Flexible Copula-Based Mixed Models in Deep Learning: A Scalable Approach to Arbitrary Marginals

Giora Simchoni
Department of Statistics
Tel Aviv University

Saharon Rosset
Department of Statistics
Tel Aviv University

Abstract

We introduce copula-based neural networks (COPNN), a novel framework that extends beyond the limitations of Gaussian marginals for random effects in mixed models. COPNN integrates the flexibility of Gaussian copulas in capturing rich dependence structures with arbitrary marginal distributions, with the expressive power of deep neural networks (DNN), allowing it to model large non-Gaussian data in both regression and classification settings, while using batch learning and stochastic gradient descent. Unlike traditional linear and non-linear mixed models, which assume Gaussianity for random effects, COPNN leverages copulas to decouple the marginal distribution from the dependence structure, caused by spatial, temporal and high-cardinality categorical features. This is achieved by minimizing a batch negative log-likelihood (NLL) loss in the continuous case, and a batch negative pairwise log-likelihood in the binary case. We demonstrate COPNN's effectiveness through extensive experiments on both simulated and real datasets. COPNN reduces NLL and MSE in the regression setting, and improves predictive accuracy in the classification setting, compared to previous state of the art methods which integrate random effects into DNN. Our real-world experiments, conducted on datasets from automotive pricing and retail traffic forecasting, further validate COPNN's ability to improve performance over traditional methods for dealing with high-cardinality categorical features.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

1 INTRODUCTION

Many real-world datasets involve dependent observations, where the data often exhibit complex dependencies that necessitate careful modeling beyond simple regression techniques. Consider a car insurance claims dataset (see e.g. Joseph Ferreira and Minikel, 2012). Each row in the dataset represents an individual insurance claim, which includes information about the car, the owner, and the circumstances of the claim. However, multiple levels of dependency naturally arise. For instance, a single car owner may file multiple claims over time, possibly for different cars. Claims are correlated by owner, reflecting consistent driving behaviors, risk profiles, or claims tendencies. Similarly, a single car might be involved in multiple claims, creating another layer of dependency, where the mechanical condition or geographic usage of the vehicle could influence the likelihood of repeated claims. In this context, linear mixed models (LMM) are essential to capture both the random effects associated with categorical features like car owners and individual cars, as well as to account for the temporal and spatial structure in the data. Furthermore, the distribution of claim sizes may not follow a simple Gaussian pattern, especially when there are extreme outliers or heavy-tailed distributions due to large claims (Frees and Valdez, 2008).

Deep neural networks (DNN), with their powerful representation learning capabilities, provide a flexible framework to model large complex data with mechanisms such as batch learning and stochastic gradient descent (SGD). However, integrating mixed models into DNN requires addressing the issue that the natural negative log-likelihood (NLL) loss for these models or its gradient do not readily decompose to sums over observations (Simchoni and Rosset, 2023). In this paper, we extend the application of DNN to mixed models with arbitrary continuous marginal distributions by leveraging copulas, a powerful tool for modeling dependencies between variables. In addition, we also extend our approach to classification problems by integrating approximation tools from generalized linear mixed models (GLMM) to DNN, combining pairwise likelihood estimation using a Gaussian copula on a probit model. This approach allows for scalable inference while accommodating both the random effects and non-trivial marginal distributions, which are common in many real-world scenarios.

The remainder of this paper is organized as follows. In Section 2, we review the relevant literature on copulas, GLMM, and prior attempts to incorporate GLMM into DNN. We discuss the limitations of these approaches and how they motivate our work. In Section 3 we introduce our novel framework for incorporating copula-based mixed models into DNN, enabling the use of diverse marginal distributions while maintaining scalability for large datasets. Section 5 presents experimental results from both simulated and real datasets, demonstrating the superior performance of our method compared to traditional approaches. Finally, in Section 6, we conclude with a discussion of our findings and suggest directions for future research.

2 BACKGROUND

Let $X_{n\times p}$ and $Z_{n\times q}$ represent data matrices where n observations are measured across p fixed-effects (FE) features and q random-effects (RE) features, respectively. In a GLMM, we aim to relate a response vector y to the explanatory variables in X and Z. Here, X contains global features that do not induce correlation between observations, while Z represents features like time or location that introduce dependency between observations. The GLMM links the conditional expectation of y to a linear combination of X and Z through a link function η, as follows:

$$\eta[E(y|b)] = X\beta + Zb \tag{1}$$

where β is a p-dimensional vector of FE coefficients, and b is a q-dimensional vector of RE variables. These RE b are typically modeled as normally distributed with mean zero and covariance matrix $D(\psi)$, such that $b \sim \mathcal{N}(0, D(\psi))$. The matrix D, a positive semidefinite $q \times q$ matrix, contains variance components ψ which are parameters that describe the structure of the RE's covariance. The form of D can vary, from a radial basis function (RBF) kernel for spatial locations to a diagonal matrix for the random intercepts model for a single categorical feature, depending on the scenario, see more details on common covariance scenarios in Appendix A. The key strength of GLMM lies in their ability to model this covariance structure, thus providing more accurate parameter estimates and improving prediction for new data, in contrast to simpler models like ordinary linear regression (Searle et al., 1992).

A natural extension of GLMM is to relax the linearity assumption in (1) and instead relate $\eta[E(y|b)]$ to X and Z through two non-linear functions, g_f and g_r, both of which can be effectively modeled by DNN:

$$\eta[E(y|b)] = g_f(X) + g_r(Z)\,b. \tag{2}$$

The corresponding marginal NLL loss, which needs to be minimized to estimate the parameters g_f, g_r, ψ is given by:

$$NLL(g_f, g_r, \psi|y) = -\log \prod_i \int f_{Y|b}(y_i|b) f_b(b|\psi)\, db. \tag{3}$$

This expression generally does not have a closed form, except in the special case where the response distribution $f_{Y|b}$ is Gaussian, i.e., $\mathcal{N}(E(y|b), \sigma^2 I_n)$, and η is the identity function:

$$\begin{aligned} &NLL(g_f, g_r, \psi|y) = \\ &\frac{1}{2}\left(y - g_f(X)\right)^T V(g_r,\psi)^{-1}\left(y - g_f(X)\right) \\ &+ \frac{1}{2}\log|V(g_r,\psi)| + \frac{n}{2}\log 2\pi, \end{aligned} \tag{4}$$

where $V(g_r, \psi) = g_r(Z)D(\psi)g_r(Z)^T + \sigma^2 I_n$ is the marginal covariance matrix of y, and we consider σ^2 part of ψ. Consequently, even in the Gaussian case, evaluating (4) involves inverting V, which is of size $n \times n$. For moderate-sized datasets, this matrix inversion becomes computationally challenging. Crucially, this loss does not easily decompose into smaller, independent components, making it incompatible with typical batch-based learning methods used in DNN (McCulloch et al., 2008).

Several recent attempts have been made to integrate GLMM into DNN. MeNets (Xiong et al., 2019) focus on the Gaussian case for the random intercepts model. DeepGLMM (Tran et al., 2020) proposes a model similar to (2), but relies on a variational approximation involving multiple components that are difficult to integrate into any given DNN architecture. Methods like sgGP (Chen et al., 2020) and deep kernel learning (Wilson et al., 2016a,b) scale Gaussian processes for the Gaussian spatial case. LMMNN (Simchoni and Rosset, 2023), which bears the closest resemblance to our work, incorporates RE into DNN to handle varied covariance structures for very large datasets. The authors utilize the natural NLL loss, demonstrating how it can be decomposed for batch learning. However, as its name suggests, LMMNN is restricted to the LMM framework and focuses primarily on continuous y with Gaussian error terms, with only a limited extension to binary y for the random intercepts model. In contrast, our approach generalizes this framework to accommodate both discrete and continuous y with a variety of marginal error distributions.

The aforementioned approaches offer insights into integrating mixed models into DNN but are limited by their reliance on Gaussian assumptions, which may impair estimation on real-world datasets (Sun et al., 2008). Copulas address this limitation by modeling the joint distribution of random variables, separating the marginal distributions from the dependency structure. Starting with a continuous response vector $y = (y_1, \dots, y_n)$, let F denote the n-dimensional joint distribution of y, which has continuous marginal distributions $F_1, \dots, F_n$. According to Sklar's theorem (Sklar, 1959), any such distribution F can be decomposed into its marginal distributions with a copula C, which captures the dependence structure. Explicitly, we have:

$$F(y_1, \dots, y_n) = C[F_1(y_1), \dots, F_n(y_n)]. \tag{5}$$

This allows us to express the likelihood of the data as:

$$f(y_1, \dots, y_n) = c[F_1(y_1), \dots, F_n(y_n)] \prod_{i=1}^{n} f_i(y_i), \tag{6}$$

where c is the copula density and f_i are the marginal densities. In this paper, we utilize the Gaussian copula with a correlation matrix $\Omega(\rho)$ holding correlation terms ρ, and we assume the marginal distributions F_θ and their corresponding densities f_θ depend on parameters θ, such as mean and variance. Taking the negative log of this likelihood, we obtain the NLL:

$$\begin{aligned} NLL(\rho, \theta|y) &= \frac{1}{2} h_\theta(y)^T (\Omega(\rho)^{-1} - I_n) h_\theta(y) \\ &+ \frac{1}{2} \log|\Omega(\rho)| + \sum_{i=1}^{n} \log\left(f_\theta(y_i)\right), \end{aligned} \tag{7}$$

where $h_\theta(z) = \Phi^{-1}(F_\theta(z))$, Φ is the standard normal cumulative distribution function, and F_θ, Φ are applied elementwise. A key observation is that if the marginal distribution F_θ is Gaussian and the correlation matrix $\Omega(\rho) = S^{-1}V(g_r, \psi)S^{-1}$, where $S = S(g_r, \psi)$ is a diagonal matrix containing the square root of the diagonal entries of the covariance matrix $V(g_r, \psi)$, then we can express Ω as $\Omega(g_r, \psi)$, and this formulation reduces to the NLL in (4), used in models like LMMNN (see e.g. Killiches and Czado, 2018).

