Integrating Random Effects in Deep Neural Networks

Giora Simchoni, Saharon Rosset

Yuankang Zhao

October 17, 2025

Why Linear Mixed Models?

The Problem: Correlated Data

Example: Student Test Scores

- 1,000 students, 50 schools
- Multiple tests per student
- Same school → correlated

Standard Linear Regression

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \varepsilon_{ij}$$

Assumes **independence**

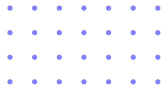
Problems:

Standard errors too small → False positives

Invalid inference (wrong p-values)

Ignores school-level effects

Standard Regression
(Independence Assumption)



Reality: Clustered Data

Solution: Linear Mixed Models

Model dependence via **random effects**

Linear Mixed Model: Mathematical Framework

Model Specification

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_j + \varepsilon_{ij}$$

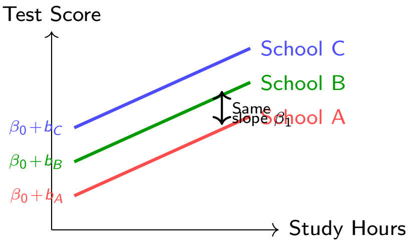
Components:

- **Fixed Effects:** $\beta_0 + \beta_1 x_{ij}$
  - $\hookrightarrow$ Population trends (same for all)
- **Random Effects:** $b_j \sim \mathcal{N}(0, \sigma_b^2)$
  - $\hookrightarrow$ School-specific deviations
- **Noise:** $\varepsilon_{ij} \sim \mathcal{N}(0, \sigma_e^2)$
  - $\hookrightarrow$ Individual measurement error

Two Sources of Variance

- σ_b^2: Between-school variance
- σ_e^2: Within-school variance

Visual Example



Marginal Covariance

Same school: $\mathrm{Cov}(y_{ij}, y_{i'j}) = \sigma_b^2$ (correlated)

Different schools: $\mathrm{Cov}(y_{ij}, y_{i'j'}) = 0$ (independent)

NLL and BLUP

In a typical LMM setting, $\mathbf{y} \in \mathbb{R}^n$ is modeled by:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon} \tag{1}$$

- $\mathbf{X}$: $n \times p$ fixed effects design matrix, $\boldsymbol{\beta} \in \mathbb{R}^p$: fixed effects
- $\mathbf{Z}$: $n \times q$ random effects design matrix, $\mathbf{b} \sim \mathcal{N}(\mathbf{0}, \mathbf{D}(\boldsymbol{\psi}))$: random effects
- $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_e^2\mathbf{I})$: i.i.d. noise, $\mathrm{cov}(\boldsymbol{\epsilon}, \mathbf{b}) = 0$

Marginal distribution: $\mathbf{y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \mathbf{V}(\boldsymbol{\theta}))$ where $\mathbf{V}(\boldsymbol{\theta}) = \mathbf{Z}\mathbf{D}(\boldsymbol{\psi})\mathbf{Z}' + \sigma_e^2\mathbf{I}$

Negative Log-Likelihood (NLL) - for estimating $\boldsymbol{\beta}, \boldsymbol{\theta}$:

$$\mathrm{NLL}(\boldsymbol{\beta}, \boldsymbol{\theta}|\mathbf{y}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\mathbf{V}(\boldsymbol{\theta})^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{1}{2}\log|\mathbf{V}(\boldsymbol{\theta})| + \frac{n}{2}\log 2\pi \tag{2}$$

Best Linear Unbiased Predictor (BLUP):

$$\hat{\mathbf{b}} = \mathbf{D}\mathbf{Z}'_{tr}\mathbf{V}(\hat{\boldsymbol{\theta}})^{-1}\left(\mathbf{y}_{tr} - \mathbf{X}_{tr}\hat{\boldsymbol{\beta}}\right) \tag{3}$$

$\mathbf{b}$ are random variables to be *predicted*, not parameters to be *estimated*

Single Categorical Feature: Random Intercepts

Real-world example: Predicting blood pressure across different clinics

The idea: Each category (clinic) gets its own random "shift" (intercept)

$$y_{lj} = \underbrace{\beta_0 + \boldsymbol{\beta}'\mathbf{x}_{lj}}_{\text{Fixed effects}} + \underbrace{b_j}_{\substack{\text{Random intercept} \\ \text{for clinic } j}} + \epsilon_{lj} \tag{4}$$

Setup:

- q clinics (categories)
- Each clinic has n_j patients
- $b_j \sim \mathcal{N}(0, \sigma_b^2)$ models clinic-specific effect

Why useful?

