Bayesian Algorithms for Practical Accelerator Control and Adaptive Machine Learning for Time-Varying Systems

Alexander Scheinker (LANL) and Ryan Roussel (SLAC)

NAPAC 2022, ABQ, NM

Part 1: Bayesian Methods

Bayesian algorithms for practical accelerator control

Ryan Roussel

8/12/2022

rroussel@slac.stanford.edu

U.S. DEPARTMENT OF ENERGY | Stanford University

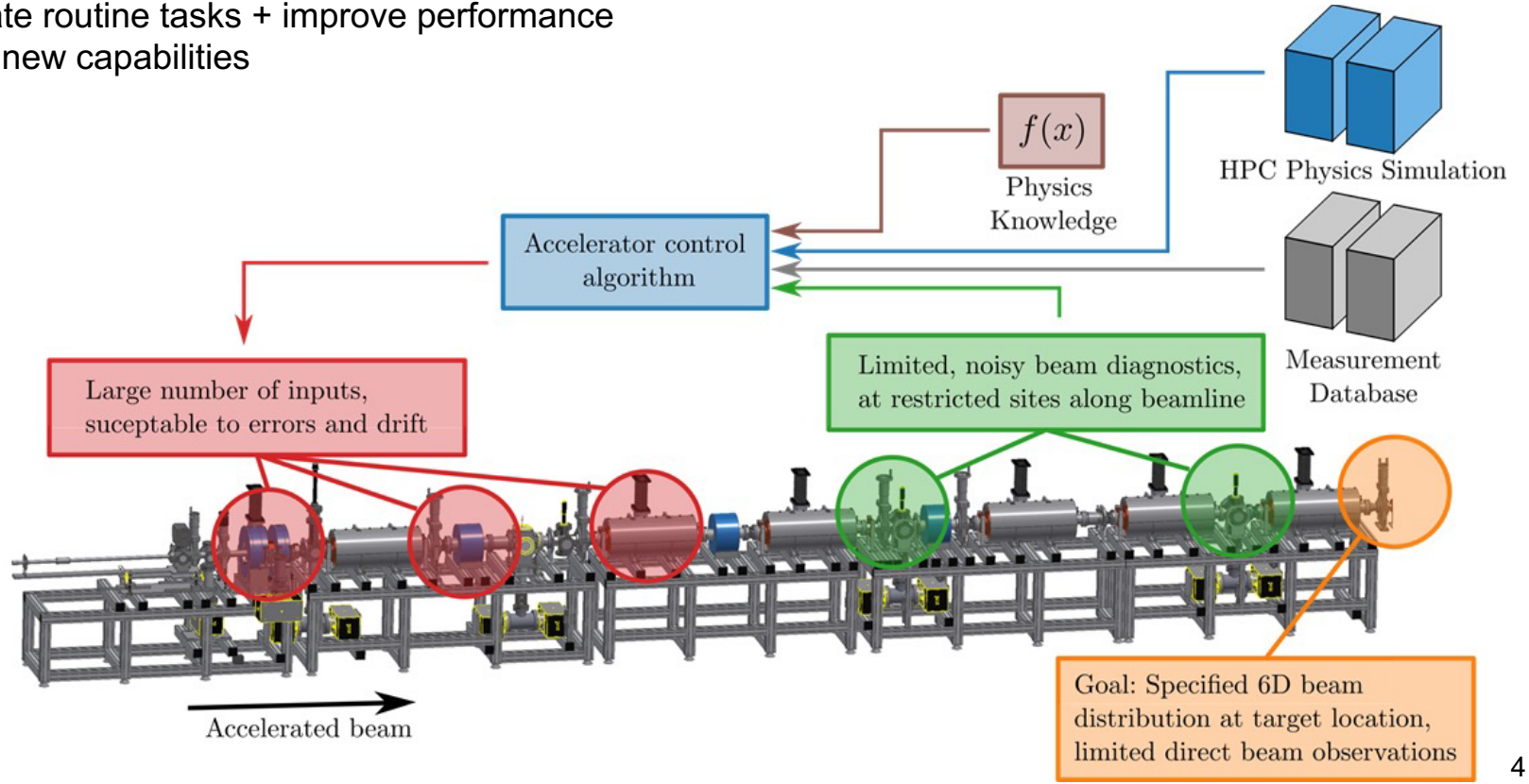
SLAC NATIONAL ACCELERATOR LABORATORY

Machine Learning Based Accelerator Control

SLAC

Goals:

