

Production: Optimizing Cell Culture Processes with Digital Twins and (Bayesian) Hybrid Modeling

NCS2026

23SEP2026

Thomas de Marchin, Jean-François Michiels, Haleh Nekoei, Pierre Lebrun

Disclaimer

This presentation has been prepared by Cencora and is not meant to serve as legal advice and may contain certain marketing statements. Cencora, Inc. strongly encourages the audience to review all available information and to rely on their own experience and expertise in making decisions with regard to the information discussed today. The contents of this presentation are owned by Cencora, and reproduction of the slides used in the presentation is not permitted without consent of Cencora.

MacroTwin: An ATMP research project



WP06: MACROTWIN

Size: 3 partners (UCL, Pharmalex Belgium and Celyntra Therapeutics)

Objective: The main objective of the mini-project is to establish a **scaled iPSC-macrophage manufacturing platform** and create a **Digital Twin** thereof for later use in allogeneic cell therapy product development and clinical trials by Cellistic clients, as well as additional ATMP production processes by Pharmalex clients.

Advanced Therapeutic Medicinal Product (ATMP): specialized class of human medicines rooted in gene therapy, somatic-cell therapy, or tissue engineering.

Results were generated with the assistance of BioWin ASBL and the financial support from the Walloon Region, in accordance with the provisions of the Grant Agreement (Convention 8881 ATMP Thérapie cellulaire).

Modelling approaches for digital twins

Mechanistic models: Based on biological/physical laws.

Strengths: Transparent, reliable in new scenarios

Limitations: May oversimplify complex systems, hard to adjust

Data-driven models: Based on function approximators (linear models, ML,...).

Strengths: Fast, Usually easy to implement

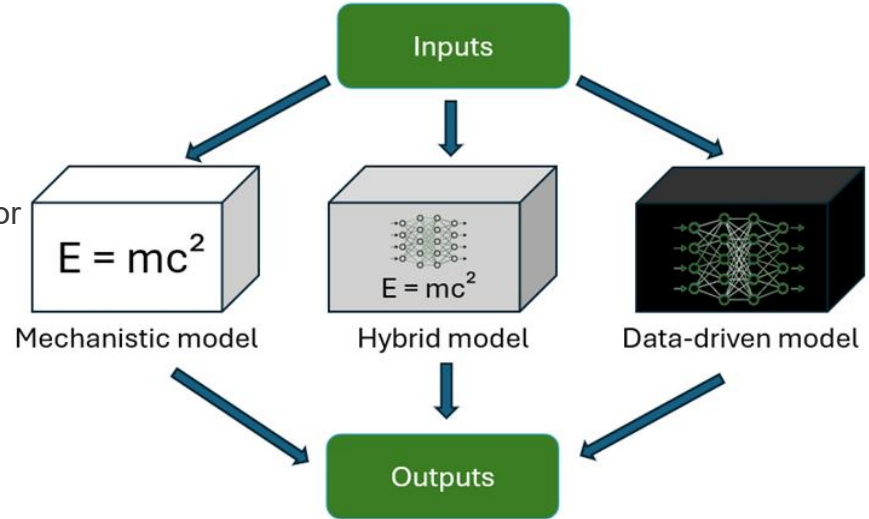
Limitations:

- Poor interpretability: do not mimic real-world biological or physical mechanisms
- Weaker extrapolation with new scenarios
- Must learn everything from data

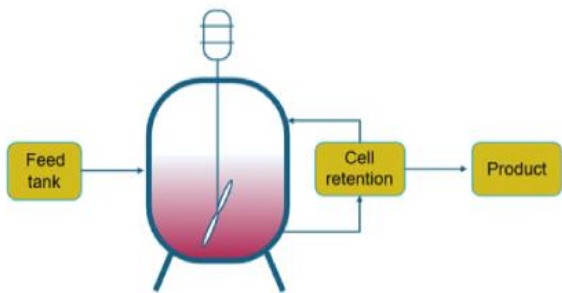
Hybrid models: Combine both approaches

Strengths:

- Combine mechanistic and data-driven modeling: use known physics/biology where well understood, and learn the unknown parts from data
- Improve extrapolation: predictions remain consistent with physical and biological constraints
- Reduce experimental needs: leverage existing mechanistic knowledge rather than learning everything from data



Cell Culture in a Perfusion reactor



● VCD (cells/mL) ● Substrate (S) (g/L) ● Product (P) (g/L) ● Harvest (Q) (g/L)

Perfusion: Cells grown in nutrient-rich medium with product excretion and cell retention

Objective: Model biomass, glucose, and product dynamics in a perfusion cell culture process.

