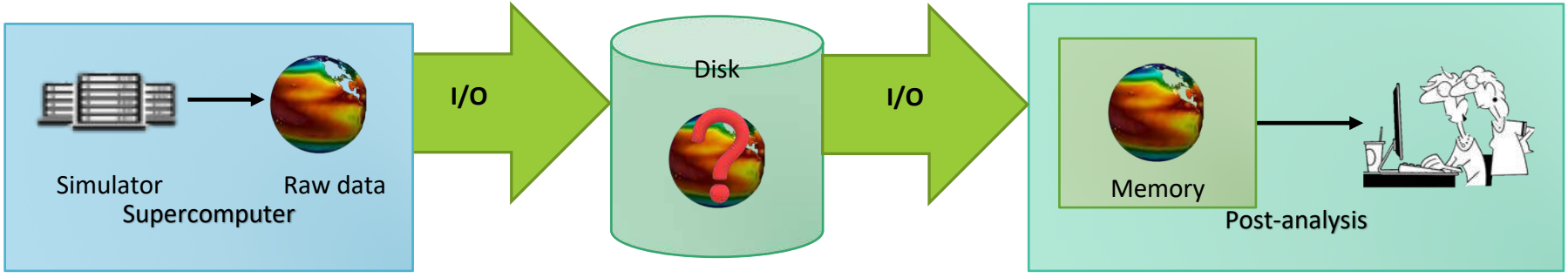
DNN-Assisted Parameter Space Exploration and Visualization for Large Scale Simulations

HAN-WEI SHEN
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
THE OHIO STATE UNIVERSITY

Traditional Visualization Pipeline



- Data are often very large
- Mainly batch mode processing; Interactive exploration is not possible
- Limited parameters to explore

Parameter Space Exploration

- Running large scale simulations is very time and storage consuming
- A physical simulation typically have a huge parameter space
- Ensemble simulations are needed for analyzing the model quality and also identify the uncertainty of the simulation
- It is not possible to exhaust all possible input parameters – time and space prohibitive

DNN Assisted Parameter Space Exploration

- Can we create visualization of the simulation outputs without saving the data?

Or

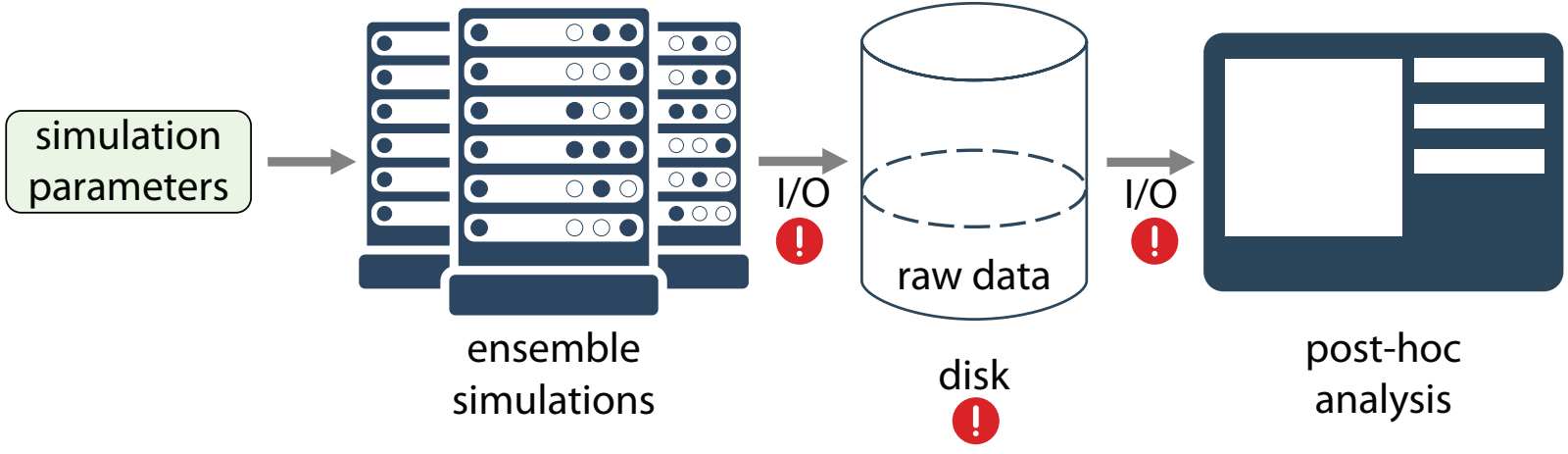
- Can we predict the simulation results without running the simulation?
- Why?
  - Identify important simulation parameters
  - Identify simulation parameter sensitivity
  - Quantify the uncertainty of the simulation models
- Methods:
  - **InSituNet** (IEEE SciVis'19 Best Paper)
  - **NNVA** (IEEE VAST'19 Best Paper Honorable Mention)

VIS2019

InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations

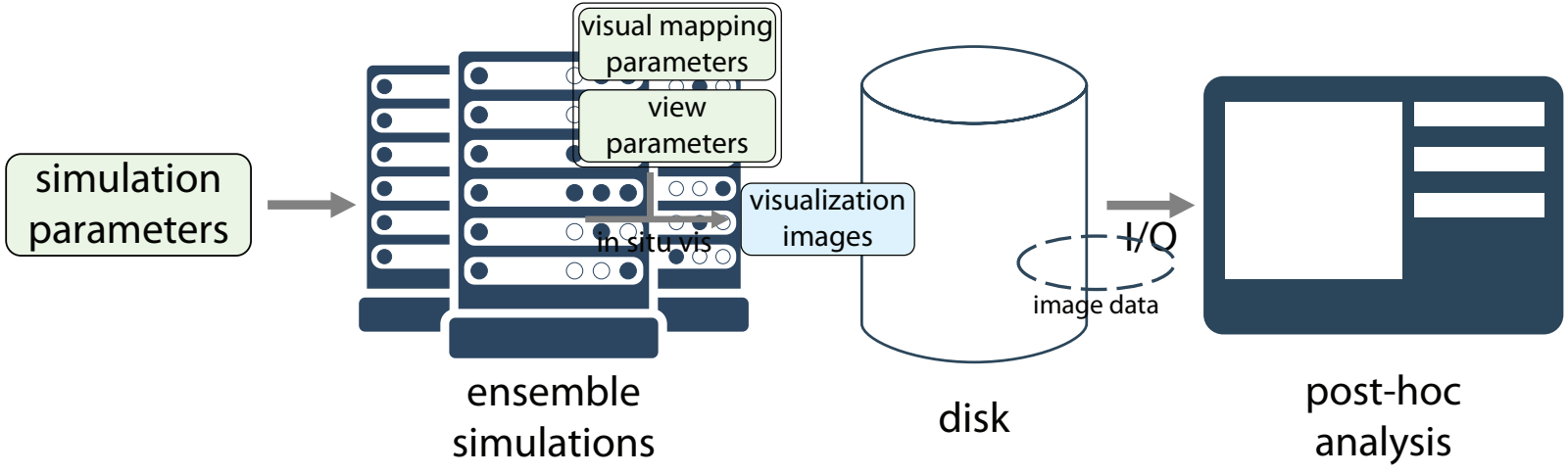
Introduction

- Ensemble data analysis workflow
- Issues
  - I/O bottleneck and storage overhead



Introduction

- In situ visualization
  - Generating visualizations at simulation time
  - Storing images for post-hoc analysis

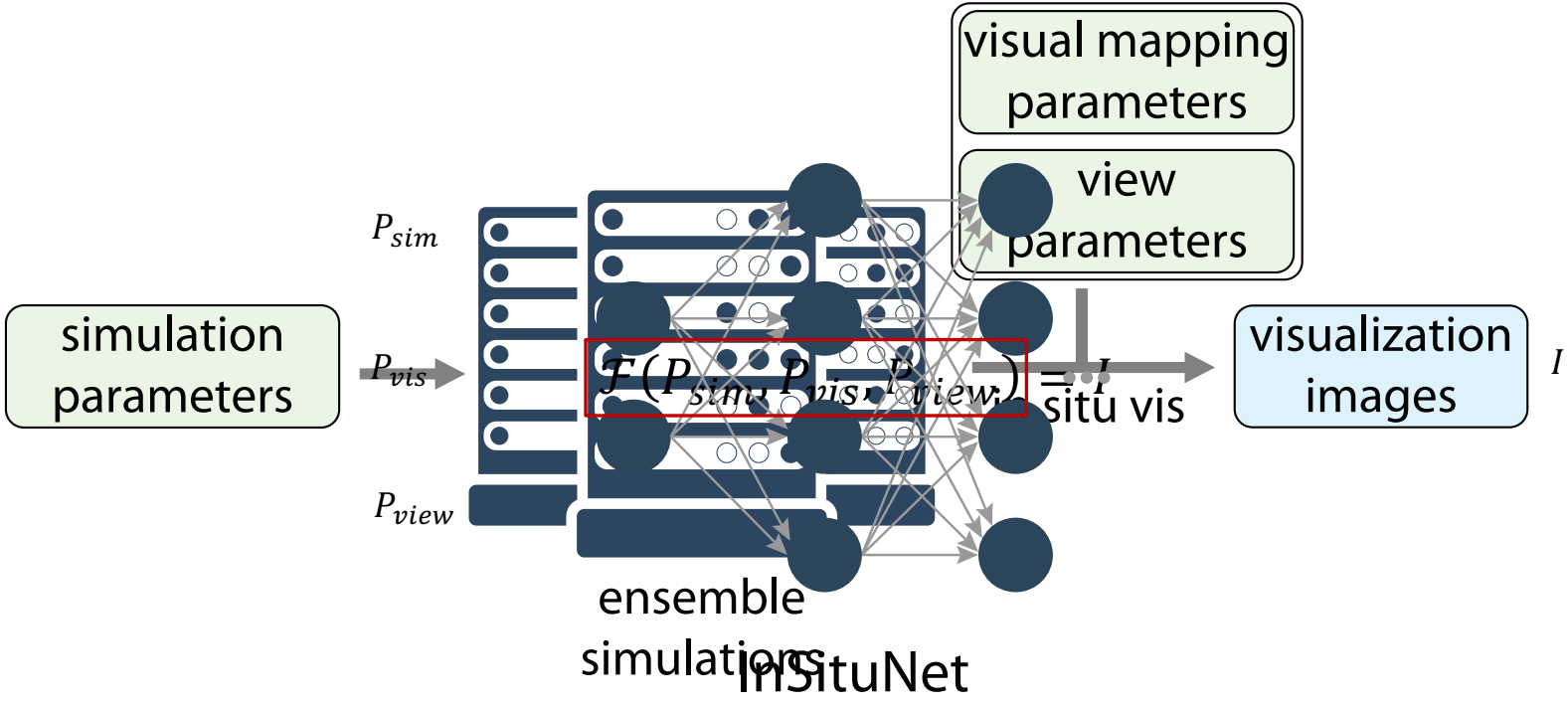


Introduction

- Challenges
  - Limiting the flexibility of post-hoc exploration and analysis
    - Raw data are no longer available
  - Incapable of exploring the simulation parameters
    - Expensive simulations need to be conducted for new parameter settings

- Our Approach
  - Studying how the parameters influence the visualization results
  - Predicting visualization results for new parameter settings

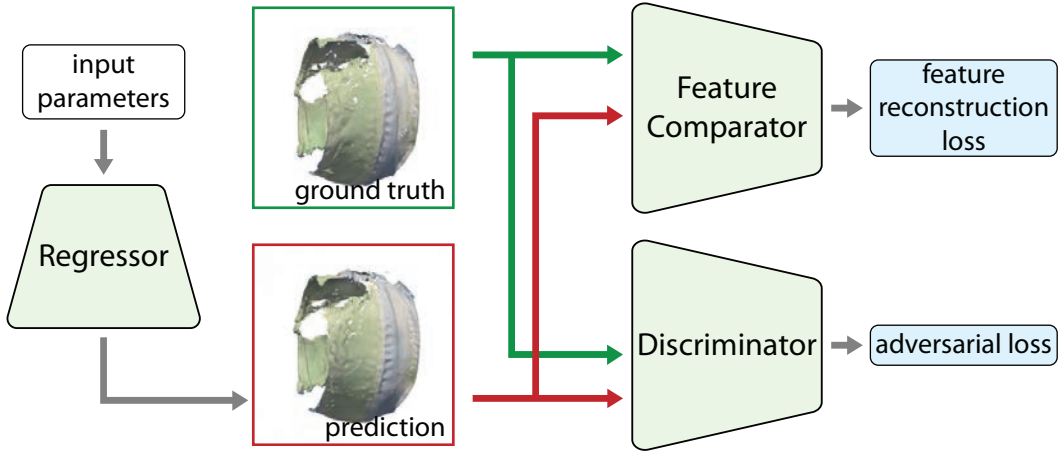
Approach Overview



A deep neural network that models the function $\mathcal{F}$

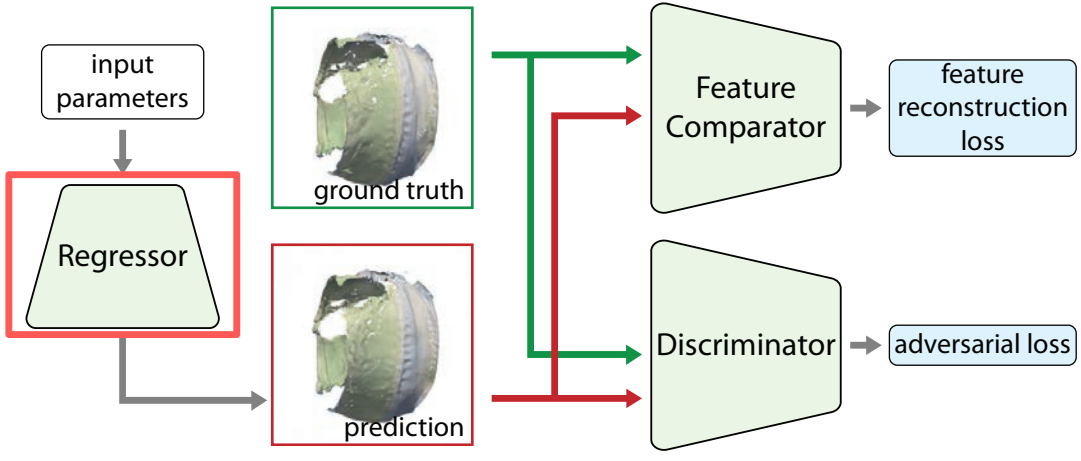
Design of InSituNet

- Three subnetworks and two losses
  - *Regressor* (mapping parameters to prediction images)
  - *Feature comparator* (computing *feature reconstruction loss*)
  - *Discriminator* (computing *adversarial loss*)



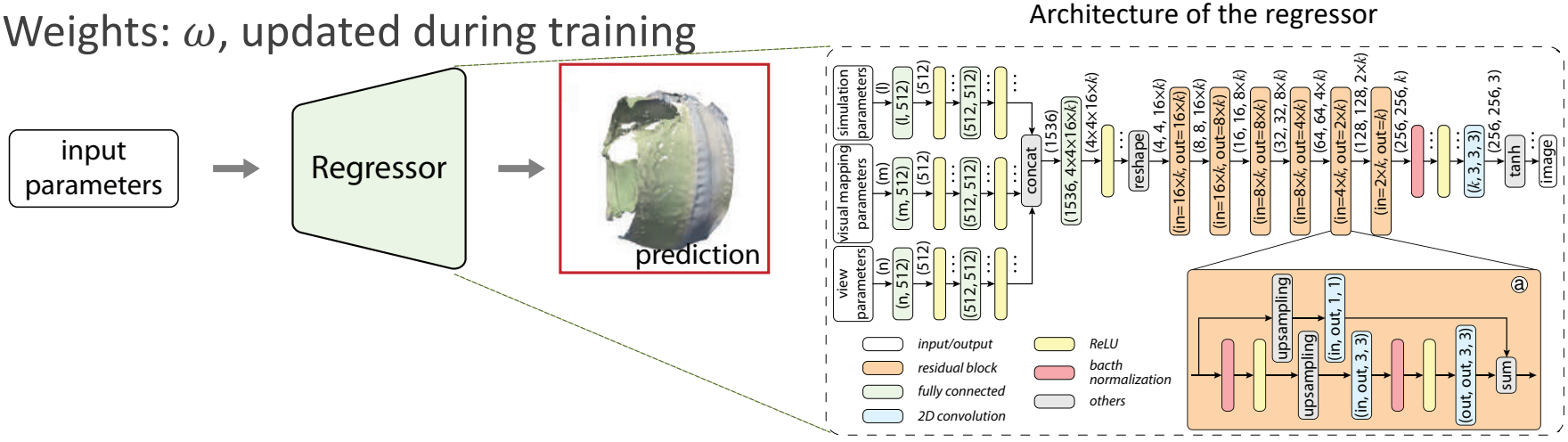
Design of InSituNet

- Three subnetworks and two losses
  - ***Regressor*** **(mapping parameters to prediction images)**
  - *Feature comparator* (computing *feature reconstruction loss*)
  - *Discriminator* (computing *adversarial loss*)