For example, for the random intercepts LMM with a single categorical feature with q levels, with g_r being the identity function, we have: $y = g_f(X) + Zb + \varepsilon$, where $b \sim \mathcal{N}(0, D(\psi))$ with $D(\psi) = \sigma_b^2 I_q$, and $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$, making $V(g_r, \psi) = \sigma_b^2 ZZ^T + \sigma^2 I_n$. That is, V has $\sigma_b^2 + \sigma^2$ on its diagonal and σ_b^2 as the covariance for two observations of the same level. Let $\Omega(\rho) = V(g_r, \psi)/(\sigma_b^2 + \sigma^2)$, so that it has 1s on its diagonal and correlation $\rho = \sigma_b^2/(\sigma_b^2 + \sigma^2)$ for two observations of the same level. If each F_θ is Gaussian with θ being expectation $g_f(x_i)$ and variance $\sigma_b^2 + \sigma^2$, then we extract the NLL of random intercepts LMM exactly. If we merely require F_θ to have expectation $g_f(X_i)$ and variance $\sigma_b^2 + \sigma^2$, we generalize the LMM framework to handle more marginal error distributions beyond Gaussian, providing greater versatility.

When the marginal distributions F_θ are discrete, we must apply the Radon-Nikodym derivative to (5) in order to derive the likelihood for y:

$$\begin{aligned} f(y_1, \dots, y_n) &= P(Y_1 = y_1, \dots, Y_n = y_n) \\ &= \sum_{j_1=1}^{2} \cdots \sum_{j_n=1}^{2} (-1)^{j_1 + \cdots + j_n} C[u_{1j_1}, \dots, u_{nj_n}], \end{aligned} \tag{8}$$

where $u_{ij_i} = F_i(y_i)$ when $j_i = 1$, and $F_i(y_i - 1)$ when $j_i = 2$ (and $F(u) = 0$ for $u < 0$) (Song, 2007). This formulation quickly becomes computationally intensive for repeated evaluations, even for relatively small n. In the next section, we introduce our method, combining a Gaussian copula with pairwise likelihood (Varin et al., 2011) to efficiently compute a loss function for discrete data, enabling batch learning and scalability.

3 COPNN: COPULA MIXED MODELS IN DNN

We now present our approach for integrating mixed models into DNN using copulas, which we refer to as COPNN. Our discussion is divided into two cases: continuous response variables and discrete response variables, each addressed with tailored methods for efficient learning.

3.1 Continuous Response Case

In the previous section, we derived the copula-based NLL loss (7) for the continuous case, which can be broken down into two components: the second term, involving the marginal terms $\log\left(f_\theta(y_i)\right)$, is easily decomposable for batch learning, while the first term, involving the copula's correlation matrix Ω, is not easily separable across observations. Nevertheless, we now write the NLL in (7) for a given mini-batch to be used in combination with SGD in DNN. Let ξ be a mini-batch (X_ξ, Z_ξ, y_ξ) of size m, where typically $m \ll n$, and mark $\Omega_\xi = \Omega(g_r, \psi)_\xi$ – the submatrix which is the correlation version of the submatrix $V(g_r, \psi)_\xi = g_r(Z_\xi) D(\psi) g_r(Z_\xi)^T + \sigma^2 I_m$:

$$\begin{aligned} NLL(g_f, g_r, \psi, \theta|y_\xi) &= \frac{1}{2} h_\theta(y_\xi)^T (\Omega_\xi^{-1} - I_m) h_\theta(y_\xi) \\ &+ \frac{1}{2} \log|\Omega_\xi| + \sum_{i=1}^{m} \log\left(f_\theta(y_{\xi,i})\right). \end{aligned} \tag{9}$$

We note that for each mini-batch ξ the inverse and log-determinant of Ω_ξ are computed on an $m \times m$ matrix, and that even this inverse needs not be explicitly calculated if we solve a linear system of equations $\Omega_\xi x = h_\theta(y_\xi)$ to get $\Omega_\xi^{-1} h_\theta(y_\xi)$ directly.

This "inversion by parts" approach is critical to make learning (and in particular SGD) practical in this setting, but it also has some theoretical grounding, as previously discussed in the context of LMMNN (Simchoni and Rosset, 2023). Firstly, when $V(g_r, \psi)$, and consequently $\Omega(g_r, \psi)$, is block-diagonal with q blocks of size $n_j \times n_j$ ($j = 1, \ldots, q$), its inverse simplifies to a block-matrix with inverses $V(g_r, \psi)_j^{-1}$ as its blocks. As a result, the loss in (7) and its gradient decompose naturally into a sum over the blocks. This occurs when g_r is the identity function and we use standard covariance structures in $D(\psi)$, such as random intercepts or longitudinal models. Even when $V(g_r, \psi)$ is not block-diagonal, Chen et al. (2020) demonstrated that as long as $V(g_r, \psi)$ exhibits fast eigendecay (e.g., exponential or polynomial), the full gradient of (4), and by extension (7), with respect to ψ converges to zero when using batch learning and SGD. This holds true for more complex covariance structures in $D(\psi)$, such as spatial models or those with multiple categorical features.

With the network loss in (9) defined, we can replace F_θ with a variety of continuous distributions, including those well-suited for modeling non-symmetric or heavy-tailed errors. A few examples of such distributions are presented in Appendix C. Imperatively, this loss function can seamlessly integrate into any appropriate DNN architecture for regression, allowing us to effectively model g_f and g_r. Figure 1 illustrates the architecture of COPNN using a simple multi-layer perceptron (MLP) as an example.

Testing In a standard learning setup, the dataset (X, Z, y) is split into train and test sets. After fitting the model on (X_{tr}, Z_{tr}, y_{tr}), we assess its performance on the test set (X_{te}, Z_{te}, y_{te}) by predicting:

$$\hat{y}_{te} = \hat{g}_f\left(X_{te}\right) + \hat{g}_r\left(Z_{te}\right)\hat{b}, \tag{10}$$

where $\hat{g}_f$ and $\hat{g}_r$ are the outputs of the DNN modeling g_f and g_r, respectively, and $\hat{b}$ represents the predicted RE. If the marginal distribution is Gaussian, making the model equivalent to LMMNN, $b|y_{tr}$ follows a normal distribution. Thus, we can estimate $\hat{b}$ using the best unbiased linear predictor (BLUP): $\hat{b} = D(\hat{\psi})\hat{g}_r\left(Z_{tr}\right)^T V(\hat{g}_r, \hat{\psi})^{-1}\left(y_{tr} - \hat{g}_f\left(X_{tr}\right)\right)$. For non-Gaussian distributions, the conditional distribution of $b|y_{tr}$ does not have a closed form. However, the distribution of $h_\theta(y_{te})\big|h_\theta(y_{tr})$ is known to be $\mathcal{N}\left(\mu_{te}, \Sigma_{te}\right)$, where $\mu_{te} = A^T\Omega_{tr}^{-1}h_\theta(y_{tr})$, $\Sigma_{te} = \Omega_{te} - A^T\Omega_{tr}^{-1}A$.

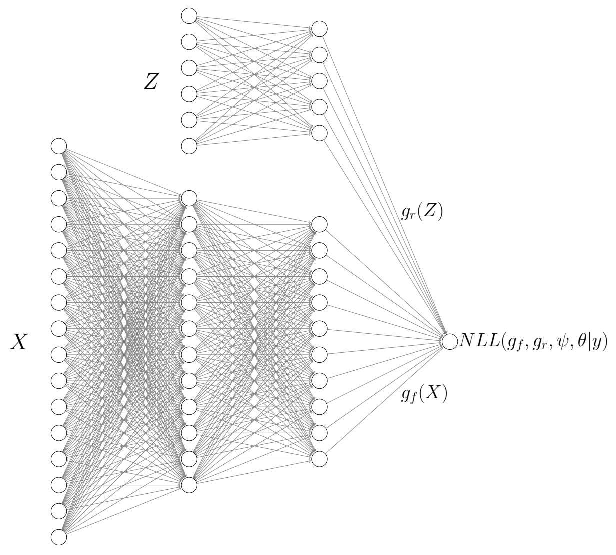

Figure 1: Schematic description of COPNN using a simple MLP for fitting g_f and g_r, and combining outputs with the NLL loss layer, in a single-stage training.

Here, $A = g_r\left(Z_{tr}\right)D^*(\psi)g_r\left(Z_{te}\right)^T$ is the correlation matrix between $h_\theta(y_{te})$ and $h_\theta(y_{tr})$, with $D^*(\psi)$ being the correlation version of $D(\psi)$.