- Automate routine tasks + improve performance
- Enable new capabilities



Machine Learning Based Accelerator Control

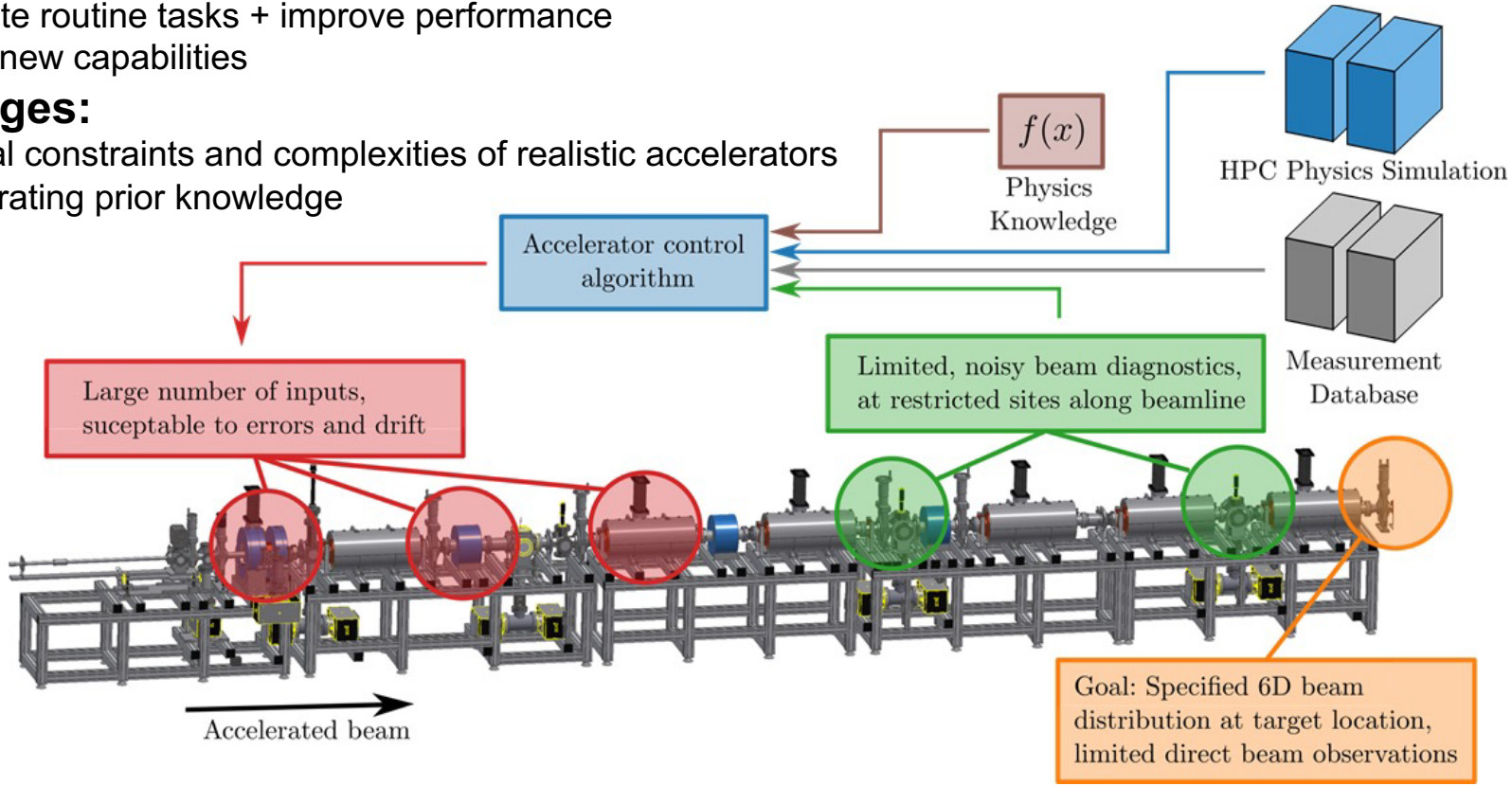
SLAC

Goals:

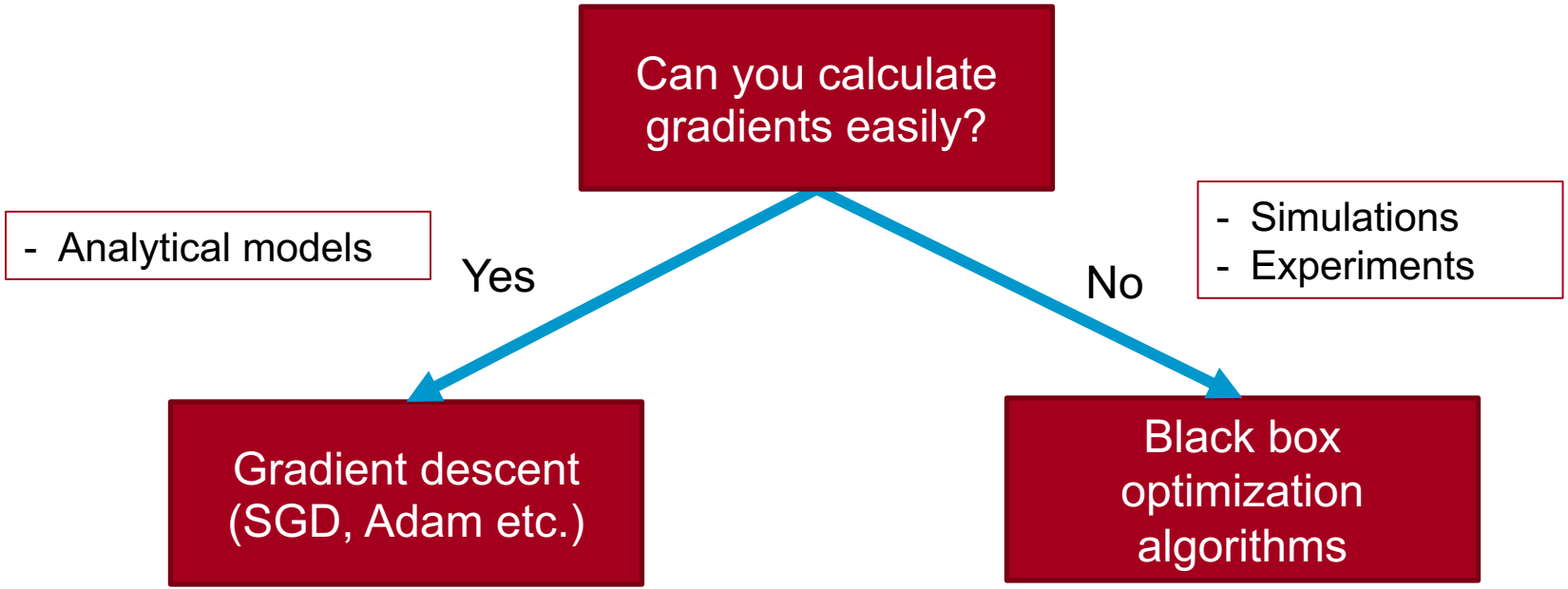
- Automate routine tasks + improve performance
- Enable new capabilities

Challenges:

- Practical constraints and complexities of realistic accelerators
- Incorporating prior knowledge
- Scaling



Global Optimization Strategies



Model Based Optimization of Accelerators

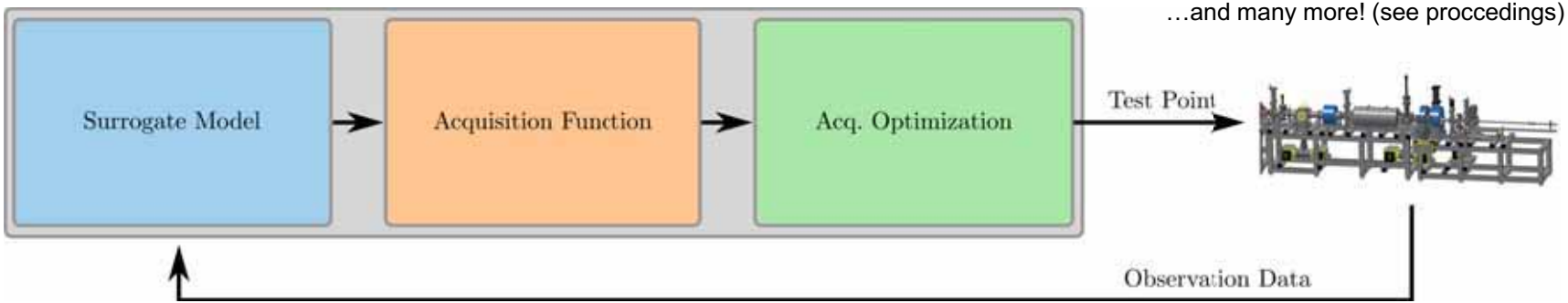
Can we improve black box optimization with surrogate models? **YES**