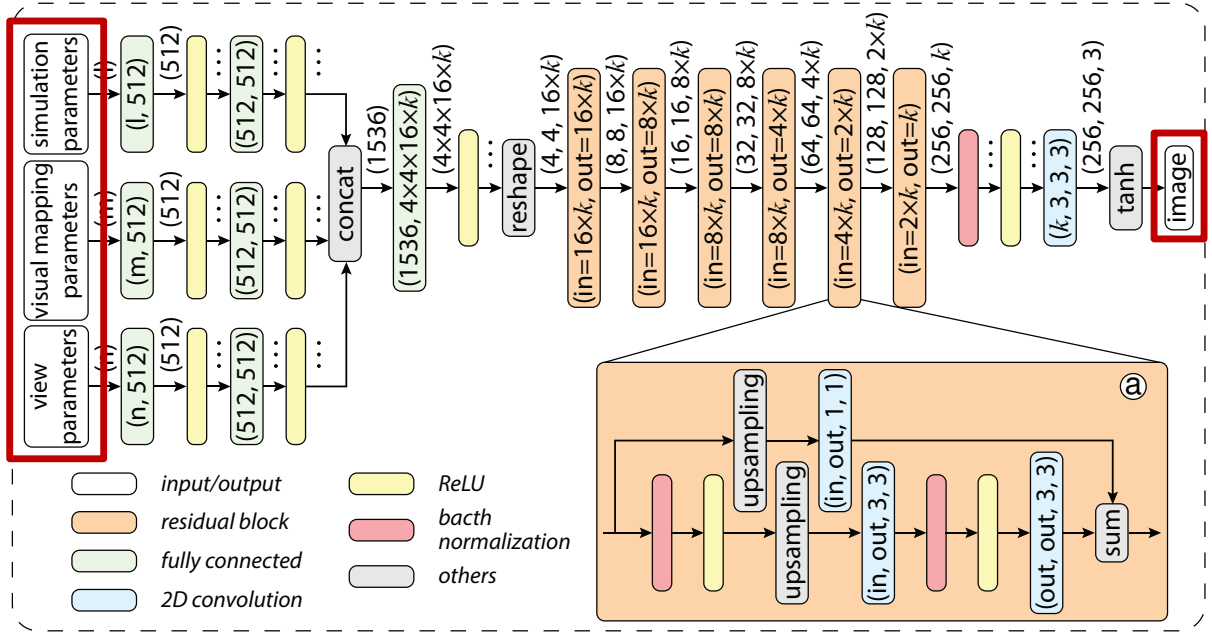


Regressor R_ω

- A convolutional neural network (CNN)
  - Input: parameters P_{sim}, P_{vis}, and P_{view}
  - Output: prediction image $\hat{I}$
  - Weights: ω, updated during training

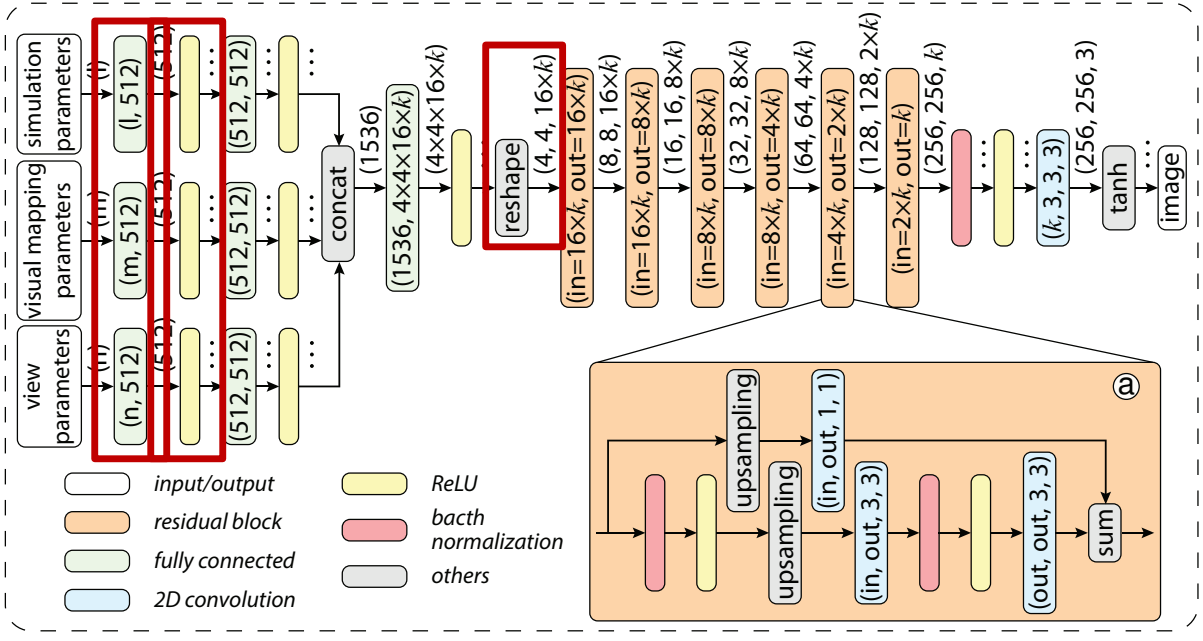


Regressor R_ω



- A convolutional neural network (CNN)
  - Input: simulation, visual mapping, view parameters
  - Output: prediction image
  - Weights ω: weights collected from all layers

Regressor R_ω

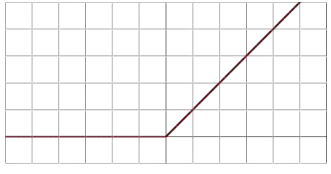


- Fully connected layer
  - Input: 1D vector $\boldsymbol{x} \in \mathbb{R}^m$
  - Output: 1D vector $\boldsymbol{z} \in \mathbb{R}^n$
  - Weights: matrix $\boldsymbol{w} \in \mathbb{R}^{m \times n}$

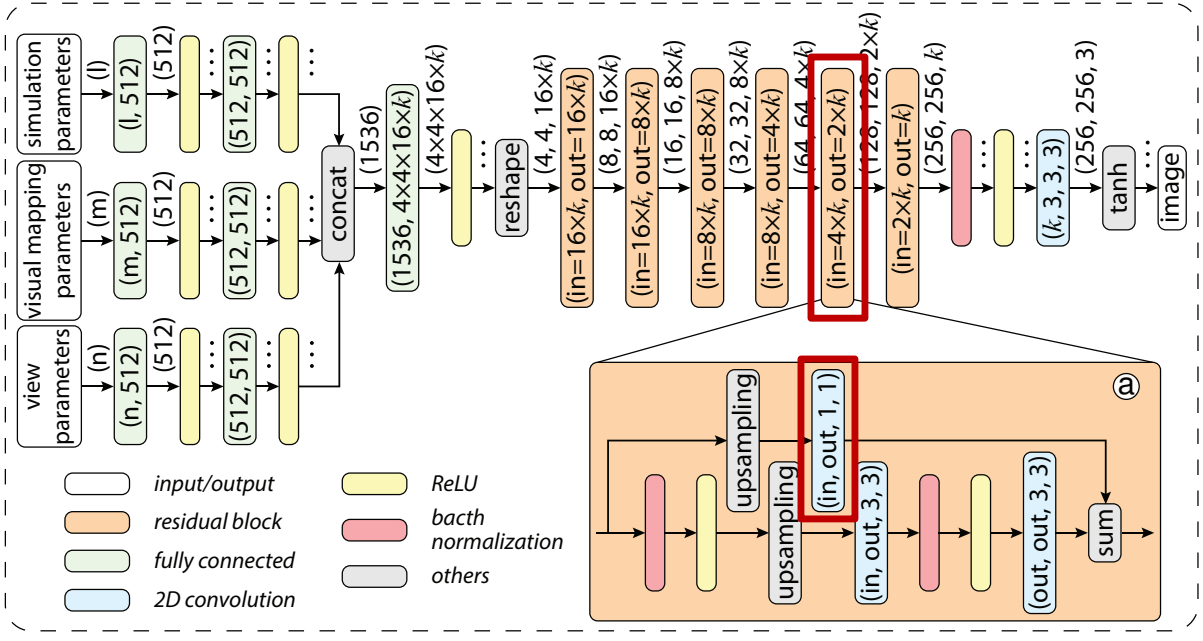
$$\boldsymbol{z} = \boldsymbol{w}\boldsymbol{x}$$

- Activation function
  - Rectified Linear Units (ReLU)

$$\boldsymbol{z}' = \max(0, \boldsymbol{z})$$



Regressor R_ω



- 2D Convolutional Layer
  - Input: tensor $\boldsymbol{x} \in \mathbb{R}^{h \times w \times c}$
  - Output: tensor $\boldsymbol{z} \in \mathbb{R}^{h \times w \times c'}$
  - Weights: kernel $\boldsymbol{w} \in \mathbb{R}^{k\_w \times k\_h \times c \times c'}$

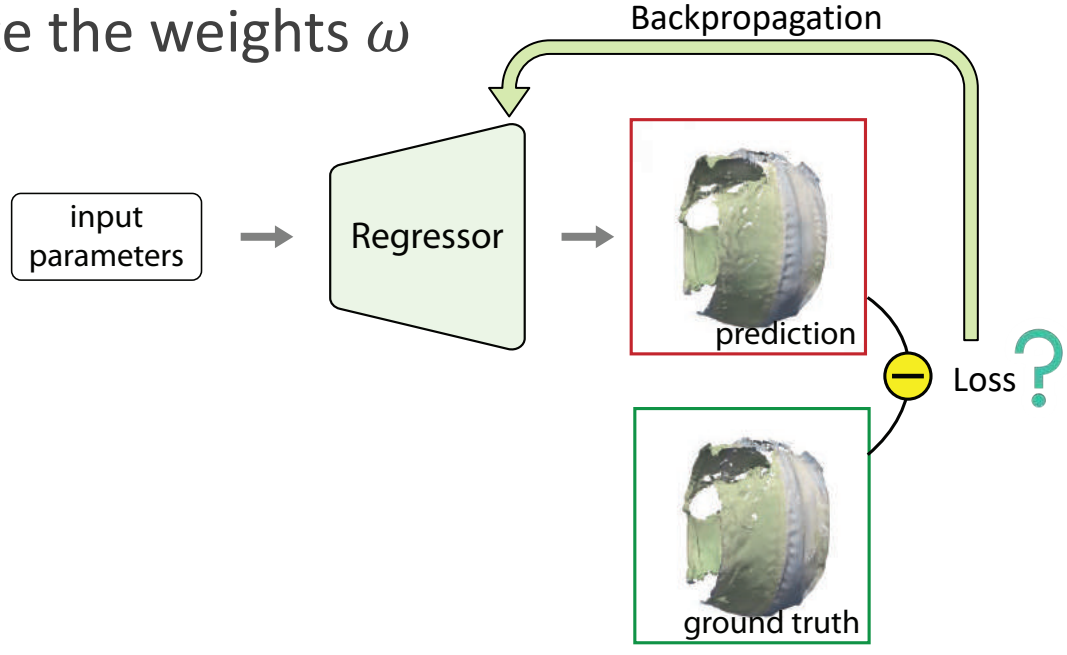
$$\boldsymbol{z} = \boldsymbol{w} * \boldsymbol{x}$$

- Residual block[1]
  - Adding input to the output of convolutional layers

[1]K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

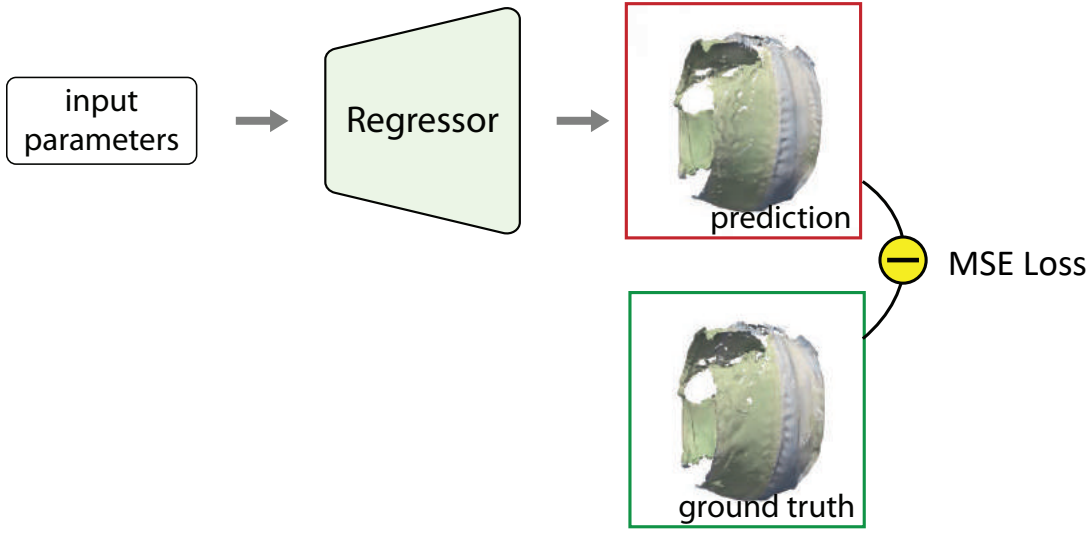
Loss Function

- Difference between the prediction and the ground truth
- Used to update the weights ω



Loss Function - Straightforward Approach

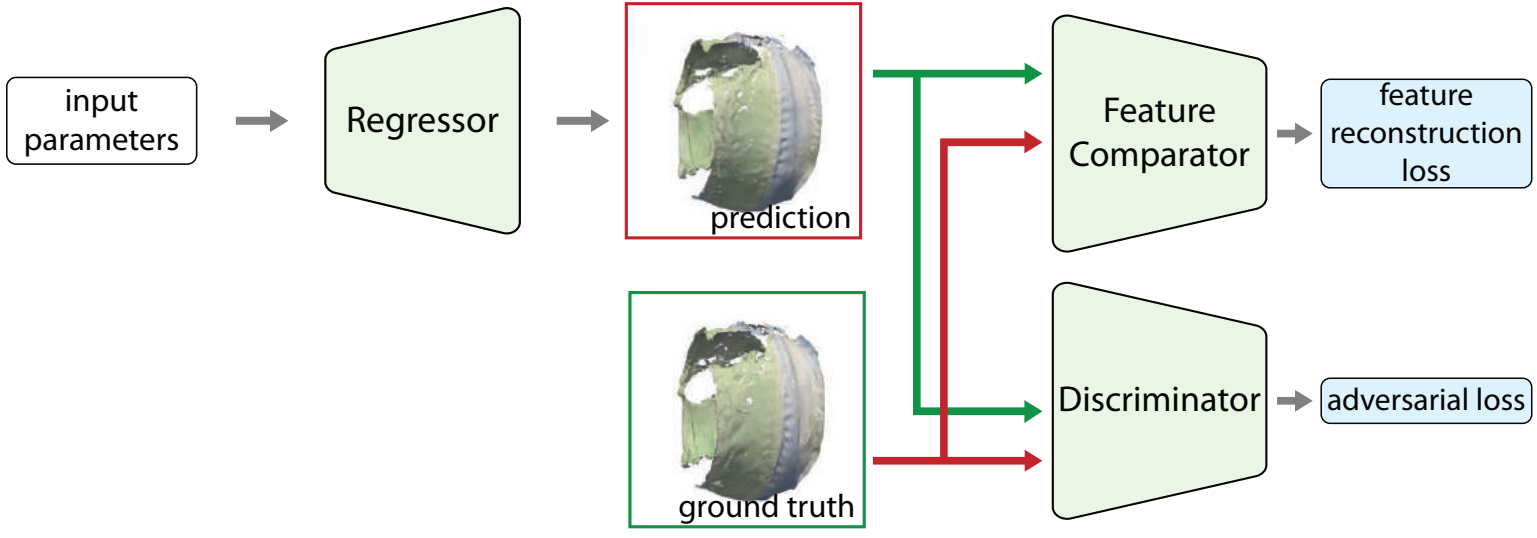
- Pixel wise loss functions
  - Example: mean squared error loss (MSE loss $\mathcal{L}_{mse}$)
  - Issue: blurry prediction images