- Patients in same clinic are correlated
- "Borrows strength" across clinics
- Better than treating each clinic independently

Good news: Covariance matrix $\mathbf{V}$ is block-diagonal $\Rightarrow$ efficient computation!

Prediction for clinic j: $\hat{b}_j = \frac{n_j\hat{\sigma}_b^2}{\hat{\sigma}^2+n_j\hat{\sigma}^2}(\bar{y}_{tr;j} - \text{predicted mean})$

Multiple Categorical Features

Real-world example: Movie ratings with director *and* genre effects

The idea: Multiple sources of correlation in the data

Setup:

- Feature 1: Director (q_1 levels)
- Feature 2: Genre (q_2 levels)
- Feature 3: Actor (q_3 levels)
- $\vdots$
- Feature K: ... (q_K levels)

Model:

- Each feature gets random effect
- Total: $M = \sum_k q_k$ random effects
- Can model correlations between features

Uncorrelated case: Each feature has its own variance

$$\mathbf{V}(\boldsymbol{\theta}) = \underbrace{\sigma^2_{b_1}\mathbf{Z}_1\mathbf{Z}_1'}_{\text{Director}} + \underbrace{\sigma^2_{b_2}\mathbf{Z}_2\mathbf{Z}_2'}_{\text{Genre}} + \cdots + \underbrace{\sigma^2_{b_K}\mathbf{Z}_K\mathbf{Z}_K'}_{\text{Feature K}} + \sigma^2_e\mathbf{I}_n \tag{5}$$

Challenge: $\mathbf{V}$ is NOT block-diagonal anymore $\Rightarrow$ harder to compute!

Longitudinal Data: Repeated Measures

Real-world example: Patient health tracked over multiple hospital visits

The idea: Each subject has their own trajectory over time

$$y_{lj} = \underbrace{\beta_0 + \boldsymbol{\beta}'\mathbf{x}_{lj}}_{\text{Fixed effects}} + \underbrace{b_{0,j}}_{\substack{\text{Random}\\\text{intercept}}} + \underbrace{b_{1,j} \cdot t_{lj}}_{\substack{\text{Random}\\\text{slope}}} + \underbrace{b_{2,j} \cdot t_{lj}^2}_{\substack{\text{Random}\\\text{curvature}}} + \epsilon_{lj} \tag{6}$$

Components:

- $b_{0,j}$: Each subject's baseline
- $b_{1,j}$: Each subject's trend
- $b_{2,j}$: Each subject's acceleration
- All $\sim \mathcal{N}(0, \sigma_{b,k}^2)$

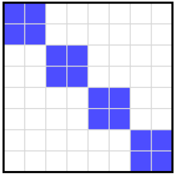
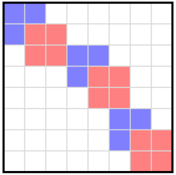
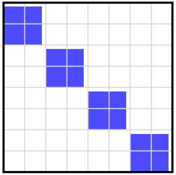
Example:

- Patient A: high baseline, steep decline
- Patient B: low baseline, gradual improvement
- Model captures individual differences

Good news: If sorted by subject, $\mathbf{V}$ is block-diagonal $\Rightarrow$ efficient!

Common in EMR data: short, irregular time series per patient

Visualizing Covariance Structures

Single Categorical	Multiple Categorical	Longitudinal
		
$\mathbf{V} = \sigma_b^2 \mathbf{Z}\mathbf{Z}' + \sigma_e^2 \mathbf{I}$	$\mathbf{V} = \sum_k \sigma_{b_k}^2 \mathbf{Z}_k \mathbf{Z}_k' + \sigma_e^2 \mathbf{I}$	$\mathbf{V}$ with random slope
Why this structure?	**Why this structure?**	**Why this structure?**
• Same clinic $\Rightarrow$ correlated	• Share director $\Rightarrow$ correlated	• Same subject $\Rightarrow$ correlated
• Different clinics $\Rightarrow$ independent	• Share genre $\Rightarrow$ correlated	• Different subjects $\Rightarrow$ independent
• Block-diagonal	• Overlapping $\Rightarrow$ NOT block-diag	• Block-diagonal (if sorted)

