Supplementary Material

Factorial Data-Driven Inverse Design of Granular Hydrogels for Targeted Therapeutic Release

Anonymous Author(s)
Affiliation
Address
email

1 Additional Text

1.1 Methods

To investigate the transport behavior of drug-laden particles through porous networks formed by granular hydrogels, we developed a coarse-grained (CG) model of the hydrogel scaffold and therapeutic particles. We varied the hydrogel bead diameter r, the intermolecular interaction ε_{TA} between therapeutic particles and the hydrogel, and the hydrogel composition, where we consider two types of hydrogels.

1.1.1 Scaffold geometries

Each hydrogel bead was modeled as a rigid sphere, with 100 mobile therapeutic particles (T) of size σ randomly distributed on its surface to represent encapsulated drug molecules, where the length scale σ corresponds to 1 μm. Hydrogel scaffold geometries were generated using SideFX Houdini following Riley et al. [2023]. A total of 70 geometry files were created, corresponding to n = 10 rigid monodisperse sphere packings for diameters of spheres of 40, 50, 60, 70, 80, 90 and 100 μm, representing randomly packed assemblies. This served as the initial configuation. For the mixture design, each hydrogel bead was randomly assigned as type A or B, with the ratio determined by the parameter ϕ_A. For the partitioned design, the hydrogel was divided into 10 segments along the z-axis (the flow direction). Each segment was assigned a binary value (0 or 1) to distinguish between the two bead subtypes (A and B), generating alternating compositional layers with controlled randomness. This binary pattern later served as an input feature for the model.

Intermolecular interactions between the therapeutic particles and the hydrogel A were modeled using a shifted 12–6 Lennard–Jones (LJ) potential:

$$U_{\text{TA}} = 4\varepsilon_{\text{TA}}[(\frac{\sigma}{x-\Delta})^{12} - (\frac{\sigma}{x-\Delta})^{6}] \quad x < r_c + \Delta \tag{1}$$

where $r_c = 4\sigma$ is the cutoff, ε_{TA} is the affinity strength, which we varied in the range of 1ε to 5ε, where ε is the characteristic energy unit, and $\Delta = (r+1)/2 - 2^{1/6}$ is the shifted amount to ensure the theraputic particle experiencing the excluded volume interaction of hydrogel at r exactly. Intermolecular interactions among therapeutic particles were represented by a Weeks–Chandler–Andersen (WCA) repulsive potential. In hydrogel B, the beads were treated as inert / phantom particles (no intermolecular interaction), allowing therapeutic molecules to diffuse freely through the matrix. This configuration mimics the hydrogel bead with high porosity.

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2025). Do not distribute.

1.1.2 MD Simulations

Molecular dynamics (MD) simulations were performed using LAMMPS [Plimpton, 1995] to study the release dynamics of therapeutic payloads from granular hydrogel scaffolds. Only the motion of therapeutic particles (T) was integrated, with their temperature controlled using a Langevin thermostat with a damping constant of 1.0 τ^{-1}, where $\tau = (m\sigma^2/\varepsilon)^{0.5}$ defines the time unit. Simulations were conducted with a time step of 0.012 for a total of 25,000 steps.

To capture the transport dynamics of particles through the hydrogel scaffolds at a reasonable computational cost, a constant driving force of 5 ε/σ was applied to each therapeutic particle in the downward direction. When a therapeutic particle reached the boundary of the simulation box, it was removed to mimic its exit from the hydrogel system (i.e., release into the contact medium). The number of released particles was recorded every 10 timesteps.

1.1.3 Simulation data processing

Therapeutic release was computed from counts of lost T particles at each timestep. Each cumulative release curve was fit to a Weibull cumulative distribution,

$$F(t) = 1 - \exp\left[-\left(\frac{t}{\alpha}\right)^{\beta}\right] \tag{2}$$

and goodness-of-fit quantified by R^2. In each case, R^2 exceeded 0.9.

1.1.4 Factorial Design of Experiment for Parameter Selection

A two-level, three-factor factorial design experiment was implemented to validate the design parameters and parameter bounds for the inverse design models. Simulations for 8 representative scaffold designs ($n >= 16$ per design) of high and low bead radii, bead–particle interaction potentials, and bead heterogeneity and fit cumulative release data from each simulation to Weibull CDF's. We then modeled each dependent variable—α (alpha) and β (beta)—as functions of the categorical factors `gel_diameter`, `epsA_NP`, and `pctA`, including all main effects and interactions. Ordinary least squares (OLS) regression models were fitted for each response variable using the formula syntax:

$$\text{response} \sim C(\text{gel\_diameter}) * C(\text{epsA\_NP}) * C(\text{pctA}) \tag{3}$$

where the `C()` function denotes categorical encoding of the factors, and the `*` operator specifies the inclusion of all main and interaction effects.

