Explainable Deep Learning for Insurance Claim Frequency: from GLM to LocalGLMNet

Huynh Sang Dao Le Ngan Quach Hong Phuc Nguyen Thi Bao Duong

Abstract

This report studies the LocalGLMNet architecture, an interpretable deep learning architecture that combines the additive structure of generalized linear models with the flexibility of neural networks through feature-dependent regression coefficients. By embedding a GLM output layer within a feedforward network, LocalGLMNet enables nonlinear effects and interactions while retaining local interpretability via regression attentions and their gradients. Using the French Motor Third-Party Liability claim frequency dataset, we compare multiple LocalGLMNet variants with classical GLMs and standard neural networks, with particular focus on different treatments of categorical variables, including one-hot encoding and entity embeddings. Empirical results based on Poisson deviance and lift charts show that LocalGLMNet consistently outperforms GLMs in predictive performance, with the self-trained embedding variant achieving the best discrimination, while being able to provide insights into variable importance and interaction effects.

Table of contents

1 Overview on Methodology **1**
1.1 Introduction to LocalGLMNet 1
1.2 Interpreting LocalGLMNet 1

2 Experimental Setup and Result **2**
2.1 Interpreting numerical features 3
2.2 Interpreting categorical features 5

3 Conclusion and Perspective **6**

Bibliography **7**

Appendix A: Dataset Description **7**

Appendix B: Neural Network Preliminaries **7**
Feature Extractors Readout Layer 8
Gradient Descent 9
Covariate Preprocessing 10

Appendix C: Additional figures **10**

1 Overview on Methodology

1.1 Introduction to LocalGLMNet

Deep learning models have become widely used in statistical modeling because they produce highly competitive regression results, frequently surpassing traditional approaches such as generalized linear models due to its ability to model complex interactions. However, a key limitation of deep learning is the lack of interpretability, which is particularly problematic in the actuarial context, where models must be transparent, explainable, and justifiable to stakeholders.

There have been many attempts to explain complex models, including global explanation methods such as SHAP (Lundberg and Lee 2017) and local explanation techniques like LIME (Ribeiro, Singh, and Guestrin 2016). LocalGLMNet, proposed by Richman and Wüthrich (2023), represents an approach that embeds a LIME-inspired structure directly into a feedforward neural network by using a GLM output layer, thereby enabling local interpretability.

Let $\mu(\mathbf{X})$ denotes the prediction of a black-box model. The core idea of LIME is to approximate $\mu(X)$ locally around a given input $\mathbf{X}_0$ by an interpretable surrogate model $\mu_0(\mathbf{X})$ via fitting a weighted local linear regression on the response. The resulting glass-box model $\mathbf{X} \mapsto \mu_0(\mathbf{X})$ serves as a local explanation of μ in the neighborhood of X_0.

LocalGLMNet incorporates this principle directly into a neural network architecture by retaining the GLM structure $\mu(X) = g^{-1}(\beta_0 + \boldsymbol{\beta}^T\mathbf{x})$, but now allowing the *regression parameters* $\boldsymbol{\beta}$ be feature $\mathbf{x}$-dependent. $\boldsymbol{\beta}(\mathbf{x})$ is then called *regression attention*, and is to be modelled by a feedforward neural network.

Concretely, if we denote $\mathbf{z}^{(m)} : \mathbb{R}^{q_{m-1}} \to \mathbb{R}^{q_m}$ to be the m-th layer in a FNN (see Appendix B), and we let the regression attention $\boldsymbol{\beta}(\mathbf{x}) = \mathbf{z}^{(d:1)} = \mathbf{z}^{(d)} \circ ... \circ \mathbf{z}^{(1)}$ be a depth-d neural network, then the LocalGLMNet is defined as

$$x \mapsto g(\mu) = g(\mu(x)) = \beta_0 + \langle \boldsymbol{\beta}, \mathbf{x} \rangle$$

This is basically a FNN architecture with a skip connection and is demonstrated in Figure 1.

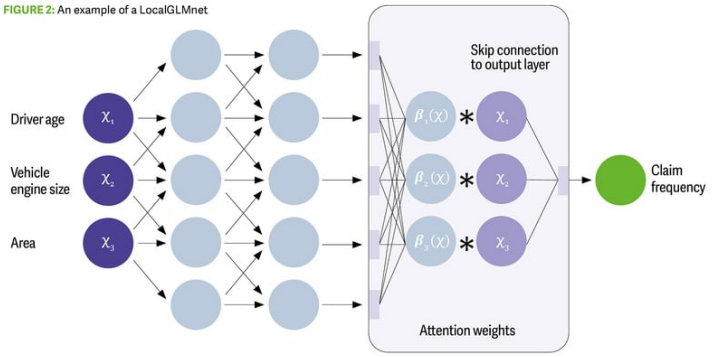

Figure 1: LocalGLMNet Architecture (Gawlowski et al. 2024)

1.2 Interpreting LocalGLMNet

LocalGLMNet admits several interpretable special cases that relate the learned regression function to classical regression models as follows: - If $\beta_j(\mathbf{x}) = \beta_j$ (i.e. feature-independent), the corresponding term reduces to the classical GLM component $\beta_j x_j$. - If $\beta_j(\mathbf{x}) \equiv 0$, the feature x_j

does not contribute to the regression function and can be excluded. - When $\beta_j(\mathbf{x})$ depends only on x_j, the term $\beta_j(x_j)x_j$ represents a nonlinear main effect without interactions. In contrast, dependence of $\beta_j(\mathbf{x})$ on multiple covariates indicates the presence of interaction effects.