Blurry image generated by a regressor trained with the MSE loss

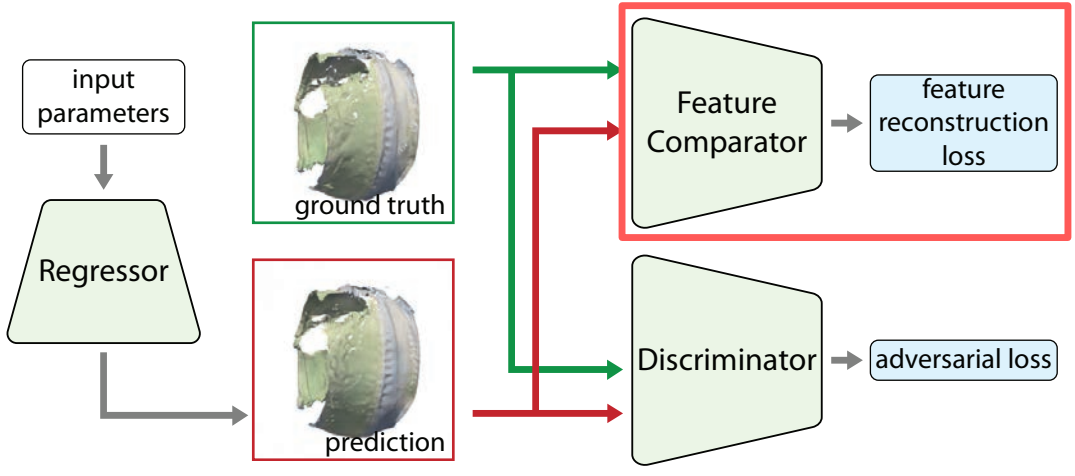
Loss Function - Our Approach

- Combining two loss functions defined by two subnetworks
  - Feature comparator -> feature reconstruction loss $\mathcal{L}_{feat}$
  - Discriminator -> adversarial loss $\mathcal{L}_{adv}$



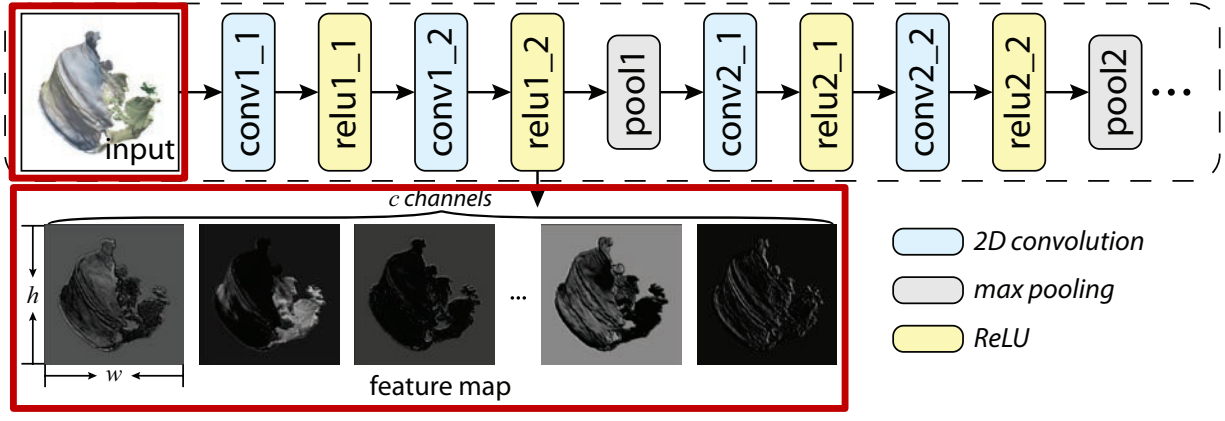
Design of InSituNet

- Three subnetworks and two losses
  - *Regressor* (mapping parameters to prediction images)
  - ***Feature comparator*** **(computing *feature reconstruction loss*)**
  - *Discriminator* (computing *adversarial loss*)



Feature Comparator F

- A pretrained CNN (e.g., VGG-19[1])
  - Input: image I
  - Output: feature map $F^l(I) \in \mathbb{R}^{h \times w \times c}$ of an intermedia layer l



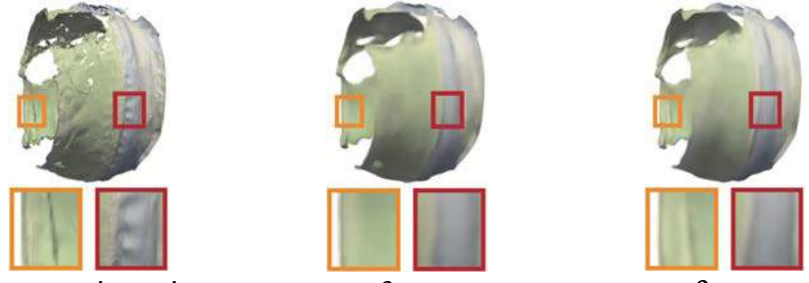
[1]K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Proceedings of International Conference on Learning Representations, 2015.

Feature Reconstruction Loss $\mathcal{L}_{feat}$

- Definition
  - MSE loss between the *feature maps* of the prediction and the ground truth
  - Given a batch of ground truth images $I_{0:b-1}$ and predictions $\hat{I}_{0:b-1}$

$$\mathcal{L}_{feat}^{F,l} = \frac{1}{hwcb}\sum_{i=0}^{b-1}\left\|F^l(I_i) - F^l(\hat{I}_i)\right\|_2^2$$

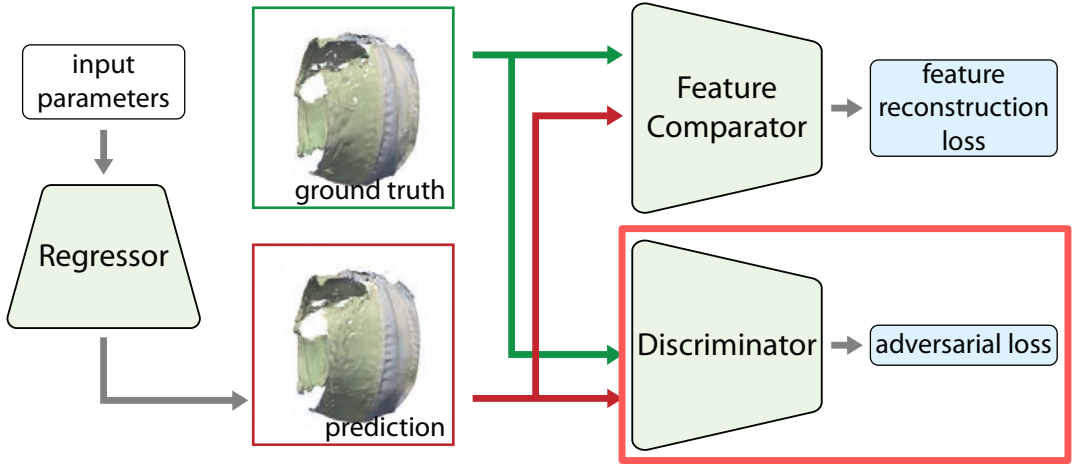
- Benefits
  - Making the regressor to generate images sharing similar features with the ground truth, which leads to images with sharper features



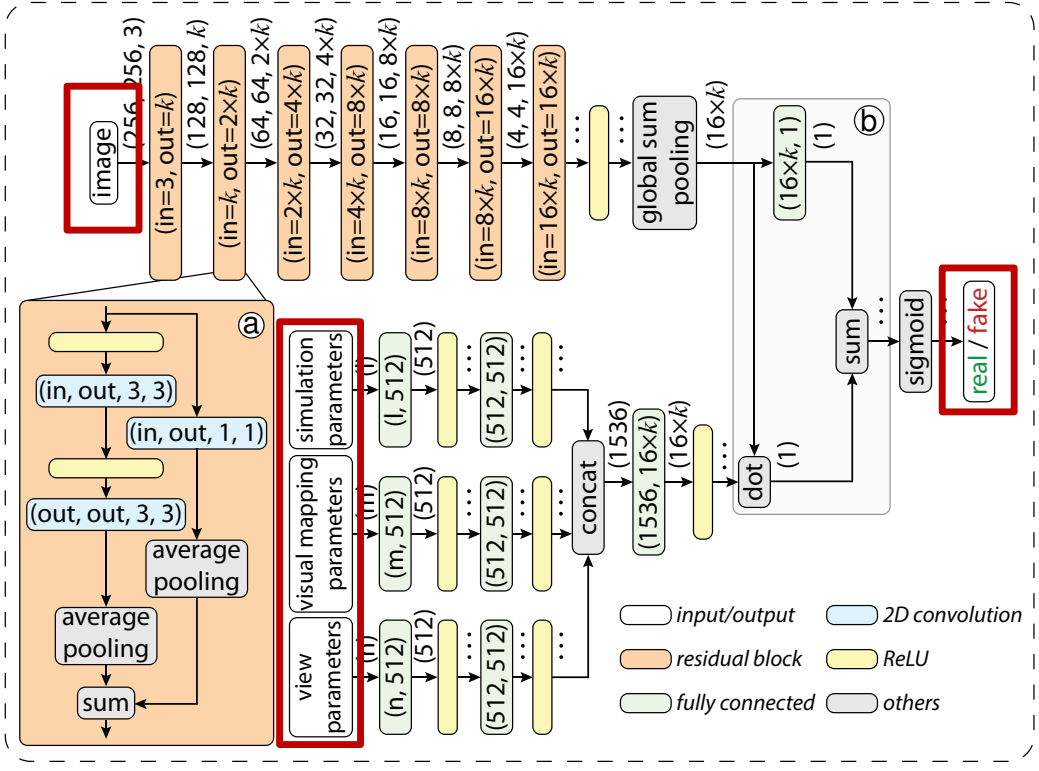
ground truth $\mathcal{L}_{mse}$ $\mathcal{L}_{feat}$

Design of InSituNet

- Three subnetworks and two losses
  - *Regressor* (mapping parameters to prediction images)
  - *Feature comparator* (computing *feature reconstruction loss*)
  - ***Discriminator*** **(computing *adversarial loss*)**



Discriminator D_υ



- A binary classifier (CNN) with weights υ
- Trained to differentiate prediction (fake) and ground truth (real) images
  - Input: real/fake image I and the input parameters P
  - Output: likelihood value $D_\upsilon(I, P) \in [0,1]$
    - 1 means real
    - 0 means fake

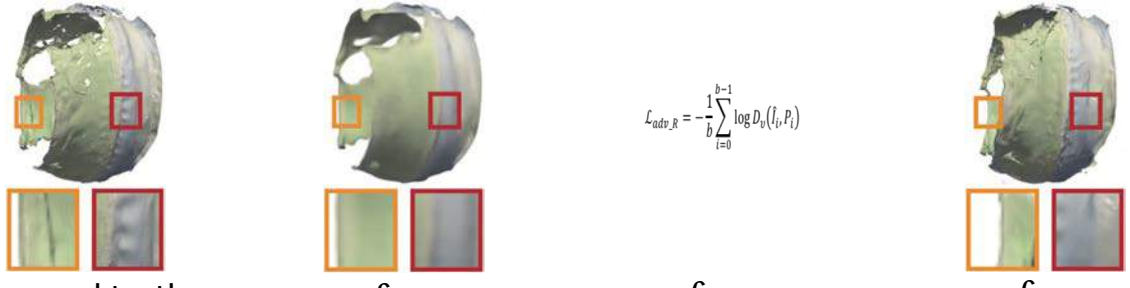
Adversarial Loss $\mathcal{L}_{adv}$

- Regressor and discriminator are trained in an *adversarial* manner[1]
- Given a batch of real images $I_{0:b-1}$, fake images $\hat{I}_{0:b-1}$, and parameter settings $P_{0:b-1}$
  - Regressor is trying to fool discriminator by minimizing

$$\mathcal{L}_{adv\_R} = -\frac{1}{b}\sum_{i=0}^{b-1} \log D_v(\hat{I}_i, P_i)$$

  - Discriminator is trying to differentiate real and fake images by minimizing

$$\mathcal{L}_{adv\_D} = -\frac{1}{b}\sum_{i=0}^{b-1} \left(\log D_v(I_i, P_i) + log\left(1 - D_v(\hat{I}_i, P_i)\right)\right)$$



ground truth $\mathcal{L}_{mse}$ $\mathcal{L}_{feat}$ $\mathcal{L}_{adv}$

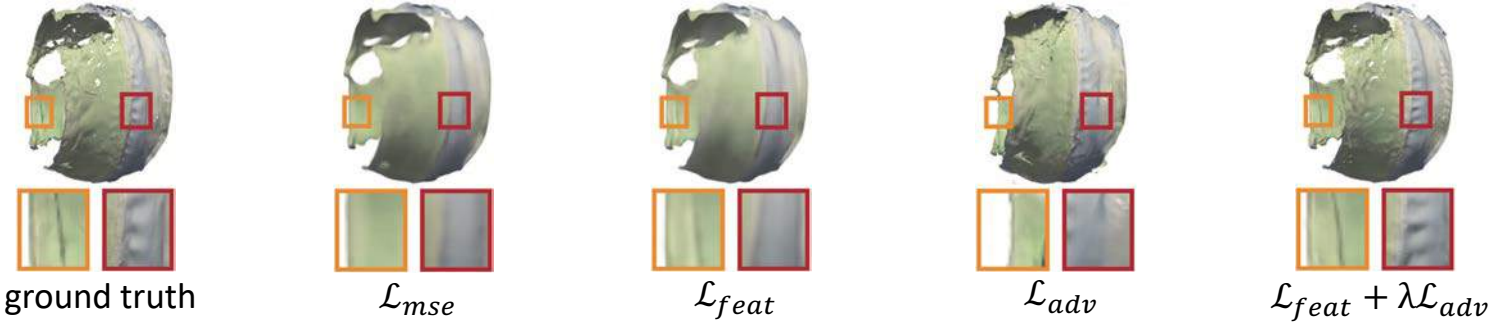
[1]I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Proceedings of Advances in Neural Information Processing Systems, pp. 2672–2680, 2014.

Total Loss

- A weighted combination of $\mathcal{L}_{feat}$ and $\mathcal{L}_{adv}$

$$\mathcal{L} = \mathcal{L}_{feat} + \lambda \mathcal{L}_{adv}$$

- $\mathcal{L}_{feat}$ and $\mathcal{L}_{adv}$ are complimentary with each other
  - $\mathcal{L}_{feat}$: overall feature level difference between image pairs
  - $\mathcal{L}_{adv}$: local details that the real and fake images differ the most



Training

- Updating D_υ and R_ω w.r.t. the loss functions iteratively

Algorithm 1 Training process of InSituNet.