One option for prediction is to use the estimated mean $\hat{\mu}_{te}$ and apply the inverse of h_θ to obtain the final estimate:

$$\hat{y}_{te} = F_{\hat{\theta}}^{-1}\left[\Phi(\hat{\mu}_{te})\right]. \tag{11}$$

Here, $\hat{\theta}$ is the required expectation for the test set $\hat{g}_f(X_{te})$ and the variance on the diagonal of $V(\hat{g}_r, \hat{\psi})_{te} = \hat{g}_r(Z_{te})D(\hat{\psi})\hat{g}_r(Z_{te})^T + \hat{\sigma}^2 I$. This gives the conditional median of $y_{te}|y_{tr}$, however, for better performance, we aim to estimate the conditional mean when feasible. We sample $v_1, \ldots, v_B$ vectors from $\mathcal{N}(\hat{\mu}_{te}, \hat{\Sigma}_{te})$, apply the inverse of h_θ, and compute their average:

$$\hat{y}_{te} = \frac{1}{B}\sum_l F_{\hat{\theta}}^{-1}\left[\Phi(v_l)\right]. \tag{12}$$

We emphasize that this step occurs only once, at the conclusion of training.

3.2 Discrete Response Case

To address the challenge in calculating the full likelihood in (8), we adopt a pairwise likelihood (PL) approach, which simplifies the computational problem by approximating the n-dimensional integral through a collection of 2-dimensional integrals on pairs of observations. Specifically, define the PL estimator $\hat{\zeta}_{PL}$ as minimizing the negative log pairwise likelihood

(NLPL):

$$NLPL(\zeta|y) = -\sum_{i<j} w_{ij} \log P(Y_i = y_i, Y_j = y_j), \quad (13)$$

where w_{ij} is some weight for pair y_i, y_j. PL works well in this context because its score functions $U(\zeta) = \partial NLPL/\partial\zeta$ are unbiased under certain regularity conditions, that is $E(U(\zeta)) = 0$ when $\zeta = \zeta_{true}$, ensuring that estimates based on pairs remain reliable (Varin, 2008). This approach is particularly suited for our model with a binary response $y_i \in \{0, 1\}$, since $P(Y_i = y_i, Y_j = y_j)$ under copula C are simple to compute (Song, 2007). Let $\pi_i = P(Y_i = 1)$, then $P(Y_i = y_i, Y_j = y_j)$ equals:

$$\begin{cases} C(1-\pi_i, 1-\pi_j), & \text{if } y_i = 0, y_j = 0 \\ 1-\pi_i - C(1-\pi_i, 1-\pi_j), & \text{if } y_i = 0, y_j = 1 \\ 1-\pi_j - C(1-\pi_i, 1-\pi_j), & \text{if } y_i = 1, y_j = 0 \\ \pi_i + \pi_j + C(1-\pi_i, 1-\pi_j) - 1, & \text{if } y_i = 1, y_j = 1 \end{cases} \quad (14)$$

As in (2) we define the canonical link function η to model π with data X, Z: $\eta(\pi_i) = g_f(X_i) + g_r(Z_i)\, b$. In this paper, we use the probit link function Φ^{-1}, which, when combined with a Gaussian copula, results in $C(1-\pi_i, 1-\pi_j) = \Phi_2(-g_f(X_i), -g_f(X_j); \rho)$. Here Φ_2 is the CDF of the bivariate normal distribution with standard normal marginals and correlation coefficient ρ, and ρ arises from the structure of $\Omega(g_r, \psi)$. This allows us to express the NLPL in (13) as $NLPL(g_f, g_r, \psi)$.

Moreover, the PL loss naturally decomposes into batches. For each batch ξ, we accumulate $NLPL(g_f, g_r, \psi)_\xi$ over correlated pairs within the batch, maintaining computational efficiency and ensuring COPNN's scalability for large datasets. In practice, we observed that even subsampling pairs within a batch performs effectively. As for pairwise weights w_{ij}, we consistently set $w_{ij} = 1$, though alternative weighting schemes might prove to be more efficient for certain applications (Joe and Lee, 2009).

Testing As in the continuous case, estimating $E(y_{te}|y_{tr})$ presents significant challenges. To address this, we follow the MCMC approach outlined by Varin et al. (2005), leveraging the Metropolis-Hastings algorithm introduced by Zhang (2004) for general covariance structures $D(\psi)$. The detailed description of this procedure is deferred to Appendix B.

4 RELATED WORK ON DEEP COPULAS

Before moving to experimental results, we position our work within prior research combining DNN and copulas. Rank-likelihood methods (Hoff, 2007; Feldman and Kowal, 2022, 2024) estimate copula dependence structures without specifying marginal distributions, treating them as nuisance parameters. In contrast, our framework prioritizes accurate prediction of y given X, Z, leveraging copulas within a supervised learning setting where the marginal distribution of y is central. Furthermore, our dependence structure aligns with classical GLMM settings, where repeated measurements of y within a cluster share a single marginal distribution, eliminating the need to model heterogeneous marginals across variables, a key aspect of rank-likelihood-based copula estimation.

Deep copula models, on the other hand, use DNN to parameterize copula functions directly, often in an unsupervised setting focused on modeling dependence between variables. While these approaches are valuable for capturing flexible dependency structures, they differ from our work in both scale and objective. For instance, deep Archimedean copulas (Ling et al., 2020) have been applied to a small number of variables, whereas our approach scales to tens of thousands of variables, such as levels of a grouping variable in mixed models. Additionally, implicit generative copulas (Janke et al., 2021) prioritize minimizing divergence between learned and true copula distributions, whereas we optimize a supervised loss to improve predictive accuracy.

By positioning our work within predictive modeling for GLMM, we complement rather than compete with these related approaches. Incorporating rank-likelihood methods or deep copulas into our framework could be an interesting direction for future research, particularly in extending copula-based inference to more flexible nonparametric settings.

5 EXPERIMENTS

We now present a series of experiments, summarizing the key results here, with additional details available in the Appendix. In Section 5.1, we primarily assess COPNN's ability to estimate the copula's ρ parameter across different covariance structures and marginal distributions, and compare its performance with LMMNN, using metrics such as NLL, MSE for continuous outcomes, and AUC for classification. In Section 5.2 we compare COPNN's performance against other methods on several real-world datasets from various applications. All experiments were implemented in Python using Keras on TensorFlow, run on Google Colab with NVIDIA Tesla V100 GPUs, and are available at `https://github.com/gsimchoni/copnn`.

5.1 Simulated Data

We start with the continuous case in model (2), where η is the identity link function, and the data is generated by:

(1) Simulating FE: $g_f(X)$

(2) Simulating RE: $g_r(Z)b$ and ε

(3) Inverse transformation to reach final y

Simulating FE X is sampled from $\mathcal{U}(-1,1)$, with $n = 10000$ rows on $p = 10$ features. g_f is given by:

$$g_f(X) = (X_1 + \cdots + X_{10}) \cdot \cos(X_1 + \cdots + X_{10}) + 2 \cdot X_1 \cdot X_2. \tag{15}$$

Simulating RE ε is a random $\mathcal{N}(0, \sigma^2 I_n)$ noise, and we fix $\sigma^2 = 1$ always. As for the $g_r(Z)b$ element, we take the identity function for g_r, that is $g_r(Z)b = Zb$. We simulate b from $\mathcal{N}(0, D(\psi))$, where we use the three covariance structures from Appendix A for $D(\psi)$ and accordingly for Z. The Categorical structure is a random intercepts model, that is a single categorical feature with $q = 100$ levels each with diverse n_j observations, with Z being a binary $n \times q$ matrix and $D(\psi) = \sigma_b^2 I_q$. Here there is a single ρ parameter in the Gaussian copula $\Omega(\rho)$, where $\rho = \sigma_b^2/(\sigma_b^2 + \sigma^2)$ represents the within-level correlation. We vary σ_b^2 to make $\rho \in \{0.1, 0.5, 0.9\}$. The Longitudinal structure is a random slope model, that is repeated measures in time t for $q = 100$ subjects, with variance components $\sigma_{b0}^2, \sigma_{b1}^2$ for an intercept and a slope, which are varied to make correlation ρ at time $t = 1$ be in $\{0.1, 0.5, 0.9\}$. See Appendix A for details on Z and $D(\psi)$ for this model. The Spatial structure is a kriging model, that is repeated measures for $q = 100$ locations, correlated with an RBF kernel $D_{jk}(\psi) = \sigma_{b0}^2 \exp\left(-\frac{|s_j - s_k|^2}{2\sigma_{b1}^2}\right)$ for a pair of longitude-latitude s_j, s_k locations. Here within-location correlation $\rho = \sigma_{b0}^2/(\sigma_{b0}^2 + \sigma^2)$, and we vary σ_{b0}^2 to make $\rho \in \{0.1, 0.5, 0.9\}$ while fixing $\sigma_{b1}^2 = 1$.

Inverse transformation In order to make y distribute according to marginal distribution F_θ where θ is the expectation $g_f(X)$ and variance on the diagonal of $V(g_r, \psi)$ (specified for each covariance structure in Appendix A), we calculate:

$$M_F = S \cdot F_0^{-1}\left[\Phi\left(S^{-1} \cdot (g_r(Z)b + \varepsilon)\right)\right], \tag{16}$$

where F_0 is parameterized to have mean 0 and variance 1, and S is as discussed above. We then sum the expectation and M_F to reach the final response: $y = g_f(X) + M_F$. We use a varied selection of distributions for F_0: Gaussian, where we expect equivalent results to LMMNN, Laplace, Gumbel, Log-Gamma, Logistic, Skew-normal with skewness parameter 1, and a 50%/50% mixture of two Gaussians. See Appendix C for the specifics of every distribution.