Classical variable selection procedures for GLMs rely on asymptotic normality of MLE, which does not hold for LocalGLMNet due to SGD and early stopping. To address this issue, LocalGLMNet employs an empirical Wald-type testing procedure for $H_0 : \beta_j(\mathbf{x}) = 0$ based on a randomly generated reference feature. An additional completely random covariate $x_{q+1} \sim \mathcal{N}(0, 1)$ is appended to each observation $\mathbf{x}^+ = (\mathbf{x}^T, x_{q+1})^T$, and an interval I_α is constructed for confidence level α:

$$I_\alpha = \left[z_{\alpha/2} \cdot \hat{s}(\hat{\beta}_{q+1}),\ z_{1-\alpha/2} \cdot \hat{s}(\hat{\beta}_{q+1}) \right]$$

The hypothesis H_0 above can be rejected if the coverage of I_α for $\hat{\beta}_j(\mathbf{x}^+)$ is substantially smaller than $1 - \alpha$.

Interactions in LocalGLMNet are analyzed by examining the gradients of the regression attentions. For each feature x_j, the gradient

$$\nabla\beta_j(\mathbf{x}) = \left(\frac{\partial\beta_j(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial\beta_j(\mathbf{x})}{\partial x_q} \right)^\top$$

indicates whether the contribution of x_j is locally linear or involves interactions with other covariates. If the off-diagonal components are negligible, the term $\beta_j(\mathbf{x})x_j$ behaves as a (possibly nonlinear) main effect; otherwise, nonzero components signal interaction effects.

Also, the model encounters some identifiability problem due to the flexibility of the FNN. That is, we can have $\beta_j(\mathbf{x}) = x_{j'}$ by learning a regression $\beta_j(\mathbf{x}) = x_{j'}/x_j$. It is recommended to initialize the network weights such that gradient descent starts from the MLE of a fitted GLM, which anchors the optimization at an interpretable solution and ensures the training model loss would at most be the GLM loss.

2 Experimental Setup and Result

We conduct all experiments on the French Motor Third-Party Liability claim frequency dataset (Dutang and Charpentier 2025), a standard benchmark in actuarial science. Appendix A provides a detailed description of the variables used in the dataset. The dataset is randomly split into a 60-20-20 training-validation-test sets, where the training set is used for model training, the validation set for model selection and early stopping in NN training, and the test set for final performance evaluation. Numerical features are simple to handle and are standardized as gradient descent converges faster. Dealing with categorical features are, however, complex: here we investigate the use of pretrained entity embeddings using a different feedforward neural network (FNN) (Guo and Berkhahn 2016) to fit GLM and LocalGLMNet, and also using dummy/one-hot encodings.

We consider a suite of models in the experimental evaluation. As a baseline, **Null** denotes an intercept-only Poisson GLM, while **GLM_num** is a standard Poisson GLM using numerical covariates only. To incorporate categorical information, we evaluate 2 Poisson GLM models, **GLM_emb** which uses entity embeddings trained by **FNN**, and **GLM_cat** which uses dummy encoding for categorical features. Beyond global models, we fit several LocalGLMNet variant: **LocalGLMNet_num**, **LocalGLMNet_emb**, and **LocalGLMNet_cat**, which extend the

corresponding GLMs by allowing locally varying coefficients, as well as **LocalGLMNet_Self-Emb**, which jointly learns entity embeddings within the LocalGLMNet framework. Finally, the **FNN** has its final layer refitted to ensure balance.

Model estimation follows a staged procedure: for each GLM fitted, we use its estimated coefficients to initialize the corresponding LocalGLMNet and use the estimated coefficients to initialize the LocalGLMNet; for models with entity embeddings, we train the FNN a priori to learn the embeddings, then fit models using these learned embeddings. For categorical features with one-hot encoding, we initialize LocalGLMNet_cat using the GLM_cat estimated coefficients, setting the reference-level coefficients to zero. All neural networks employ four hidden layers with widths 64, 48, 32, and 16 and are trained for up to 100 epochs with early stopping using the validation set.

Model performance is assessed using the Poisson deviance loss, and results are summarized in Table 1. A multiple lift chart Figure 2 is also used to compare models – a model with higher lift (last decile minus first decile) is better discriminating between risky and less risky policyholders.

Table 1: Poisson deviance losses for different models

Model	In / out sample loss
Null model	72.827 / 73.074
GLM_num	70.296 / 70.746
GLM_emb	70.222 / 70.652
GLM_cat	70.194 / 70.637
FNN	66.953 / 68.145
LGLMNet_num	67.363 / 68.366
LGLMNet_cat	66.953 / 68.230
LGLMNet_emb	66.658 / 68.009
LGLMNet_Self-emb	66.782 / 67.974

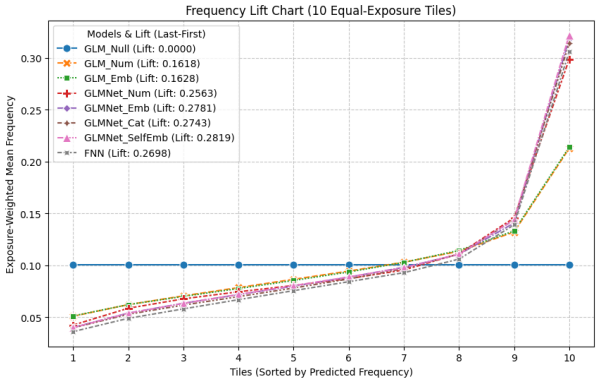

Figure 2: Multiple lift chart comparing models

Overall, all LocalGLMNet variants consistently outperform the corresponding GLM models, particularly in the upper risk tiles, demonstrating superior risk discrimination. Among all approaches, LocalGLMNet with self-trained embeddings achieves the highest lift, indicating that end-to-end learned embeddings provide more informative representations. These results confirm that LocalGLMNet improves predictive performance over classical models like GLMs.