Input: Training data includes parameters $\{P_{sim}, P_{vis}, P_{view}\}_{0:N-1}$ and the corresponding images $I_{0:N-1}$. Initial weights ω and υ of R_ω and D_υ, respectively. The feature comparator F.

Output: Optimized weights ω and υ

Repeat:

  $\{P_{sim}, P_{vis}, P_{view}\}_{0:b-1}, I_{0:b-1}$ sampled from training data

  $\hat{I}_{0:b-1} \leftarrow R_\omega(\{P_{sim}, P_{vis}, P_{view}\}_{0:b-1})$

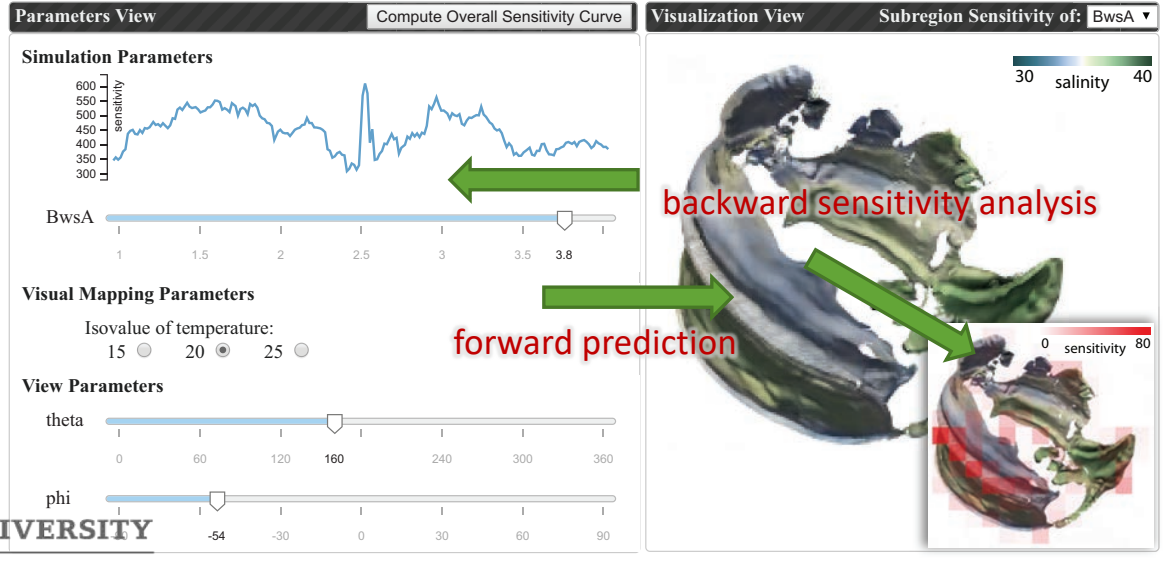
  $\upsilon \leftarrow \text{Adam}(\nabla_\upsilon \mathcal{L}_{adv\_D}(I_{0:b-1}, \hat{I}_{0:b-1}; \upsilon), \upsilon, \alpha_D, \beta_1, \beta_2)$

  $\omega \leftarrow \text{Adam}(\nabla_\omega \mathcal{L}(I_{0:b-1}, \hat{I}_{0:b-1}; \omega), \omega, \alpha_R, \beta_1, \beta_2)$

Until exit criterion is satisfied

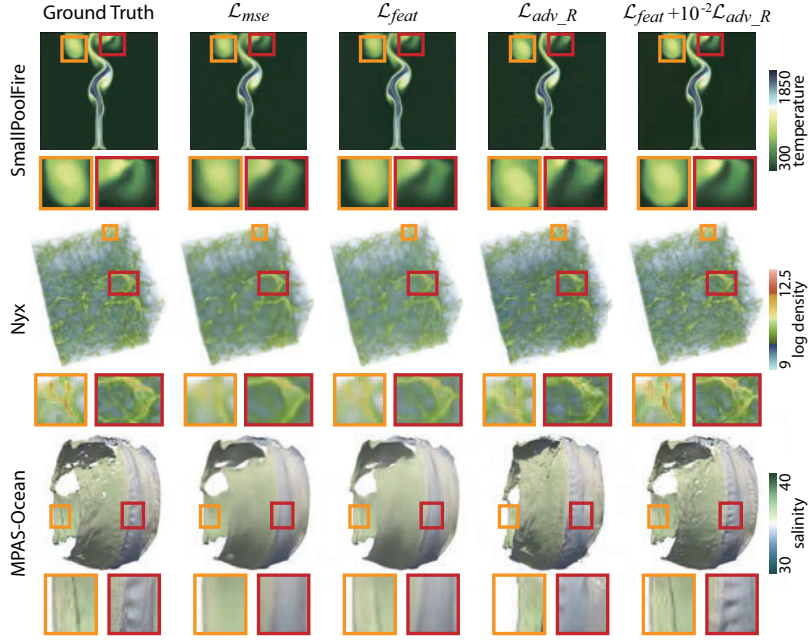
Parameter Space Exploration with InSituNet

- Forward prediction
  - Predicting image for new parameter settings
- Backward sensitivity analysis
  - Computing the sensitivity of the parameters



Results

- Three simulations: SmallPoolFire, Nyx, and MPAS-Ocean
- Comparing the prediction with the ground truth



Results

- Nyx (cosmological simulation)
- MPAS-Ocean (ocean simulation)

Case Study 1: Nyx

Case Study 2: MPAS-Ocean

Results

Simulation	P_{sim}		P_{vis}	P_{view}		k	Diversity	Size (GB)			Performance				
	Name	Number		Name	Number			Raw	Image	Network	t_{sim} (hr)	t_{vis} (hr)	t_{tr} (hr)	t_{fp} (s)	t_{bp} (s)
SmallPoolFire	Ck, C	4,000	pseudo-coloring with 5 color schemes	N/A	N/A	32	2.72	≈25.0	0.43	0.06	1,420.0	6.45	16.40	0.031	0.19
Nyx	OmM, OmB, h	500	volume rendering with a transfer function	θ, ϕ	100	48	1.72	≈30.0	3.92	0.12	537.5	8.47	18.02	0.033	0.22
MPAS-Ocean	$BwsA$	300	isosurface visualization with 3 isovalues	θ, ϕ	100	48	1.75	≈300.0	3.46	0.15	229.5	10.73	18.13	0.033	0.23

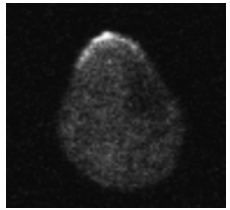
Datasets and timings: t_{sim}, t_{vis}, and t_{tr} are timings for running ensemble simulations, visualizing data in situ, and training InSituNet, respectively; t_{fp} and t_{bp} are timings for a forward and backward propagation of the trained InSituNet, respectively.

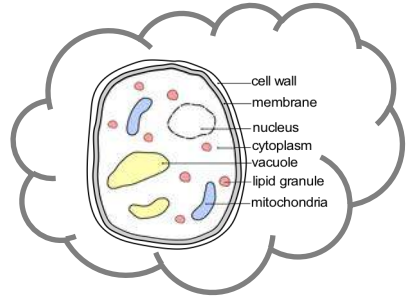
Conclusion

- We propose InSituNet, a deep learning-based image synthesis model for parameter space exploration of large-scale ensemble simulations
- We evaluate the effectiveness of InSituNet in analyzing ensemble simulations that model different physical phenomena

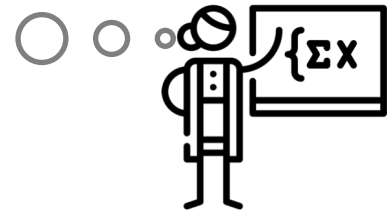
Neural **N**etwork Assisted **V**isual **A**nalysis of Yeast Cell Polarization Simulation

Experimental Biologist

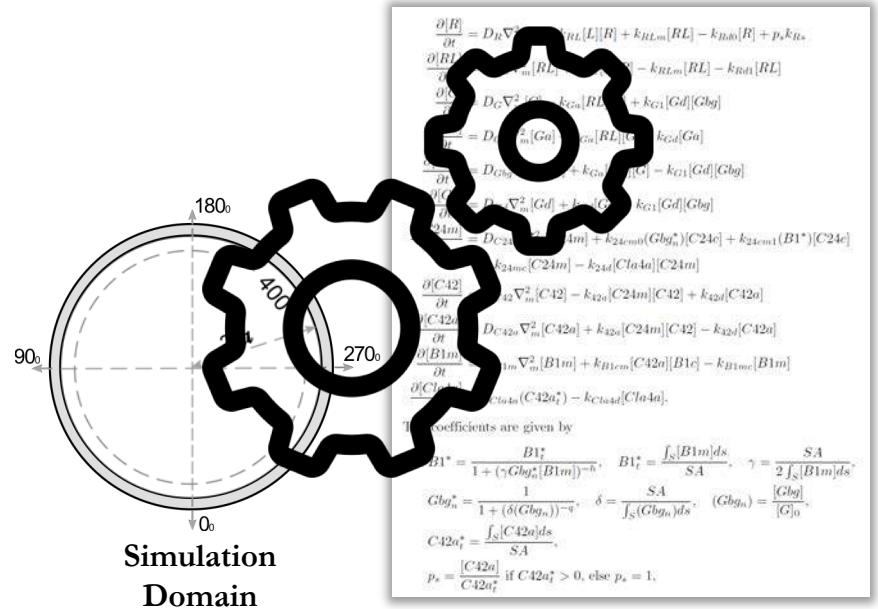




Computational Biologist



Mathematical Simulation Model



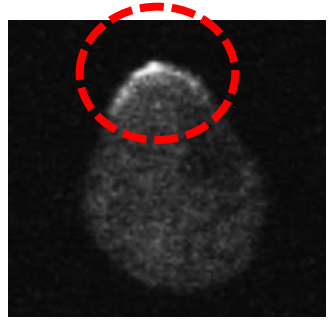
Simulation Domain

Simulation Background

- Protein species of interest : Cdc42

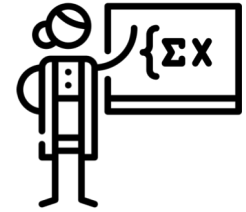
- Identify parameters configurations that can simulate high Cdc42 polarization

- <u>Polarization:</u> Asymmetric localization of protein concentration in a small region of the cell membrane

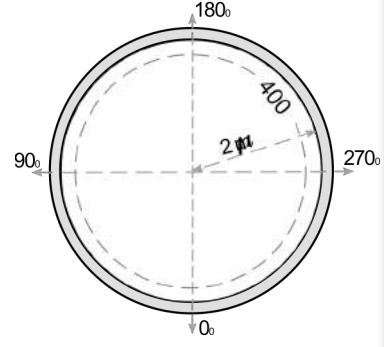


Microscopic Image

<u>Computational Biologist</u>



Mathematical Simulation Model



Simulation Domain

$$\frac{\partial[R]}{\partial t} = D_R\nabla^2_m[R] - k_{RL}[L][R] + k_{RLm}[RL] - k_{Rd0}[R] + p_s k_{Rs}$$

$$\frac{\partial[RL]}{\partial t} = D_{RL}\nabla^2_m[RL] + k_{RL}[L][R] - k_{RLm}[RL] - k_{RLd1}[RL]$$

$$\frac{\partial[G]}{\partial t} = D_G\nabla^2_m[G] - k_{Ga}[RL][G] + k_{G1}[Gd][Gbg]$$

$$\frac{\partial[Ga]}{\partial t} = D_{Ga}\nabla^2_m[Ga] + k_{Ga}[RL][G] - k_{Gd}[Ga]$$

$$\frac{\partial[Gbg]}{\partial t} = D_{Gbg}\nabla^2_m[Gbg] + k_{Ga}[RL][G] - k_{G1}[Gd][Gbg]$$

$$\frac{\partial[Gd]}{\partial t} = D_{Gd}\nabla^2_m[Gd] + k_{Gd}[Ga] - k_{G1}[Gd][Gbg]$$

$$\frac{\partial[C24m]}{\partial t} = D_{C24m}\nabla^2_m[C24m] + k_{24cm0}(Gbg^*_n)[C24c] + k_{24cm1}(B1^*)[C24c] - k_{24mc}[C24m] - k_{24d}[Cla4a][C24m]$$

$$\frac{\partial[C42]}{\partial t} = D_{C42}\nabla^2_m[C42] - k_{42a}[C24m][C42] + k_{42d}[C42a]$$

$$\frac{\partial[C42a]}{\partial t} = D_{C42a}\nabla^2_m[C42a] + k_{42a}[C24m][C42] - k_{42d}[C42a]$$

$$\frac{\partial[B1m]}{\partial t} = D_{B1m}\nabla^2_m[B1m] + k_{B1cm}[C42a][B1c] - k_{B1mc}[B1m]$$

$$\frac{\partial[Cla4a]}{\partial t} = k_{Cla4a}(C42a^*_t) - k_{Cla4d}[Cla4a].$$

The coefficients are given by

$$B1^* = \frac{B1^*_t}{1+(\gamma Gbg^*_n[B1m])^{-h}}, \quad B1^*_t = \frac{\int_S[B1m]ds}{SA}, \quad \gamma = \frac{SA}{2\int_S[B1m]ds},$$

$$Gbg^*_n = \frac{1}{1+(\delta(Gbg_n))^{-q}}, \quad \delta = \frac{SA}{\int_S(Gbg_n)ds}, \quad (Gbg_n) = \frac{[Gbg]}{[G]_0},$$

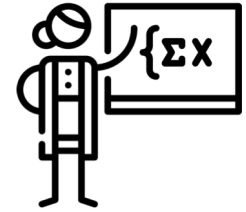
$$C42a^*_t = \frac{\int_S[C42a]ds}{SA},$$

$$p_s = \frac{[C42a]}{C42a^*_t} \text{ if } C42a^*_t > 0, \text{ else } p_s = 1.$$

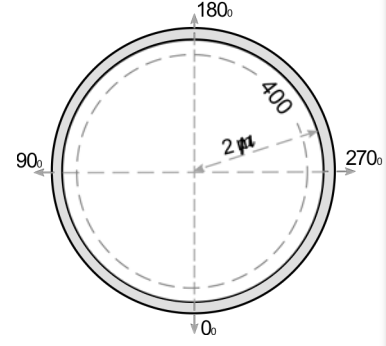
Challenges

- High-dimensional input/output spaces
  - 35 uncalibrated simulation input parameters
  - 400-dimensional output
- Computationally expensive
  - ~2.3 hrs/execution
- Challenging to perform interactive exploratory analysis of the parameter space

Computational Biologist



Mathematical Simulation Model



Simulation Domain

$$\frac{\partial[R]}{\partial t} = D_R\nabla^2_m[R] - k_{RL}[L][R] + k_{RLm}[RL] - k_{Rd0}[R] + p_s k_{Rs}$$

$$\frac{\partial[RL]}{\partial t} = D_{RL}\nabla^2_m[RL] + k_{RL}[L][R] - k_{RLm}[RL] - k_{Rd1}[RL]$$

$$\frac{\partial[G]}{\partial t} = D_G\nabla^2_m[G] - k_{Ga}[RL][G] + k_{G1}[Gd][Gbg]$$

$$\frac{\partial[Ga]}{\partial t} = D_{Ga}\nabla^2_m[Ga] + k_{Ga}[RL][G] - k_{Gd}[Ga]$$

$$\frac{\partial[Gbg]}{\partial t} = D_{Gbg}\nabla^2_m[Gbg] + k_{Ga}[RL][G] - k_{G1}[Gd][Gbg]$$

$$\frac{\partial[Gd]}{\partial t} = D_{Gd}\nabla^2_m[Gd] + k_{Gd}[Ga] - k_{G1}[Gd][Gbg]$$

$$\frac{\partial[C24m]}{\partial t} = D_{C24m}\nabla^2_m[C24m] + k_{24cm0}(Gbg^*_n)[C24c] + k_{24cm1}(B1^*)[C24c] - k_{24mc}[C24m] - k_{24d}[Cla4a][C24m]$$

$$\frac{\partial[C42]}{\partial t} = D_{C42}\nabla^2_m[C42] - k_{42a}[C24m][C42] + k_{42d}[C42a]$$

$$\frac{\partial[C42a]}{\partial t} = D_{C42a}\nabla^2_m[C42a] + k_{42a}[C24m][C42] - k_{42d}[C42a]$$

$$\frac{\partial[B1m]}{\partial t} = D_{B1m}\nabla^2_m[B1m] + k_{B1cm}[C42a][B1c] - k_{B1mc}[B1m]$$

$$\frac{\partial[Cla4a]}{\partial t} = k_{Cla4a}(C42a^*_t) - k_{Cla4d}[Cla4a].$$

The coefficients are given by

$$B1^* = \frac{B1^*_t}{1+(\gamma Gbg^*_n[B1m])^{-h}}, \quad B1^*_t = \frac{\int_S[B1m]ds}{SA}, \quad \gamma = \frac{SA}{2\int_S[B1m]ds},$$

$$Gbg^*_n = \frac{1}{1+(\delta(Gbg_n))^{-q}}, \quad \delta = \frac{SA}{\int_S(Gbg_n)ds}, \quad (Gbg_n) = \frac{[Gbg]}{[G]_0},$$