Inputs: Temperature, agitation rate

Hybrid Modeling: Bayesian deep neural network (DNN) with ordinary differential equations (ODE).

- DNN allow to capture nonlinear relationships between process inputs and the ODE parameter (without having to specify a model).
- Bayesian version provide predictions and uncertainty. Essential in pharma and drug development.

Process parameters (inputs)

Temperature

Nutrients concentration

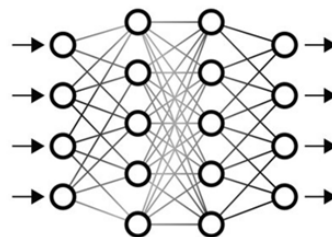
Agitation

...



AI

(Bayesian probabilistic deep neural network)



Predicted ODE inputs

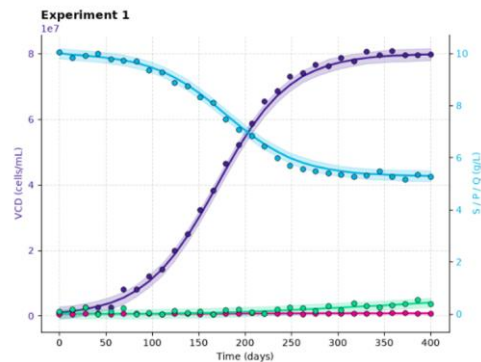
parameters

Rates

Yields

Plateau

Responses



Mechanistic model (ODE)

$$\frac{d(\text{VCD})}{dt} = \mu \cdot \text{VCD} \cdot \frac{S}{S + K_S} \cdot \left(1 - \frac{\text{VCD}}{\text{VCD}_{\max}}\right) - \delta \cdot \text{VCD}$$

$$\frac{dS}{dt} = D_{\text{medium}}(S_0 - S) - \frac{\mu \cdot \text{VCD} \cdot \frac{S}{S + K_S}}{Y_{X/S}} - \frac{\rho \cdot \text{VCD}}{Y_{P/S}}$$

$$\frac{dP}{dt} = \rho \cdot \text{VCD} - D_{\text{medium}} \cdot P$$

$$\frac{dQ}{dt} = D_{\text{medium}} \cdot P$$

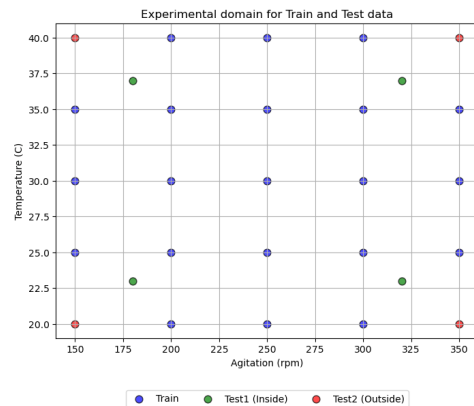
Cell Culture in Perfusion reactor: Validation and Results

Simulation setup

- 29 simulated experiments: 21 training, 4 interpolation test, 4 extrapolation test
- Nonlinear kinetics for growth and product rates: Gaussian temperature effect + sigmoid agitation effect