2.1 Interpreting numerical features

For simplicity, we interpret the distribution of regression attentions of LocalGLMNet_num model; see Figure 3 and Figure 4

BonusMalus is the most influential variable in the model, with regression attention weights clearly shifted away from zero, confirming the central role of the Bonus–Malus system in explaining claim frequency. The attention weights are large and positive at low values, then decrease sharply before stabilizing. This reflects a strong penalization for unfavorable claim histories, with a diminishing marginal effect as the Bonus–Malus level increases.

DrivAge tends to have positive attention weights, suggesting that driver age generally contributes positively to the predicted risk intensity, with effects that depend on the overall insurance profile. The attention weights for DrivAge exhibit a smooth, non-monotonic pattern. They are higher

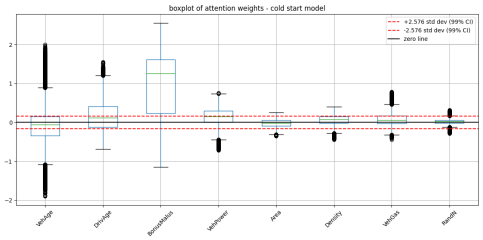

Figure 3: Box plot of attention weights

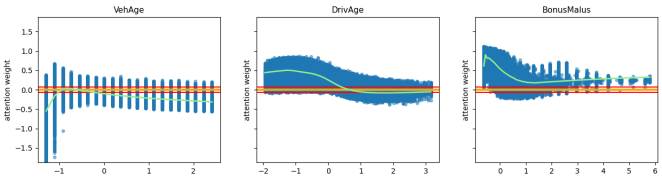

Figure 4: Scatter plot of attention weights

for younger drivers, peak at intermediate ages, and gradually decline for older drivers. This confirms that the effect of driver age is nonlinear and varies across age groups.

In contrast, VehAge shows a predominantly negative shift, indicating that older vehicles are typically assigned lower risk intensity, although the presence of extreme values reflects heterogeneity across different segments.For VehAge, the attention weights are predominantly negative for newer vehicles and gradually increase toward zero as vehicle age rises. This indicates that newer cars tend to be associated with higher predicted risk intensity, while the influence of vehicle age diminishes for older vehicles.

In contrast, the remaining numerical variables exhibit attention weights concentrated around zero with nearly flat trends, suggesting weak or approximately constant effects (see Figure 12)

These results suggest a strong interaction between DrivAge and BonusMalus, which will be examined in more detail in the next section.

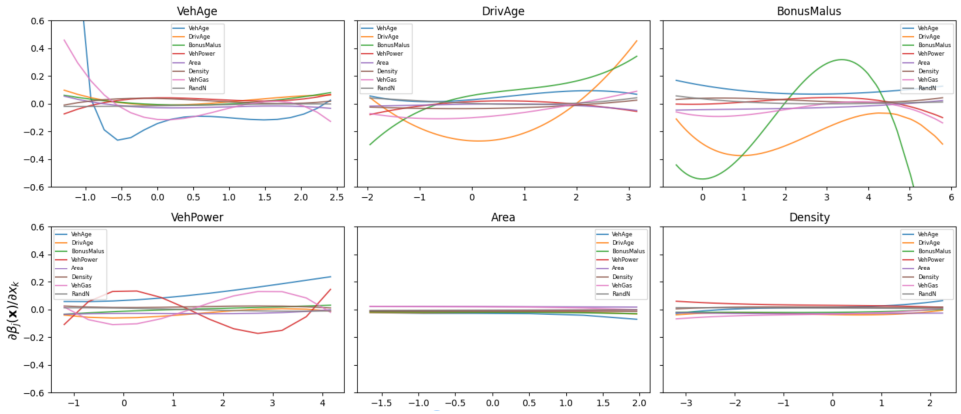

Figure 5: Interaction

Strong interaction effects are mainly observed among BonusMalus, DrivAge, and VehAge. In particular, the gradients of BonusMalus vary substantially with respect to DrivAge and VehAge, indicating that the effect of claim history depends on both driver age and vehicle age, rather than acting independently.

In the **DrivAge** interaction panel, the curve corresponding to **BonusMalus** changes sign across the age range. For younger drivers, the gradient $\partial\beta_{\text{BonusMalus}}(\mathbf{x})/\partial x_{\text{DrivAge}} < 0$, indicating that increases in driver age tend to reduce the local importance of BonusMalus within this group. In contrast, for older drivers the gradient becomes positive, implying that further increases in driver age strengthen the role of claim history in risk assessment.

Moreover, the curve associated with **DrivAge** itself is clearly non-zero and varies with age, confirming that $\beta_{\text{DrivAge}}(\mathbf{x})$ is non-constant and nonlinear, rather than a fixed GLM-like coefficient.