Training Details The training procedure for our networks follows a consistent pipeline across all experiments. The data is first split into an 80/20 train-test ratio, where the network is fitted on the training set and tested on the held-out test set. The entire process – data generation and training – is repeated 10 times, where for each run we use batches of 100 observations and train the networks over 200 epochs. The same DNN architecture is employed to model g_f, consisting of four hidden layers with 100, 50, 25, and 12 neurons, respectively, and a single output neuron. Each hidden layer is followed by a 25% dropout, and a ReLU activation function. All variance variables are initialized to 1 before training. We further assume in this set of experiments F_θ is known by COPNN, and conduct a separate set of experiments where it is not, see below.

Results Table 1 summarizes our results for the continuous case, where for each combination of covariance structure, marginal distribution, and ρ, we report the mean estimated ρ and the mean percentage change in NLL and MSE loss of COPNN versus LMMNN (which assumes normality) on the test data. As seen in the table, COPNN consistently achieves acceptable estimates of ρ with a slight negative bias also appearing in previous works (Simchoni and Rosset, 2021), and leads to a significant reduction in NLL compared to LMMNN, particularly for distributions like the Skew-normal and the mixture of two Gaussians, which deviate most from normality. The reduction in NLL does not always directly translate into a similar reduction in MSE, but overall, MSE is significantly reduced in most cases. These results demonstrate the superiority of COPNN in handling non-normal data, especially under more varied marginal distributions. In Appendix D, we further provide comparisons to alternative strategies for handling RE features, such as ignoring dependencies, applying one-hot encoding (OHE), or using entity embeddings. These additional results reinforce the advantages of our method over traditional approaches.

We now turn to a similar group of experiments for the discrete case, focusing on binary outcomes $y \in \{0, 1\}$ and using the probit link function for η. After simulating $g_f(X), g_r(Z)b, \varepsilon$ as described above, we calculate the latent variable $l = g_f(X) + g_r(Z)b + \varepsilon$ and apply a threshold of zero to generate the response: $y_i = \mathbb{I}[l_i \leq 0]$ (see e.g. Song, 2007). As with the continuous case, we set g_r to the identity function and explore different covariance structures for $D(\psi)$. A

Table 1: Mean estimated ρ and percent change in test NLL and MSE with COPNN and true distribution assumed known vs. LMMNN, for $y \in \mathbb{R}$ in simulated datasets with various covariance structures. All NLL reductions are statistically significant except for the Gaussian and Logistic distributions, in a paired t-test.

		$\rho = 0.1$			$\rho = 0.5$			$\rho = 0.9$		
	Dist.	$\hat{\rho}$	NLL	MSE	$\hat{\rho}$	NLL	MSE	$\hat{\rho}$	NLL	MSE
Categorical	Gaussian	0.08	-0.1%	-0.1%	0.45	-0.3%	-1%	0.89	+0.0%	-0.3%
	Laplace	0.08	-4%	-2%	0.43	-3%	-2%	0.88	-3%	-4%
	Gumbel	0.08	-4%	+0.0%	0.43	-3%	-1%	0.87	-4%	-2%
	LogGamma	0.09	-3%	-2%	0.45	-4%	-4%	0.88	-3%	-4%
	Logistic	0.08	-1%	+0.2%	0.47	-1%	-1%	0.89	-1%	-2%
	SkewNorm	0.09	-12%	+0.3%	0.44	-11%	+0.2%	0.89	-8%	+0.4%
	GaussMix	0.07	-64%	-7%	0.44	-61%	-8%	0.81	-51%	-7%
Longitudinal	Gaussian	0.08	+0.3%	+1%	0.45	+0.0%	+0.0%	0.89	+0.0%	-1%
	Laplace	0.08	-4%	-2%	0.49	-4%	-4%	0.87	-4%	-0.5%
	Gumbel	0.07	-4%	+0.5%	0.44	-3%	-2%	0.88	-5%	-0.1%
	LogGamma	0.15	-3%	-2%	0.44	-4%	-4%	0.88	-3%	-2%
	Logistic	0.04	-0.4%	+1%	0.43	-1%	-1%	0.88	-1%	-2%
	SkewNorm	0.05	-13%	+1%	0.44	-12%	-1%	0.89	-10%	+0.1%
	GaussMix	0.13	-63%	-7%	0.45	-59%	-9%	0.84	-56%	-8%
Spatial	Gaussian	0.08	-0.3%	-1%	0.47	+0.3%	+1%	0.87	+0.0%	+0.0%
	Laplace	0.08	-4%	-2%	0.43	-4%	-2%	0.89	-4%	-3%
	Gumbel	0.07	-3%	+1%	0.44	-4%	-2%	0.86	-3%	-2%
	LogGamma	0.08	-4%	-4%	0.43	-4%	-3%	0.89	-3%	-3%
	Logistic	0.08	-1%	+0.1%	0.47	-1%	+0.5%	0.88	-1%	-1%
	SkewNorm	0.08	-13%	-1%	0.45	-11%	-0.3%	0.88	-9%	+0.1%
	GaussMix	0.07	-63%	-7%	0.41	-61%	-8%	0.81	-52%	-7%

direct comparison to LMMNN is challenging here, as LMMNN is limited to handling a single categorical feature and we are employing the NLPL loss instead of NLL. Still, we provide a comparison to LMMNN by having LMMNN assume a single categorical feature with $q = 100$ levels (Categ1) for all covariance scenarios, rather than spatial or longitudinal settings, and reporting the test AUC. While LMMNN is expected to perform well under Categ1, we introduce a more complex covariance structure, Categ3, featuring three uncorrelated categorical variables, each with $q = 100$ levels. The training setup mirrors that of the continuous case, where in each batch of 100 observations we sample 50 correlated pairs to compute the batch NLPL loss. Moreover, for the discrete case we also implement the Metropolis-Hastings algorithm for prediction, as described in Section 3.2, with 1000 iterations and the first 200 iterations discarded.

Table 2 presents the results for the discrete case, examining four different covariance structures and various values of ρ. COPNN demonstrates good estimation of ρ, though with a downward bias in the Spatial and Categ3 cases. As expected, its performance is comparable to LMMNN in the simpler Categ1 scenario and when ρ is small, indicating low dependence in the data. However, for moderate to high correlations in the more complex covariance structures, COPNN consistently improves the AUC by 2-6% over the mis-specified LMMNN, highlighting its effectiveness in handling intricate dependencies. See detailed results in Appendix D.

Table 2: Mean estimated ρ and percent change in test AUC vs. LMMNN with a single categorical feature for $y \in \{0, 1\}$ in simulated datasets with various covariance structures. All increases are statistically significant except for the single categorical feature scenario (Categ1) and when $\rho = 0.1$, with a paired t-test.

Struct.	$\rho = 0.1$	$\rho = 0.5$	$\rho = 0.9$
Categ1	0.10, +0%	0.47, +1%	0.86, +0%
Categ3	0.09, +1%	0.46, +6%	0.80, +5%
Longitud.	0.12, +1%	0.50, +4%	0.84, +3%
Spatial	0.13, +0%	0.47, +3%	0.78, +2%

Additional Experiments We present two additional sets of experiments: (i) Figure 2 shows a report on the mean runtime of COPNN and other methods in the spatial scenario, with sample sizes (n) up to 10^7 and up to 10^4 locations (q); (ii) Table 3, illustrates the change in mean test MSE when mis-specifying the marginal distribution compared to assuming the true

Table 3: Change in mean test MSE when the marginal distribution is mis-specified, versus fitting the true distribution; for $y \in \mathbb{R}$, single categorical feature with $n = 10000; q = 100; \rho = 0.5$.

		Fitted Distribution						
		Gaussian	Laplace	Gumbel	LogGamma	Logistic	SkewNorm	GaussMix
True Distribution	Ga.	–	2%	5%	2%	3%	1%	33%
	La.	1%	–	11%	9%	-2%	6%	97%
	Gu.	7%	5%	–	9%	2%	4%	17%
	LG	2%	-.1%	17%	–	5%	1%	8%
	Log	2%	1%	9%	2%	–	2%	69%
	SN	.5%	.2%	6%	2%	-1%	–	36%
	GM	7%	23%	4%	4%	8%	8%	–

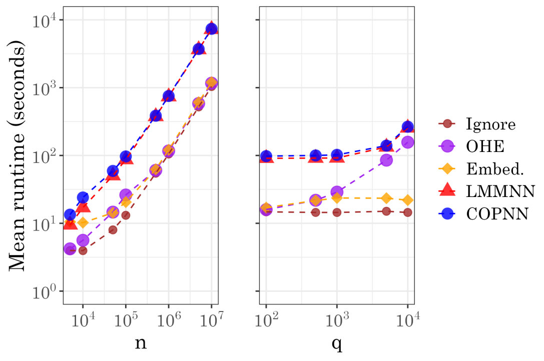

Figure 2: Mean log runtime for $y \in \mathbb{R}$ in the spatial scenario with within-location $\rho = 0.5$. Left: n up to 10^7 (with q fixed at 1000 locations), Right: q up to 10^4 (with n fixed on 10^6).

distribution. The results show that while using the correct marginal distribution improves COPNN's performance, the Gaussian distribution remains quite robust, delivering reasonable MSEs even when the true distribution differs.