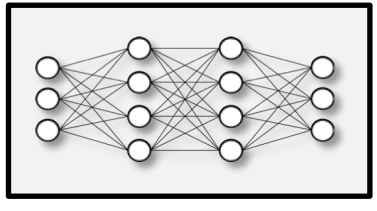
$$C42a^*_t = \frac{\int_S[C42a]ds}{SA},$$

$$p_s = \frac{[C42a]}{C42a^*_t} \text{ if } C42a^*_t > 0, \text{ else } p_s = 1.$$

Proposed Approach

- Neural network-based surrogate model
- Mimics the expensive simulation during analysis
- Facilitates interactive visual analytics
- Quick preview of predicted results for new parameters configurations

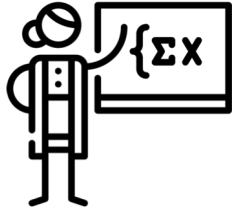
Data Scientist



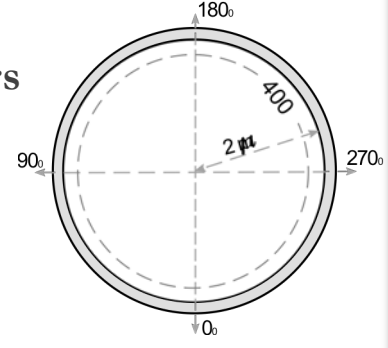
Surrogate Model

≈

Computational Biologist



Mathematical Simulation Model



Simulation Domain

$$\frac{\partial[R]}{\partial t} = D_R\nabla^2_m[R] - k_{RL}[L][R] + k_{RLm}[RL] - k_{Rd0}[R] + p_s k_{Rs}$$

$$\frac{\partial[RL]}{\partial t} = D_{RL}\nabla^2_m[RL] + k_{RL}[L][R] - k_{RLm}[RL] - k_{Rd1}[RL]$$

$$\frac{\partial[G]}{\partial t} = D_G\nabla^2_m[G] - k_{Ga}[RL][G] + k_{G1}[Gd][Gbg]$$

$$\frac{\partial[Ga]}{\partial t} = D_{Ga}\nabla^2_m[Ga] + k_{Ga}[RL][G] - k_{Gd}[Ga]$$

$$\frac{\partial[Gbg]}{\partial t} = D_{Gbg}\nabla^2_m[Gbg] + k_{Ga}[RL][G] - k_{G1}[Gd][Gbg]$$

$$\frac{\partial[Gd]}{\partial t} = D_{Gd}\nabla^2_m[Gd] + k_{Gd}[Ga] - k_{G1}[Gd][Gbg]$$

$$\frac{\partial[C24m]}{\partial t} = D_{C24m}\nabla^2_m[C24m] + k_{24cm0}(Gbg^*_n)[C24c] + k_{24cm1}(B1^*)[C24c] - k_{24mc}[C24m] - k_{24d}[Cla4a][C24m]$$

$$\frac{\partial[C42]}{\partial t} = D_{C42}\nabla^2_m[C42] - k_{42a}[C24m][C42] + k_{42d}[C42a]$$

$$\frac{\partial[C42a]}{\partial t} = D_{C42a}\nabla^2_m[C42a] + k_{42a}[C24m][C42] - k_{42d}[C42a]$$

$$\frac{\partial[B1m]}{\partial t} = D_{B1m}\nabla^2_m[B1m] + k_{B1cm}[C42a][B1c] - k_{B1mc}[B1m]$$

$$\frac{\partial[Cla4a]}{\partial t} = k_{Cla4a}(C42a^*_t) - k_{Cla4d}[Cla4a].$$

The coefficients are given by

$$B1^* = \frac{B1^*_t}{1+(\gamma Gbg^*_n[B1m])^{-h}}, \quad B1^*_t = \frac{\int_S[B1m]ds}{SA}, \quad \gamma = \frac{SA}{2\int_S[B1m]ds},$$

$$Gbg^*_n = \frac{1}{1+(\delta(Gbg_n))^{-q}}, \quad \delta = \frac{SA}{\int_S(Gbg_n)ds}, \quad (Gbg_n) = \frac{[Gbg]}{[G]_0},$$

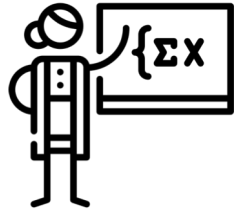
$$C42a^*_t = \frac{\int_S[C42a]ds}{SA},$$

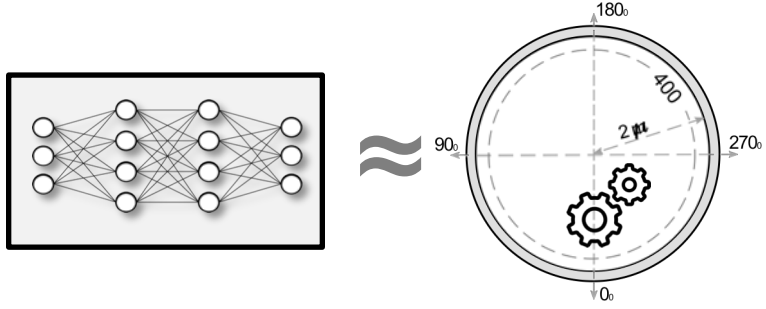
$$p_s = \frac{[C42a]}{C42a^*_t} \text{ if } C42a^*_t > 0, \text{ else } p_s = 1.$$

VA System Requirements

- Quickly preview results for new parameters
- Perform parameter sensitivity analysis
- Discover interesting parameter configurations
- Validate the surrogate model
- Extract insights from trained surrogate

Computational Biologist

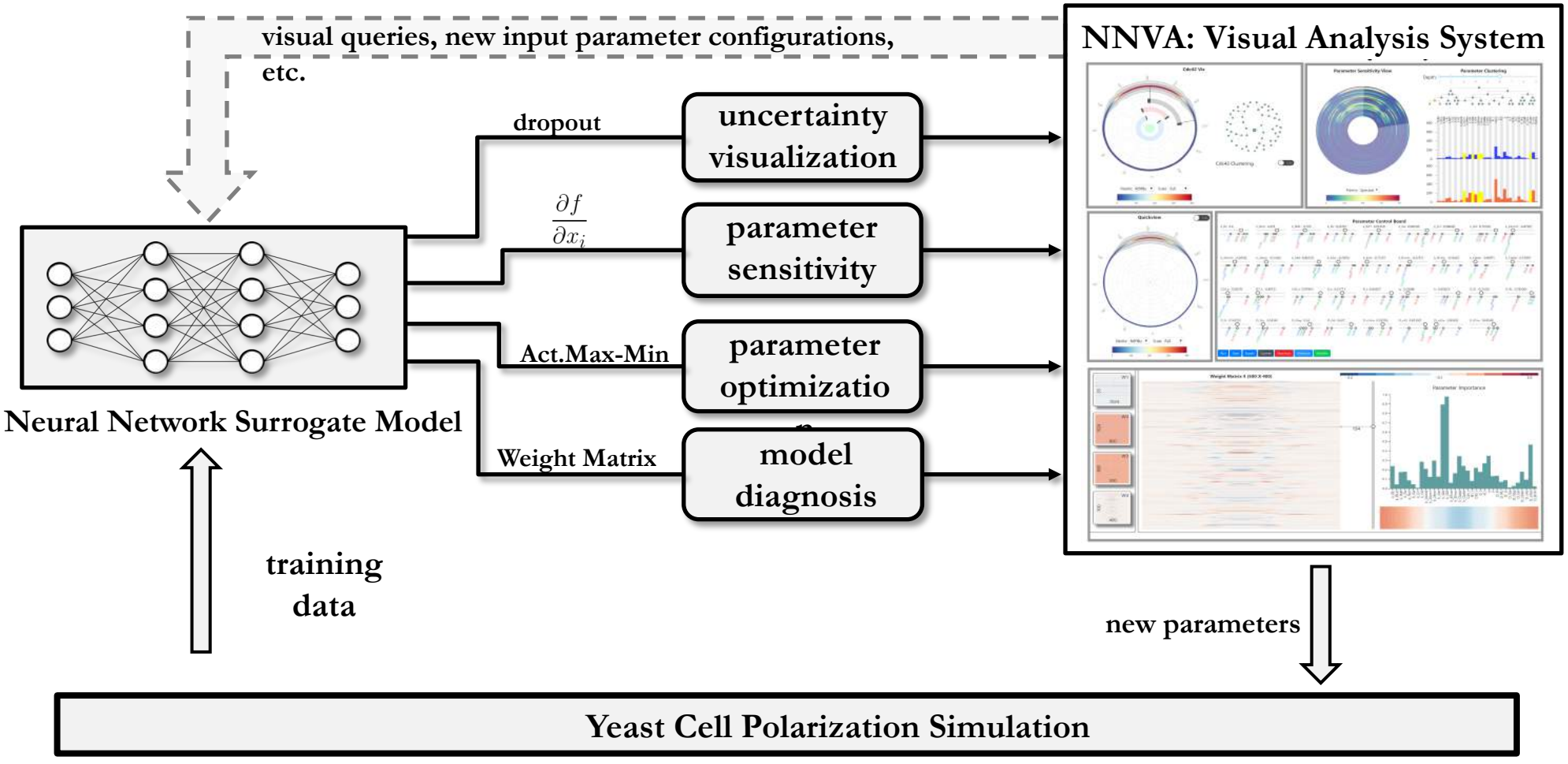




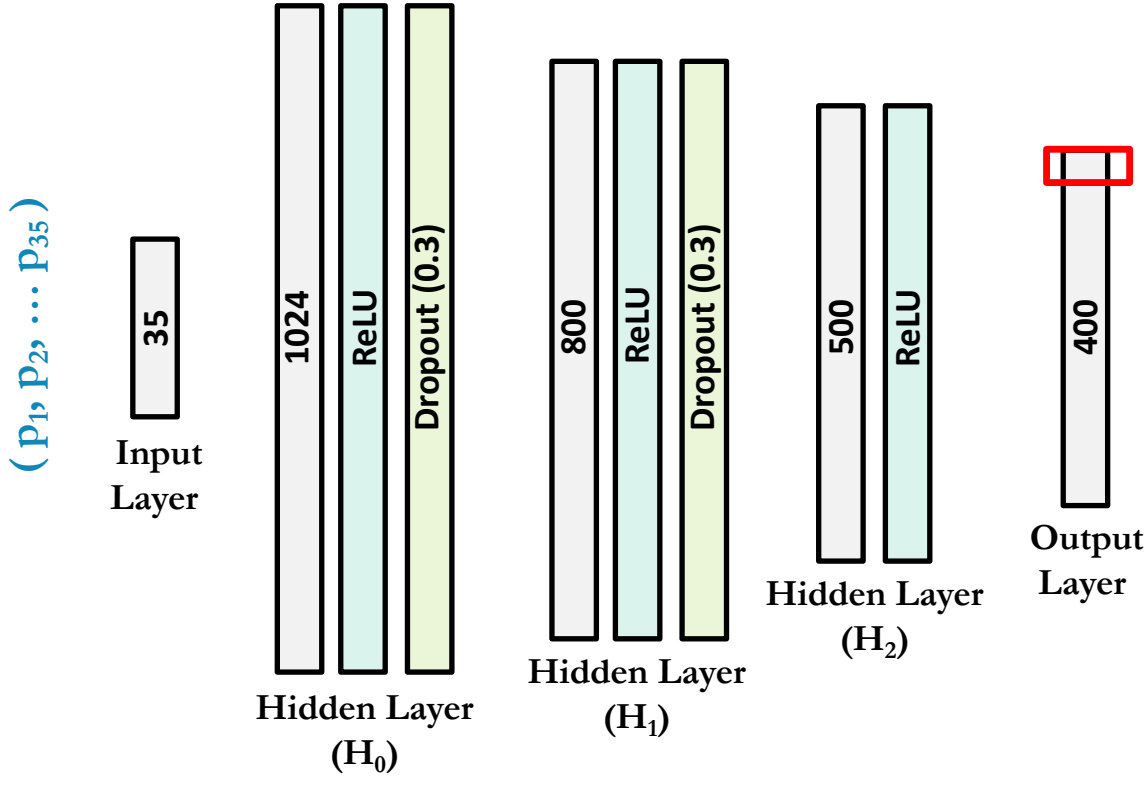
Surrogate

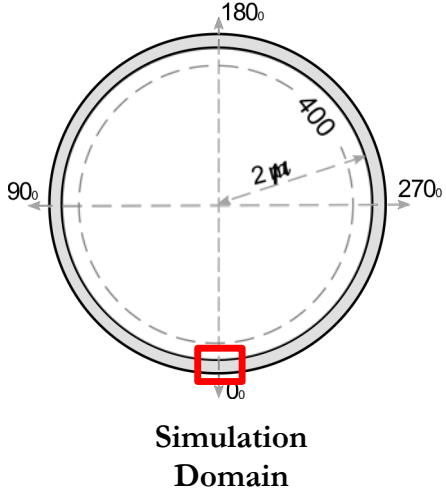
Simulation

Visual Analysis Workflow

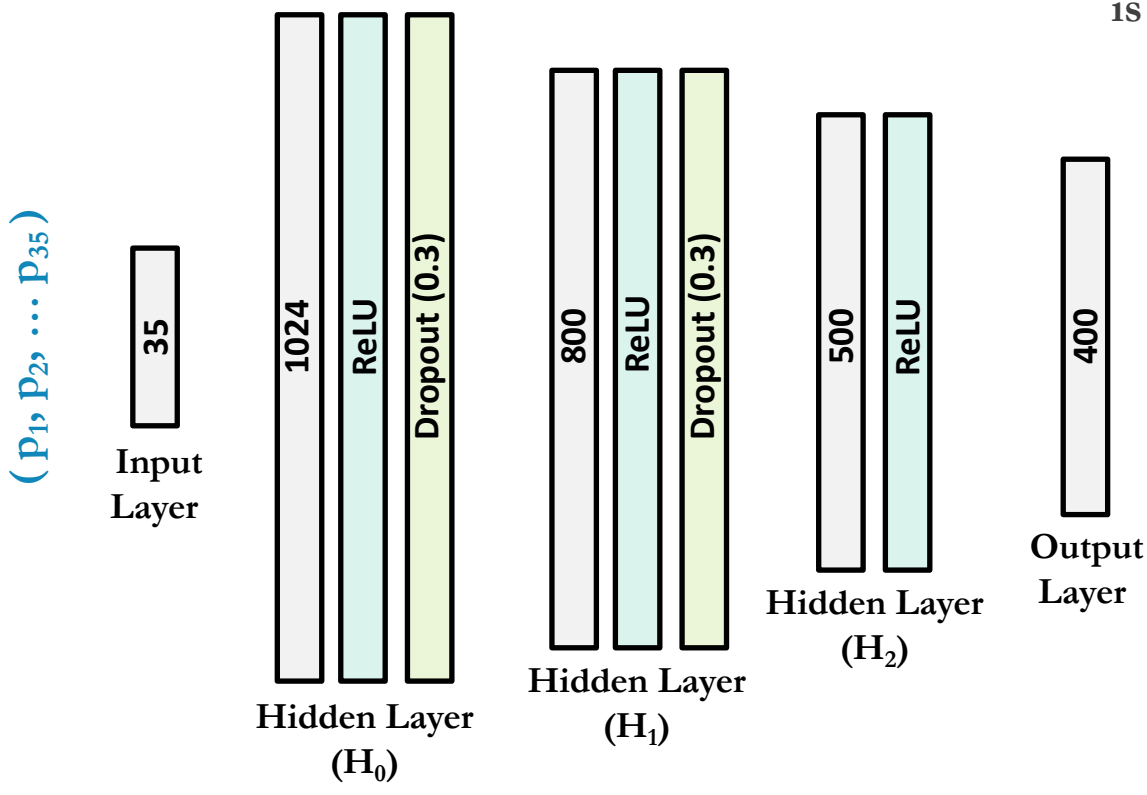

visual queries, new input parameter configurations, etc.
NNVA: Visual Analysis System
dropout
uncertainty visualization
$\frac{\partial f}{\partial x_i}$
parameter sensitivity
Act.Max-Min
parameter optimizatio
Weight Matrix
model diagnosis
Neural Network Surrogate Model
training data
new parameters
Yeast Cell Polarization Simulation