The interaction patterns involving DrivAge further confirm its role as a contextual variable, as changes in driver age modulate the contribution of BonusMalus and other numerical features across different age groups.

In contrast, interactions involving VehPower, Area, Density and baseline RandN remain close to zero, indicating weak or negligible interaction effects.

2.2 Interpreting categorical features

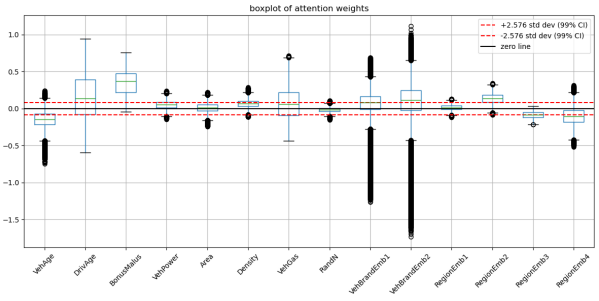

Figure 6: Box plot of LocalGLMNet_emb attention weight

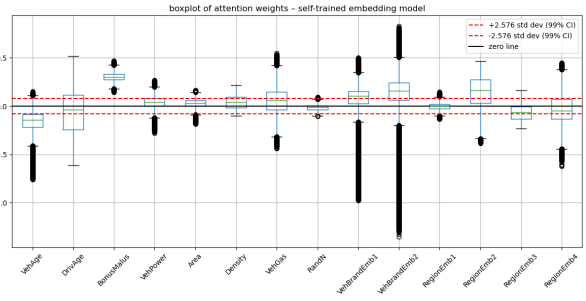

Figure 7: Box plot of LocalGLMNet_SelfEmb attention weight

We consider two models: one using embeddings pre-trained with a FNN and another in which self embeddings model.

For both models, the attention weights associated with the embedding dimensions are largely centered around zero, with moderate dispersion for a subset of components. This indicates that categorical variables contribute to the predictions, but their influence remains secondary compared to dominant numerical features such as BonusMalus and DrivAge.

Overall, the distributions of embedding attention weights are qualitatively similar across the two models, suggesting that the learned importance of categorical embeddings does not differ substantially between the FNN-pretrained and self-trained approaches. Despite these similarities in attention patterns, the model with self-trained embeddings achieves superior predictive performance.

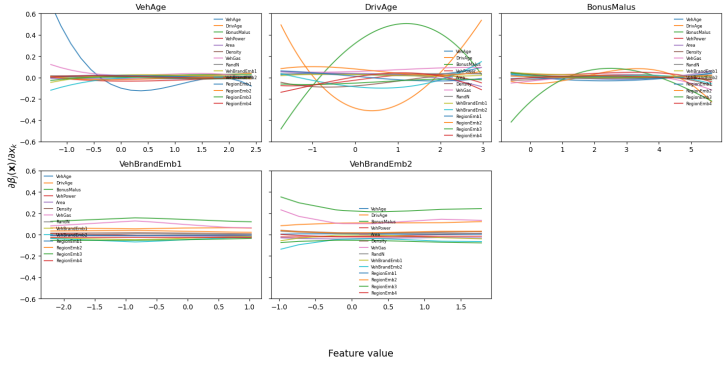

Figure 8: Interaction of LocalGLMNet_emb with categorical embeddings

Regarding interaction effects, VehAge, DrivAge, and BonusMalus continue to exhibit meaningful interactions, consistent with the numerical-only model. In contrast, the categorical embedding dimensions show only weak interaction patterns, indicating that embeddings do not introduce substantial additional interactions. We only plot the interaction of features with significantly non-zero gradient

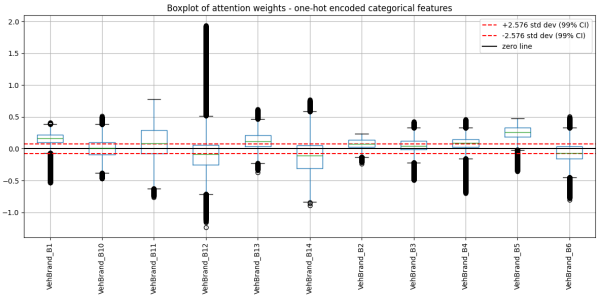

Figure 9: Box plot of Brand in LocalGLMNet_cat

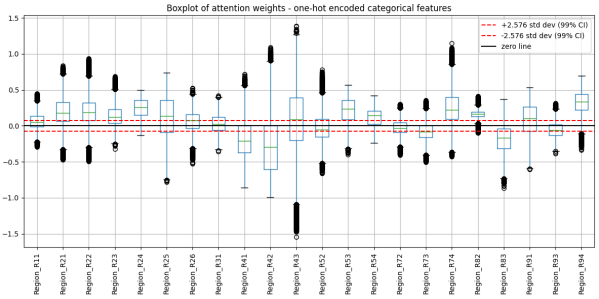

Figure 10: Box plot of Region in LocalGLMNet_cat

The boxplots of Vehicle Brand and Region illustrate the heterogeneous contributions of categorical levels in the LocalGLMNet_cat. Differences in medians across levels indicate that certain brands and regions are systematically associated with higher or lower predicted claim frequency.