5.2 Real Data

In this section, we demonstrate the performance of COPNN on two real large-scale datasets: one continuous and one discrete. The continuous dataset comprises $n = 97,729$ cars and trucks sourced from Craigslist (Reese, 2020), with prices ranging from \$1K to \$300K. The dataset includes $p = 73$ features such as manufacturer, year of make, size, and condition. What makes this dataset particularly interesting for our model is its combination of spatial and categorical dependencies: latitude and longitude represent $q_{sp} = 3,367$ unique locations across the U.S., while the high-cardinality categorical feature "car model" includes $q_{cat} = 15,226$ models, each having between 1 and 632 cars. Ignoring these dependencies or applying default methods like OHE could negatively impact model performance. Additionally, the skewed distribution of prices, even after a log transformation, suggests that using a Gaussian marginal (as in LMMNN) may not be ideal.

The second dataset is a longitudinal dataset with a binary outcome: the Rossmann stores dataset (FlorianKnauer, 2015), which contains monthly customer data from $q = 1,115$ stores across Europe over 30-31 months, for a total of $n = 33,485$ rows. There are $p = 23$ features, including the number of holiday and promotional days. Our objective here is to predict whether a store will experience an extreme month, with customer totals in the 90th percentile or higher, a situation where the store may need to hire more staff. We explore two interesting scenarios for the retail chain: *Random Mode*, where customer data is missing for random months, simulating a data imputation task, and *Future Mode*, where we predict customer trends for the last six months of the time series.

Beginning with the Cars dataset, for both LMMNN and COPNN, we incorporate spatial and categorical dependencies into a single block-diagonal covariance matrix, $D(\psi)$. This matrix consists of two blocks: a $q_{sp} \times q_{sp}$ RBF kernel $\sigma^2_{b1} \exp\left(-\frac{|s_j - s_k|^2}{2\sigma^2_{b2}}\right)$ representing spatial dependencies, and a $q_{cat} \times q_{cat}$ diagonal matrix $\sigma^2_{b3} I$ capturing the categorical dependency. The dimensions of $D(\psi)$ are therefore $(q_{sp}+q_{cat}) \times (q_{sp}+q_{cat})$, with three variance components in ψ not including the error component σ^2: $\sigma^2_{b1}, \sigma^2_{b2}$(the RBF parameters), and σ^2_{b3}(the categorical feature variance). For two cars from the same location and model, the correlation is given by $\rho = \frac{\sigma^2_{b0}+\sigma^2_{b2}}{\sigma^2_{b0}+\sigma^2_{b2}+\sigma^2}$. The matrix $Z_{n\times(q_{sp}+q_{cat})}$ is formed by concatenating the binary spatial matrix $Z_{n\times q_{sp}}$ with the binary categorical matrix $Z_{n\times q_{cat}}$.

We compare COPNN, using different marginal distributions, to LMMNN and traditional methods like entity embeddings for the RE features, or ignoring these dependencies altogether. While COPNN and LMMNN optimize for NLL, the traditional methods are optimized for MSE. OHE was not feasible here due to the dataset's high cardinality, which would introduce over 18,000 additional columns to the model. We use the same DNN architecture for all methods: an MLP with two hidden layers of 10 and 3 neurons, respectively. Training is done with a batch size of 100 for 200 epochs. A 10-fold cross-validation (10-CV) procedure is applied, and we report the mean test MSE and mean test NLL for both COPNN and LMMNN. As shown in Table 4, ignoring the spatial and categori-

Table 4: The Cars spatial-categorical dataset: mean test MSE and NLL of predicting a car's log(price), and estimated within-location-model ρ. Bold results are within 2 SEs of the best result in a paired test.

Method	Test MSE	Test NLL	$\hat{\rho}$
Ignore	0.164 (.013)	–	–
Embedding	0.155 (.012)	–	–
LMMNN	0.135 (.011)	36.1 (2.4)	.76
COP-Logistic	0.127 (.011)	**26.9 (1.8)**	.70
COP-Laplace	0.127 (.011)	**25.8 (1.7)**	.72
COP-LogGam	0.129 (.012)	34.1 (2.0)	.75
COP-SkewNor	**0.123 (.011)**	32.4 (2.5)	.76

Table 5: The Rossmann longitudinal dataset: mean test AUC of predicting if a store will surpass its 90th quantile monthly no. of customers. Bold results are within 2 SEs of the best result in a paired test.

Method	Random mode	Future mode
Ignore	0.87 (.004)	0.69 (.018)
OHE	0.78 (.012)	0.59 (.019)
Embedding	0.77 (.008)	0.51 (.005)
LMMNN	0.90 (.005)	0.70 (.005)
COPNN	**0.91 (.004)**	**0.73 (.004)**

cal RE features significantly degrades performance on this dataset. While LMMNN performs better compared to ignoring the features or using entity embeddings, COPNN with an appropriate marginal distribution performs even better, yielding lower test MSE and NLL. This suggests that a Gaussian marginal is not the most suitable for this data, and distributions like Logistic or Laplace allowing heavier tails might be more appropriate.

In the Rossmann stores dataset, we aim to predict for the jth store at time t_{lj} whether y_{lj} represents an extreme month (1) or not (0). Here t_{lj} is essentially the month transformed into the $[0, 1]$ segment. LMMNN can only implement a random intercepts model, treating each store as a high-cardinality feature. However, COPNN, with its NLPL loss, is able to model a much richer longitudinal structure, incorporating a random slope and quadratic term: $\eta(y_{lj}) = g_f(x_{lj}) + b_{0j} + b_{1j}t_{lj} + b_{2j} \cdot t_{lj}^2$. Assuming no correlation between terms, the covariance matrix $D(\psi)$ consists of three diagonal $q \times q$ blocks: $\sigma^2_{b0}I$, $\sigma^2_{b1}I$, and $\sigma^2_{b2}I$, for a total size of $3q \times 3q$, representing three variance components. Additional details regarding the Z matrix structure can be found in Appendix A.

The training process follows similar details as the continuous case, including DNN architecture and number of epochs, with COPNN using the NLPL loss. For the Future mode, we reserve the last 6 months of data from each store as the test set, with the remaining data used for training via a 5-CV procedure. Table 5 reports the bottom-line AUC. As can be seen, the Future mode poses greater prediction challenges compared to the Random mode. Nevertheless, COPNN consistently achieves the highest AUC on unseen data, reflecting its superior capacity to model dependencies across time.

6 CONCLUSION

In this work, we introduced COPNN, a method for mixed modeling in DNN with flexible marginal distributions. Our experiments on both simulated and real datasets showed COPNN's ability to capture covariance structures and improve predictive performance, especially when Gaussian assumptions are inappropriate. COPNN also achieved significant reductions in NLL and MSE compared to LMMNN, particularly with skewed data, and demonstrated scalability to large datasets in practical applications like retail and automotive pricing.

Future work will aim to improve COPNN's interpretability and scalability for even larger datasets, broadening its potential in machine learning applications. We would also address the challenge of selecting the marginal distribution and explore alternative copula families to enhance the handling of dependencies. For instance, within the Archimedean family, the multivariate Clayton copula could be readily employed due to its closed-form density, enabling a natural NLL formulation for DNN in the spirit of COPNN. Specifically, Equation (6) could be adapted by replacing $c[F_1(y_1), \ldots, F_n(y_n)]$ with $(\prod_{i=1}^n F(y_i)^{-(1+\eta)}) \cdot (\sum_{i=1}^n F(y_i)^{-\eta} - n + 1)^{-(n+1)/\eta}$, where η is the copula's correlation parameter. However, there are inherent limitations. Unlike the Gaussian copula, which supports a complex and rich correlation matrix, the Clayton copula is restricted to a single correlation parameter, making it suitable for specific scenarios, such as modeling a single categorical feature with many levels, but less effective for spatial, temporal, or mixed correlations.

Other multivariate Archimedean copulas, such as the Gumbel copula, lack a closed-form density, complicating their use in constructing NLL losses for mini-batch optimization. If the bivariate density is simple this might be treated with our pairwise likelihood approach. The focus of this work is to extend and scale non-linear GLMM using DNN for traditional correlation scenarios, which aligns naturally with the Gaussian copula.

Acknowledgements

This study was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University, by Israel Science Foundation grant 2180/20 and by Israel Council for Higher Education Data-Science Centers.

References

Chen, H., Zheng, L., AL Kontar, R., and Raskutti, G. (2020). Stochastic gradient descent in correlated settings: A study on gaussian processes. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2722–2733. Curran Associates, Inc.

Feldman, J. and Kowal, D. R. (2022). Bayesian data synthesis and the utility-risk trade-off for mixed epidemiological data. *The Annals of Applied Statistics*, 16(4):2577 – 2602.

Feldman, J. and Kowal, D. R. (2024). Nonparametric copula models for multivariate, mixed, and missing data. *Journal of Machine Learning Research*, 25(164):1–50.

FlorianKnauer, W. C. (2015). Rossmann store sales.

