30%	2.07%	2.37%	2.37%
			MVNB			
Poisson deviance	5.36%	5.65%	5.79%	5.47%	5.90%	5.90%
Logarithmic score	4.07%	4.27%	4.38%	4.15%	4.46%	4.46%
Squared error	2.13%	2.29%	2.34%	2.08%	2.41%	2.42%

Table 8. *Comparison of the sum of predicted values and the sum of actual values for CANN models using telematics*

Model	Sum of predicted values (1)	Sum of actual values (2)	(1)/(2)
	In-sample (learning set)		
Poisson CANN	6618	6184	107.0%
Negative binomial CANN	6630	6184	107.2%
MVNB CANN	6622	6184	107.1%
	Out-of-sample (testing set)		
Poisson CANN	1592	1538	103.5%
Negative binomial CANN	1594	1538	103.6%
MVNB CANN	1595	1538	103.7%

5.2. Permutation feature importance and PDPs

One substantial drawback of neural networks is their difficulty of interpretation. However, researchers have developed tools to shed light on the inner workings of these black box algorithms. Two particularly useful tools in this context are permutation feature importance and PDPs.

Permutation feature importance is a model-agnostic technique that computes an importance score for each input (or variable) in a supervised learning algorithm. It achieves this by randomly permuting the values of a specific input while holding the other inputs constant and observing the resulting effect on the model's performance. By comparing the original model's performance with the permuted performance, we can determine the variable's importance relative to the chosen performance metric. Suppose we have a trained model and a holdout sample for evaluation purposes. We can initially score the model on this sample and measure its performance using a chosen metric, such as the average loss. Let us denote the average loss obtained with the original holdout sample as ℓ_{original}. To assess the importance of input j in

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

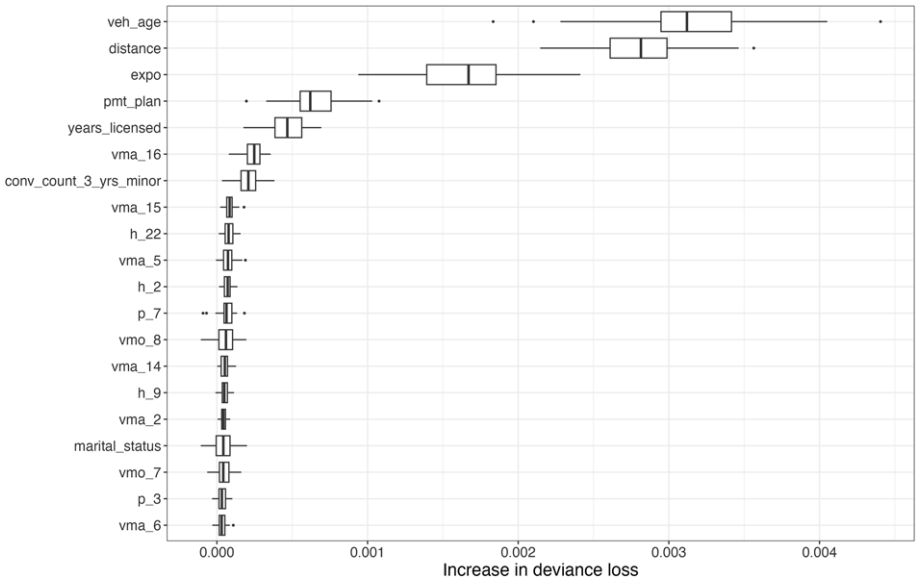

Figure 5. *Importance scores of the 20 most important variables obtained for the MVNB CANN model.*

the prediction process, we randomly shuffle the values of input j in the holdout sample and rescore the model. This process yields a new average loss, denoted as $\ell^{(j)}_{\text{permuted}}$, where the superscript j indicates that input j has been permuted. If input j is indeed important for the model's prediction, the permuted average loss $\ell^{(j)}_{\text{permuted}}$ is expected to be greater than the original average loss ℓ_{original}. This suggests that permuting the values of input j has a detrimental effect on the model's performance. To obtain an importance score for input j, we can compute the difference between the permuted average loss and the original average loss, resulting in the feature importance score FI_j:

$$\text{FI}_j = \ell^{(j)}_{permuted} - \ell_{original}.$$

To obtain a more reliable estimate of the importance score, this procedure can be repeated a certain number of times for input j, creating a distribution of the increase or decrease in the average loss. The whole procedure can then be repeated for all inputs. In Figure 5, the importance scores of the 20 most important variables for our best model, the MVNB CANN, are visualized using boxplots. Please note that the names used for the telematics inputs in Figure 5 differ from the introduced notation. However, a translation table is provided in Table A.1 of Appendix A to clarify the correspondence between the names used and the introduced notation. Each boxplot represents the distribution of the 100 importance scores assigned to a specific input obtained by shuffling and assessing the model 100 times. The performance metric used is the deviance loss (up to a factor 2). The analysis reveals interesting findings regarding the claim count model. As can be seen, the top 5 most important variables are from our set of 11 traditional risk factors. Notably, `veh_age`, `distance`, and `expo` play a significant role in the model's performance. When it comes to telematics inputs, those related to maximum speed demonstrate a substantial impact on the model's performance. Particularly, `vma_16`, representing the fraction of trips made at a maximum speed exceeding 150 kilometers per hour, stands out as the most important input. In general, the fraction of trips made at high maximum speeds, such as `vma_14`, `vma_15`, and `vma_16`, proves to be valuable for predicting claims. Additionally, it is interesting to observe that `h_22` and `h_2`, which represent the fraction of driving during night hours, contribute substantially to the assessment of risk. Importantly, the `gender` variable, often used by insurers as a risk factor, is rendered useless in the presence of telematics inputs. It ranks as the 70$^{\text{th}}$ most important variable (not showed in Figure 5), indicating its insignificance in the model's predictive power.

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

PDPs are valuable tools for understanding the relationship between a specific input variable and the output of a supervised learning model. PDPs are also model-agnostic, meaning they can be applied to different types of models. They provide insights into how changes in a particular input variable influence the model's predictions while keeping all other variables at fixed values. In other words, they illustrate the marginal effect of an input variable on the predicted outcome. To compute a PDP for a specific input variable j, the process involves the following steps. First, a grid of values is defined to cover the entire or plausible range of the variable's values. Next, while holding all other variables fixed, the input vector in the holdout dataset is sequentially replaced with each value from the defined grid. Subsequently, predictions are obtained using the trained model on the modified holdout dataset for each value. By plotting the input variable values on the x-axis and the corresponding average prediction on the y-axis, the resulting PDP visually showcases the relationship between the input variable and the model's predictions. Figure S1 displays the PDPs of the eight most important telematics inputs in the MVNB CANN model. The plots reveal that the risk, expressed as the expected number of claims, appears to increase in a linear fashion with the proportion of trips made at high maximum speeds, indicated by the input variables `vma_14`, `vma_15`, and `vma_16`. Additionally, there appears to be a positive linear association between the expected number of claims and the proportion of driving taking place during nighttime hours, specifically between 9 p.m. and 10 p.m. (`h_22`) and between 1 a.m. and 2 a.m. (`h_2`). It is important to emphasize that when interpreting PDPs, caution must be exercised, as the procedure assumes that the input variables are independent of each other. In particular, the interpretation of the PDPs related to the fraction of driving on Tuesdays (`p_2`) is challenging due to the correlation between the proportions of driving on different days of the week. For instance, if an insured individual drives in smaller proportions on Tuesdays, they will systematically drive in larger proportions on other days of the week.

6. Conclusions

In this study, we presented three claim count regression models leveraging telematics data in the form of trip summaries. Our models are based on the CANN architecture, specifically designed to address actuarial problems and harness rich and complex information such as data provided by telematics technology. One key aspect of our work is the integration of a longitudinal specification into the CANN framework. This integration allows us to effectively capture the time dependence between insurance contracts, which is important for accurately modeling claim counts. The superior results obtained with longitudinal models suggest that residual dependencies remain even after incorporating traditional and telematics data. This underscores the preference for longitudinal models, even when considering telematics information. While we specifically tested the implementation of the MVNB distribution in the CANN framework, it is important to note that the approach can be readily extended to other longitudinal distributions, such as the beta negative binomial distribution, detailed in Turcotte and Boucher (2023). Furthermore, our findings highlight the importance of telematics inputs related to the maximum speed reached during trips in the claim count models. With PDPs, we found that claim frequency is positively correlated with the fraction of trips made at high maximum speeds. Overall, the new approaches developed in this article represent a significant advancement in accurately modeling claim counts and enhancing the performance of predictive models in the context of UBI. Remarkably, the CANN regression models consistently outperform traditional log-linear models using handcrafted telematics features, as demonstrated by the superior performance across three performance metrics. These results are further supported by the use of a proper machine learning methodology that effectively prevents data leakage and mitigates the risk of producing falsely optimistic results. All the code necessary to replicate the analysis is available on the project's GitHub repository (see Duval, 2023).