Moreover, the varying dispersion of the boxes reflects different degrees of interaction with other features: levels with wider distributions exhibit stronger context-dependent effects, while tighter boxes suggest more stable, near-additive behavior. Overall, these results help us interpret different category levels better than the entity embedding approach; however, it sacrifices some predictive accuracy comparing to **LocalGLMNet_SelfEmb** model which learns the embedding in a more optimal way than pure one-hot embedding.

3 Conclusion and Perspective

This study demonstrates that LocalGLMNet provides an effective and principled compromise between the interpretability of GLMs and the predictive accuracy of neural networks. Across all experimental settings, LocalGLMNet variants consistently outperform their corresponding GLM baselines in terms of Poisson deviance and lift, with particularly strong improvements observed in the upper risk deciles. Detailed analyses of regression attentions reveal meaningful nonlinear main effects and interaction patterns, especially among key numerical variables such as BonusMalus, DrivAge, and VehAge, whereas less relevant features exhibit weak and near-zero contributions. Additional figures and results are provided in the Appendix C for reference.

Regarding categorical variables, our empirical comparison shows that entity embeddings substantially improve predictive performance over one-hot encoding, particularly when embeddings are learned end-to-end within the LocalGLMNet framework. Although the self-trained embedding model achieves the best overall performance, the categorical LocalGLMNet with dummy variables offers more direct interpretability at the level of individual categories, highlighting an inherent trade-off between predictive accuracy and interpretability.

A natural direction for future research is the incorporation of regularization schemes inspired by the LASSO and ElasticNet frameworks into LocalGLMNet (Richman and Wüthrich 2022). Penalizing regression attentions could enable automatic variable selection, promote sparsity in local coefficients, and further stabilize model estimation, especially in high-dimensional settings with many correlated features.

Bibliography

Dutang, Christophe, and Arthur Charpentier. 2025. *CASdatasets: Insurance Datasets*. https://doi.org/10.57745/P0KHAG.

Gawlowski, Karol, John Condon, Jack Harrington, and Davide Ruffini. 2024. "Gladiators, Ready! Pricing Models Fight It Out." *The Actuary*. https://www.theactuary.com/2024/03/05/gladiators-ready-pricing-models-fight-it-out.

Guo, Cheng, and Felix Berkhahn. 2016. "Entity Embeddings of Categorical Variables." *arXiv Preprint arXiv:1604.06737*.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. "Multilayer Feedforward Networks Are Universal Approximators." *Neural Networks* 2 (5): 359–66. https://doi.org/https://doi.org/10.1016/0893-6080(89)90020-8.

Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." In. Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. ""Why Should i Trust You?": Explaining the Predictions of Any Classifier." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–44. ACM. https://doi.org/10.1145/2939672.2939778.

Richman, Ronald, and Mario V. Wüthrich. 2022. "LASSO Regularization Within the LocalGLMnet Architecture." *Adv. Data Anal. Classif.* 17 (4): 951–81. https://doi.org/10.1007/s11634-022-00529-z.

———. 2023. "LocalGLMnet: Interpretable Deep Learning for Tabular Data." *Scandinavian Actuarial Journal* 2023 (1): 71–95. https://doi.org/10.1080/03461238.2022.2081816.

Wuthrich, Mario V., Ronald Richman, Benjamin Avanzi, Mathias Lindholm, Marco Maggi, Michael Mayer, Jürg Schelldorfer, and Salvatore Scognamiglio. 2026. "AI Tools for Actuaries." *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5162304.

Appendix A: Dataset Description

Table 2: Description of the main variables in the `freMTPL2freq` dataset{#tab-data-description}

Variable	Type	Actuarial Meaning
`ClaimNb`	Count (integer)	Number of claims during the exposure period.
`Exposure`	Numerical	The period of exposure for a policy, in years.
`VehAge`	Numerical	The vehicle age, in years.
`DrivAge`	Numerical	The driver age, in years.
'VehPower	Numerical / Ordinal	The power level of the vehicle.
`VehGas`	Categorical	Fuel type (diesel or gasoline).
`BonusMalus`	Numerical	Regulatory bonus–malus coefficient.
`Density`	Numerical	Population density of the policyholder's city.
`Area`	Categorical	Urban versus rural classification.
`Region`	Categorical (high-cardinality)	The policyholder's region in France.
`VehBrand`	Categorical (high-cardinality)	The car brand.

Appendix B: Neural Network Preliminaries

This section is an extract of Chapter 5 in Wuthrich et al. (2026).

Feature Extractors Readout Layer

In a nutshell, networks perform representation learning, meaning that multi-layer networks learn in each layer of the architecture a new representation of the covariates X (inputs). This is done by massaging the covariates X through a feature extractor so that eventually the newly learned representation of the covariates has a suitable structure to enter a (generalized) linear model. This is illustrated by the feature extractor (in blue) and the (generalized) linear model readout (in green) in Figure 11.