A Type II analysis of variance (ANOVA) was then performed on each fitted model to evaluate the significance of the factors and their interactions on α and β, respectively.

Table 1: ANOVA results for α (256 rows matching factorial design)

Source	**Sum Sq**	**df**	**F**	**PR($> F$)**
C(gel_diameter)	1.06×10^{5}	1	985.01	2.42×10^{-88}
C(epsA_NP)	1.99×10^{7}	1	1.85×10^{5}	0.00
C(pctA)	5.05×10^{6}	1	4.70×10^{4}	1.05×10^{-284}
C(gel_diameter):C(epsA_NP)	1.58×10^{4}	1	147.35	6.35×10^{-27}
C(gel_diameter):C(pctA)	1.27×10^{2}	1	1.19	2.77×10^{-1}
C(epsA_NP):C(pctA)	2.85×10^{6}	1	2.66×10^{4}	3.19×10^{-254}
C(gel_diameter):C(epsA_NP):C(pctA)	2.51×10^{4}	1	233.94	1.20×10^{-37}

Table 2: ANOVA results for β (256 rows matching factorial design)

Source	Sum Sq	df	F	PR($> F$)
C(gel_diameter)	0.0331	1	63.23	6.51×10^{-14}
C(epsA_NP)	3.5007	1	6693.61	1.92×10^{-181}
C(pctA)	1.2339	1	2359.36	1.07×10^{-128}
C(gel_diameter):C(epsA_NP)	0.1405	1	268.68	2.07×10^{-41}
C(gel_diameter):C(pctA)	0.0592	1	113.22	5.01×10^{-22}
C(epsA_NP):C(pctA)	1.5745	1	3010.51	1.04×10^{-140}
C(gel_diameter):C(epsA_NP):C(pctA)	0.0399	1	76.27	3.74×10^{-16}

1.1.5 Factorially vs Randomly Sampled Simulations

We establish two datasets, one factorially sampled and one randomly sampled, to further explore the efficacy of factorial sampling in training inverse design models on limited data sets. The factorially sampled data set is defined by the discrete values specified in Supplemental Table 3, and the randomly sampled data is defined by the ranges of values defined in Supplementary Table 4. The randomly sampled data uses the same r values as the factorial sampled dataset due to the available geometry files, but the ε_{AT} and ϕ_A values can be drawn from any decimal values that fall in the value range. We then generate n=1664 simulations from the combinatorial space of both sets.

Parameter	Factorially Sampled Possible Values
Bead diameter (LJ units)	40, 50, 60, 70 ,80, 90, 100
Interaction potentials (ε_{AT})	1, 3, 5, 10
Proportion Bead A (ϕ_A, %)	25, 75

Table 3: Factorially Sampled Simulation parameter space

Parameter	[Value Range]
Bead diameter (LJ units)	[40, 50, 60, 70, 80, 90, 100]
Interaction potentials (ε_{AT})	[1-10]
Proportion Bead A (ϕ_A, %)	[25-75]

Table 4: Randomly Sampled Simulation parameter space

1.1.6 Forward Model Architectures

Random Packing Forward Model

For the random geometry, the model was trained to predict the cumulative release profile, parameterized using as α and β. The forward model maps the design parameters $\boldsymbol{\theta} = [r, \varepsilon_{AT}, \phi_A] \in \mathbb{R}^3$ to the Weibull release parameters $\boldsymbol{y} = [\alpha, \beta] \in \mathbb{R}^2$. A multilayer perceptron (MLP) architecture with layer normalization was implemented using the PyTorch deep learning framework. The model consisted of an input layer, four hidden layers, and an output layer. Hidden layer sizes were set to 64, 128, 128, and 64 units, respectively. Each hidden layer applied the following sequence of operations: a fully connected (linear) transformation, layer normalization, a LeakyReLU activation function with a negative slope of 0.1, and a dropout layer with a rate of 0.1 to reduce overfitting. The final linear layer projected the last hidden representation to the output dimension. Formally, for an input vector $\mathbf{x}$, the network output $\hat{\mathbf{y}}$ was computed as:

$$\hat{\mathbf{y}} = f(\mathbf{x}) = W_n\, \phi(\mathrm{LN}(W_{n-1}\, \phi(\cdots \mathrm{LN}(W_1 \mathbf{x} + b_1) \cdots))) \tag{4}$$

where W_i and b_i denote the weight matrices and biases of layer i, $\mathrm{LN}(\cdot)$ is layer normalization, $\phi(\cdot)$ is the LeakyReLU activation, and dropout is applied after each activation during training.This architecture was selected as it showed significant performance improvements over a standard linear

regression (Supplemental Figure 1).