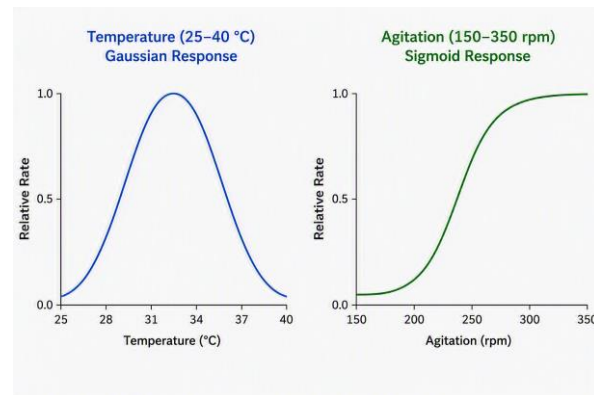
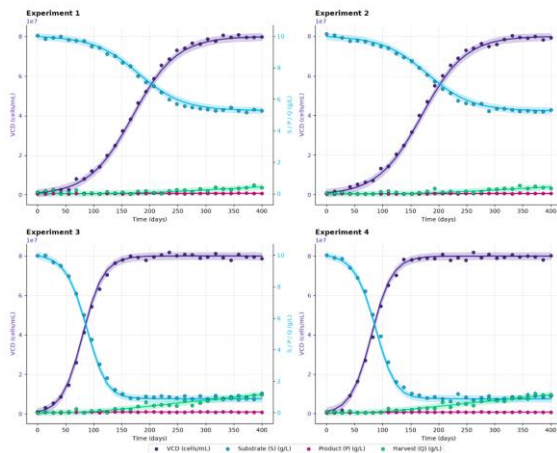
Results

- High agreement between predictions and experimental data
- Uncertainty bands are consistent with observed variability



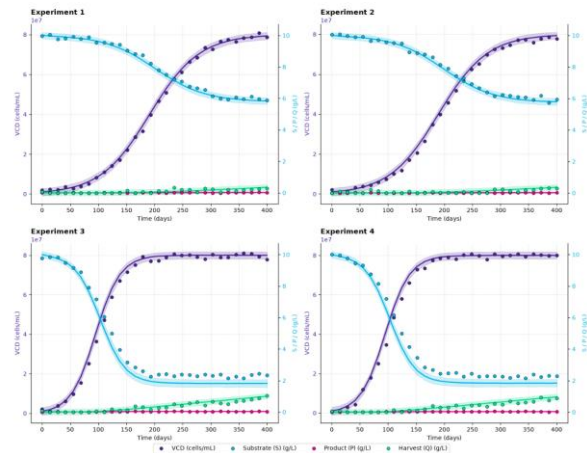
Test inside (MCMC)

Posterior Predictive — All States per Experiment (95% Credible Interval)



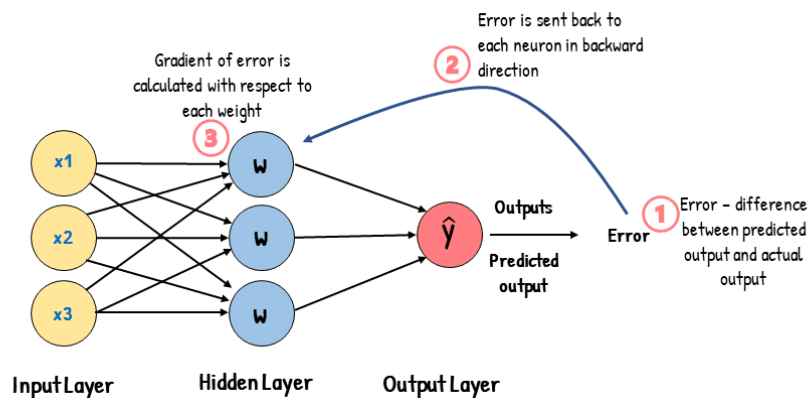
Test outside (MCMC)

Posterior Predictive — All States per Experiment (95% Credible Interval)



Why Deep neural network ?

- Deep Neural Networks (DNNs) excel at capturing complex nonlinear relationships and learning from high-dimensional data.
- Why not use alternatives such as SVMs or Random Forests? DNNs enable end-to-end learning and can be seamlessly combined with differentiable external components (e.g., mechanistic ODE models) within a single optimization framework.
- Gradients propagate through all components of the hybrid model, allowing model parameters to be estimated using gradient descent and backpropagation.

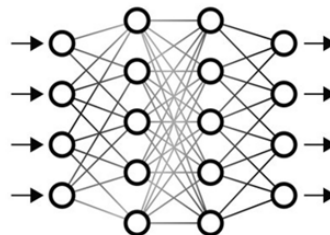


Process parameters (inputs)

Temperature
Nutrients concentration
Agitation
...