Frees, E. W. and Valdez, E. A. (2008). Hierarchical insurance claims modeling. *Journal of the American Statistical Association*, 103(484):1457–1469.

Hoff, P. D. (2007). Extending the rank likelihood for semiparametric copula estimation. *The Annals of Applied Statistics*, 1(1):265 – 283.

Janke, T., Ghanmi, M., and Steinke, F. (2021). Implicit generative copulas. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 26028–26039. Curran Associates, Inc.

Joe, H. and Lee, Y. (2009). On weighting of bivariate margins in pairwise likelihood. *Journal of Multivariate Analysis*, 100(4):670–685.

Joseph Ferreira, J. and Minikel, E. (2012). Measuring per mile risk for pay-as-you-drive automobile insurance. *Transportation Research Record*, 2297(1):97–103.

Killiches, M. and Czado, C. (2018). A d-vine copula-based model for repeated measurements extending linear mixed models with homogeneous correlation structure. *Biometrics*, 74(3):997–1005.

Ling, C. K., Fang, F., and Kolter, J. Z. (2020). Deep archimedean copulas. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1535–1545. Curran Associates, Inc.

McCulloch, C. E., Searle, S. R., and Neuhaus, J. M. (2008). *Generalized, Linear, and Mixed Models*. John Wiley and Sons, Inc.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Reese, A. (2020). Used cars dataset - vehicles listings from craigslist.org.

Searle, S. R., Casella, G., and McCulloch, C. (1992). *Variance components*. Wiley Series in Probability and Statistics. John Wiley & Sons.

Simchoni, G. and Rosset, S. (2021). Using random effects to account for high-cardinality categorical features and repeated measures in deep neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25111–25122. Curran Associates, Inc.

Simchoni, G. and Rosset, S. (2023). Integrating random effects in deep neural networks. *Journal of Machine Learning Research*, 24(156):1–57.

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231.

Song, P. X.-K. X.-K. (2007). *Correlated data analysis : modeling, analytics, and applications*. Springer series in statistics. Springer, New York, NY.

Sun, J., Frees, E. W., and Rosenberg, M. A. (2008). Heavy-tailed longitudinal data modeling using copulas. *Insurance: Mathematics and Economics*, 42(2):817–830.

Tran, M.-N., Nguyen, N., Nott, D., and Kohn, R. (2020). Bayesian deep net glm and glmm. *Journal of Computational and Graphical Statistics*, 29(1):97–113.

Varin, C. (2008). On composite marginal likelihoods. *AStA Advances in Statistical Analysis*, 92(1):1–28.

Varin, C., Høst, G., and Øivind Skare (2005). Pairwise likelihood inference in spatial generalized linear mixed models. *Computational Statistics & Data Analysis*, 49(4):1173–1191.

Varin, C., Reid, N., and Firth, D. (2011). An overview of composite likelihood methods. *Statistica Sinica*, 21(1):5–42.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016a). Deep kernel learning. In Gretton, A. and Robert, C. C., editors, *Proceedings of the*

19th International Conference on Artificial Intelligence and Statistics, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain. PMLR.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. (2016b). Stochastic variational deep kernel learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Xiong, Y., Kim, H. J., and Singh, V. (2019). Mixed effects neural networks (menets) with applications to gaze estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, H. (2004). On Estimation and Prediction for Spatial Generalized Linear Mixed Models. *Biometrics*, 58(1):129–136.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, see a scalability simulation in Appendix D]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes, all code is available in a public Github repository, see link in paper]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes, all code is available in a public Github repository, see link in paper]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes, see detailed mean and SEs in Appendix D]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes, real datasets are freely available from Kaggle, see sources]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Common Covariance Structures

Single categorical feature: random intercepts The random intercepts model is used for a categorical RE variable with q distinct levels. The Z matrix, which has dimensions $n \times q$, is a binary matrix where $Z_{ij} = 1$ indicates that observation i corresponds to level j of the categorical variable, and $Z_{ij} = 0$ otherwise. This ensures that each row has exactly one non-zero entry. We can represent the lth observation from level j ($j = 1, \dots q; l = 1, \dots, n_j$) as y_{lj}, and the scalar form of model (1) with $g_f(x) = \beta^T x$ and $g_r(z) = z$ is written as:

$$y_{lj} = \beta_0 + \beta^T x_{lj} + b_j + \varepsilon_{lj}. \tag{17}$$

In this formulation, each level j of the categorical variable has its own random intercept b_j, which gives the model its name. The random intercepts b_j are assumed to follow a normal distribution with mean 0 and variance σ_b^2, making $\psi = \sigma_b^2$ (not including the error term σ^2) and the covariance matrix $D(\psi) = \sigma_b^2 I$ diagonal. Consequently, the marginal covariance matrix $V(\psi)$ for y becomes block diagonal, where $V(\psi) = \sigma_b^2 ZZ^T + \sigma_e^2 I_n$. This block structure simplifies the inversion of $V(\psi)$ when calculating the NLL or the BLUP. Specifically, the BLUP for any level j can be computed as $\hat{b}_j = \frac{n_j \hat{\sigma}_b^2}{\hat{\sigma}^2 + n_j \hat{\sigma}_b^2} \left(\bar{y}_{tr;j} - \overline{X_{tr}\beta}_j \right)$, where $\hat{\sigma}^2$ and $\hat{\sigma}_b^2$ are the estimated variance components, n_j is the number of observations in level j, and $\bar{y}_{tr;j}$ and $\overline{X_{tr}\beta}_j$ are the observed and predicted average values of y within level j, respectively.

Multiple categorical features For K categorical RE variables, each with q_k levels, the matrix Z can be expressed as the concatenation of K binary matrices Z_k, each of size $n \times q_k$. This results in a larger binary matrix of size $n \times M$, where $M = \sum_k q_k$. The RE vector b has length M and is distributed as $\mathcal{N}(0, D(\psi))$, where $D(\psi)$ is an $M \times M$ covariance matrix. If the RE are correlated across the K variables, these correlations are captured in the off-diagonal elements of $D(\psi)$, forming part of the variance components to estimate. In the absence of such correlations, $D(\psi)$ remains diagonal, with $\psi = [\sigma_{b1}^2, \dots, \sigma_{bK}^2]$, not including σ^2. The marginal covariance matrix of y, when assuming the K categorical variables are uncorrelated, takes the form: $V(\psi) = \sum_k \sigma_{bk}^2 Z_k Z_k^T + \sigma_e^2 I_n$. In this case, $V(\psi)$ is no longer block-diagonal, reflecting the combined contribution of the multiple categorical features to the marginal covariance of y.

Longitudinal data and repeated measures In many practical situations, repeated measurements are taken for the same units of interest, often corresponding to q subjects, each being monitored over time for a continuous response y. It is common to assume temporal correlation among these observations, and a longitudinal LMM model is employed to predict y at various time points. In scalar form, the l-th measurement of subject j can be modeled using a polynomial function of time t_{lj} as follows:

$$y_{lj} = \beta_0 + \beta^T x_{lj} + b_{0,j} + b_{1,j} \cdot t_{lj} + b_{2,j} \cdot t_{lj}^2 + \dots + b_{K-1,j} \cdot t^{K-1} lj + \varepsilon_{lj}. \tag{18}$$

Here, each subject j ($j = 1, \dots, q$) has a random intercept $b_{0,j}$, random slope $b_{1,j}$, and so on up to the $(K-1)$-th polynomial term. Each random effect $b_{k,j}$ is assumed to follow $\mathbb{N}(0, \sigma_{b,k}^2)$. This model is also flexible enough to incorporate time-varying covariates from X or fixed effects for time through the β coefficients.

Let t represent the full n-length vector of time points. The matrix Z_0 is a binary $n \times q$ matrix, where $Z_0[l, j] = 1$ if subject j was measured at time t_l. The full design matrix Z is of dimension $n \times Kq$, formed by concatenating K matrices: $[Z_0 \vdots Z_1 \vdots \dots \vdots Z_{K-1}]$, where each $Z_k = \text{diag}(t^k) \cdot Z_0$ for $k = 0, \dots, K-1$. Thus, $Z_k[l, j] = t_{lj}^k$ if subject j was measured at time t_l, and 0 otherwise. The random effects vector b is of length Kq and follows a normal distribution with covariance matrix $D(\psi)$ of size $Kq \times Kq$, where the diagonal contains the variance components $\sigma_{b,0}^2 I_q, \dots, \sigma_{b,K-1}^2 I_q$. If random effects are correlated, additional correlation parameters are included in the off-diagonal of $D(\psi)$, but if uncorrelated, we have $\psi = [\sigma_{b,0}^2, \dots, \sigma_{b,K-1}^2]$, and $D(\psi)$ remains diagonal. The marginal covariance matrix $V(\psi)$ of y is block-diagonal in this case, with correlations between observations arising both from the variance components and through the time variable t. For example, in a random slopes model with uncorrelated intercept and slope terms, the correlation between two observations y_{ij} and y_{lj} from the same subject j is:

$$\text{Corr}(y_{ij}, y_{lj}) = \frac{\sigma_{b,0}^2 + \sigma_{b,1}^2 t_{ij} t_{lj}}{\sqrt{(\sigma_{b,0}^2 + \sigma_{b,1}^2 t_{ij}^2 + \sigma^2)(\sigma_{b,0}^2 + \sigma_{b,1}^2 t_{lj}^2 + \sigma^2)}} \tag{19}$$

For further details on this approach, see McCulloch et al. (2008).