Key for LMMNN: Block-diagonal $\Rightarrow$ invert block-by-block $\Rightarrow$ mini-batch SGD works naturally
Not block-diagonal $\Rightarrow$ mini-batch approximation needed

Spatial Data: Geographic Correlations

Real-world example: Air pollution levels across different locations

The idea: Nearby locations are more similar than distant ones

$$y(s) = \underbrace{\mu(\mathbf{x}, s)}_{\text{Deterministic part}} + \underbrace{e(s)}_{\text{Spatial random effect}} + \epsilon \tag{7}$$

Key principle: Correlation decays with distance

$$\text{Covariance}(s_i, s_j) = \tau^2 \cdot \exp\left(-\frac{\text{distance}_{ij}^2}{2l^2}\right) \tag{8}$$

- τ^2: overall variance
- l^2: how fast correlation decays
- Small l $\Rightarrow$ only very close locations correlated
- Large l $\Rightarrow$ distant locations still correlated

Examples:

- Weather: neighboring cities similar
- Disease: local outbreaks
- House prices: location matters

Challenge: $\mathbf{V}$ is dense (not sparse) $\Rightarrow$ computationally expensive!

From LMM to LMMNN: The Key Idea

Standard LMM: Only linear relationships

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{b} + \epsilon \tag{9}$$

LMMNN: Replace with neural networks

$$\mathbf{y} = \underbrace{f(\mathbf{X})}_{\text{DNN for fixed}} + \underbrace{g(\mathbf{Z})\mathbf{b}}_{\text{DNN for random}} + \epsilon \tag{10}$$

Flexibility:

- f, g: any DNN (MLP, CNN, RNN)
- Can be same or different
- g can be identity: $g(\mathbf{Z}) = \mathbf{Z}$

Examples:

- f: MLP for tabular data
- $g =$ identity for categorical
- g: embedding for spatial

Best of both worlds: DNN flexibility + Principled correlation handling

LMMNN Visualization

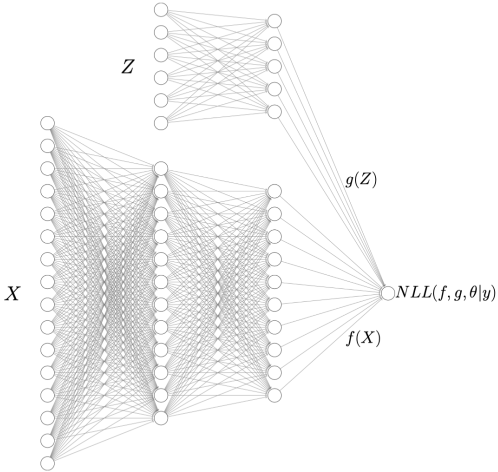


Figure 1: Schematic description of LMMNN using a simple deep MLP for fitting f and g, and combining outputs with the NLL loss layer, in a single-stage training.

LMMNN Training: NLL Loss with Mini-batch SGD

Modified NLL loss:

$$\mathrm{NLL}(f, g, \boldsymbol{\theta}|\mathbf{y}) = \frac{1}{2}(\mathbf{y} - f(\mathbf{X}))'\mathbf{V}^{-1}(\mathbf{y} - f(\mathbf{X})) + \frac{1}{2}\log|\mathbf{V}| + \frac{n}{2}\log 2\pi \tag{11}$$

where $\mathbf{V}(g, \boldsymbol{\theta}) = g(\mathbf{Z})\mathbf{D}(\psi)g(\mathbf{Z})' + \sigma_e^2\mathbf{I}_n$

Challenge: Large $n \Rightarrow$ inverting $n \times n$ matrix $\mathbf{V}$ is infeasible!