AI

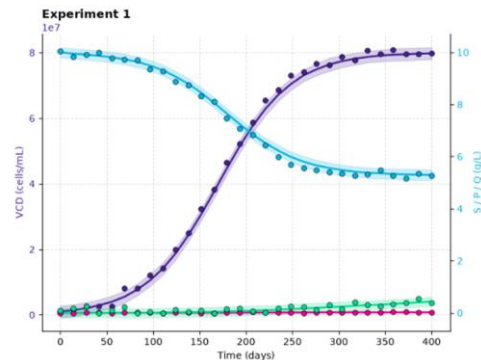
(Bayesian probabilistic
deep neural network)



Predicted ODE inputs

Rates
Yields
Plateau

Responses



Mechanistic model (ODE)

$$\frac{d(\text{VCD})}{dt} = \mu \cdot \text{VCD} \cdot \frac{S}{S + K_S} \cdot \left(1 - \frac{\text{VCD}}{\text{VCD}_{\max}}\right) - \delta \cdot \text{VCD}$$

$$\frac{dS}{dt} = D_{\text{medium}}(S_0 - S) - \frac{\mu \cdot \text{VCD} \cdot \frac{S}{S + K_S}}{Y_{X/S}} - \frac{\rho \cdot \text{VCD}}{Y_{P/S}}$$

$$\frac{dP}{dt} = \rho \cdot \text{VCD} - D_{\text{medium}} \cdot P$$

$$\frac{dQ}{dt} = D_{\text{medium}} \cdot P$$

The non-Bayesian NN model in JAX (python)

1. Input process parameters: agitation and temperature

2. 3 layers with 16 neurons predict ODE parameters: growth rate and product rate

3. Solve the ODE system and generate predictions

Process Parameters → Neural Network → ODE Model → Predicted Trajectories

$$\frac{d(\text{VCD})}{dt} = \mu \cdot \text{VCD} \cdot \frac{S}{S + K_S} \cdot \left(1 - \frac{\text{VCD}}{\text{VCD}_{\max}}\right) - \delta \cdot \text{VCD}$$

$$\frac{dS}{dt} = D_{\text{medium}}(S_0 - S) - \frac{\mu \cdot \text{VCD} \cdot \frac{S}{S + K_S}}{Y_{X/S}} - \frac{\rho \cdot \text{VCD}}{Y_{P/S}}$$

$$\frac{dP}{dt} = \rho \cdot \text{VCD} - D_{\text{medium}} \cdot P$$

$$\frac{dQ}{dt} = D_{\text{medium}} \cdot P$$

```
def forward(params, x, t, dilution_rate, substrate_feed, initial_VCD):
    n_exp, _ = x.shape
```

— NN forward pass —

```
h = jnn.elu(x @ params["fc1_weight"].T + params["fc1_bias"])
```

```
h = jnn.elu(h @ params["fc2_weight"].T + params["fc2_bias"])
```

```
raw_out = h @ params["fc3_weight"].T + params["fc3_bias"]
```

```
growth_rates = jnn.sigmoid(raw_out[:, 0]) * GROWTH_RATE_MAX
```

```
product_rates = jnn.sigmoid(raw_out[:, 1]) * PRODUCT_RATE_MAX
```

— Initial conditions —

```
ic = jnp.stack([jnp.full((n_exp,), initial_VCD), jnp.full((n_exp,),
    substrate_feed), jnp.zeros(n_exp), jnp.zeros(n_exp),], axis=1)
```

— ODE args —

```
ode_args = (jnp.full((n_exp,), dilution_rate), jnp.full((n_exp,),
    substrate_feed), growth_rates, product_rates)
```

— Solve ODEs in parallel —

```
pred_raw = solve_ode_batch(ic, t, ode_args)
```

```
return pred_raw
```