Create a computational model of the system

Determine the value of potential future measurements

Pick the point that maximizes value

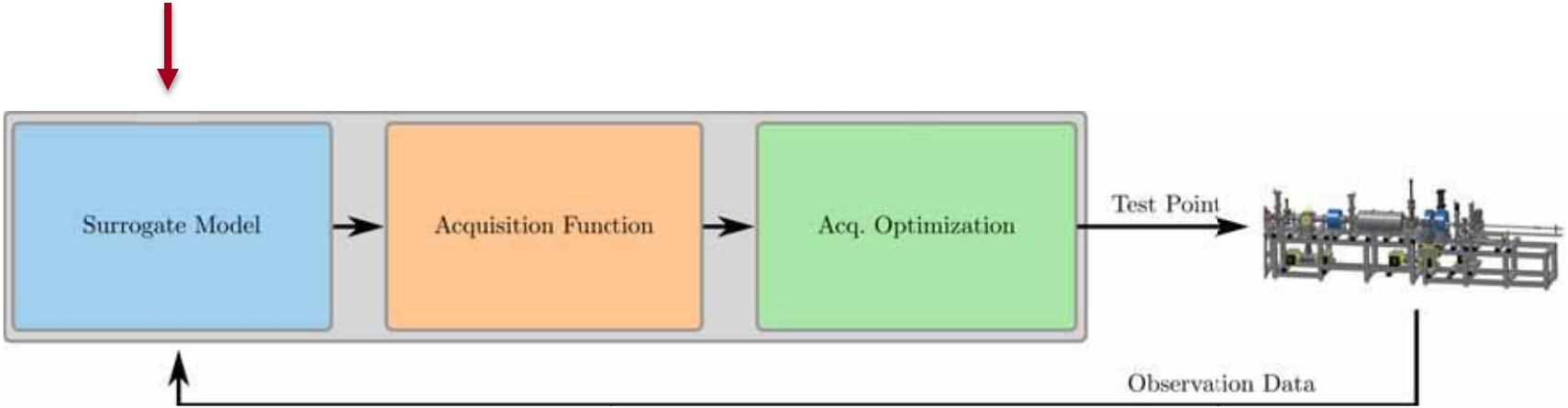
Duris et al. PRL 2020
Roussel et al. PRAB 2021
Hanuka et al. PRAB 2021
Miskovich et al. PRAB 2021
…and many more! (see proccedings)



… **but** we must consider the cost of creating models with enough information

Model Based Optimization of Accelerators

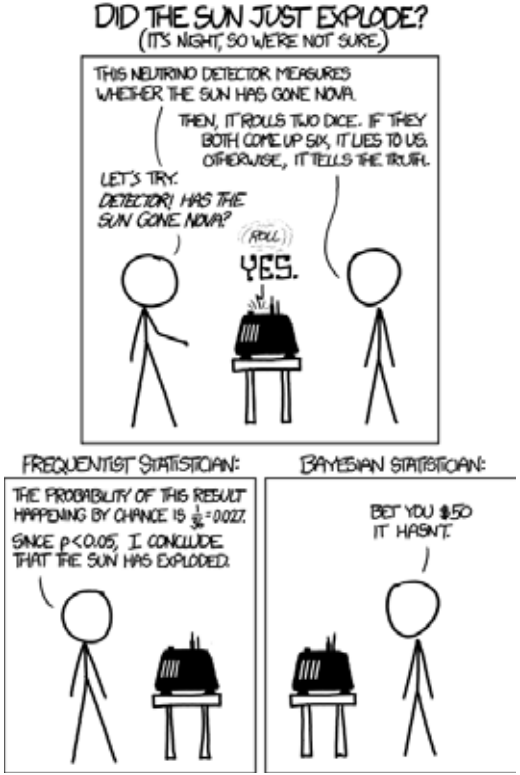
Gaussian processes (GPs)



Why?

- Extracts a lot of information from a small number of data points → efficient
- Produces explicit uncertainty estimations -> advantageous for global optimization

Bayesian Statistics: A Practical Example



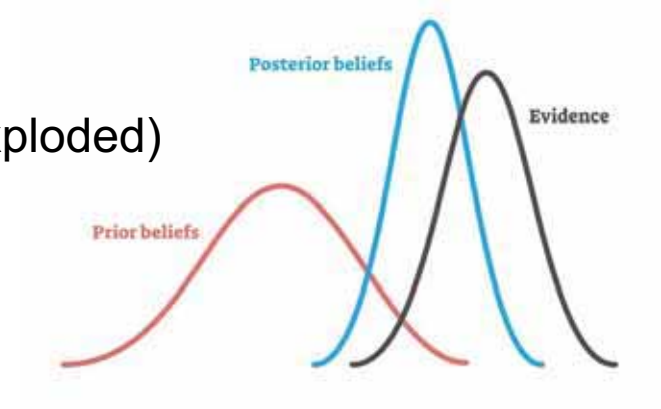
https://xkcd.com/1132/

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

posterior: $p(B|A)$ — likelihood: $p(A|B)$ — prior: $p(B)$ — marginal likelihood: $p(A)$

A: our measurement
(we measure that the sun has exploded)

B: our prediction
(the sun has exploded)



Bayesian Statistics: A (More) Practical Example

Model:

$$y = w_0 + w_1 x + \epsilon$$

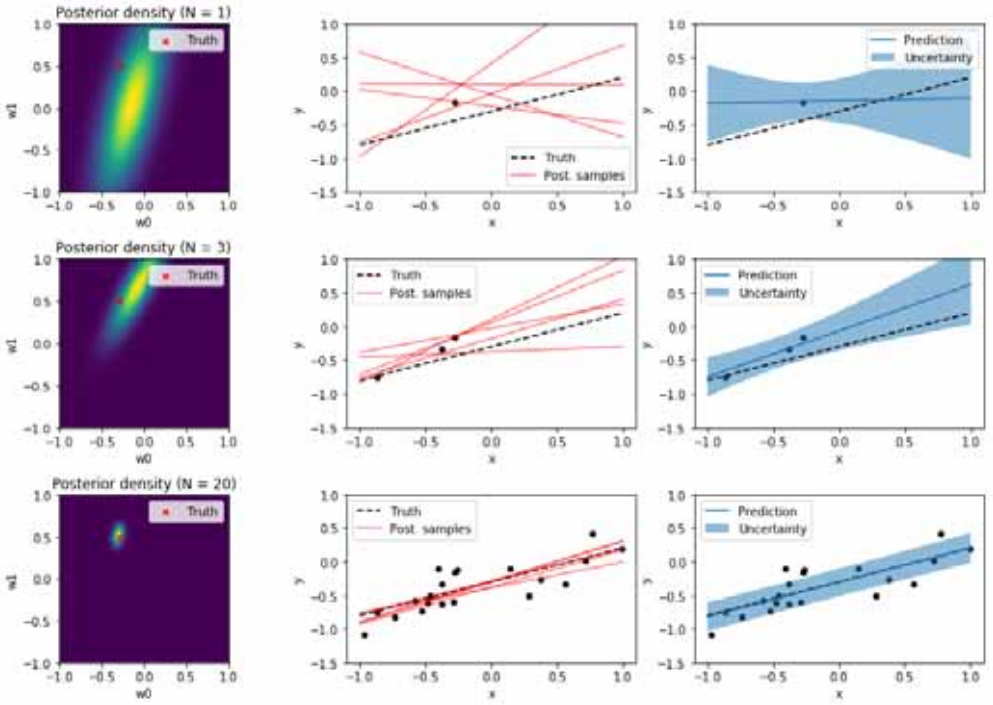
Bayesian regression (determining the weights):

$$p(w_0, w_1 | \boldsymbol{y}_N, \boldsymbol{x}_N) \propto p(\boldsymbol{y}_N | w_0, w_1, \boldsymbol{x}_N) p(w_0, w_1)$$

Making predictions:

$$p(\boldsymbol{y}_M | w_0, w_1, \boldsymbol{x}_M)$$

Note: least squares fitting is equivalent to using a uniform prior and Gaussian likelihood

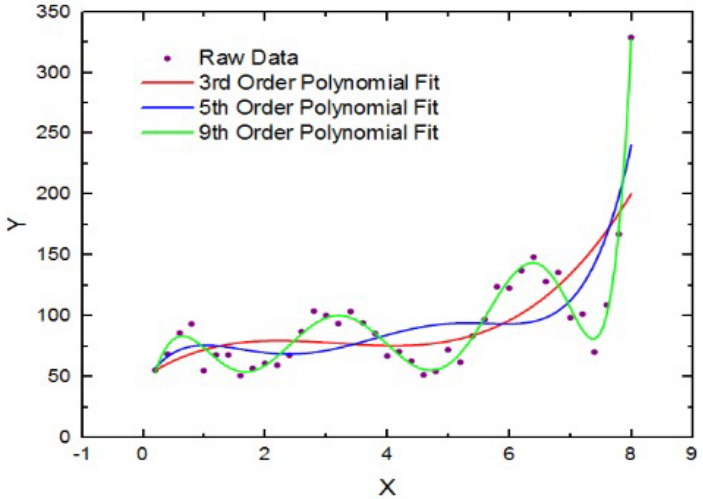


M. Krasser. https://nbviewer.org/github/krasserm/bayesian-machine-learning/blob/dev/bayesian-linear-regression/bayesian_linear_regression.ipynb