Training data was derived from processed simulation data. Both input features and target variables were standardized independently using the z-score normalization scheme implemented by the `StandardScaler` class from the *scikit-learn* library. Specifically, the scaler was fit separately on the training data to compute the mean (μ) and standard deviation (σ) for each feature or target dimension, following the transformation:

$$z = \frac{x - \mu}{\sigma}$$

For the input features and target values, the scalers were defined and fitted as:

```
sc_x = StandardScaler().fit(X_train)
sc_y = StandardScaler().fit(y_train)
```

The fitted scalers were then applied to the validation and test sets to ensure that all data splits were transformed using statistics derived solely from the training set:

```
Xn_train, Xn_val, Xn_test = sc_x.transform(X_train),sc_x.transform(X_val),sc_x.transform(X_test)
yn_train, yn_val, yn_test = sc_y.transform(y_train),sc_y.transform(y_val),sc_y.transform(y_test)
```

This procedure centers each feature (and target) to zero mean and scales it to unit variance according to its own distribution, thereby preserving the internal structure of each variable while preventing information leakage between datasets.

The training, validation, and test data was split by a random seed at 70%, 10%, and 20% respectively. To validate the effectiveness of the factorially sampled dataset vs the randomly sampled, we trained the model on both and saw that the factorially sampled dataset proved to be much more effective in identifying the appropriate weights for mapping the inputs to the output (Supplemental Figure 2).

Layered Packing Forward Model

For the partitioned geometry, an XGBoost multi-output regressor was trained to predict the instantaneous release profile from the interfacial interaction strengths (ε_{AT}) and the layer configuration. The prediction for a single output can be expressed as the sum over K regression trees:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i), \quad f_k \in \mathcal{F}, \tag{5}$$

where $\boldsymbol{x}_i$ is the input feature vector, f_k is the function represented by the k-th regression tree, and $\mathcal{F}$ is the space of regression trees. The initial training on the baseline dataset achieved a masked mean absolute error (MAE) of 13.931 and coefficients of determination (R^2) of 0.947 ± 0.003 across the first five timesteps. Subsequently, Bayesian hyperparameter optimization was performed over 25 candidate configurations with 3-fold cross-validation, resulting in a total of 75 fits. The optimal hyperparameters are summarized in Table 5. Retraining the multi-output XGBoost model using the tuned parameters yielded a masked MAE of 14.899 and improved R^2 values of $\mathbf{0.953 \pm 0.004}$ for the first five timesteps, indicating enhanced predictive stability and consistency across temporal sequences.

1.2 Inverse Design Approach

To identify hydrogel parameters $(\varepsilon_{AT}, \mathbf{z})$ that produce a desired release profile ΔY_{target}, we implement a Bayesian optimization-based inverse design framework. Starting from a randomly chosen initial parameter set, the pre-trained forward model predicts the resulting release profile, and the discrepancy from the target is quantified using the mean squared error (MSE):

Table 5: Optimal hyperparameters and performance metrics for the XGBoost multi-output regressor.

Hyperparameter	Description	Optimal Value
colsample_bytree	Fraction of features sampled per tree	0.508
gamma	Minimum loss reduction required to make a split	0.423
max_depth	Maximum tree depth	3
n_estimators	Number of boosting iterations	571

$$\text{MSE}(\boldsymbol{x}) = \frac{1}{L}\sum_{i=1}^{L}\left(f_{\text{forward}}(\boldsymbol{x})_i - \Delta Y_{\text{target},i}\right)^2, \tag{6}$$

where $\boldsymbol{x} = [\varepsilon_{AT}, \mathbf{z}]$ is the parameter vector, L is the number of timesteps or release points, and f_{forward} is the pre-trained surrogate model (MLP for random geometries or XGBoost for partitioned geometries).