The Bayesian NN Model with NumPyro

The challenge

- A standard neural network gives a single point estimate: no measure of confidence, no uncertainty quantification.
- In pharma, knowing *how confident* the model is matters as much as the prediction itself.

```
# — NN weights and bias —
W1 = numpyro.sample("fc1_weight", dist.Normal(jnp.zeros((H, input_size)), 0.1))
b1 = numpyro.sample("fc1_bias", dist.Normal(jnp.zeros((H,)), 0.1))

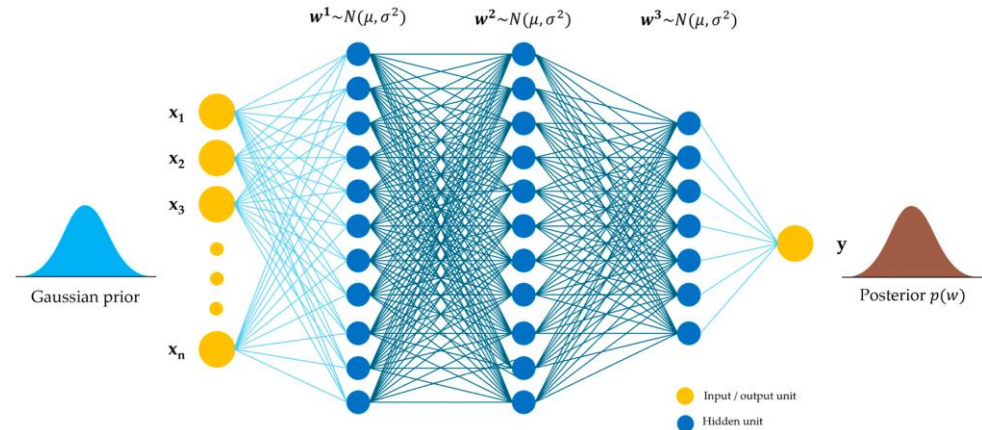
W2 = numpyro.sample("fc2_weight", dist.Normal(jnp.zeros((H, H)), 0.1))
b2 = numpyro.sample("fc2_bias", dist.Normal(jnp.zeros((H,)), 0.1))

W3 = numpyro.sample("fc3_weight", dist.Normal(jnp.zeros((2, H)), 0.1))
b3 = numpyro.sample("fc3_bias", dist.Normal(B3_PRIOR_LOC, 0.5))

# — Observation noise —
obs_sigma_std = numpyro.sample("obs_sigma_std", dist.Normal(sigma_prior_std, 1))
```

NumPyro

- Probabilistic programming library built on JAX.
- Neural network weights and biases are modeled as probability distributions (instead of fixed values), producing posterior distributions that propagate uncertainty through the ODE model and onto predicted trajectories.



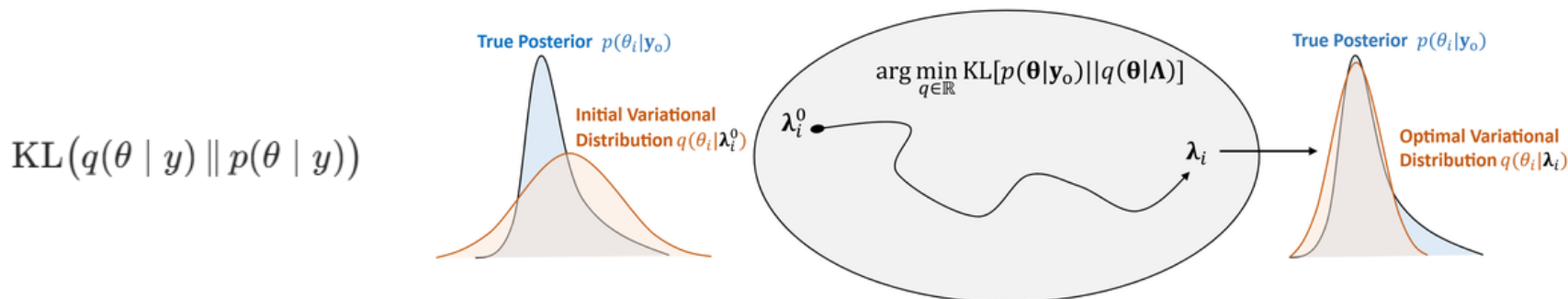
Making Bayesian inference computationally feasible for complex models: SVI

The Challenge

- Bayesian inference aims to estimate the posterior distribution: $p(\theta | y) = \frac{p(y|\theta)p(\theta)}{p(y)}$
- For complex models, exact inference can become computationally expensive: 3h in this case with MCMC

Stochastic Variational Inference (SVI)

- Instead of computing the true posterior, SVI approximates it with a simpler distribution $q(\theta)$, typically a Gaussian.
- The parameters of $q(\theta)$ are optimized to make it as close as possible to the exact posterior $p(\theta | y)$.
- Similarity between the two distributions is measured using the Kullback-Leibler (KL) divergence, transforming Bayesian inference into an optimization problem rather than a sampling problem.