Network Structure and Training

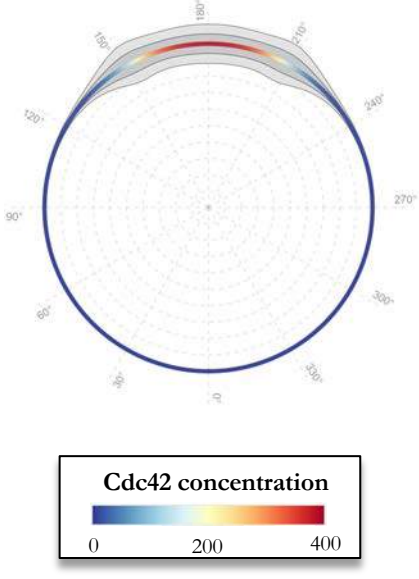




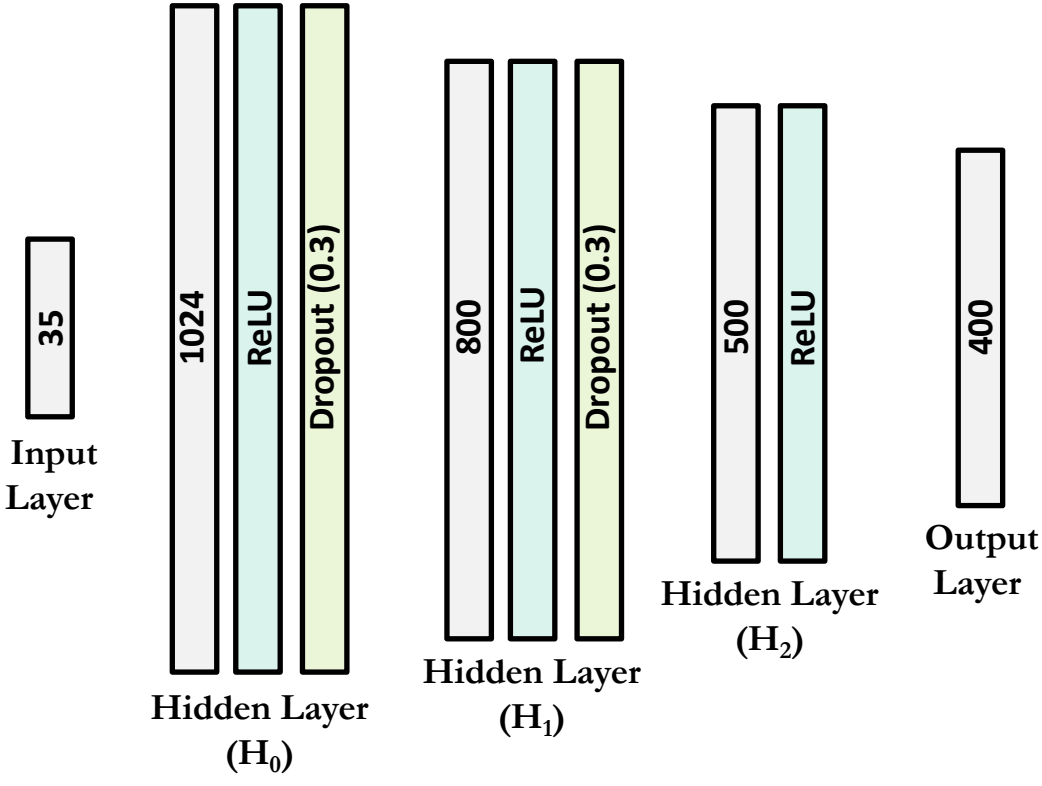
Network Structure and Training



<u>Visualization:</u> Predicted protein concentration is color-mapped and laid-out radially



Network Structure and Training

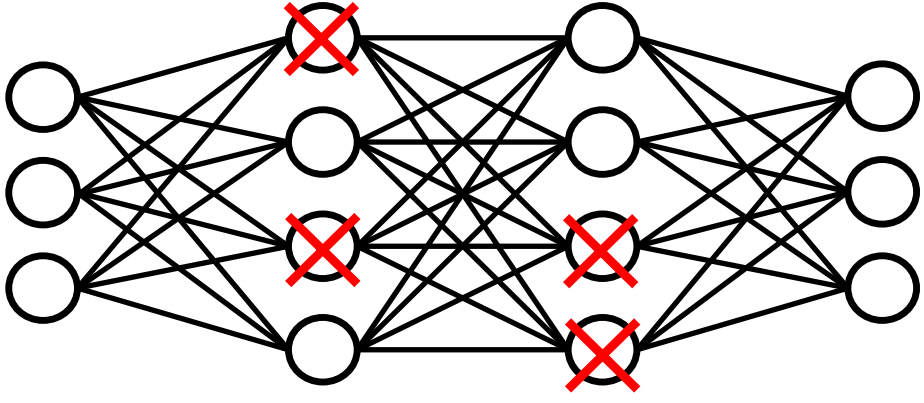


- Loss Function: MSE
- Training data size: 3000
- Uniformly sampled parameter space
- Validation data size: 500
- Final Accuracy: 87.6%

Uncertainty in Neural Networks using Dropout

- Important to visualize the prediction uncertainty in the exploration process

- <u>Dropout:</u> Randomly ignoring the output of neurons in a layer



- Training phase:
  - Acts as regularizer to avoid overfitting

- Prediction phase[1]:
  - Uncertainty quantification of predicted result

[1] Gal et al. : Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. ICML 2016

Uncertainty in Neural Networks using Dropout

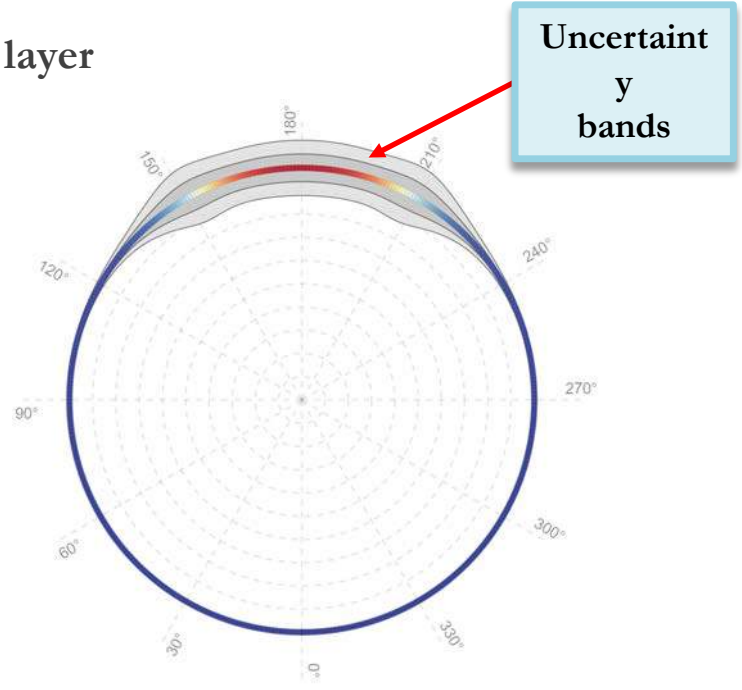
- Important to visualize the prediction uncertainty in the exploration process

- <u>Dropout:</u> Randomly ignoring the output of neurons in a layer

- Training phase:
  - Acts as regularizer to avoid overfitting

- Prediction phase[1]:
  - Uncertainty quantification of predicted result

- Visualized using standard deviation bands



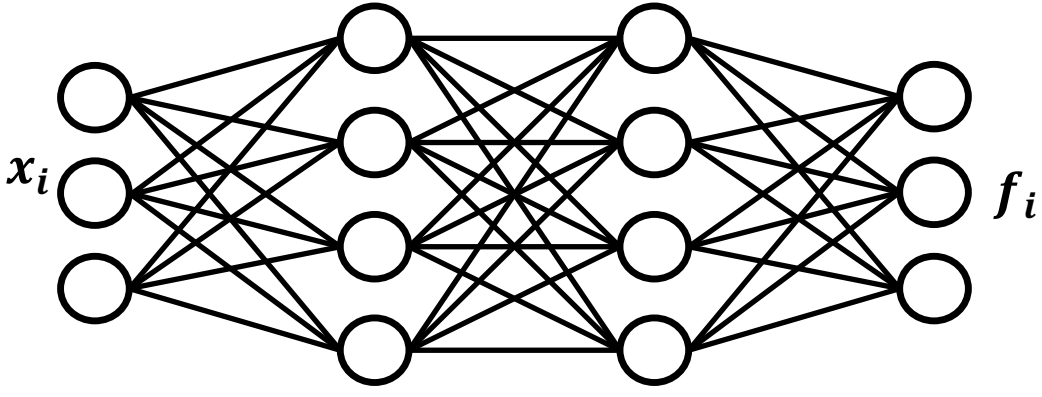
[1] Gal et al. : Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. ICML 2016

Parameter Sensitivity Analysis

- Influence of individual parameters on different areas of output domain
  - Measure of how much the output changes for a small change in the input
  - Useful for parameter tuning

- Partial derivative of output w.r.t input: $\frac{\partial f_i}{\partial x_i}$

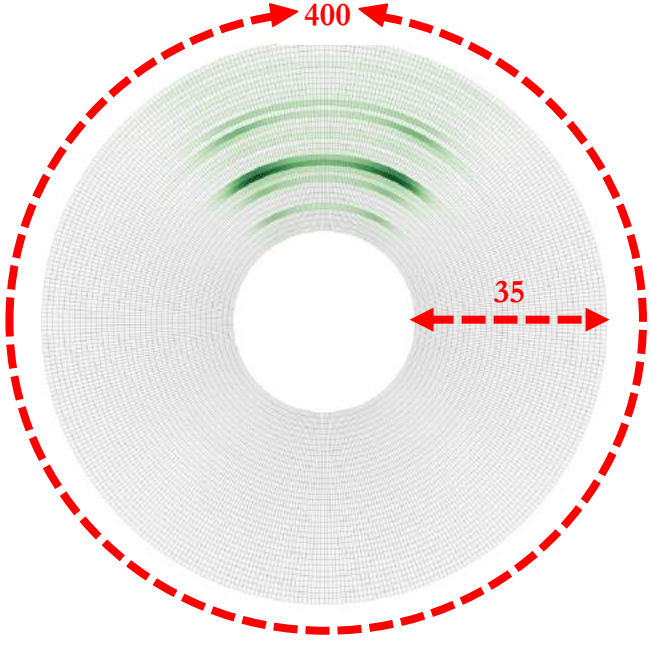
- Computed using backpropagation



Parameter Sensitivity Analysis

- Influence of individual parameters on different areas of output domain
  - Measure of how much the output changes for a small change in the input
  - Useful for parameter tuning

- Visualize detail parameter sensitivity across the cell membrane

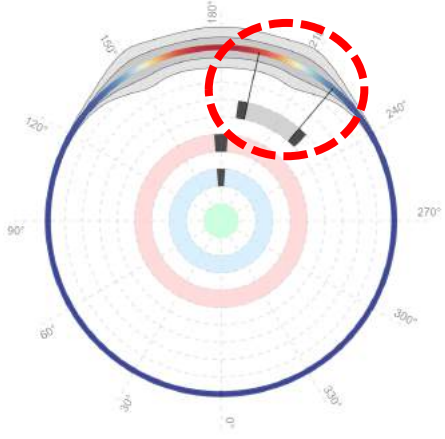


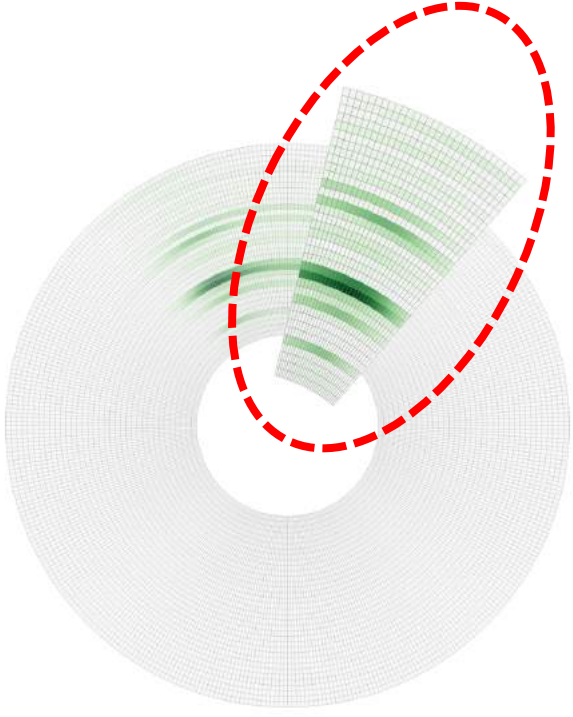
Parameter Sensitivity Analysis

- Influence of individual parameters on different areas of output domain
  - Measure of how much the output changes for a small change in the input
  - Useful for parameter tuning

- Visualize detail parameter sensitivity across the cell membrane

- Interactive sensitivity brush to select area of interest





Parameter Optimization

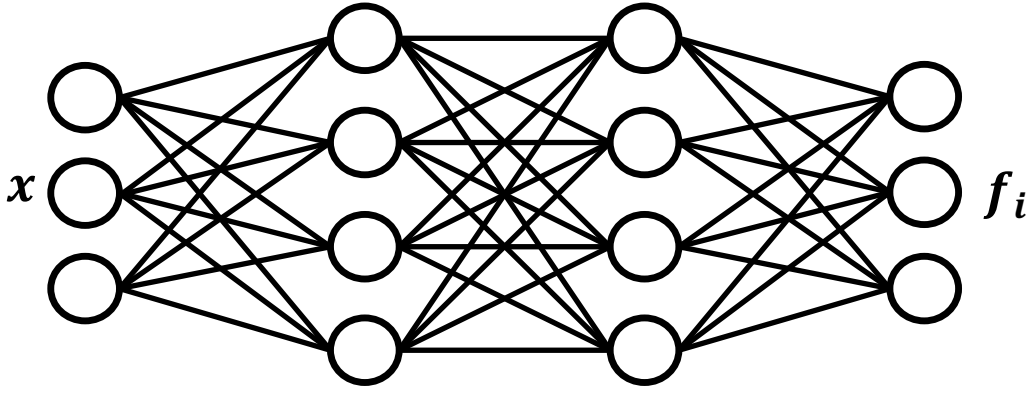
- Recommend parameter configurations which maximizes/minimizes the predicted protein concentration values at different regions