Parametric vs. Non-Parametric Modeling

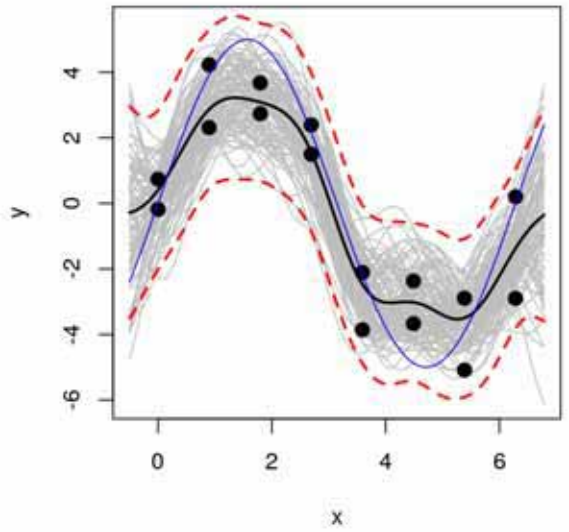
SLAC

Parametric modeling



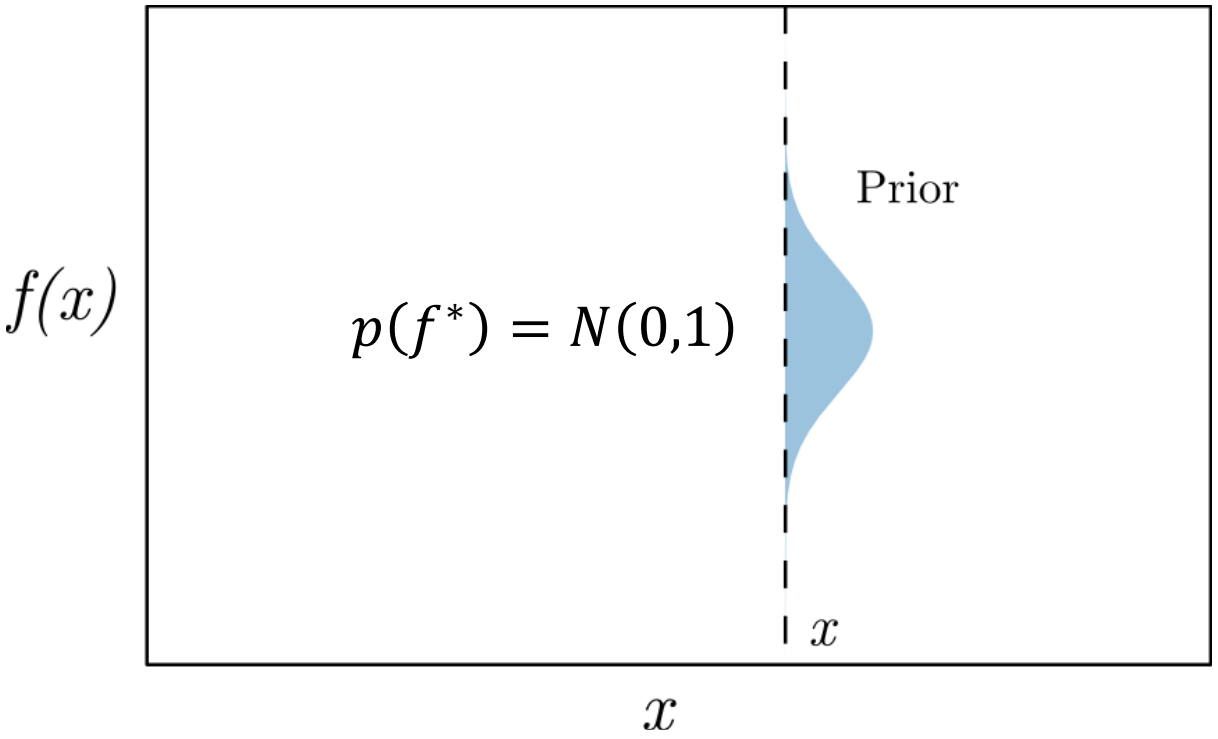
$$f(x) = f(x;\theta)$$

Non-Parametric modeling



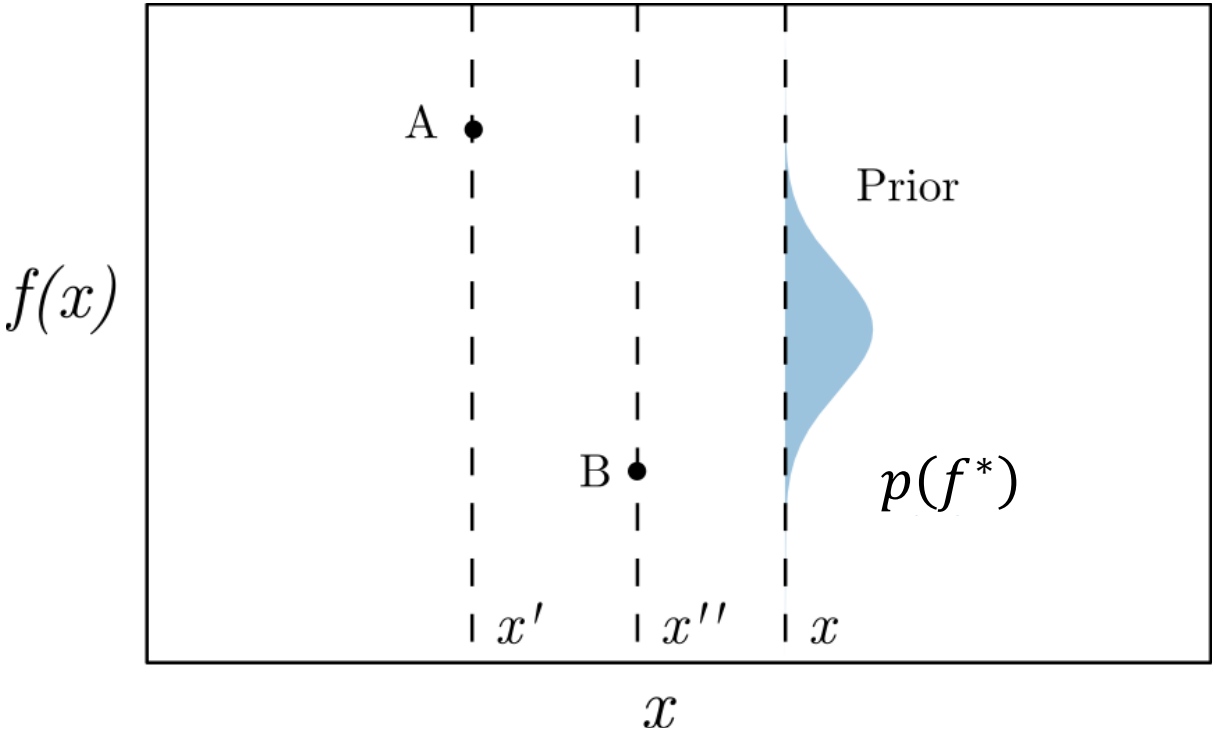
$$f = \{\theta_0(x_0), \theta_1(x_1), \ldots, \theta_N(x_N)\}$$