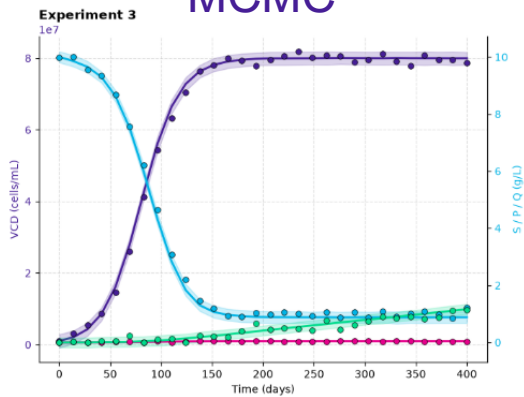


MCMC/SVI comparison

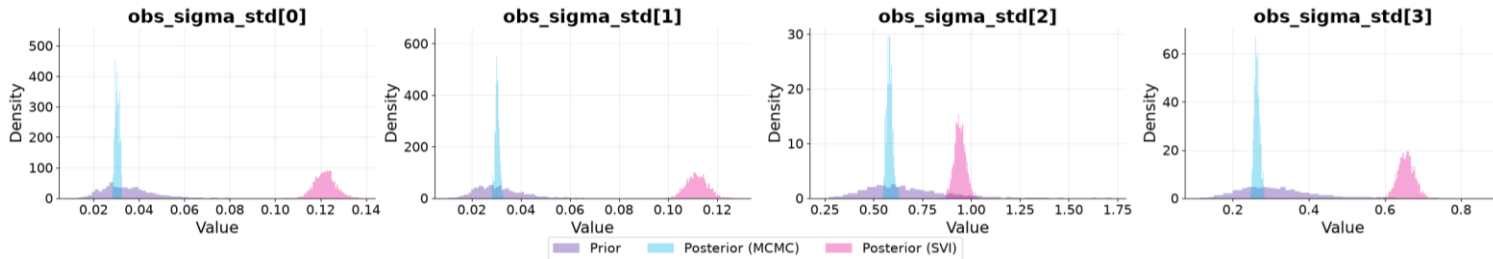
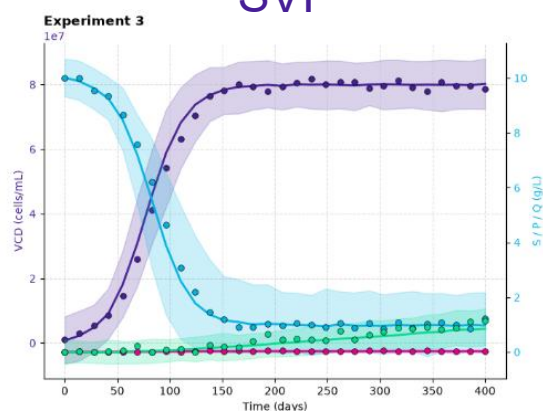
MCMC — 3 hrs / 1,500 iterations → perfect fit, exact posterior samples

SVI — 5 min / 3,000 epochs → good fit, but observational noise is overestimated

MCMC



SVI



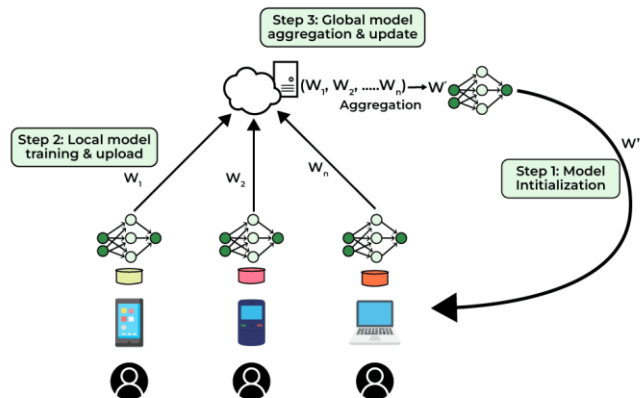
SVI for speed, MCMC for confidence: use SVI to develop models rapidly, then validate with MCMC when needed.