The *search space* consists of:

- Continuous variables: ε_{12}.
- Discrete binary variables: $\mathbf{z} = [z_0, z_1, \ldots, z_{N-1}]$ representing the layer configuration.

Bayesian optimization (implemented using `gp_minimize` from `scikit-optimize`) iteratively proposes new candidate parameters by minimizing the MSE while accounting for the uncertainty in the surrogate model. At each iteration, the forward model evaluates the proposed parameters:

$$\boldsymbol{x}^{(t+1)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \text{Acquisition}(\boldsymbol{x}; \text{MSE}, \sigma_{\text{pred}}), \tag{7}$$

where the acquisition function (e.g., Expected Improvement, Probability of Improvement, or Lower Confidence Bound) balances exploration and exploitation, and σ_{pred} represents the predictive uncertainty.

The optimization proceeds until the specified number of evaluations is reached. The algorithm records the best-performing parameter set:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \text{MSE}(\boldsymbol{x}), \tag{8}$$

and also stores the top-5 solutions ranked by MSE. This approach enables efficient inversion of the forward model, allowing rapid identification of parameter combinations that closely match the target release profile without the need for repeated molecular simulations.

2 Additional Figures

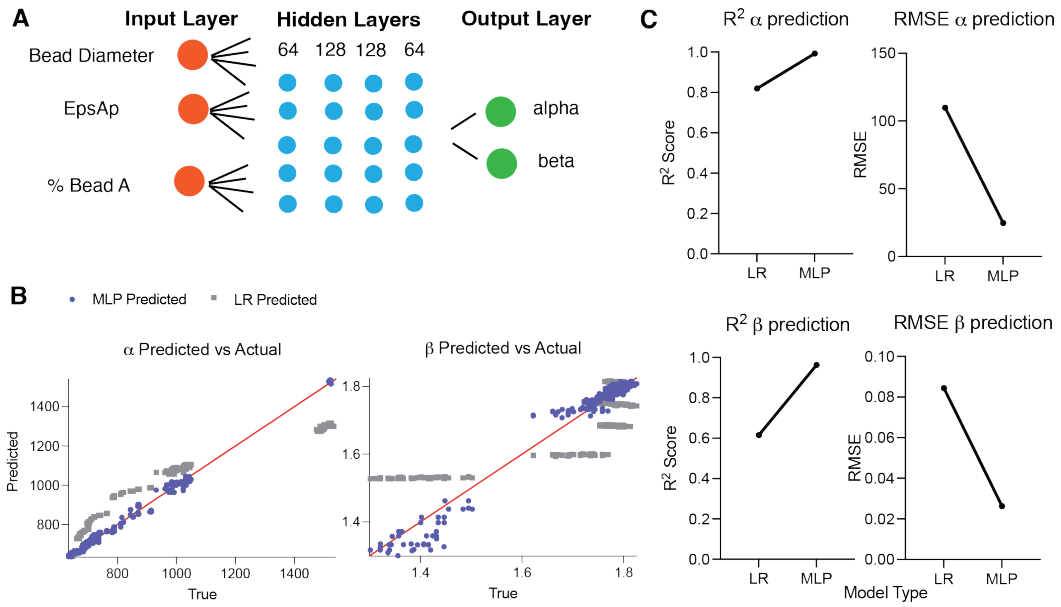

Figure 1: MLP Forward Performance for Randomly Packed Scaffolds vs Linear Regression