Some intuition…



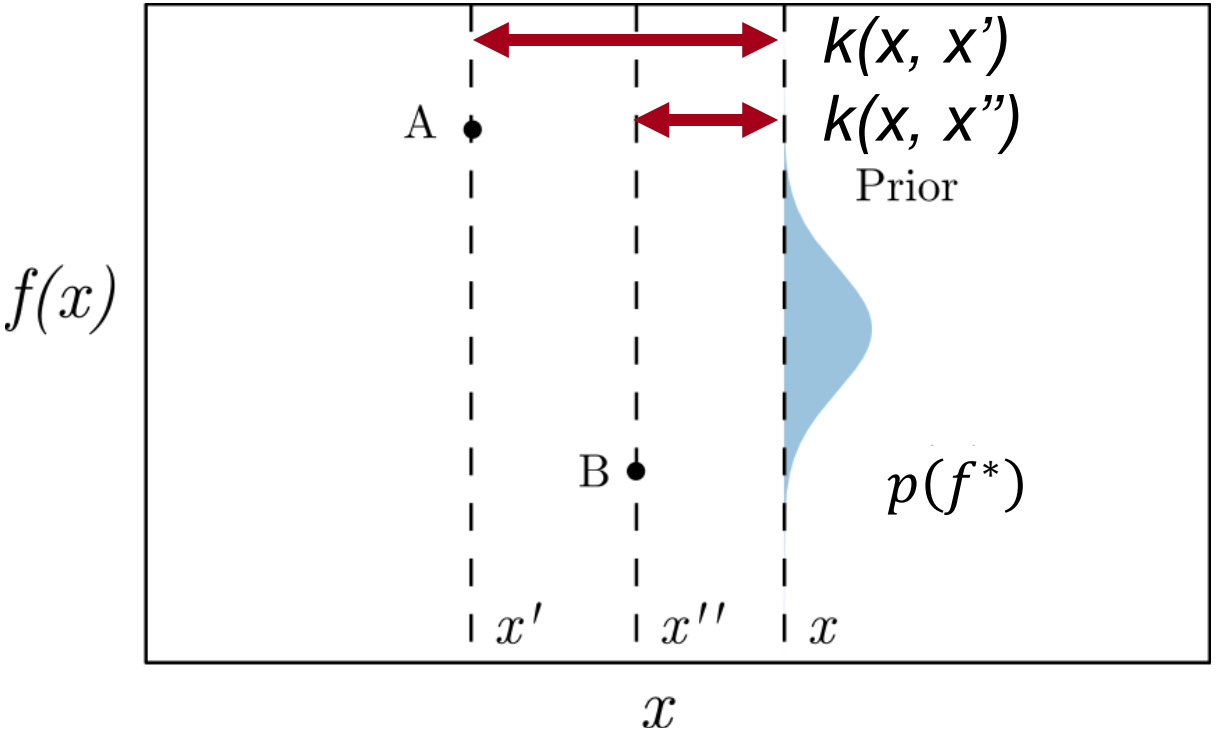
Some intuition…

Which observation will have a larger impact on changing *p(f)*?