While the available telematics data have been instrumental in improving our claim count models, we believe that further improvement can be achieved with access to richer data. For instance, if second-by-second data or additional information such as harsh acceleration/braking and distracted driving were accessible, we believe the performance could be further improved. Depending on the data format, different types of neural networks, such as CNN and RNN, could be used as the network component in the

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press

CANN models. Additionally, we acknowledge that with more time and computational power, a more comprehensive fine-tuning process of the CANN models could yield even better results than what we achieved. Notably, we were constrained in adjusting the number of hidden layers and units in the MLP components of the CANN models due to time and computational limitations. Moreover, a more advanced tuning method, beyond the grid search approach used in this study, could be employed to optimize model performance. Finally, it would be interesting to conduct further research investigating the impact of using a longitudinal model on telematics variables. It is expected that the importance of certain telematics variables would decrease when considering past claim history, as this historical data can provide insights into the claiming risk of an insured.

Supplementary material. To view supplementary material for this article, please visit https://doi.org/10.1017/asb.2024.4

Acknowledgments. The authors gratefully acknowledge *The Co-operators* for their generous financial support and for providing the data used in this paper through the *Co-operators Chair in Actuarial Risk Analysis*. Furthermore, they express their thanks to the editor and the reviewers for their valuable contributions that have helped improve the present article. Finally, the authors would like to extend their sincere appreciation to Marc Morin from the Research and Innovation team at *The Co-operators* for his invaluable assistance with the `torch` library.

Funding Statement. The authors thank *The Co-operators*, the *Natural Sciences and Engineering Research Council of Canada* and *Les fonds de recherche du Québec* for funding.

Competing interests. None.

References

Blier-Wong, C., Baillargeon, J.-T., Cossette, H., Lamontagne, L. and Marceau, E. (2020) Encoding neighbor information into geographical embeddings using convolutional neural networks. In *The Thirty-Third International Flairs Conference*.

Blier-Wong, C., Baillargeon, J.-T., Cossette, H., Lamontagne, L. and Marceau, E. (2021) Rethinking representations in P&C actuarial science with deep neural networks. arXiv preprint arXiv:2102.05784.

Bordoff, J.E. and Noel, P.J. (2008) *The Impact of Pay-as-You-Drive Auto Insurance in California*. Brookings Institution.

Boucher, J.-P., Côté, S. and Guillen, M. (2017) Exposure as duration and distance in telematics motor insurance using generalized additive models. *Risks*, **5**(4), 54.

Boucher, J.-P., Denuit, M. and Guillén, M. (2008) Models of insurance claim counts with time dependence based on generalization of Poisson and negative binomial distributions. *Variance*, **2**(1), 135–162.

Cameron, A.C. and Trivedi, P.K. (2013) *Regression Analysis of Count Data*, Vol. **53**. Cambridge University Press.

Denuit, M., Charpentier, A. and Trufin, J. (2021) Autocalibration and Tweedie-dominance for insurance pricing with machine learning. *Insurance: Mathematics and Economics*, **101**, 485–497.

Denuit, M., Maréchal, X., Pitrebois, S. and Walhin, J.-F. (2007) *Actuarial Modelling of Claim Counts: Risk Classification, Credibility and Bonus-Malus Systems*. John Wiley & Sons.

Dionne, G. and Vanasse, C. (1989) A generalization of automobile insurance rating models: The negative binomial distribution with a regression component. *ASTIN Bulletin: The Journal of the IAA*, **19**(2), 199–212.

Duval, F. (2023) Project's github repository. https://github.com/francisduval/article_3_count.

Embrechts, P. and Wüthrich, M.V. (2022) Recent challenges in actuarial science. *Annual Review of Statistics and Its Application*, **9**, 119–140.

Gabrielli, A., Richman, R. and Wüthrich, M.V. (2020) Neural network embedding of the over-dispersed Poisson reserving model. *Scandinavian Actuarial Journal*, **2020**(1), 1–29.

Gao, G., Meng, S. and Wüthrich, M.V. (2019) Claims frequency modeling using telematics car driving data. *Scandinavian Actuarial Journal*, **2019**(2), 143–162.

Gao, G., Wang, H. and Wüthrich, M.V. (2022) Boosting Poisson regression models with telematics car driving data. *Machine Learning*, 1–30.

Gao, G. and Wüthrich, M.V. (2018) Feature extraction from telematics car driving heatmaps. *European Actuarial Journal*, **8**(2), 383–406.

Gao, G. and Wüthrich, M.V. (2019) Convolutional neural network classification of telematics car driving data. *Risks*, **7**(1), 6.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. MIT Press.