Kriging or spatial data Consider a continuous measurement y that varies across an N-dimensional random field $\mathcal{S}$. For each point $s \in \mathcal{S}$ (representing a location in space and time), the value of $y(s)$ is expressed as the sum of a deterministic component μ and a stochastic component e, both functions of the location s and some features $x \in \mathbb{R}^p$, such that:

$$y(s) = \mu(x, s) + e(s) + \varepsilon. \tag{20}$$

Here, μ may represent a constant mean or a regression-like term $x^T\beta$, independent of s, while $e(s)$ is a Gaussian-distributed additive term with zero mean and a spatial covariance structure. Typically, the covariance decays as the distance between two locations $h_{ij} = |s_i - s_j|$ increases. In the isotropic case, where the covariance depends only on the distance h_{ij} and not on direction, we can express the covariance as $\text{cov}(y(s_i), y(s_j)) = f(h_{ij})$, where f is the kernel function, commonly denoted as $k(s_i, s_j)$. One of the most widely used kernels is the radial basis function (RBF) or squared exponential kernel:

$$\text{Cov}(y(s_i), y(s_j)) = \tau^2 \cdot \exp\left(-\frac{h_{ij}^2}{2l^2}\right), \tag{21}$$

where τ^2 is the variance, and l^2 is the lengthscale (or range) parameter governing the rate of decay of covariance as the distance increases. The covariance diminishes rapidly with increasing h_{ij}, depending on the specific kernel and parameters used. This forms the basis of kriging, Gaussian processes (GP), and spatial analysis, all of which share similar core structures (see e.g. Rasmussen and Williams (2005)). Additionally, this setup is analogous to model (1), where $Z_{n\times q}$ represents a binary matrix for q locations, and b has a covariance matrix $D(\psi)$ of size $q \times q$:

$$D_{ij}(\psi) = \sigma_{b0}^2 \cdot \exp\left(-\frac{|s_i - s_j|^2}{2\sigma_{b1}^2}\right), \tag{22}$$

where $\psi = [\sigma_{b0}^2, \sigma_{b1}^2]$, and s_i, s_j are N-dimensional locations. Commonly, N is 2 (latitude and longitude) or 3 (latitude, longitude, and time). In this setting, the marginal covariance matrix of y does not exhibit a sparse structure.

B Metropolis-Hastings Algorithm for Predicting the RE in Classification

In this section, we provide a detailed explanation of the Metropolis-Hastings (MH) algorithm used to predict the vector of random effects $b \in \mathbb{R}^q$ given the observed binary outcomes y, as referenced in Section 3.2. This approach allows us to estimate the posterior distribution of b, given the observed data and the estimated g_f, g_r, ψ, under the specified covariance structure $D_{q\times q}$. Then the final prediction vector is: $P(y_{te} = 1) = \Phi\left(\hat{g}_f(X_{te}) + \hat{g}_r(Z_{te})\hat{b}\right)$.

Given the latent variable formulation for the binary case, with $y \in \{0, 1\}$ and the probit link function, we define the latent variable $l_i = g_f(X_i) + g_r(Z_i)b + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, 1)$. The goal is to sample from the posterior distribution $P(b|y)$, which is not available in closed form due to the non-conjugate nature of the probit link. We therefore resort to MCMC methods, specifically the Metropolis-Hastings algorithm from Zhang (2004).

Algorithm steps:

1. Start with an initial guess $b^{(0)}$, from $\mathcal{N}(0, 1)$
2. For each $m = 1, \dots, n_{iter}$:
   (a) Initialize $b^{(m)} = b^{(m-1)}$
   (b) For each b_j $(j = 1, \dots q)$, sample candidate b_j^* from $\mathcal{N}\left(-\sum_{k\neq j}(D_{jk}/D_{jj})b_k^{(m-1)}, 1/D_{jj}\right)$
   (c) Define $\mathcal{J}$, the set of all observations i impacted by b_j, calculate $P(y_j|b_j^*)$ and $P(y_j|b_j^{(m-1)})$, where $P(y_j|b_j) = \prod_{i\in\mathcal{J}} \Phi(l_i)^{y_i}[1 - \Phi(l_i)]^{1-y_i}$
   (d) Calculate acceptance ratio $\alpha = \min\left(1, P(y_j|b_j^*)/P(y_j|b_j^{(m-1)})\right)$
   (e) Accept $b_j^{(m)} = b_j^*$ w.p. α
3. Calculate mean of posterior samples after $n_{burn-in}$ iterations: $\hat{b} = \frac{1}{n_{iter}-n_{burn-in}} \sum_{m=n_{burn-in}}^{n_{iter}} b^{(m)}$

C Marginal Distributions

Table 6 details the marginal distributions considered here, Figure 3 visualizes their densities and below is a list of important parameters and functions used in their formulations. All distributions are formulated so that they have $E(X) = 0$ and $Var(X) = \tau^2$, where for the visualization we use $\tau^2 = 1$.

- γ is the Euler constant
- ψ, ψ_1 are the digamma and trigamma functions
- $\kappa = 1.42625512$ to make $\psi_1(\kappa) = 1$
- Γ is the gamma function
- $\alpha = 1$ is a skewness parameter assumed known
- ϕ, Φ the $\mathcal{N}(0,1)$ density and CDF functions
- The Gaussian mixture is a 50%/50% mixture of two Gaussians $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, where $\mu_1 = -\mu_2$ and they are distant from the zero mean by $a = 3$ (assumed known) standard deviations from each side of the zero mean, where $\sigma_1 = \sigma_2$ are set so that the total variance is τ^2 as required

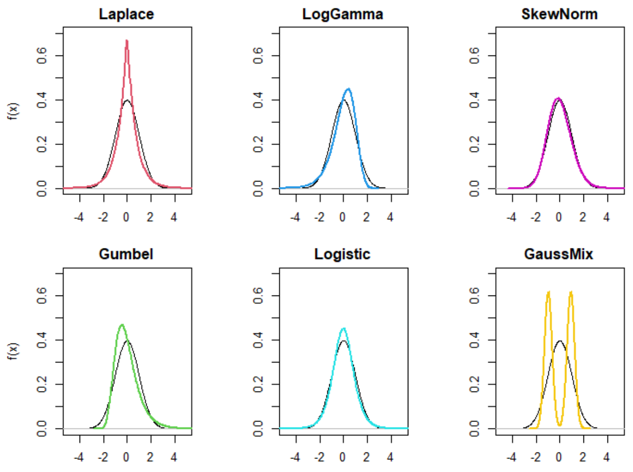

Figure 3: Visualization of the marginal distributions density $f(x)$ where $E(X) = 0$ and $Var(X) = 1$.

Table 6: Some continuous marginal distributions, parameterized so that $E(X) = 0$ and $Var(X) = \tau^2$.

Distribution	$f_\theta(x)$
Laplace	$\frac{1}{2\sqrt{\tau^2/2}} \exp\left(-\frac{\lvert x\rvert}{\sqrt{\tau^2/2}}\right)$
Gumbel	$\frac{\pi}{\tau\sqrt{6}} e^{-\left(\frac{x\pi}{\tau\sqrt{6}}+\gamma+e^{-\left(\frac{x\pi}{\tau\sqrt{6}}+\gamma\right)}\right)}$
Log-Gamma	$\frac{\sqrt{\psi_1(\kappa)}}{\tau\Gamma(\kappa)} e^{\kappa\sqrt{\psi_1(\kappa)}\frac{x+\psi(\kappa)}{\tau}} e^{-e^{\sqrt{\psi_1(\kappa)}\frac{x+\psi(\kappa)}{\tau}}}$
Logistic	$\frac{e^{-\frac{x\pi}{\tau\sqrt{3}}}}{\frac{\tau\sqrt{3}}{\pi}\left(1+e^{-\frac{x\pi}{\tau\sqrt{3}}}\right)^2}$
Skew-Normal	$2\phi\left(\frac{x\sqrt{\pi(1+\alpha^2)-2\alpha^2}+\alpha\sqrt{2\tau^2}}{\sqrt{\pi\tau^2(1+\alpha^2)}}\right)\Phi\left(\alpha\left(\frac{x\sqrt{\pi(1+\alpha^2)-2\alpha^2}+\alpha\sqrt{2\tau^2}}{\sqrt{\pi\tau^2(1+\alpha^2)}}\right)\right)$
GaussMix	$\sqrt{\frac{1+a^2}{8\pi\tau^2}}\left[\exp\left(-0.5\left(\frac{x\sqrt{1+a^2}}{\tau}-a\right)^2\right)+\exp\left(-0.5\left(\frac{x\sqrt{1+a^2}}{\tau}+a\right)^2\right)\right]$

D Additional Experiments

Table 7: Mean test MSE for $y \in \mathbb{R}$ in simulated datasets with various covariance structures, marginal distributions and ρ, standard errors in parentheses.