Solution: Mini-batch approximation (batch size $m \ll n$)

$$\mathrm{NLL}_\xi = \frac{1}{2}(\mathbf{y}_\xi - f(\mathbf{X}_\xi))'\mathbf{V}_\xi^{-1}(\mathbf{y}_\xi - f(\mathbf{X}_\xi)) + \frac{1}{2}\log|\mathbf{V}_\xi| + \frac{m}{2}\log 2\pi \tag{12}$$

- Invert small $m \times m$ sub-matrix $\mathbf{V}_\xi$ instead of full $n \times n$ matrix
- Variance components $\boldsymbol{\theta} = [\sigma_e^2, \psi]$ are learnable parameters
- Use standard backpropagation + SGD

Key innovation: "Inversion in parts" enables scalability

LMMNN Prediction: Modified BLUP

Training: Optimize f, g, and θ using mini-batch NLL on $(\mathbf{X}_{tr}, \mathbf{Z}_{tr}, \mathbf{y}_{tr})$

Prediction on test set:

$$\hat{\mathbf{y}}_{te} = \underbrace{\hat{f}(\mathbf{X}_{te})}_{\text{Fixed effects}} + \underbrace{\hat{g}(\mathbf{Z}_{te})\hat{\mathbf{b}}}_{\text{Random effects}} \tag{13}$$

Modified BLUP:

$$\hat{\mathbf{b}} = \mathbf{D}(\hat{\psi})\hat{g}(\mathbf{Z}_{tr})'\mathbf{V}(\hat{g}, \hat{\theta})^{-1}\left(\mathbf{y}_{tr} - \hat{f}(\mathbf{X}_{tr})\right) \tag{14}$$

Implementation Strategies

- **Single categorical** ($g =$ identity): Use closed-form BLUP, no inversion needed
- **Multiple categorical / Longitudinal**: Sparse $\mathbf{V} \Rightarrow$ solve linear system efficiently
- **Spatial** (dense $\mathbf{V}$, large n): Use sampling approximation or inducing points

Advantage: Prediction only requires ONE matrix inversion (at test time), not at every training iteration!

Why Does Mini-batch SGD Work?

Recall the challenge: mini-batch approximation:

$$\text{NLL}_{\xi} = \frac{1}{2}(\mathbf{y}_{\xi} - f(\mathbf{X}_{\xi}))'\mathbf{V}_{\xi}^{-1}(\mathbf{y}_{\xi} - f(\mathbf{X}_{\xi})) + \frac{1}{2}\log|\mathbf{V}_{\xi}| \tag{15}$$

Invert $\mathbf{V}_{\xi}$ (size $m \times m$) instead of sub-matrix of $\mathbf{V}^{-1}$ (from full $n \times n$)

The question: Is this valid? Does it converge?

Three Justifications

1. When $\mathbf{V}$ is block-diagonal $\Rightarrow$ gradient decomposes exactly
2. When $\mathbf{V}$ is approximately block-diagonal $\Rightarrow$ good approximation
3. Theoretical convergence guarantees (Chen et al. 2020)

Key insight: Block-diagonal structure (exact or approximate) makes "inversion in parts" mathematically sound

Block-diagonal Case: Gradient Decomposes Exactly

Key observation: When $\mathbf{V}(\theta)$ is block-diagonal, the gradient naturally decomposes!

Example: Random intercepts model (single categorical, q levels)

- $\mathbf{V}(\theta) = \sigma_b^2\mathbf{ZZ}' + \sigma_e^2\mathbf{I} = \mathrm{diag}(\mathbf{V}_1, \ldots, \mathbf{V}_q)$
- Each block $\mathbf{V}_j$ corresponds to one category (size $n_j \times n_j$)

Consequence: NLL and its gradient decompose as sums

$$\mathrm{NLL} = \sum_{j=1}^{q} \left[\frac{1}{2}(\mathbf{y}_j - f(\mathbf{X}_j))'\mathbf{V}_j^{-1}(\mathbf{y}_j - f(\mathbf{X}_j)) + \frac{1}{2}\log|\mathbf{V}_j|\right] \tag{16}$$

$$\frac{\partial \mathrm{NLL}}{\partial \boldsymbol{\theta}} = \sum_{j=1}^{q} \frac{\partial \mathrm{NLL}_j}{\partial \boldsymbol{\theta}} \tag{17}$$

Implication: If batch size $m = n_j$ and we sample by category $\Rightarrow$ computing gradient on mini-batch = computing true gradient! No approximation needed.