How can these models be deployed?

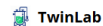
User accessibility: The platform must be highly intuitive and accessible to non-experts, with a user-friendly interface (e.g., Shiny applications).

Lowering training data requirements:

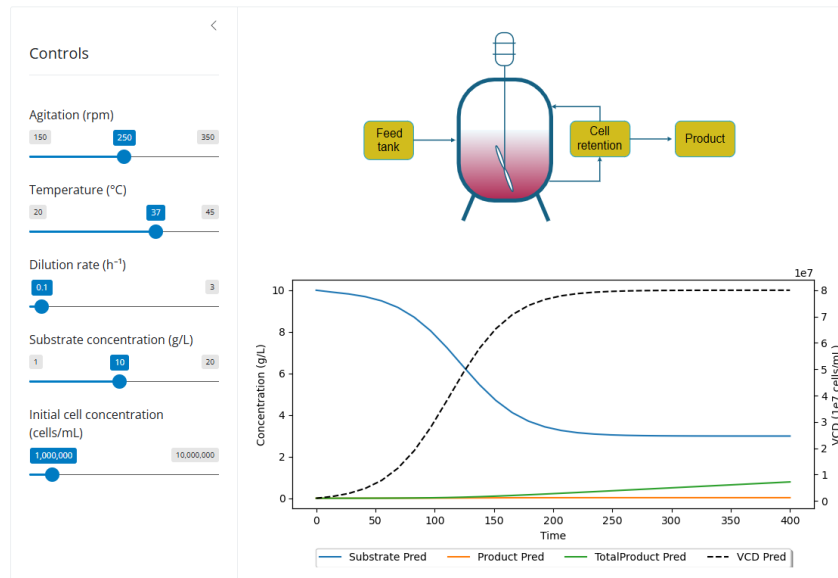
- Hybrid modeling allows modeling Intensified DoE (iDoE): Introduce controlled process shifts within a single run to capture information equivalent to multiple conventional experiments.
- Pre-trained digital twins: Pre-train hybrid models on large aggregated datasets (like LLMs), then fine-tune them using limited local data.
- Federated learning: Enable collaborative model improvement while keeping proprietary experimental data confidential.



<https://www.geeksforgeeks.org/machine-learning/collaborative-learning-federated-learning/>



Simulations Mechanistic Model About



Challenges and next steps

What We Learned

- Hybrid modeling is a promising approach combining process knowledge with flexible data-driven learning
- Bayesian neural networks can provide both predictions and uncertainty
- SVI makes Bayesian hybrid models more computationally feasible than MCMC, although fit was better with MCMC

Challenges

- Selecting neural network hyperparameters remains difficult (number of layers and neurons, activation function, learning rate...) → Use SVI for that step ?
- Selecting appropriate priors for Bayesian neural networks is a complex task requiring further investigation.

Next Steps

- Assess sensitivity to prior assumptions
- Investigate why SVI overestimates observational noise
- Evaluate how intensified DoE can reduce experimental burden
- Explore the deployment strategies