		$\rho = 0.1$			$\rho = 0.5$			$\rho = 0.9$		
	Method	Categ.	Longit.	Spatial	Categ.	Longit.	Spatial	Categ.	Longit.	Spatial
Gaussian	Ignore	1.25 (.01)	1.17 (.02)	1.23 (.01)	2.29 (.07)	2.07 (.04)	1.49 (.03)	10.7 (.44)	5.39 (.19)	2.51 (.11)
	OHE	1.38 (.01)	1.38 (.01)	1.34 (.01)	1.44 (.02)	1.43 (.01)	1.41 (.02)	1.67 (.03)	0.94 (.02)	1.78 (.03)
	Embed.	1.28 (.01)	1.28 (.01)	1.27 (.01)	1.35 (.02)	1.35 (.02)	1.32 (.01)	1.65 (.04)	0.89 (.02)	1.77 (.04)
	LMMNN	1.17 (.02)	1.17 (.01)	1.14 (.01)	1.18 (.01)	1.17 (.01)	1.17 (.01)	1.20 (.02)	0.62 (.01)	1.17 (.02)
	COPNN	1.16 (.02)	1.18 (.01)	1.13 (.02)	1.17 (.01)	1.17 (.02)	1.18 (.01)	1.19 (.02)	0.62 (.01)	1.17 (.02)
Laplace	Ignore	1.22 (.02)	1.18 (.01)	1.26 (.02)	2.19 (.14)	2.12 (.06)	1.68 (.06)	10.3 (.61)	5.05 (.40)	2.71 (.15)
	OHE	1.33 (.02)	1.37 (.02)	1.35 (.02)	1.47 (.05)	1.45 (.03)	1.50 (.04)	1.78 (.04)	0.97 (.06)	1.93 (.11)
	Embed.	1.27 (.02)	1.30 (.01)	1.28 (.02)	1.36 (.04)	1.36 (.02)	1.41 (.04)	1.65 (.05)	0.88 (.04)	1.85 (.08)
	LMMNN	1.15 (.02)	1.13 (.02)	1.14 (.01)	1.19 (.01)	1.17 (.02)	1.17 (.02)	1.27 (.04)	0.63 (.02)	1.23 (.04)
	COPNN	1.13 (.02)	1.10 (.02)	1.12 (.01)	1.17 (.01)	1.13 (.02)	1.15 (.01)	1.22 (.03)	0.62 (.02)	1.19 (.04)
Gumbel	Ignore	1.25 (.02)	1.20 (.02)	1.25 (.01)	2.24 (.08)	2.12 (.11)	1.53 (.04)	10.5 (.71)	5.27 (.30)	2.66 (.13)
	OHE	1.37 (.02)	1.40 (.03)	1.36 (.01)	1.43 (.02)	1.44 (.05)	1.42 (.03)	1.73 (.06)	0.93 (.03)	1.90 (.10)
	Embed.	1.30 (.02)	1.32 (.02)	1.30 (.02)	1.34 (.02)	1.35 (.04)	1.34 (.03)	1.62 (.06)	0.85 (.04)	1.85 (.09)
	LMMNN	1.17 (.02)	1.18 (.02)	1.12 (.02)	1.20 (.03)	1.19 (.03)	1.22 (.04)	1.23 (.05)	0.65 (.01)	1.24 (.04)
	COPNN	1.18 (.01)	1.19 (.01)	1.13 (.02)	1.18 (.03)	1.17 (.03)	1.20 (.03)	1.20 (.04)	0.65 (.02)	1.22 (.04)
LogGamma	Ignore	1.28 (.02)	1.16 (.01)	1.23 (.03)	2.34 (.11)	2.12 (.10)	1.59 (.04)	10.2 (.57)	4.95 (.35)	2.65 (.13)
	OHE	1.40 (.03)	1.34 (.01)	1.32 (.03)	1.51 (.04)	1.44 (.01)	1.52 (.04)	1.72 (.06)	0.95 (.03)	1.81 (.06)
	Embed.	1.32 (.02)	1.30 (.01)	1.26 (.03)	1.39 (.04)	1.35 (.02)	1.40 (.04)	1.67 (.04)	0.89 (.03)	1.92 (.12)
	LMMNN	1.17 (.01)	1.17 (.02)	1.14 (.01)	1.21 (.02)	1.18 (.02)	1.19 (.02)	1.20 (.03)	0.61 (.01)	1.22 (.04)
	COPNN	1.14 (.01)	1.14 (.02)	1.10 (.01)	1.16 (.02)	1.14 (.03)	1.16 (.02)	1.16 (.03)	0.59 (.01)	1.18 (.05)
Logistic	Ignore	1.24 (.02)	1.16 (.01)	1.22 (.02)	2.16 (.05)	2.16 (.05)	1.56 (.04)	10.8 (.73)	5.39 (.28)	2.48 (.10)
	OHE	1.36 (.02)	1.35 (.02)	1.33 (.02)	1.41 (.02)	1.43 (.02)	1.45 (.02)	1.75 (.04)	0.91 (.02)	1.85 (.08)
	Embed.	1.28 (.02)	1.28 (.02)	1.26 (.01)	1.32 (.02)	1.33 (.01)	1.35 (.02)	1.68 (.05)	0.87 (.03)	1.87 (.10)
	LMMNN	1.13 (.01)	1.13 (.01)	1.17 (.01)	1.17 (.02)	1.18 (.02)	1.15 (.02)	1.17 (.01)	0.63 (.01)	1.15 (.01)
	COPNN	1.14 (.01)	1.14 (.01)	1.17 (.01)	1.16 (.02)	1.17 (.02)	1.16 (.02)	1.15 (.01)	0.62 (.01)	1.13 (.01)
SkewNorm	Ignore	1.29 (.01)	1.18 (.01)	1.26 (.01)	2.22 (.06)	2.10 (.07)	1.49 (.02)	10.9 (.75)	5.52 (.23)	2.47 (.07)
	OHE	1.40 (.01)	1.37 (.02)	1.36 (.02)	1.44 (.02)	1.41 (.02)	1.43 (.02)	1.75 (.04)	0.95 (.01)	1.82 (.05)
	Embed.	1.33 (.01)	1.31 (.02)	1.30 (.01)	1.34 (.04)	1.30 (.02)	1.33 (.01)	1.70 (.05)	0.89 (.01)	1.76 (.04)
	LMMNN	1.17 (.01)	1.11 (.01)	1.13 (.01)	1.16 (.01)	1.17 (.01)	1.16 (.01)	1.16 (.01)	0.62 (.01)	1.15 (.01)
	COPNN	1.17 (.01)	1.12 (.02)	1.12 (.01)	1.17 (.01)	1.16 (.02)	1.15 (.01)	1.17 (.01)	0.62 (.01)	1.15 (.01)
GaussMix	Ignore	1.26 (.01)	1.17 (.01)	1.25 (.01)	2.19 (.03)	2.20 (.02)	1.60 (.04)	10.7 (.09)	5.06 (.11)	3.06 (.15)
	OHE	1.42 (.02)	1.42 (.01)	1.39 (.01)	1.59 (.02)	1.56 (.03)	1.54 (.02)	2.29 (.05)	1.28 (.03)	2.27 (.08)
	Embed.	1.33 (.01)	1.33 (.01)	1.31 (.01)	1.48 (.02)	1.43 (.02)	1.43 (.02)	2.16 (.04)	1.22 (.03)	2.21 (.06)
	LMMNN	1.17 (.01)	1.15 (.01)	1.20 (.01)	1.28 (.02)	1.31 (.02)	1.31 (.02)	1.60 (.06)	0.93 (.02)	1.84 (.06)
	COPNN	1.09 (.01)	1.07 (.00)	1.12 (.01)	1.19 (.01)	1.20 (.02)	1.21 (.02)	1.49 (.05)	0.86 (.02)	1.71 (.05)

Table 8: Mean test AUC for $y \in \{0, 1\}$ in simulated datasets with various covariance structures and ρ, standard errors in parentheses.

	Method	$\rho = 0.1$	$\rho = 0.5$	$\rho = 0.9$
Categ1	Ignore	0.72 (.004)	0.72 (.004)	0.72 (.004)
	OHE	0.69 (.004)	0.82 (.004)	0.94 (.003)
	Embed.	0.70 (.003)	0.84 (.004)	0.95 (.002)
	LMMNN	0.74 (.002)	0.85 (.003)	0.95 (.002)
	COPNN	0.74 (.003)	0.86 (.003)	0.95 (.002)
Categ3	Ignore	0.72 (.002)	0.72 (.004)	0.72 (.005)
	OHE	0.65 (.004)	0.79 (.007)	0.92 (.003)
	Embed.	0.68 (.005)	0.80 (.006)	0.94 (.003)
	LMMNN	0.74 (.004)	0.81 (.001)	0.90 (.002)
	COPNN	0.74 (.005)	0.86 (.001)	0.94 (.002)
Longitudinal	Ignore	0.73 (.003)	0.75 (.004)	0.77 (.005)
	OHE	0.68 (.004)	0.79 (.002)	0.92 (.003)
	Embed.	0.70 (.003)	0.80 (.004)	0.93 (.003)
	LMMNN	0.74 (.003)	0.81 (.003)	0.92 (.003)
	COPNN	0.74 (.003)	0.84 (.003)	0.94 (.002)
Spatial	Ignore	0.72 (.004)	0.80 (.007)	0.89 (.006)
	OHE	0.69 (.004)	0.82 (.006)	0.94 (.003)
	Embed.	0.71 (.003)	0.84 (.006)	0.96 (.002)
	LMMNN	0.74 (.004)	0.84 (.004)	0.93 (.004)
	COPNN	0.74 (.004)	0.86 (.003)	0.95 (.002)