Also applies to: Longitudinal model (if sorted by subject), nested categorical features

Approximate Block-diagonal Structure

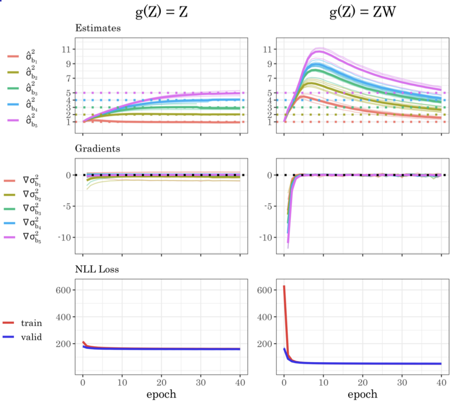


Figure: A LMMNN simulation with 5 uncorrelated categorical features each with $q = 1000$ and $\sigma^2_{b_j} = j$ for $j = 1, \ldots, 5$. $n = 100000$, $\sigma^2_e = 1$, there are $p = 10$ fixed features in X and $f(X)$ and network architecture are as described in Section 5.1. From top to bottom: $\sigma^2_{b_j}$ estimates, $\sigma^2_{b_j}$ gradients and NLL through epochs. The experiment was repeated five times, and the five results are shown as light lines, bold lines are average. Left: $g(Z) = Z$, Right: $g(Z) = ZW$, where W is a $5{,}000 \times 500$ random

Approximate Block-diagonal Structure

Problem: Many cases are NOT exactly block-diagonal:

- Multiple categorical features (overlapping patterns)
- Spatial data (nearby locations correlated)

Key insight: In practice, $\mathbf{V}$ is often approximately block-diagonal

Evidence from simulations:

Figure 2 shows $K = 5$ categorical features with $q = 1000$ each, $n = 100K$

- Variance estimates converge
- Gradients approach zero
- NLL decreases

Works even when $g(\mathbf{Z}) \neq \mathbf{Z}$!

Theoretical connection:

Bickel & Levina (2008): Banding covariance matrices

- Set distant correlations to zero
- Form of regularization
- Bounds on $\|\mathbf{B}_k(\boldsymbol{\Sigma}) - \boldsymbol{\Sigma}\|$
- Ideal when data is sorted

Mini-batch SGD implicitly does this!

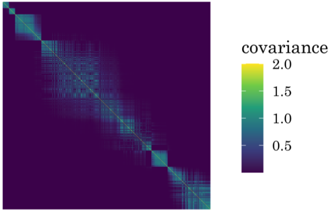
Practical takeaway: Mini-batch approximation works well empirically, even without exact block-diagonal structure

Evidence from Real Data: Figure 3 - UK Biobank

Experiment: Visualize covariance matrix $\mathbf{V}(\boldsymbol{\theta})$ on real data

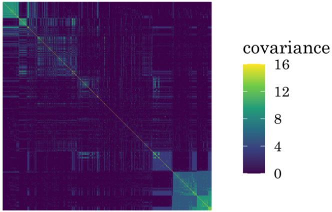
Dataset: Sample of $n = 1000$ UK Biobank cancer patients

Spatial Model (Left)



- RE: Patient location on UK map
- $q = 900$ unique locations
- RBF kernel: $\sigma^2_{b_0} = \sigma^2_{b_1} = 1$

Multiple Categorical (Right)



- 5 categorical REs: diagnosis (338), operation (304), treatment (211), cancer type (151), histology (104)
- $\sigma^2_{b_k} = k$ for $k = 1, \ldots, 5$

Implication: Real-world $\mathbf{V}$ matrices exhibit approximate block-diagonal or banded structure, even when not theoretically block-diagonal $\Rightarrow$ Mini-batch approximation is effective

PCA Sorting in Figure 3: Purpose and Necessity

Purpose: Visualization technique to reveal covariance structure

Procedure:
Spatial: PCA on distance matrix $\rightarrow$ sort by PC1
Categorical: PCA on $\mathbf{V}(\boldsymbol{\theta})$ $\rightarrow$ sort by PC1

Why PC1? Captures dominant structure, groups similar observations

Effect:

- With: Block patterns visible
- Without: Appears random

Is Sorting Required?
For Figure 3: Yes (visualization)
For LMMNN: NO!