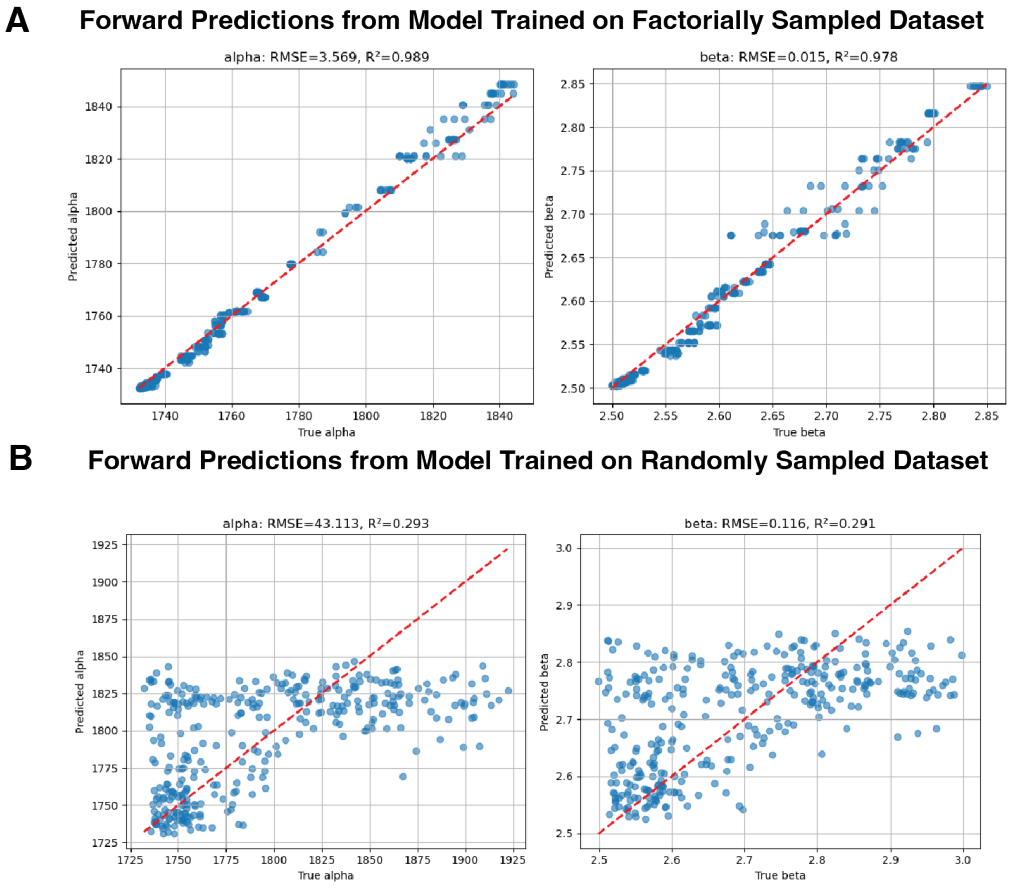

Figure 2: MLP Forward Performance for Factorially vs Randomly Sampled Datasets

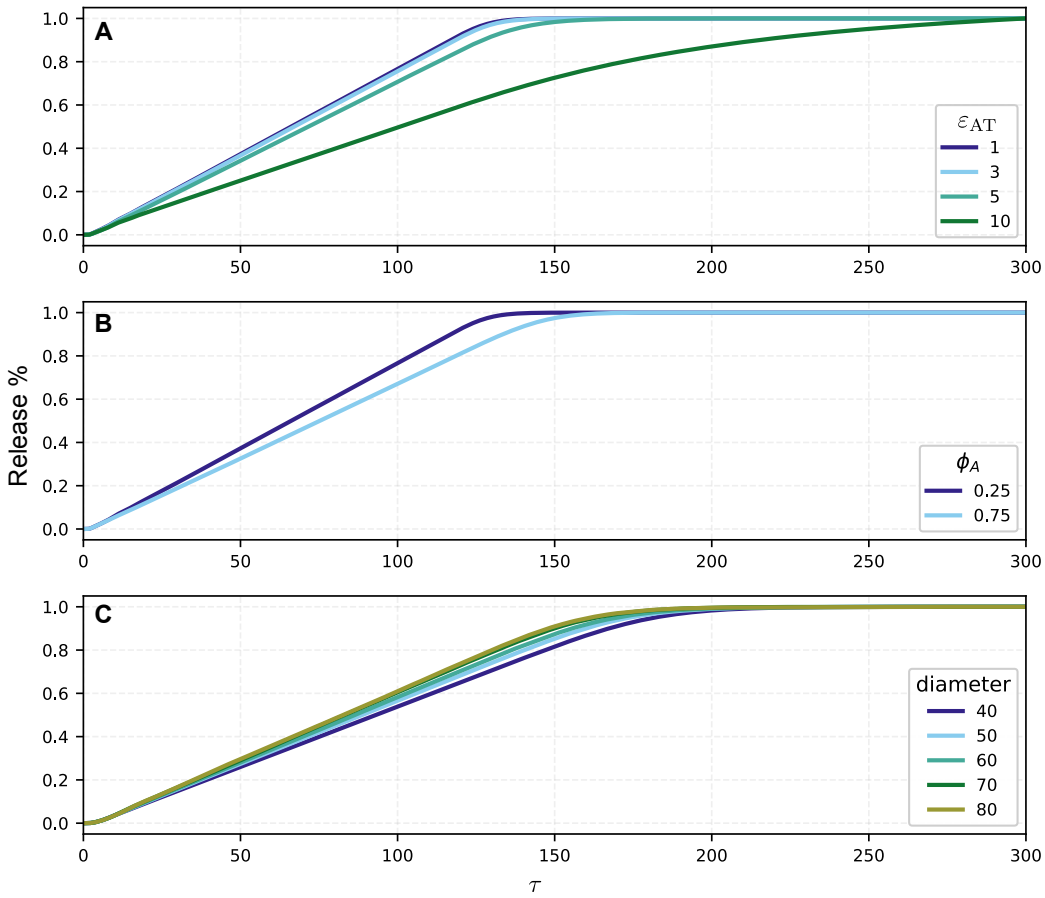

Figure 3: Release profiles of drug-laden hydrogels with heterogeneous random mixture designs under different parameter variations.(A) Effect of varying ε_{12} while fixing $\phi_A = 0.25$ and diameter = 40 μm. (B) Effect of varying ϕ_A while fixing $\varepsilon_{12} = 1.0$ and diameter = 40 μm. (C) Effect of varying diameter while fixing $\varepsilon_{12} = 5.0$ and $\phi_A = 0.75$.

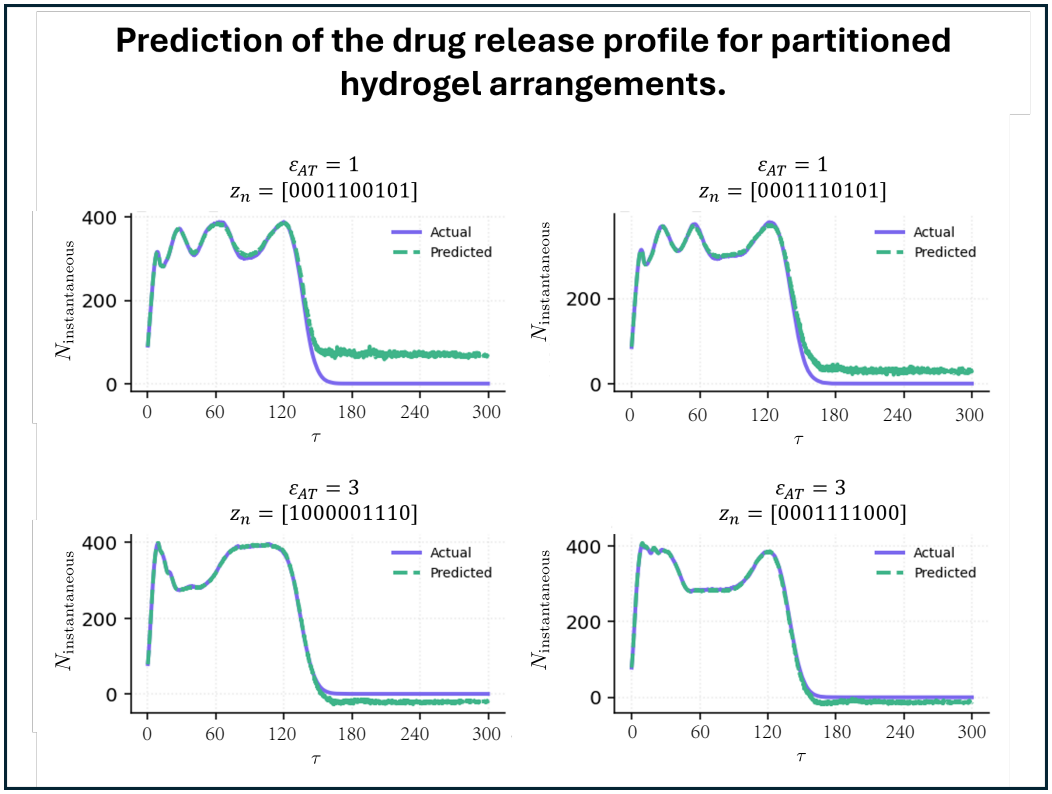

Figure 4: Few examples of the XGBoost model performance

3 Author Contributions

YS lead the project, developed Figures 1 and 2, wrote the manuscript, developed forward model for random mixture designs, and ran the factorial design of experiment under supervision of TS. PA developed the simulation for both the random mixture and partitioned designs, assisted with model development, developed Figure 3, wrote the manuscript, and conducted the literature review under the supervision of GA. JS devised and validated the inverse design framework and developed Figure 4 under the supervision of AJ. TA assisted with literature review and project discussions.

References

Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995.