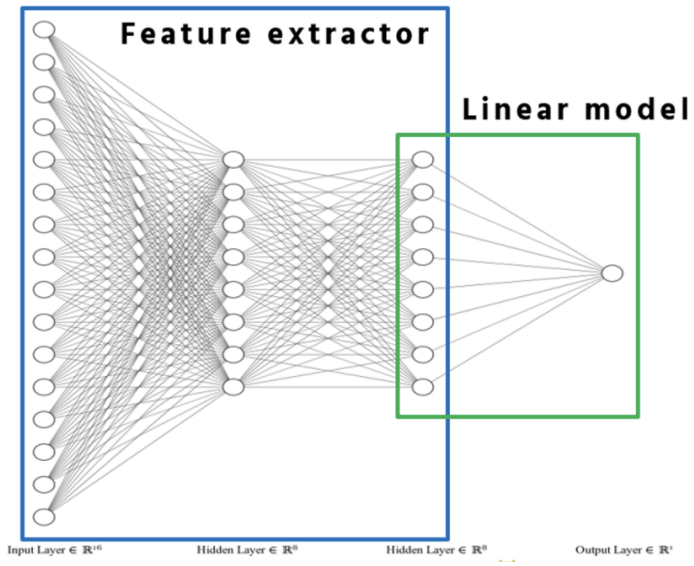

Figure 11: Feed-forward neural network architecture

The universality theorems state that any compactly supported continuous regression function can be approximated arbitrarily well by a sufficiently large FNN, under mild assumptions. See Hornik, Stinchcombe, and White (1989) for a statement and proof.

This implies FNNs are highly flexible: any continuous, compactly supported regression function can be approximated within the class. However, on finite samples, there is no parsimonious model; infinitely many near-equivalent candidates exist. Model selection is high-dimensional and non-convex, often yielding local optima.

Feed-forward neural network architectures (feature extractors) consist of (hidden) FNN layers

$$\mathbf{z}^{(m)} : \mathbb{R}^{q_{m-1}} \to \mathbb{R}^{q_m}, \quad m \geq 1.$$

Each FNN layer performs a non-linear transformation of the covariates. The main ingredients of such a FNN layer $\mathbf{z}^{(m)}$ are:

(a) the number $q_m \in \mathbb{N}$ of neurons (also called units);

(b) the non-linear activation function $\phi : \mathbb{R} \to \mathbb{R}$; and

(c) the network weights (representing part of the model parameter ϑ).

Items (a) and (b) are hyper-parameters selected by the modeler, and the network weights in item (c) are parameters that are learned during network training (model fitting).

Select $d \in \mathbb{N}$ of these FNN layers $(\mathbf{z}^{(m)})_{m=1}^{d}$ with matching input and output dimensions. A

feature extractor of depth d is given by the composition

$$X \mapsto \mathbf{z}^{(d:1)}(X) := (\mathbf{z}^{(d)} \circ ... \circ \mathbf{z}^{(1)})(X) \in \mathbb{R}^{q_d}.$$

The input dimension of the 1st FNN layer $\mathbf{z}^{(1)}$ is the dimension of the covariates X, i.e., $q_0 = q$.

Because feature extractors should be able to learn non-linear structure, non-linear activation functions ϕ are needed. Commonly used examples are:

Name	Activation function ϕ	Derivative ϕ'
Sigmoid (logistic)	$\phi(x) = \sigma(x) = (1 + e^{-x})^{-1}$	$\phi(1 - \phi)$
Hyperbolic tangent (tanh)	$\phi(x) = \tanh(x) = 2\sigma(2x) - 1$	$1 - \phi^2$
Rectified linear unit (ReLU)	$\phi(x) = x\mathbf{1}_{\{x \geq 0\}}$	$\mathbf{1}_{\{x>0\}}$, $x \neq 0$

The final step of the FNN architecture is to use the learned representation $\mathbf{z}^{(d:1)}(X)$ of the covariates X as input to a GLM. This gives a FNN architecture of depth d

$$X \mapsto \mu(X) = \mathbb{E}[Y \mid X] = g^{-1}\left\langle \mathbf{w}^{(d+1)}, \mathbf{z}^{(d:1)}(X) \right\rangle,$$

with readout parameter $\mathbf{w}^{(d+1)} \in \mathbb{R}^{q_{d+1}}$, including a bias term $w_0^{(d+1)}$, and g^{-1} is the chosen inverse link function.

Compared to the GLM, the difference is that we replace the original covariates $X \in \mathbb{R}^{q+1}$ by the newly learned representation $\mathbf{z}^{(d:1)}(X) \in \mathbb{R}^{q_{d+1}}$ from the feature extractor.

Gradient Descent

FNN fitting relies on stochastic gradient descent (SGD) to minimize the empirical risk defined by the loss function. Choose a strictly consistent scoring function L for mean estimation, such as the deviance loss or squared error. Given a learning sample $\mathcal{L} = \{(Y_i, \mathbf{X}_i, v_i)\}_{i=1}^n$, the empirical loss for network parameters $\vartheta \in \mathbb{R}^r$ is

$$L(\vartheta; \mathcal{L}) = \frac{1}{n} \sum_{i=1}^{n} \frac{v_i}{\varphi} L(Y_i, \mu_\vartheta(\mathbf{X}_i)),$$

where $\mu_\vartheta(\mathbf{X}_i)$ denotes the FNN prediction, v_i are exposure weights, and φ is a fixed dispersion parameter (if applicable).

The gradient descent algorithm iteratively updates ϑ to reduce this loss. Starting from a random initialization $\vartheta^{[0]}$, the basic update rule at iteration t is

$$\vartheta^{[t+1]} = \vartheta^{[t]} - \varrho_{t+1} \nabla_\vartheta L(\vartheta^{[t]}; \mathcal{L}),$$

where $\varrho_{t+1} > 0$ is the learning rate, chosen small to ensure stability but large enough for efficient convergence. The gradient $\nabla_\vartheta L$ is computed via backpropagation, which efficiently applies the chain rule layer-by-layer to propagate errors backward from the output.