- Algorithm works unsorted
- Figure 2: converges without sorting
- May help efficiency, not mandatory

$$\boxed{\text{Sorting} = \text{visualization tool, not algorithmic requirement}}$$

Theoretical Guarantees (Brief Overview)

Question: Can we prove convergence for non-block-diagonal cases?

Answer: Yes! Using results from Chen et al. (2020) on stochastic GP optimization

Key Result: Convergence of Full Gradient (Theorem 1)

Under certain conditions (exponential eigendecay of kernel), the full gradient magnitude is bounded:

$$\|\nabla \mathrm{NLL}(\boldsymbol{\theta}^K)\|_2^2 \leq C\left(\frac{G^2}{K} + m^{-\frac{1}{2}+\epsilon}\right) \tag{18}$$

where K = number of SGD iterations, m = batch size

What this means: As training progresses ($K \to \infty$), gradient $\to 0 \Rightarrow$ convergence!

Applicability:

- **Spatial data:** RBF kernel has exponential eigendecay
- **Multiple categorical:** Each $\mathbf{Z}_k\mathbf{Z}_k'$ has fast eigendecay (Appendix 1)

Related Methods: Categorical Features in DNNs (4.1)

1. **One-Hot Encoding (OHE)**
   - Create q binary columns: $\mathbf{Z}_{ij} = 1$ if obs i has level j, else 0
   - Feed directly into network as wide sparse input
   - $+$ Simple, no learning needed - Memory explodes for large q
2. **Entity Embeddings**
   - Learn lookup table $\mathbf{E} \in \mathbb{R}^{q \times d}$ ($d \ll q$, e.g., $d = 50$)
   - For level j: retrieve row $\mathbf{E}_j$ as dense embedding vector
   - Train embedding jointly with network using task loss (e.g., MSE)
   - $+$ Compact representation - Must retrain for each task, dimension q $\times$ d may be a problem
3. **MeNets (Xiong et al. 2019)**
   - Model: $y = f(\mathbf{X})\beta + f(\mathbf{X})\mathbf{b}$ where $\mathbf{b} \sim \mathcal{N}(0, \mathbf{D})$
   - **E-step:** Update $\hat{\beta}, \hat{\mathbf{b}}$ via SGD with MSE loss
   - **M-step:** Update $\hat{\theta}$ via NLL loss
   - $+$ Statistical foundation - inverte all q levels $n_j * n_j$ matrix, only diagonal $\mathbf{D}$

DeepGLMM (Tran et al. 2020): Model Setup

Context: Handling longitudinal/temporal data with repeated measures in DNNs

Data structure:

- Subject i measured at same set of times $t_1, \ldots, t_T$ (balanced design)
- Response y_{i,t_j} can be continuous or discrete (GLM framework)

Key idea: Learn features from DNN, then add random slopes for each feature

Model formulation:

Extract m features from last hidden layer via DNN $z_{it;j} = z(x_{it;j}), \quad j = 1, \ldots, m$
Then, for each subject i, add random intercept a_{i0} and random slopes a_{ij}

$$g(\mu_{it}) = \beta_0 + a_{i0} + (\beta_1 + a_{i1}) z_{it;1} + \cdots + (\beta_m + a_{im}) z_{it;m} = f(x_{it}^{(1)}, w, \boldsymbol{\beta}^{(1)}) \tag{19}$$

where $g(\cdot)$ is link function (e.g., identity for regression, logit for classification)

Further can be rewritten as:

$$g(\mu_{it}) = f(x_{it}^{(1)}, w, \boldsymbol{\beta}^{(1)}) + (\boldsymbol{\beta}^{(2)} + \mathbf{a}_i)' \mathbf{x}_{it}^{(2)} \tag{20}$$

where $\mathbf{x}^{(1)}$ = fixed features (nonlinear), $\mathbf{x}^{(2)}$ = random features (linear)

Simulation: Single Categorical Feature

Setup:

- $n = 100K$ observations, single categorical RE with $q \in \{100, 1000, 10000\}$ levels
- RE variance $\sigma_b^2 \in \{0.1, 1, 10\}$, noise $\sigma_e^2 = 1$ (27 scenarios $\times$ 5 repeats)
- Level sizes: Poisson(30) normalized (Pareto-like distribution)
- Fixed features: $p = 10$ from $U(-1, 1)$
- Nonlinear model: $y = (X_1 + \cdots + X_{10})\cos(X_1 + \cdots + X_{10}) + 2X_1X_2 + g(\mathbf{Z})\mathbf{b} + \epsilon$
- Three g transforms: (1) Identity, (2) Linear $\mathbf{ZW}_{q\times 0.1q}$, (3) Nonlinear $\mathbf{ZW} * \cos(\mathbf{ZW})$
- Network: 4 layers (100-50-25-12 neurons), ReLU, 25% dropout, batch=100
- Competitors: Ignore, OHE, Embeddings, lme4, MeNets

Key Results (Test MSE):

$g(\mathbf{Z})$	σ_b^2	q	OHE	Embed.	LMMNN
Identity	10	10^4	2.37	2.12	**1.29**
Linear	10	10^4	81.3	153.5	**13.9**
Nonlinear	10	10^4	36.3	90.7	**14.1**

Findings: LMMNN best in all scenarios, especially high q, high σ_b^2, complex g

Simulation: Multiple Categorical Features

Setup:

- $n = 100K$ observations, $K = 3$ uncorrelated categorical REs
- Cardinalities: $q_1 = 1000, q_2 = 2000, q_3 = 3000$ (total $M = 6000$ REs)
- Variances: $[\sigma^2_{b_1}, \sigma^2_{b_2}, \sigma^2_{b_3}] \in \{0.3, 3.0\}$ (8 combinations), $\sigma^2_e = 1$
- Same nonlinear f as 5.1.1: $(X_1 + \cdots + X_{10})\cos(\cdots) + 2X_1X_2$
- Three g transforms: Identity, Linear, Nonlinear (24 scenarios $\times$ 5 repeats)
- Same MLP architecture: 4 layers (100-50-25-12), batch=100
- Key challenge: $\mathbf{V}(\boldsymbol{\theta}) = \sum_k \sigma^2_{b_k}\mathbf{Z}_k\mathbf{Z}'_k + \sigma^2_e\mathbf{I}$ is NOT block-diagonal
- Competitors: Ignore, OHE, Embeddings, lme4 (MeNets N/A)

Key Results (Test MSE):

$g(\mathbf{Z})$	$[\sigma^2_{b_1}, \sigma^2_{b_2}, \sigma^2_{b_3}]$	OHE	Embed.	lme4	LMMNN
Identity	[3.0, 3.0, 3.0]	2.17	1.92	3.07	**1.17**
Linear	[3.0, 3.0, 3.0]	23.7	31.4	3.25	**2.50**

Findings: LMMNN handles overlapping correlations; gap widens with complexity

Setup:

- $n = 100K$ observations, $q = 10K$ subjects, variable n_j measurements per subject
- Time points: equally spaced in [0,1], max 6 measurements
- Model: $y_{ij} = f(x_{ij}) + b_{0,j} + b_{1,j}t_{ij} + b_{2,j}t_{ij}^2 + \epsilon_{ij}$
- Random effects: intercept + slope + quadratic with correlations $\rho_{01} = \rho_{02} = 0.3$
- Variances: $[\sigma^2_{b_0}, \sigma^2_{b_1}, \sigma^2_{b_2}] \in \{0.3, 3.0\}$ (8 combinations), $\sigma^2_e = 1$
- Two evaluation modes:
  - **Random:** Standard 80/20 random split
  - **Future:** Train on earliest 80%, test on latest 20% (temporal extrapolation)
- Same MLP (100-50-25-12), batch=100; LSTM: 5 neurons (grid-searched)
- Key: $\mathbf{V}(\boldsymbol{\theta})$ is block-diagonal (one block per subject)

Key Results (Test MSE):

Mode	$[\sigma^2_{b_0}, \sigma^2_{b_1}, \sigma^2_{b_2}]$	Embed.	LSTM	lme4	LMMNN
Random	[3.0, 3.0, 3.0]	1.88	4.55	5.10	**1.29**
Future	[3.0, 3.0, 3.0]	2.25	5.47	4.82	**1.47**