- <u>Activation Maximization:</u>
  - Keeping the weights fixed, update the input (via gradient ascent) such that it maximize the model output i,e.

$$\max_{\boldsymbol{x}} f_i(\boldsymbol{x}) - \underbrace{\lambda\|\boldsymbol{x} - \boldsymbol{x}'\|^2}_{\text{L}_2\text{-regularizer}}$$

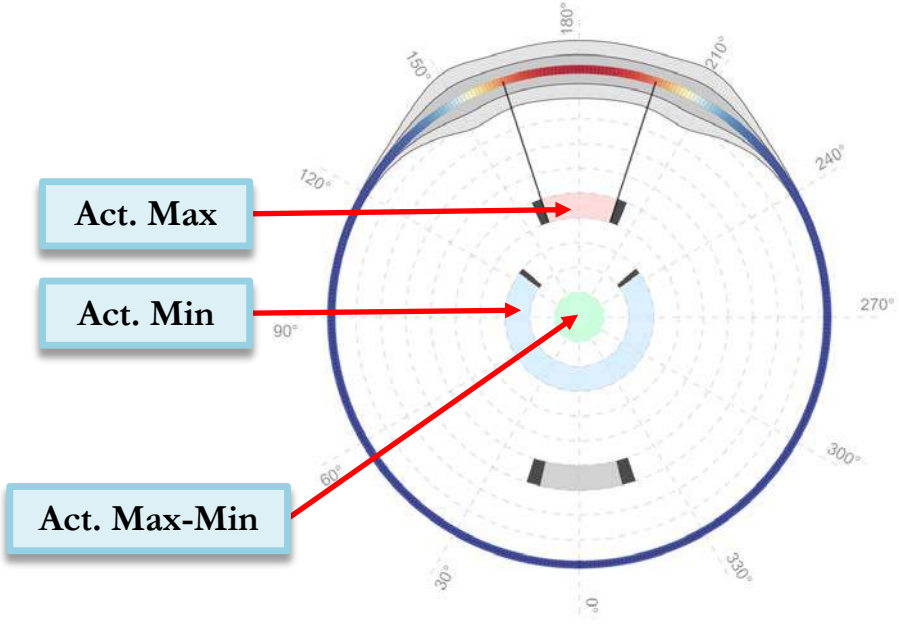
- <u>Activation Minimization:</u>

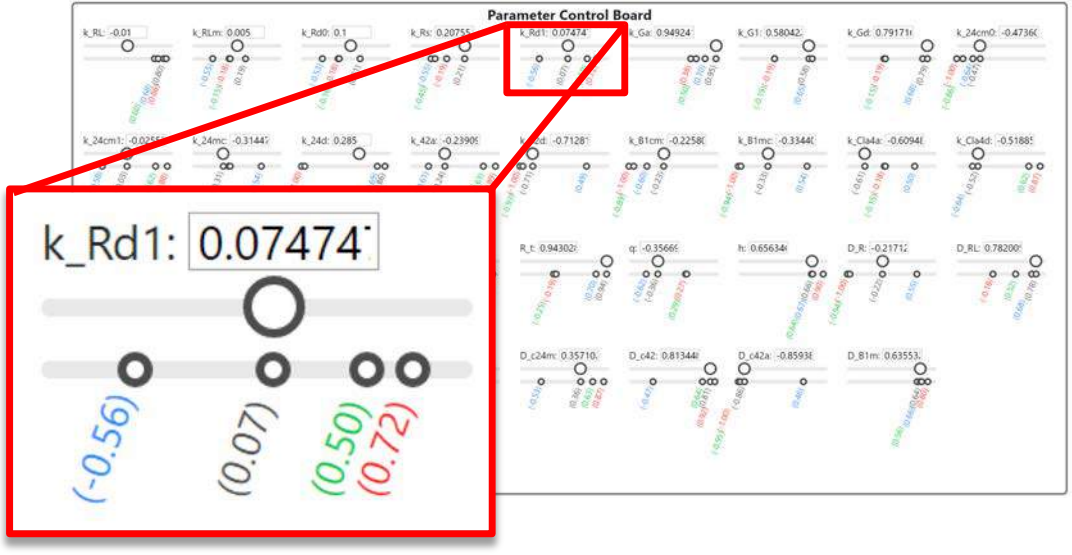
$$\min_{\boldsymbol{x}} f_i(\boldsymbol{x}) + \lambda\|\boldsymbol{x} - \boldsymbol{x}'\|^2$$



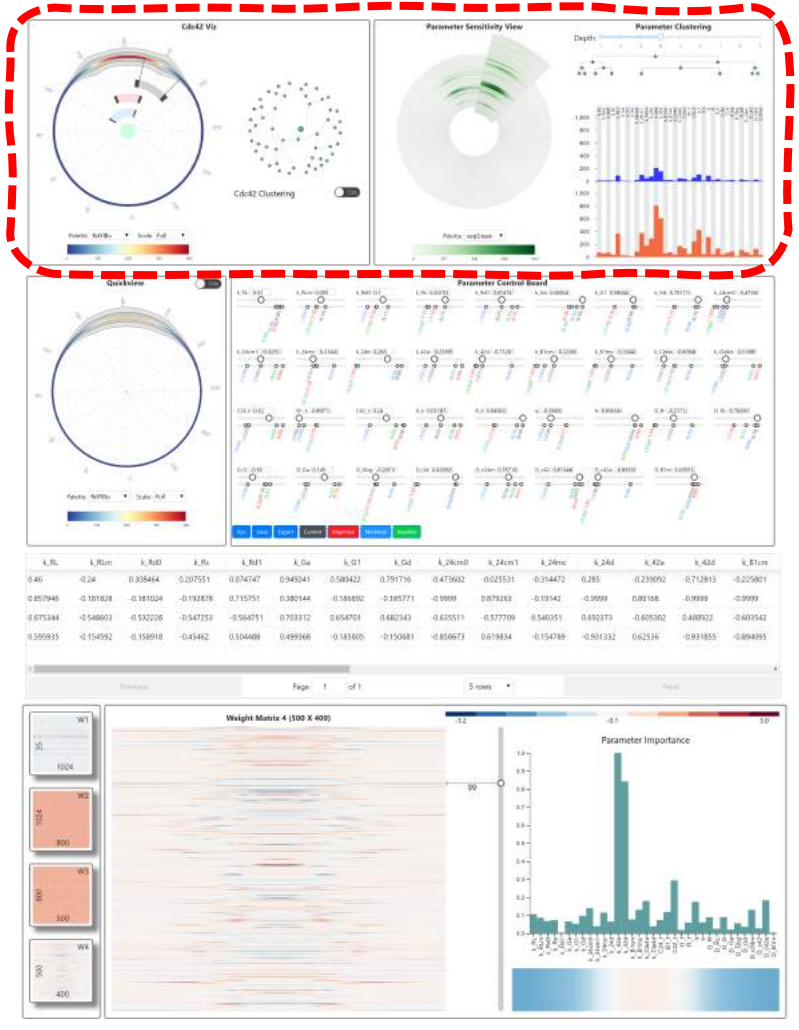
Parameter Optimization

- Recommend parameter configurations which maximizes/minimizes the predicted protein concentration values at different regions
- Interactive brushes to select specific areas of the cell membrane

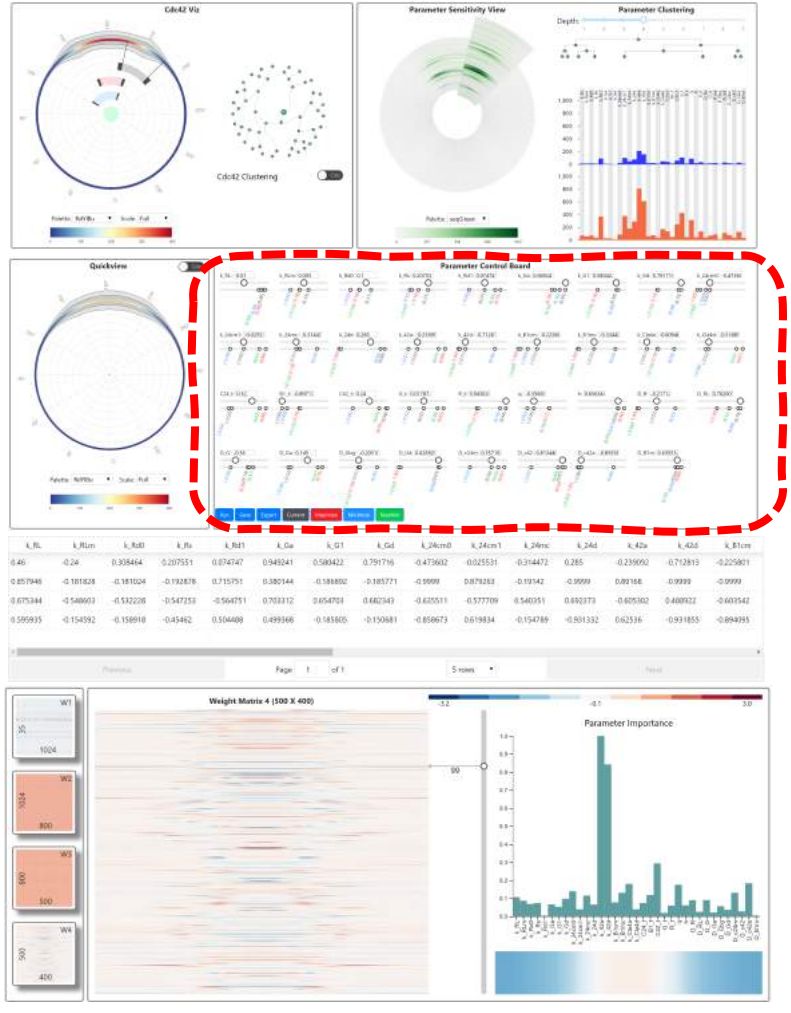




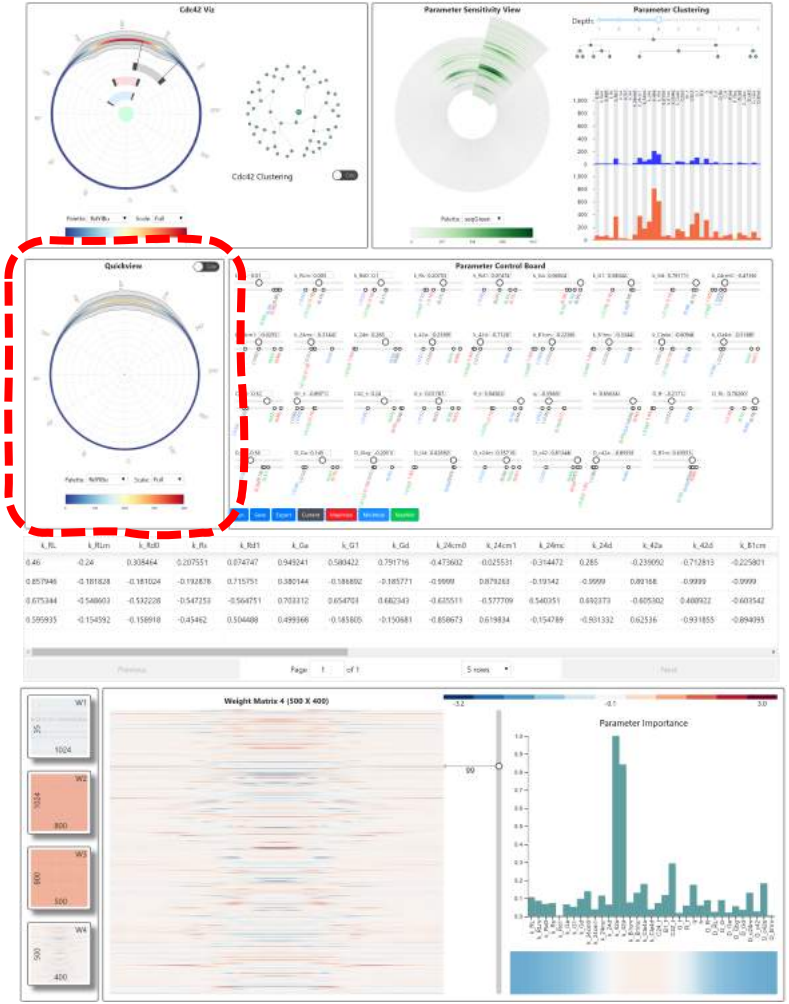
Visual Analysis System



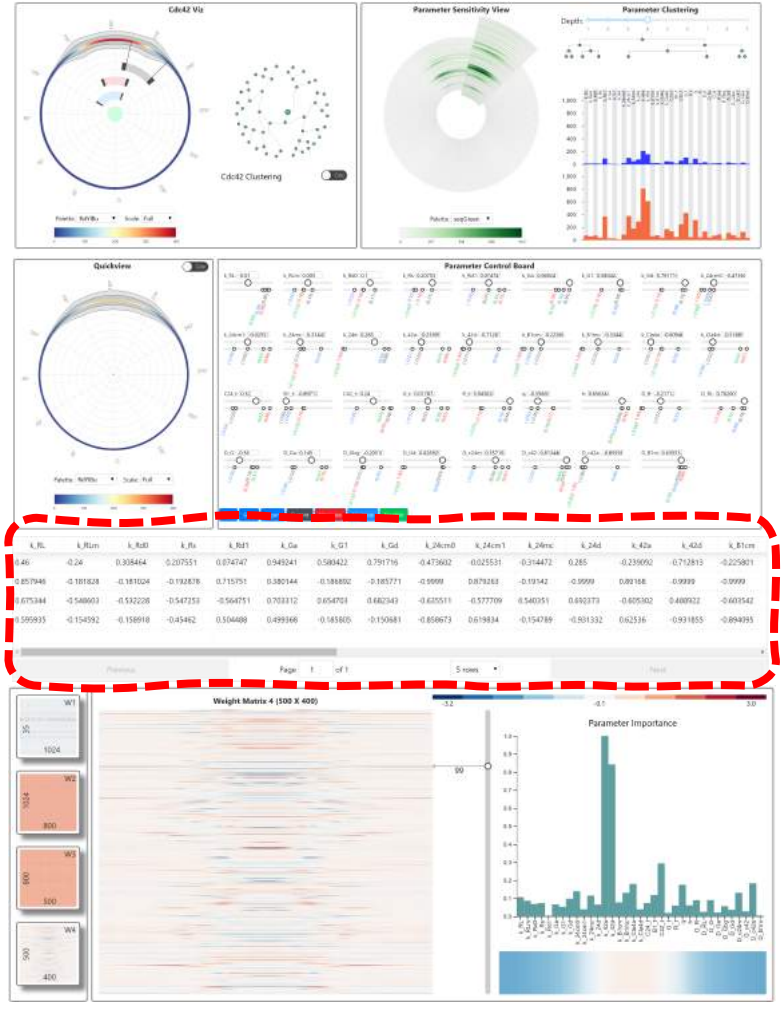
- **Instance View:**
  - Detail analysis of the predicted result for a specific parameter configuration of interest