To incorporate approximate second-order information and accelerate convergence, momentum-based methods are used. The update becomes

$$\mathbf{v}^{[t+1]} = \nu \mathbf{v}^{[t]} - \varrho_{t+1} \nabla_\vartheta L(\vartheta^{[t]}; \mathcal{L}), \quad \vartheta^{[t+1]} = \vartheta^{[t]} + \mathbf{v}^{[t+1]},$$

with momentum parameter $\nu \in (0, 1)$ (typically 0.9) and $\mathbf{v}^{[0]} = \mathbf{0}$. Popular adaptive optimizers include RMSprop (which normalizes gradients by root-mean-square estimates), Adam (combining momentum with adaptive per-parameter learning rates), Nadam (Adam with Nesterov acceleration to lookahead).

To prevent overfitting, split $\mathcal{L}$ randomly into a training set $\mathcal{U}$ (e.g., 80-90%) for computing gradients and a validation set $\mathcal{V}$ for monitoring generalization. Perform updates using $\nabla_\vartheta L(\vartheta^{[t]}; \mathcal{U})$, and track the validation loss $L(\vartheta^{[t]}; \mathcal{V})$. Implement early stopping: halt at iteration t^* when the validation loss begins to increase (indicating transition from learning signal to noise), and revert to the weights minimizing $L(\cdot; \mathcal{V})$ via callbacks.

For scalability with large n, employ mini-batch SGD: partition $\mathcal{U}$ into mini-batches $\{\mathcal{U}_k\}_{k=1}^B$ of size s (e.g., 32-512), and cycle through them:

$$\vartheta^{[t+1]} = \vartheta^{[t]} - \varrho_{t+1} \nabla_\vartheta L(\vartheta^{[t]}; \mathcal{U}_k).$$

One full cycle (epoch) approximates the full gradient; smaller s introduces noise that aids escaping local minima but may slow convergence.

Covariate Preprocessing

It is important for the gradient descent algorithm to work properly that all covariate components live on the same scale. Otherwise some covariate components will dominate the gradient, and the algorithm may not be able to adapt to the right scales for all covariate components. For this reason, continuous covariates should be standardized or min-max scaled.

For categorical covariates one usually starts with one-hot encoding. In FNNs, a full rank design matrix is not an essential property because, in general, identifiability and parsimony is not given in FNNs. High-cardinality categorical covariates lead to large input dimensions q_0 to the feature extractor. This is generally problematic in network fitting as it gives a high potential for over-fitting. A general recommendation in such situations is to use an entity embedding with a low-dimensional embedding dimension b. The entity embedded variables are concatenated with the continuous ones, to jointly enter the feature extractor of the FNN architecture. Entity embedding adds $b \cdot K$ embedding weights to the network parameter ϑ, if K is the number of levels of the categorical covariate. These embedding weights are also learned during gradient descent training.