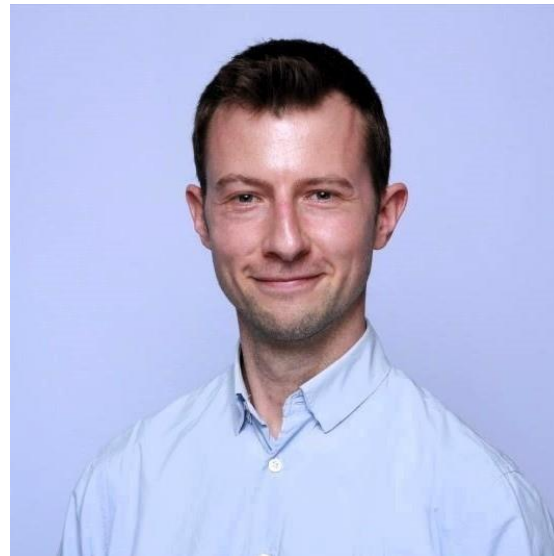
Contact

Thomas de Marchin

Associate Director Statistics & Predictive AI

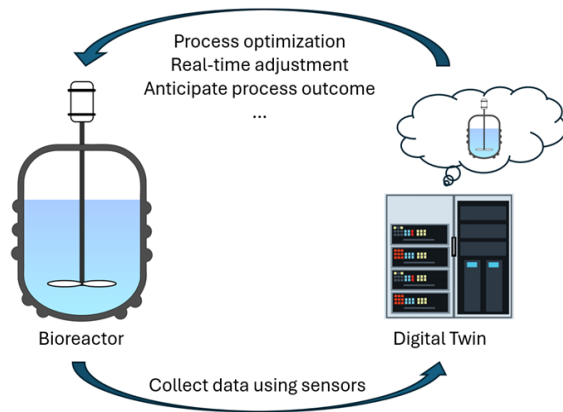
Thomas.deMarchin@cencora.com

<https://www.linkedin.com/in/tdemarchin/>



What is a Digital Twin ?

A digital twin is a dynamic virtual replica of a physical system such as a bioreactor or a chromatographic purification step



Key benefits:

- Real-time process optimization: closed-loop control systems that self-adjust to maintain optimal conditions.
- Process understanding and risk mitigation: simulate "what-if" scenarios to avoid costly production failures.
- Reduces the need for physical experiments by testing hypotheses virtually
- Crucial for personalized medicine and cell and gene therapies: enable adaptable production of small batches tailored to individualized treatments.

The term “digital twin” is broad and often used inconsistently. In practice, it can range from a standalone predictive model to an integrated platform connected to equipment, data streams, and user interfaces.

Stochastic Variational Inference (SVI)

Model: defines the assumptions

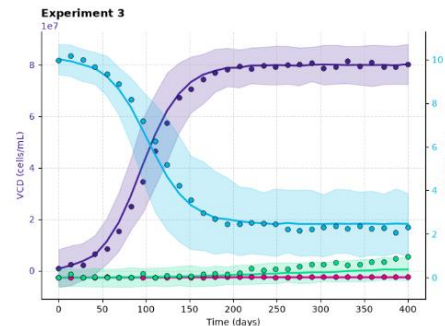
- Specifies the priors, likelihood, and forward model
- Encodes assumptions about how the data are generated
- Remains unchanged during training
- Used only to evaluate the plausibility of parameter values

Guide: learns to match them

- A learnable approximation of the posterior
- Defines a simple distribution $q(\theta)$, typically Gaussian
- Learns the mean and variance of each parameter
- Updated at every optimization step
- Represents the current estimate of posterior uncertainty

Stochastic Variational Inference (SVI): connect the two

- Maximizes the Evidence Lower Bound (ELBO): $\text{ELBO} = \mathbb{E}_q[\log p(y | \theta)] - KL(q(\theta) || p(\theta))$
- At each iteration:
 1. Sample parameters θ from the guide
 2. Evaluate the model likelihood
 3. Compute the ELBO gradient
 4. Update the guide parameters
- After convergence: $q(\theta) \approx p(\theta | y)$
- The learned guide provides both parameter estimates and uncertainty quantification.



SVI: how can we minimize KL divergence if we don't know $p(\theta | y)$?

Minimizing $KL[q(\theta) || p(\theta | y)]$ is mathematically equivalent to maximizing the Evidence Lower BOund (ELBO):

$$\begin{aligned} KL(q(z|x) || p(z|x)) \\ &= \mathbb{E}_q \log \frac{q(z|x)}{p(z|x)} \\ &= \mathbb{E}_q \log q(z|x) - \mathbb{E}_q \log p(z|x) \\ &= \mathbb{E}_q \log q(z|x) - \mathbb{E}_q [\log p(z, x) - \log p(x)] \\ &= \mathbb{E}_q [\log q(z|x) - \log p(z, x)] + \log p(x) \\ &= -\mathbb{E}_q [\log p(z, x) - \log q(z|x)] + \log p(x) \end{aligned}$$


<https://www.cs.toronto.edu/~madras/presentations/svi-tutorial.pdf>

These 3 terms (likelihood, prior and variational distribution) are known, we therefore never need to evaluate the exact posterior directly; optimizing the ELBO implicitly drives the variational posterior toward it.

