Pharmacometric Machine Learning: Integrating Neural Networks for Flexible, Advanced Covariate Analysis

Ahmed Elmokadem, Ph.D, Matthew Wiens, M.A, Hillary Husband, Ph.D, Samuel P. Callisto, Ph.D, Kiersten Utsey, Ph.D, Timothy Knab, Ph.D, and Daniel Kirouac, Ph.D
Metrum Research Group, Boston, MA, USA

Abstract

Background: Integrating covariates in traditional pharmacometric modeling involves complex steps and assumptions about covariate-parameter relationships, which may oversimplify or miss important clinical variables impacting pharmacology and response. Deep Compartment Modeling (DCM) addresses these challenges by using artificial neural networks (ANN) to characterize these relationships in a single step, eliminating the need for covariate selection and enabling complex data inputs and non-linearities. We demonstrate DCM in pharmacometric workflows using two open-source platforms: A Julia-based workflow incorporating hierarchical random effects through Bayesian analysis [1], and an R workflow using Keras and TensorFlow for handling larger datasets [2].
Methods: Population Pharmacokinetic (PK) data with clinically relevant covariates was synthesized using a two compartment model to test the DCM frameworks. For the Julia-based workflow, we assessed whether it could identify covariate relationships and quantify interindividual versus residual variability (random effects) from a small dataset (10:20 train:test split). The R workflow was trained on a large (10,000 subject) dataset with non-linear covariates, exploring various network architectures with minimal code using Keras.
Results: The Julia framework (a hierarchical DCM) successfully identified PK model parameters, random effects, and ANN weights while quantifying uncertainty. This was evident from Bayesian model diagnostics and posterior predictive checks (PPCs) which accurately characterized the training and testing of PK datasets. The R workflow effectively captured non-linear covariate relationships, highlighted by Shapley additive explanations (SHAP), while also estimating the Residual Unexplained Variability (sigma) conditional on covariates.
Conclusion: Neural networks can be integrated with traditional pharmacometric models using several free open-source programming languages. Both Julia and R environments are suitable platforms, but there are tradeoffs regarding development speed, built-in capabilities, and documentation. DCM simplifies the covariate modeling process and uncovers complex, non-linear relationships in computationally efficient workflows.

Methods

The Julia Workflow

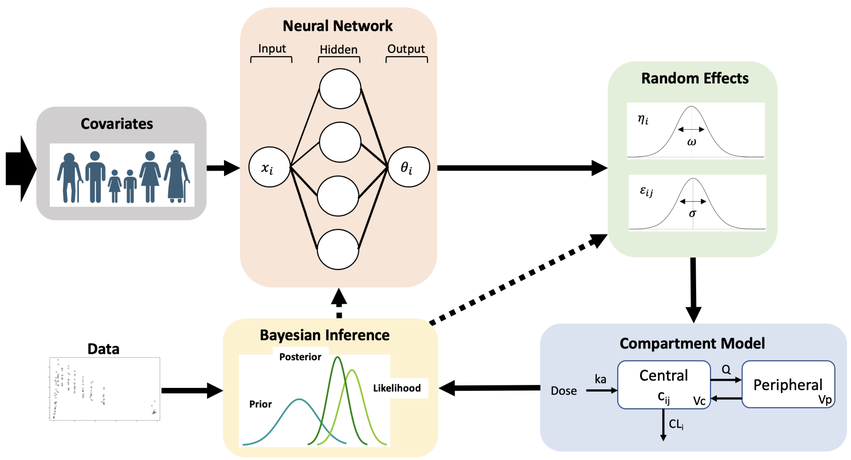

Figure 1. Hierarchical Deep Compartment Modeling (HDCM) workflow. The individual covariates x_i are used as inputs to the ANN that would then output the typical individual parameters θ_i. IIV (η_i) and residual error (ϵ_{ij}) parameters are added to the hierarchical compartmental model structure. The model makes predictions that are compared to the observed data within a Bayesian analysis framework to infer the posterior distributions of ANN parameters as well as the random effects.

Setup

Synthetic PK data for 30 subjects followed a two-compartment model. Data was split into training (n = 10) and test (n = 20) datasets.
Covariates used as ANN inputs: age, weight, EGFR, and albumin. ANN output θ represented PK parameters: CL, V1, Q, V2, and ka. IIV was added to CL such that $CL_i = \theta_i.e^{\eta_i}$ where $\eta_i \sim N(0, \omega^2)$.The model was built using DifferentialEquations.jl [3].

ANN structure

The ANN had 4 input nodes (covariates), 6 hidden nodes, and 5 output nodes (PK parameters). Activation functions: *swish* and *CELU*. The output layer was initialized at the MAP Bayes estimate from an initial naive pooled fit. Flux.jl and DiffEqFlux.jl were used to build the ANN and integrate with ODE solvers.