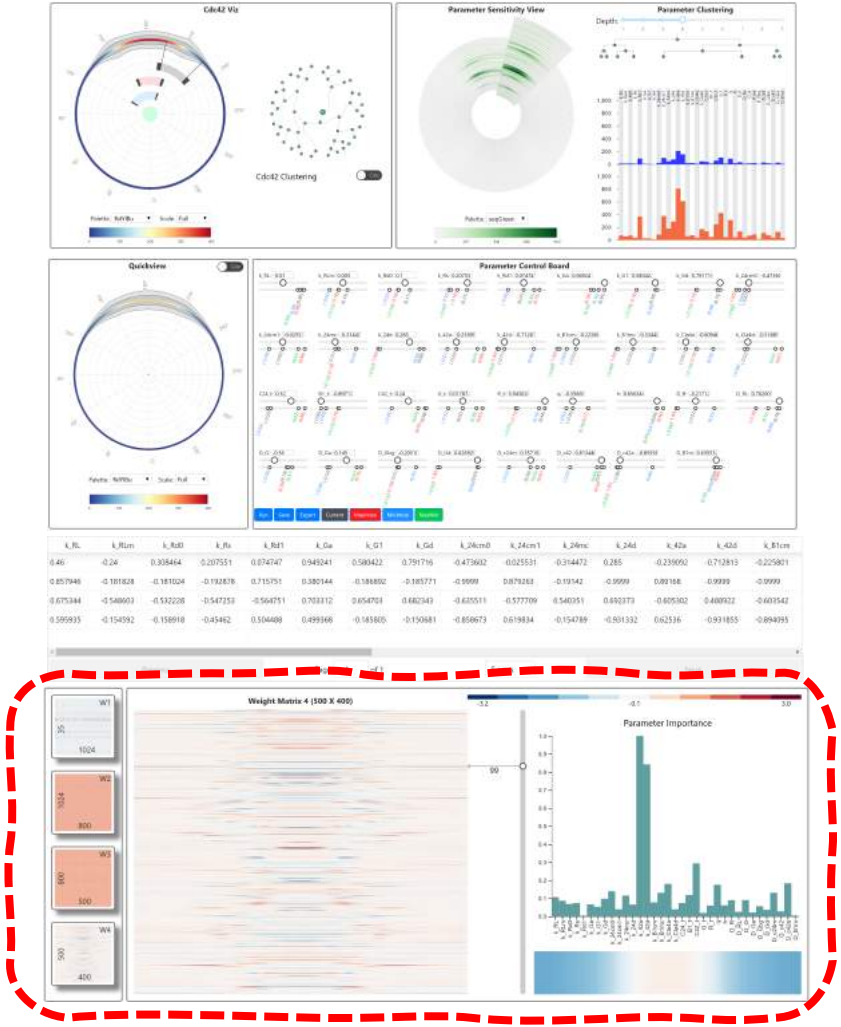
- **Instance View:**
  - Detail analysis of the predicted result for a specific parameter configuration of interest
- **Parameter Control Board:**
  - Interactively calibrate the 35 simulation parameters



- **Instance View:**
  - Detail analysis of the predicted result for a specific parameter configuration of interest
- **Parameter Control Board:**
  - Interactively calibrate the 35 simulation parameters
- **Quick View**
  - Visualize predicted protein concentration



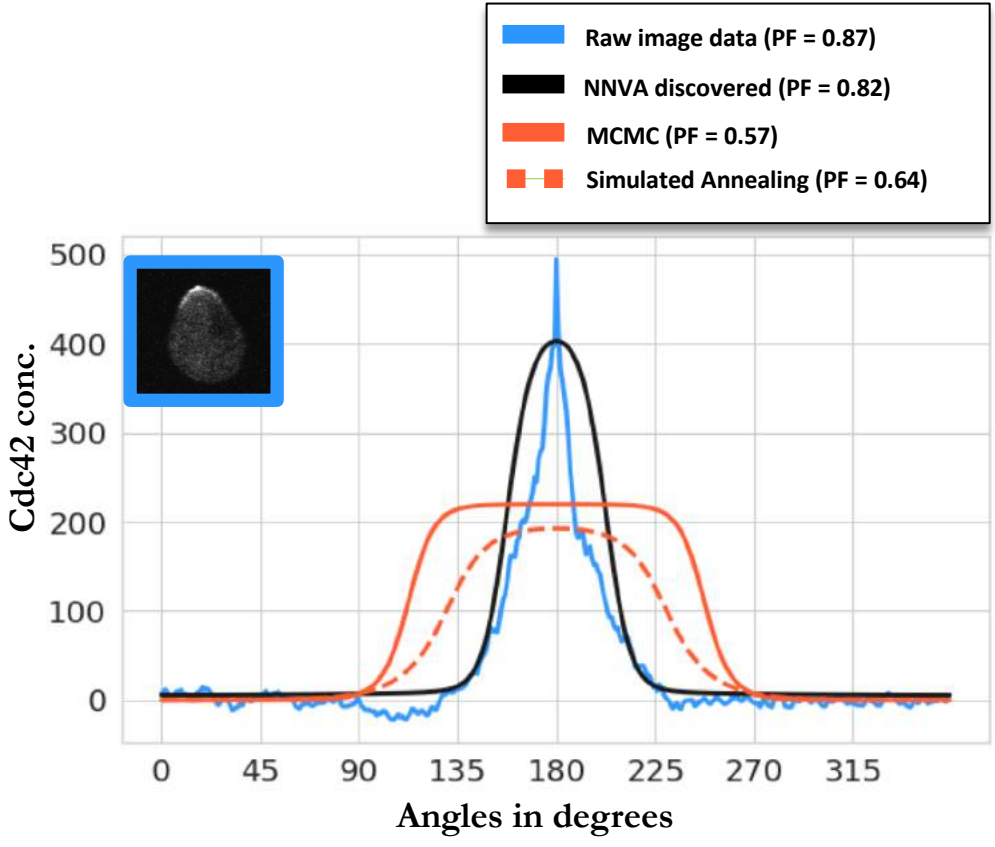
- **Instance View:**
  - Detail analysis of the predicted result for a specific parameter configuration of interest
- **Parameter Control Board:**
  - Interactively calibrate the 35 simulation parameters
- **Quick View**
  - Visualize predicted protein concentration
- **Parameter List View:**
  - Progressively store newly discovered sets of parameters



- **Instance View:**
  - Detail analysis of the predicted result for a specific parameter configuration of interest
- **Parameter Control Board:**
  - Interactively calibrate the 35 simulation parameters
- **Quick View**
  - Visualize predicted protein concentration
- **Parameter List View:**
  - Progressively store newly discovered sets of parameters
- **Model Analysis View:**
  - Analysis the weight matrices of the trained network

Use Case 1: Discover New Parameter Configurations

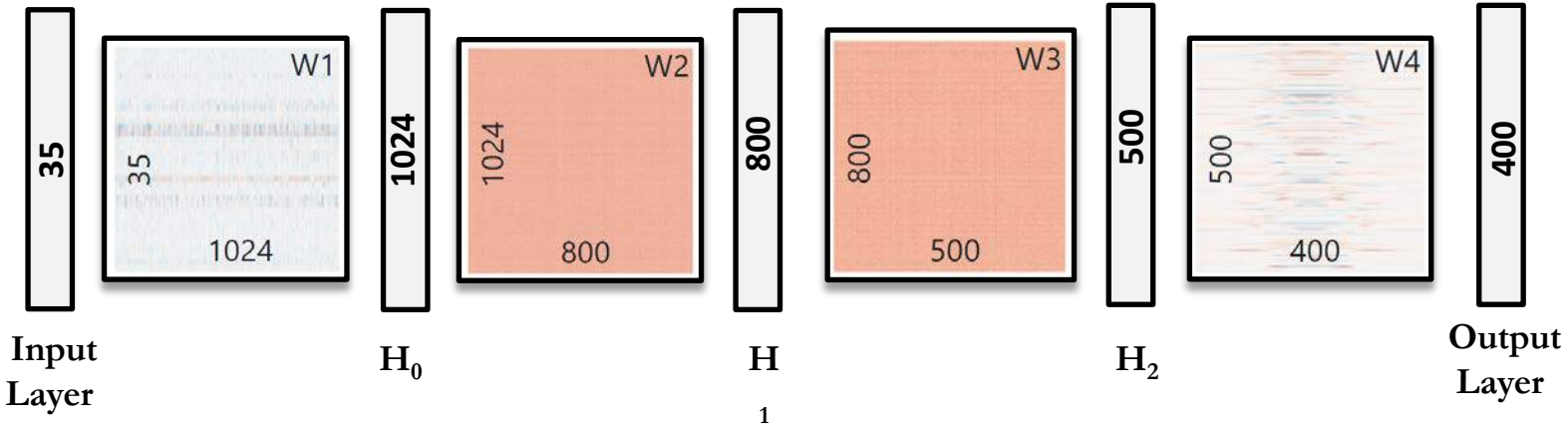
- Identify parameters configurations that can simulate high Cdc42 polarization
- Polarization Factor (PF): Measure of the extent of polarization [0.0, 1.0]
- Found several parameter configurations that simulated high PF (>0.8)
  - Highest PF = 0.82
- Compared with previous analysis efforts (using polynomial surrogate models)



Use Case 2: Analyzing the trained surrogate

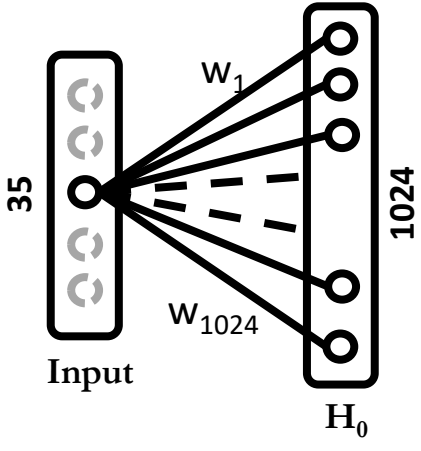
- To extract and validate the knowledge learned by the trained network

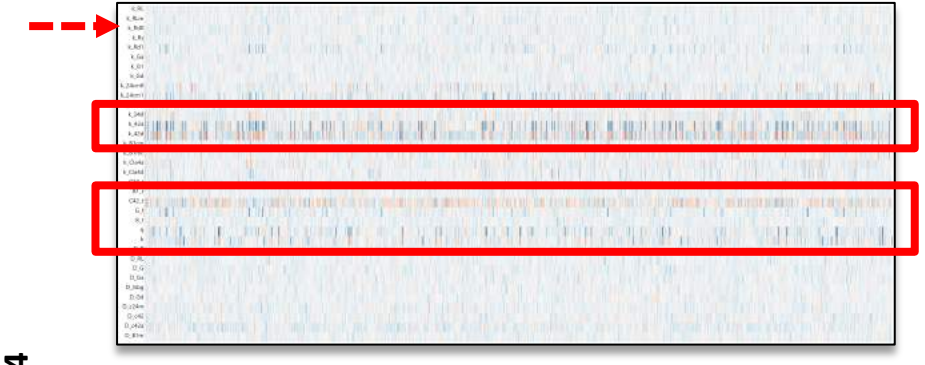
- Visualize and analyze the weight matrices



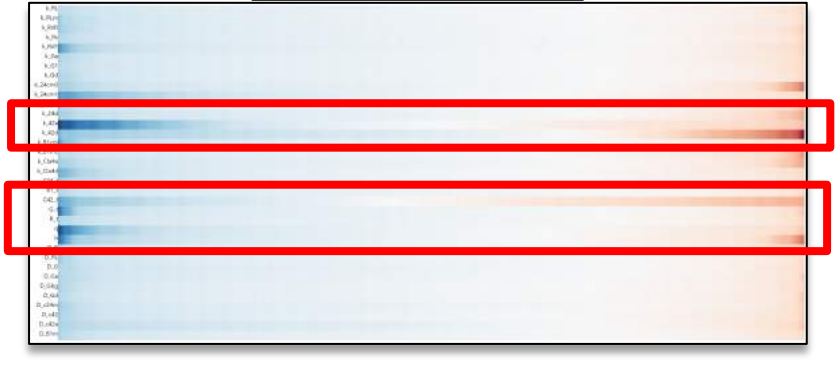
Use Case 2: Analyzing the trained surrogate

- First Weight Matrix





Row-wise Sorted

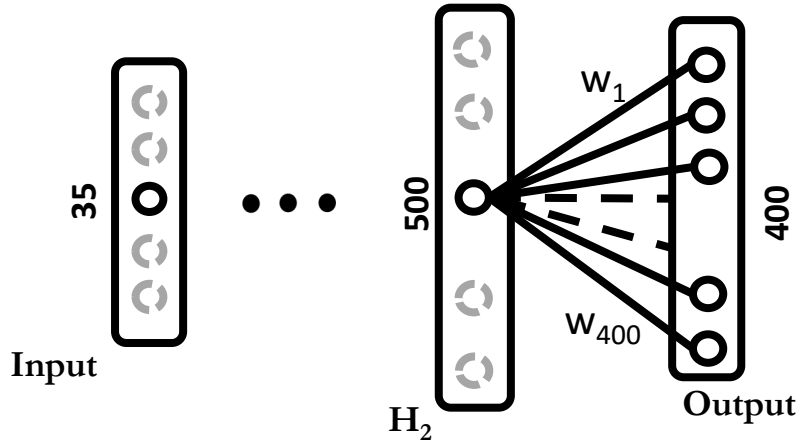


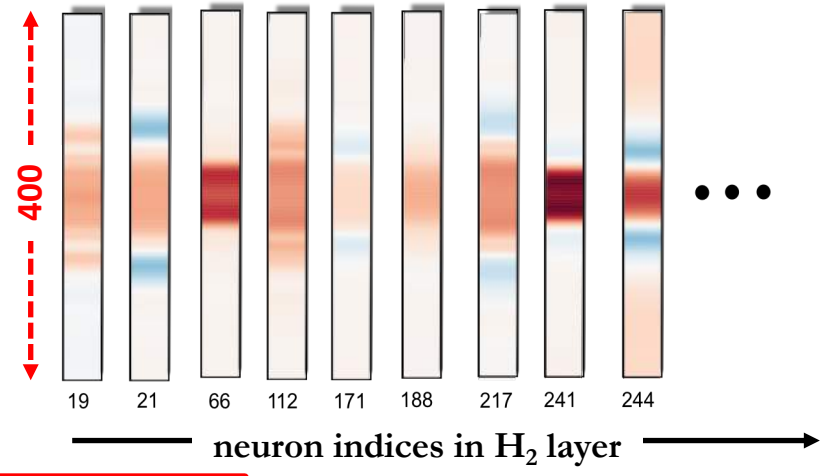
Correlated Parameters
• k_24cm0, k_24cm1
• k_42a, k_42d
• q, h

Insufficient parameter ranges
• k_24cm0, k_24cm1, k_42a, k_42d, q, h, C42_t

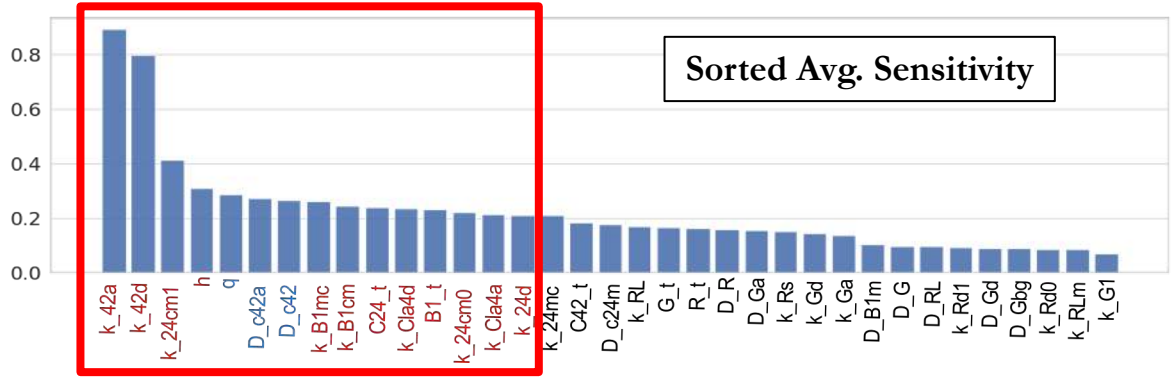
Case Study 2: Analyzing the trained surrogate

- Final Weight Matrix





- Patterns with high weights to the center neurons
- Avg. Parameter sensitivity to such patterns
- Top 15 aligns with the domain knowledge



Conclusion and Future Work

- Neural network-assisted analysis backend for interactive visual analytics
- Utilized post-hoc analysis operation on trained model to facilitate scientific enquires
- [Future] Investigate more effective model interpretation techniques. Currently, weight matrix analyses require ML knowledge
- [Future] Explore additional post-hoc analysis techniques for neural networks to interpret abstract domain level scientific concepts from the model