Findings: LMMNN superior in both modes; LSTM overfits short sequences

Real Data: Multiple Categorical Features

Datasets: 5 real-world tabular datasets with $K = 2$ to 5 high-cardinality categorical features

Dataset	n, K, p	**Categorical features (levels)**	**Target** y
Imdb	86K, 2, 159	Director (38K), Movie type (1.7K)	Movie score (1-10)
News	81K, 2, 176	Source (5.4K), Title (72K)	FB shares (log)
InstEval	73K, 3, 3	Student (2.9K), Teacher (1.1K), Dept (14)	Teacher rating (1-5)
Spotify	28K, 4, 14	Artist (10K), Album (22K), Playlist (2.3K), Subgenre (553)	Danceability (0-1)
UKB-blood	42K, 5, 19	Treatment (1.1K), Operation (2.0K), Diagnosis (2.1K), Cancer type (446), Histology (359)	Triglycerides (mmol/L)

Setup: MLP (2 layers: 10-3 neurons), 5-fold CV, batch=100, early stopping

Results (Mean Test MSE):

Dataset	Ignore	OHE	Embeddings	lme4	LMMNN
Imdb	1.44	–	1.26	0.99	**0.97**
News	3.22	–	1.89	1.80	**1.81**
InstEval	1.77	1.48	1.50	1.45	**1.45**
Spotify	0.015	–	0.016	0.011	**0.009**
UKB-blood	0.88	1.01	1.04	0.88	**0.86**

Real Data: Longitudinal & Repeated Measures

Datasets: 3 longitudinal datasets with varying time series lengths

Dataset	n, q, p	**Structure**	**Target** y
Rossmann	33K, 1.1K, 23	1.1K stores, 25-31 monthly measurements (2013-2015)	Total sales ($100K)
AUimport	125K, 5K, 8	5K commodities, 1-29 yearly measurements (1988-2016)	Total import (log $)
UKB-SBP	528K, 469K, 50	469K patients, 1-4 measurements at different ages (38-83)	Systolic BP ($\times$100)

Setup: Random intercept + slope (+ quadratic for Rossmann/AUimport), two modes

Mode: Random						
Dataset	Ignore	OHE	Embed.	lme4	LSTM	LMMNN
Rossmann	.179 (.01)	.052 (.01)	.052 (.01)	.013 (.00)	.505 (.01)	**.010 (.00)**
AUimport	7.78 (.70)	4.91 (.30)	3.35 (.45)	**0.72 (.01)**	8.44 (.35)	**0.71 (.01)**
UKB SBP	.0321 (.00)	–	.0327 (.00)	.0310 (.00)	–	**.0307(.00)**
Mode: Future						
Rossmann	.215 (.01)	.067 (.01)	.087 (.02)	.026 (.00)	.336 (.00)	**.020 (.00)**
AU Import	7.69 (.48)	5.60 (1.22)	3.70 (.12)	1.77 (.00)	11.7 (1.1)	**1.48 (.02)**
UKB SBP	.0387 (.00)	–	.0396 (.00)	.0383 (.00)	–	**.0379 (.00)**

Summary: LMMNN - Bridging Statistical Rigor and Deep Learning

The Problem

- Modern DNNs assume independence
- Real data has correlations:
  - Categorical (schools, doctors)
  - Temporal (patients over time)
  - Spatial (geographic locations)
- Standard approaches fail or don't scale

The Solution: LMMNN

$$\mathbf{y} = \underbrace{f(\mathbf{X})}_{\text{DNN}} + \underbrace{g(\mathbf{Z})\mathbf{b}}_{\text{Random effects}} + \epsilon$$

- Nonlinear fixed effects via DNNs
- Principled correlation handling via random effects

Key Innovation

- **Training:** Invert small $m \times m$ matrices instead of full $n \times n$
- **Theory:** Block-diagonal (exact or approximate) structure justifies mini-batch approximation
- **Prediction:** One matrix inversion at test time

Empirical Results

- ✓ Outperforms OHE, embeddings, MeNets, LSTM
- ✓ Handles multiple covariance structures
- ✓ Scales to 100K+ observations, 10K+