Statistical model

Likelihood: $c_{ij} \sim N(\hat{c}_{ij}, \sigma^2)$
where c_{ij} = concentration of the drug for individual i at timepoint j and $\hat{c}_{ij}$ = the corresponding prediction.

Prior distributions: $\omega \sim half-Cauchy(0, 0.5)$; $\sigma \sim half-Cauchy(0, 0.5)$; $w \sim N(0, 0.75)$
where ω and σ represent the standard deviations of IIV and residual error, respectively. w represents the ANN weights.
NUTS drew 3 chains of posterior samples (500 warmup, 500 sampling) with a 0.65 acceptance ratio. Bayesian inference was performed using Turing.jl [4].

The R Workflow

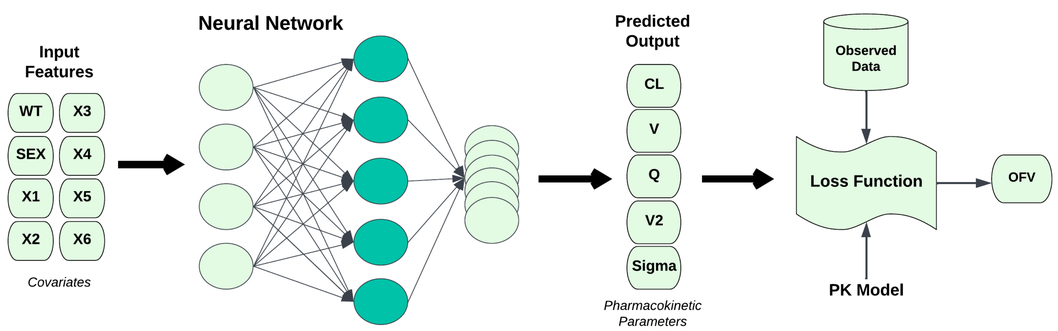

Figure 2. Workflow Representation Diagram. Schematic of the Keras model, with each PK parameter, including RUV, estimated by the model for each individual. Random effects are not included.

- DCMs were implemented in R using the Keras interface for TensorFlow [2]. PK parameters were estimated for each individual in this approach using the observed covariates and drug concentrations as inputs to a neural network
- Alternative network structures were compared to a base neural network defined by one input layer, two hidden layers with nonlinear activation functions, and one output layer of size five (corresponding to the estimated PK parameters: CL, V, V2, Q, and Sigma)
- Run times to fit identical DCMs were compared between GPU and 16-core CPU architectures on the flexible and autoscaling Metworx® platform
- A custom loss function representing the compartmental model was implemented using an ODE solver available in Keras, which can be extended to additional ODE-based population PK structural models

Results

The hierarchical DCM framework was successfully applied in Julia to train an ANN to characterize the functional relationship between the tested covariates and the typical PK parameters. The approach was evaluated using standard Bayesian inference diagnostics including trace and density plots that showed convergence of all chains to the same distributions for the select parameters (Figure 3). The posterior estimates for the select parameters are shown in Table 1 with the MCMC diagnostics: the Gelman-Rubin statistic ($\hat{R}$) showing values < 1.05 and effective sample size (ESS) showing reasonably large values for ESS_{bulk} and ESS_{tail}. Table 2 displays the interpretable posterior predictions for the typical PK parameter values together with the uncertainty around the estimates. Posterior predictive checks (PPCs) showed good characterization of the test data (no overfitting; Figure 4). DCMs optimized in Keras with complex covariate relationships showed an ability to recover the simulated relationships with SHAP, an interpretable ML method (Figure 5). Estimation was substantially faster than the HDCM framework in Julia, and for larger datasets performance was improved with the use of GPUs (Table 3).

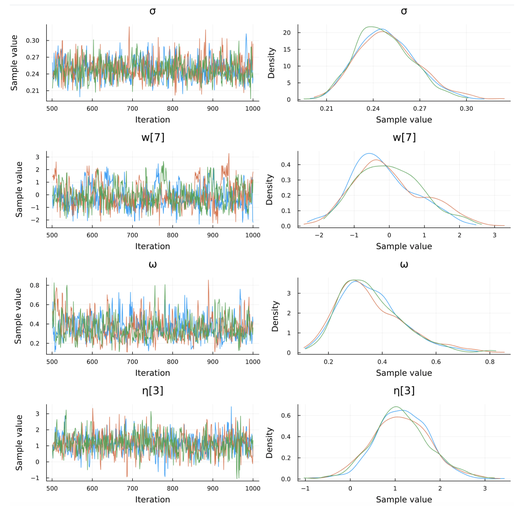

Figure 3. Diagnostic plots. Trace and density plots of select parameters: σ = residual error, w = NN weight, ω = standard deviation for IIV on CL, and η = individual random effect on CL. Number of samples = 500, number of chains = 3.