Hausman, J., Hall, B. and Griliches, Z. (1984) Econometric models for count data with an application to the patents-R&D relationship. *Econometrica*, **52**, 909–938.

Jeong, H. (2022) Dimension reduction techniques for summarized telematics data. *The Journal of Risk Management, Forthcoming*.

Laporta, A.G., Levantesi, S. and Petrella, L. (2023) Neural networks for quantile claim amount estimation: A quantile regression approach. *Annals of Actuarial Science*, 1–21.

Lemaire, J., Park, S.C. and Wang, K. (2015) The use of annual mileage as a rating variable. *Astin Bulletin*, **46**(1), 39.
Litman, T. (2007) Distance-based vehicle insurance feasibility, costs and benefits. *Victoria*, 11.
Meng, S., Wang, H., Shi, Y. and Gao, G. (2022) Improving automobile insurance claims frequency prediction with telematics car driving data. *ASTIN Bulletin: The Journal of the IAA*, **52**(2), 363–391.
Nelder, J.A. and Wedderburn, R.W. (1972) Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, **135**(3), 370–384.
Schelldorfer, J. and Wuthrich, M.V. (2019) Nesting classical actuarial models into neural networks. Available at SSRN 3320525.
So, B., Boucher, J.-P. and Valdez, E.A. (2021) Synthetic dataset generation of driver telematics. *Risks*, **9**(4), 58.
Turcotte, R. and Boucher, J.-P. (2023) Gamlss for longitudinal multivariate claim count models. *North American Actuarial Journal*, 1–24.
Tzougas, G. and Kutzkov, K. (2023) Enhancing logistic regression using neural networks for classification in actuarial learning. *Algorithms*, **16**(2), 99.
Wüthrich, M.V. (2017) Covariate selection from telematics car driving data. *European Actuarial Journal*, **7**(1), 89–108.
Wüthrich, M.V. (2020) Bias regularization in neural network models for general insurance pricing. *European Actuarial Journal*, **10**(1), 179–202.
Wüthrich, M.V. (2022) The balance property in neural network modelling. *Statistical Theory and Related Fields*, **6**(1), 1–9.
Wüthrich, M.V. and Merz, M. (2019) Yes, we CANN! *ASTIN Bulletin: The Journal of the IAA*, 49(1), 1–3.
Wüthrich, M.V. and Merz, M. (2023) *Statistical Foundations of Actuarial Learning and its Applications*. Springer Nature.
Ziakopoulos, A., Petraki, V., Kontaxi, A. and Yannis, G. (2022) The transformation of the insurance industry and road safety by driver safety behaviour telematics. *Case Studies on Transport Policy*, **10**(4), 2271–2279.

A. Telematics Input Names Translation

Table A.1. *Telematics inputs names translation*

Introduced notation	Notation in the plots	Description
$x_{it,1}^{(h)}$	`h_1`	Fraction of driving between midnight and 1 a.m.
$x_{it,2}^{(h)}$	`h_2`	Fraction of driving between 1 a.m. and 2 a.m.
⋮	⋮	⋮
$x_{it,24}^{(h)}$	`h_24`	Fraction of driving between 11 p.m. and midnight
$x_{it,1}^{(d)}$	`p_1`	Fraction of driving on Mondays
$x_{it,2}^{(d)}$	`p_2`	Fraction of driving on Tuesdays
⋮	⋮	⋮
$x_{it,7}^{(d)}$	`p_7`	Fraction of driving on Sundays
$x_{it,1}^{(a)}$	`vmo_1`	Fraction of trips with average speed between 0 and 10 kph
$x_{it,2}^{(a)}$	`vmo_2`	Fraction of trips with average speed between 10 and 20 kph
⋮	⋮	⋮
$x_{it,14}^{(a)}$	`vmo_14`	Fraction of trips with average speed exceeding 130 kph
$x_{it,1}^{(m)}$	`vma_1`	Fraction of trips with maximum speed between 0 and 10 kph
$x_{it,2}^{(m)}$	`vma_2`	Fraction of trips with maximum speed between 10 and 20 kph
⋮	⋮	⋮
$x_{it,16}^{(m)}$	`vma_16`	Fraction of trips with maximum speed exceeding 150 kph
$x_{it,1}^{(k)}$	`d_1`	Fraction of trips with distance between 0 and 5 km
$x_{it,2}^{(k)}$	`d_2`	Fraction of trips with distance between 5 and 10 km
⋮	⋮	⋮
$x_{it,10}^{(k)}$	`d_10`	Fraction of trips with distance exceeding 45 km

https://doi.org/10.1017/asb.2024.4 Published online by Cambridge University Press