Appendix C: Additional figures

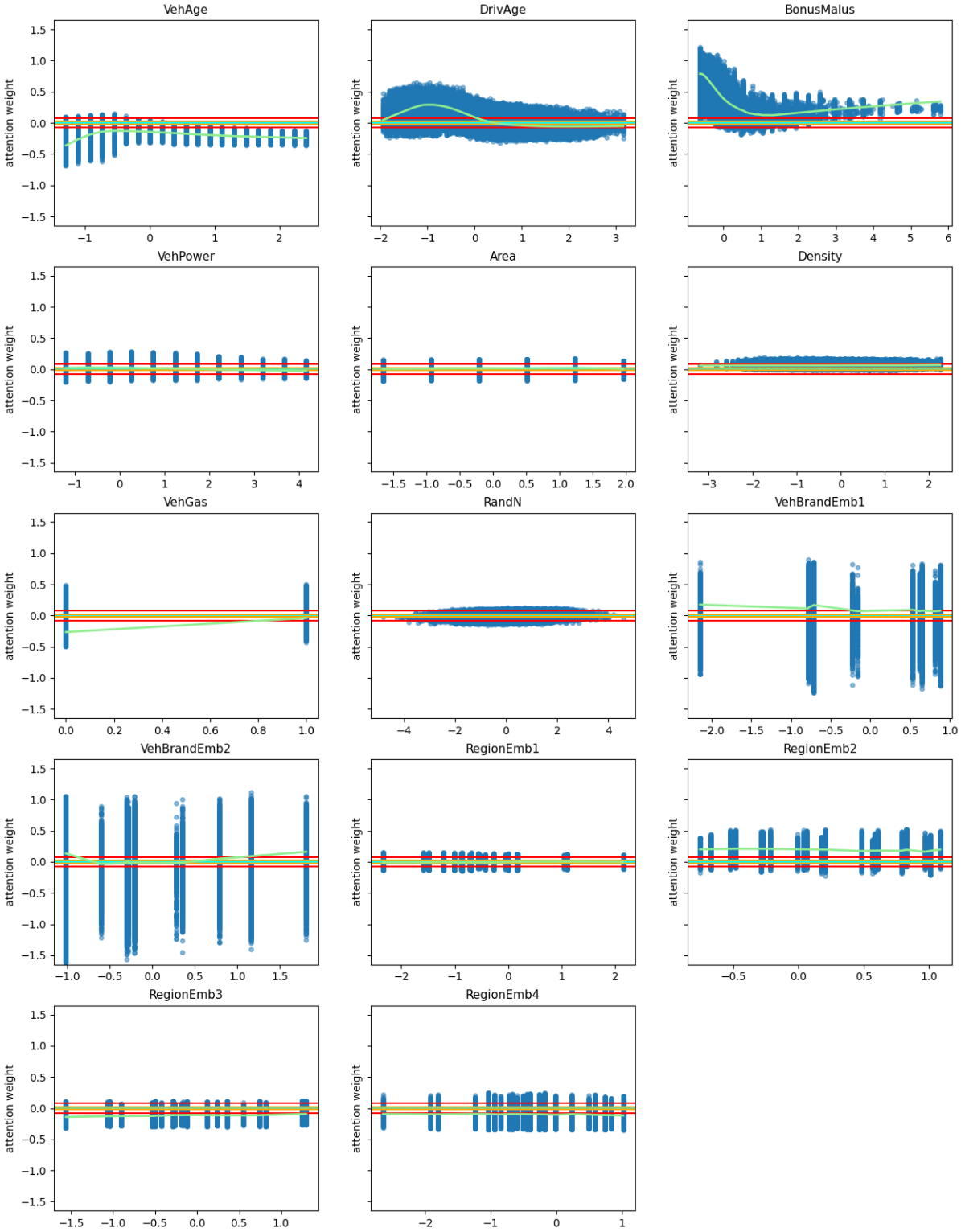

Figure 12: Scatter plot of attention weight of LocalGLMNet_emb

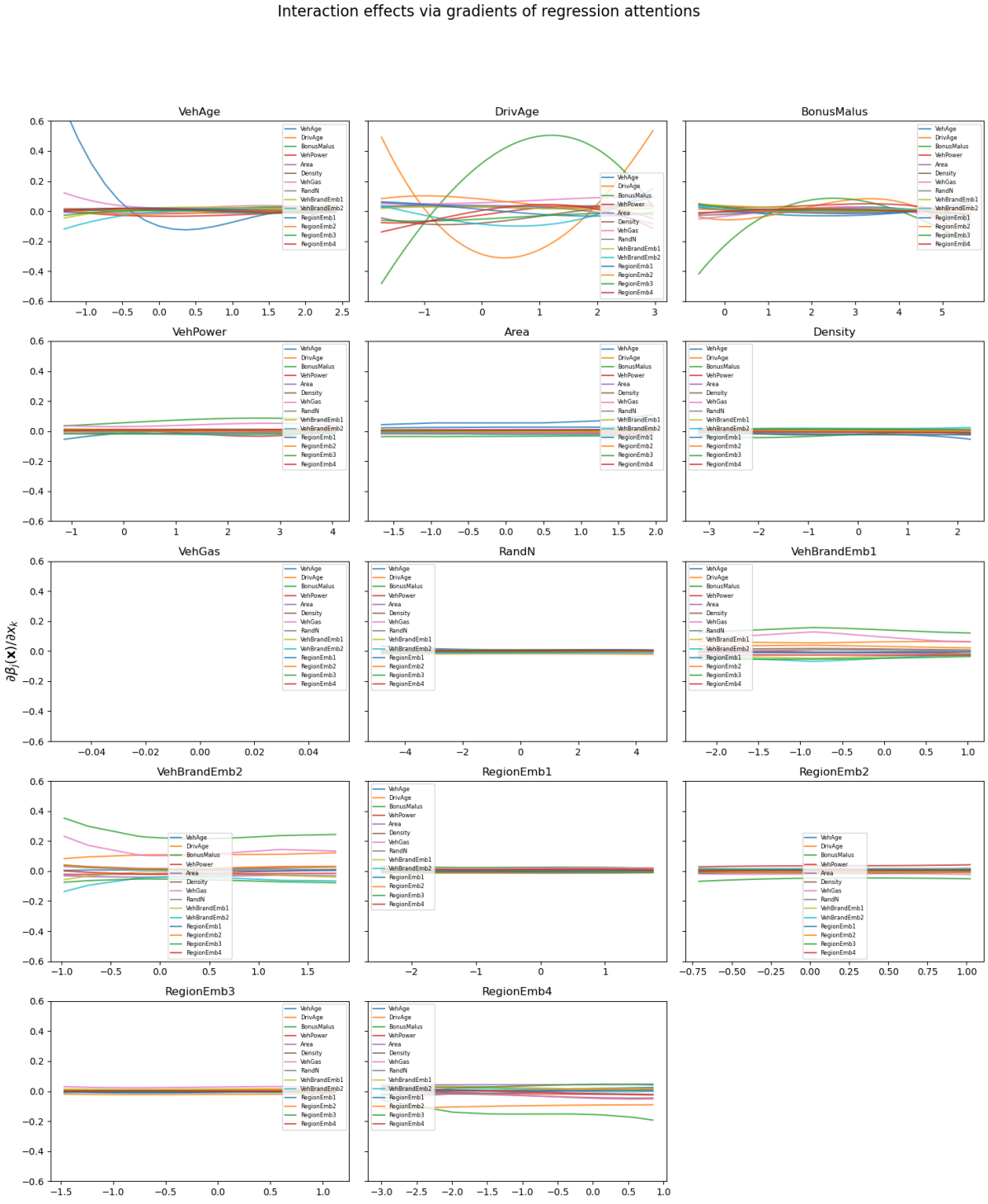

Figure 13: Full interaction of LocalGLMNet_emb with categorical embeddings