Parameter	Mean (95% CI)	$\hat{R}$	ESS_{bulk}	ESS_{tail}
σ	0.25 (0.22, 0.29)	1.002	887	1013
w[7]	-0.06 (-1.65, 1.96)	1.004	260	316
ω	0.356 (2.24, 4.65)	1.004	427	529
η[3]	1.14 (-0.07, 2.41)	1.007	819	891

Table 1. Parameter table. CI = credible interval, σ = residual error, w = NN weight, ω = standard deviation for IIV on CL, and η = individual random effect on CL. $\hat{R}$ = Gelman-Rubin statistic, ESS_{bulk} and ESS_{tail} are the effective sample size at bulk and tail of the distribution, respectively.

Parameter	Mean (95% CI)	SD (95% CI)
CL (L/h)	3.64 (2.79, 4.7)	1.25 (0.471, 2.29)
V1 (L)	69.3 (60.4, 76.5)	29.6 (19.7, 38.2)
Q (L/h)	3.3 (2.24, 4.65)	1.16 (0.303, 2.67)
V2 (L)	70.9 (50.9, 106.0)	18.6 (5.44, 41.9)
ka (/h)	2.27 (1.58, 3.61)	0.47 (0.144, 1.21)

Table 2. PK parameter table. SD = standard deviation, CI = credible interval, CL = central clearance, V1 = central volume, Q = intercompartmental clearance, V2 = peripheral volume, ka = absorption rate constant.

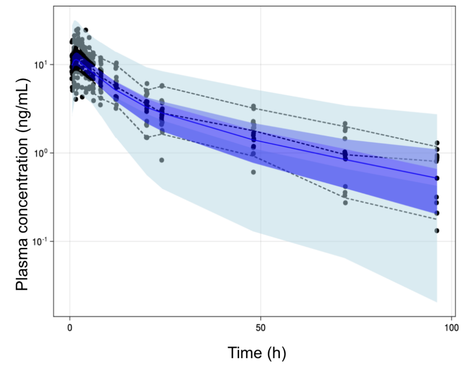

Figure 4. PPCs for test data. Summary PPCs for test data. Black dots = observed data, blue line = median of prediction, blue bands = 95% CI for the predictions around distribution statistics, namely, median, 5%, and 95% of predictions. Dashed lines are median, 5%, and 95% through observed data.

Table 3. Comparison of epoch run times between CPU and GPU architectures.

N	CPU	GPU	Difference	% Difference
N=100	2.12 sec	3.82 sec	1.7 sec	44.5%
N=500	11.57 sec	10.03 sec	-1.54 sec	-15.35%
N=750	16.75 sec	14.32 sec	-2.43 sec	-16.97%
N=1000	17.98 sec	19.43 sec	1.45 sec	7.46%
N=2000	36.39 sec	34.86 sec	-1.53 sec	-4.39%
N=5000	108.52 sec	88.22 sec	-20.3 sec	-23.01%
N=7500	152.95 sec	128.2 sec	-24.75 sec	-19.31%

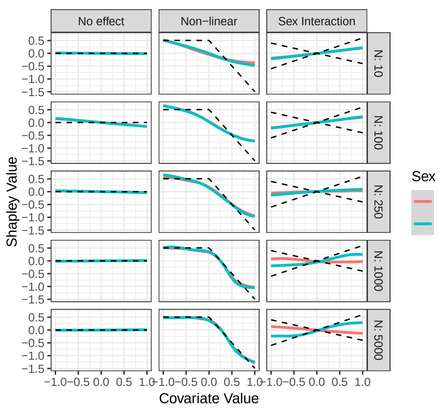

Figure 5. SHAP analysis of covariate effects. Visualization of the estimated covariate relationship for three types of relationships: No effect, a "hockey-stick" non-linear relationship, and a different linear relationship by sex. The NN captures each of these relationships as the sample size increases.

Conclusion

- Demonstrated open-source R & Julia workflows integrating neural networks into pharmacometrics for covariate modeling
- Julia: Bayesian hierarchical modeling with uncertainty quantification
- R: SHAP analysis for nonlinear covariate relationships

References

[1] Bezanson, J., Edelman, A., Karpinski, S. and Shah, V.B. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.* **59** (2017):65–98.
[2] Abadi, M, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015). Software available from tensorflow.org.
[3] Rackauckas, C. and Nie, Q. DifferentialEquations.Jl – A performant and feature-rich ecosystem for solving differential equations in Julia. *J. Open Res. Softw.* **5** (2017):15.
[4] Ge, H., Xu, K. and Ghahramani, Z. Turing: A Language for Flexible Probabilistic Inference. In A. Storkey and F. Perez-Cruz, (eds.) *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, *Proceedings of Machine Learning Research*, volume 84 (PMLR, 2018), pages 1682–1690.

Metrum Research Group Publications/Posters