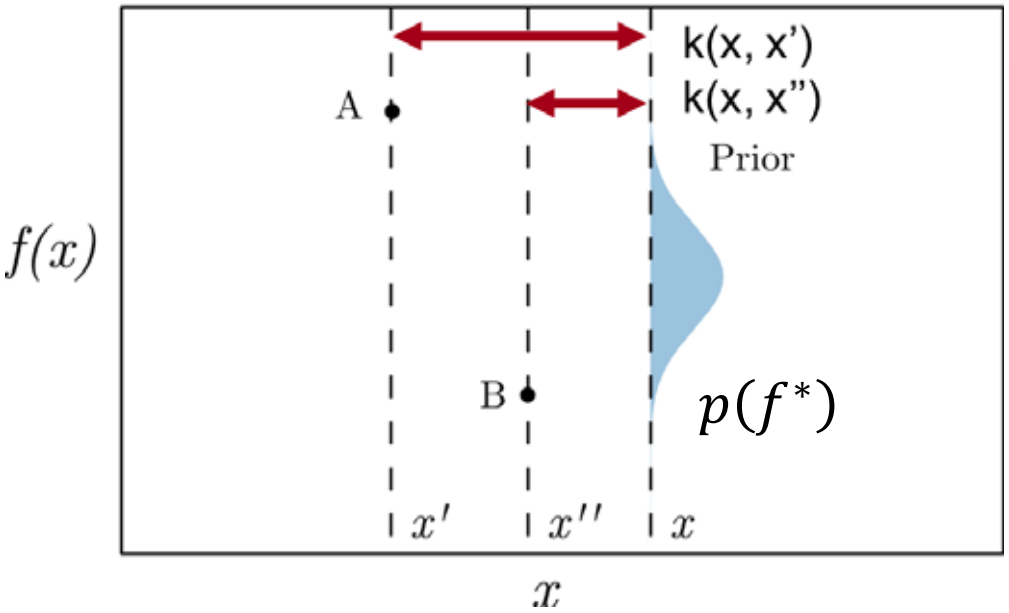
Adding some math

Which observation will have a larger impact on changing *p(f)*?

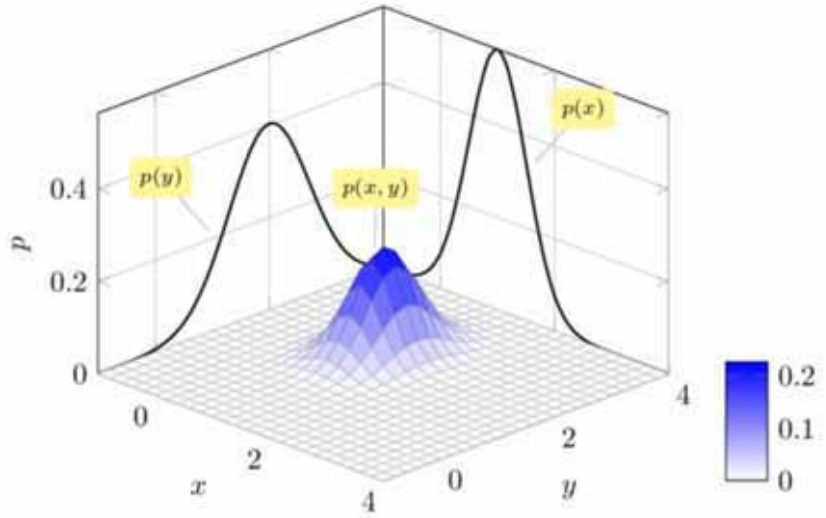


$k(x, x') < k(x, x'')$

Adding some math

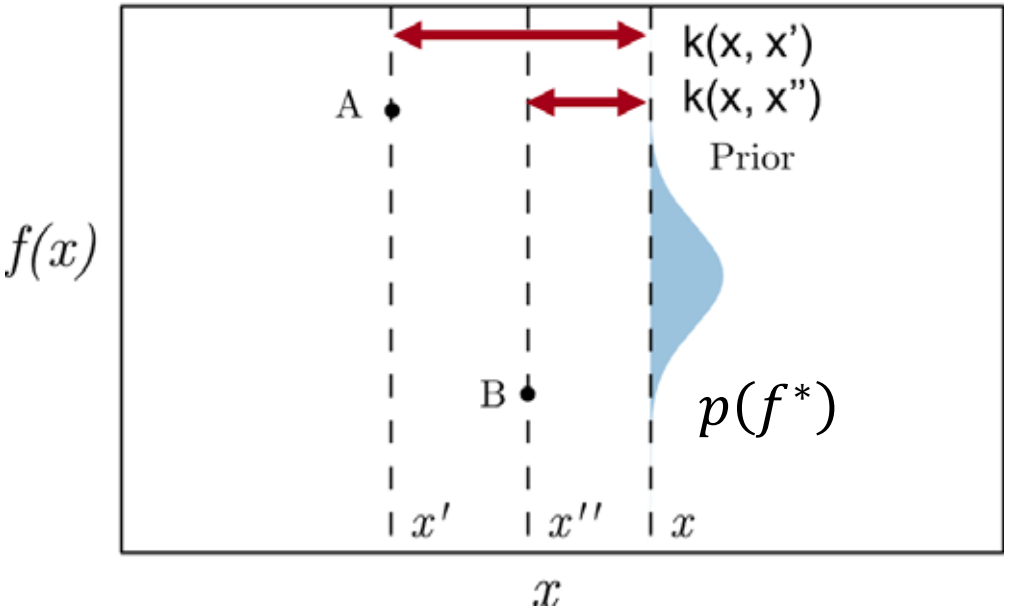


$$p(f_A, f_B, f^*) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$



$$p(f^* \mid f_A, f_B) = \frac{p(f_A, f_B \mid f^*)p(f^*)}{p(f_A, f_B)}$$

Adding some math

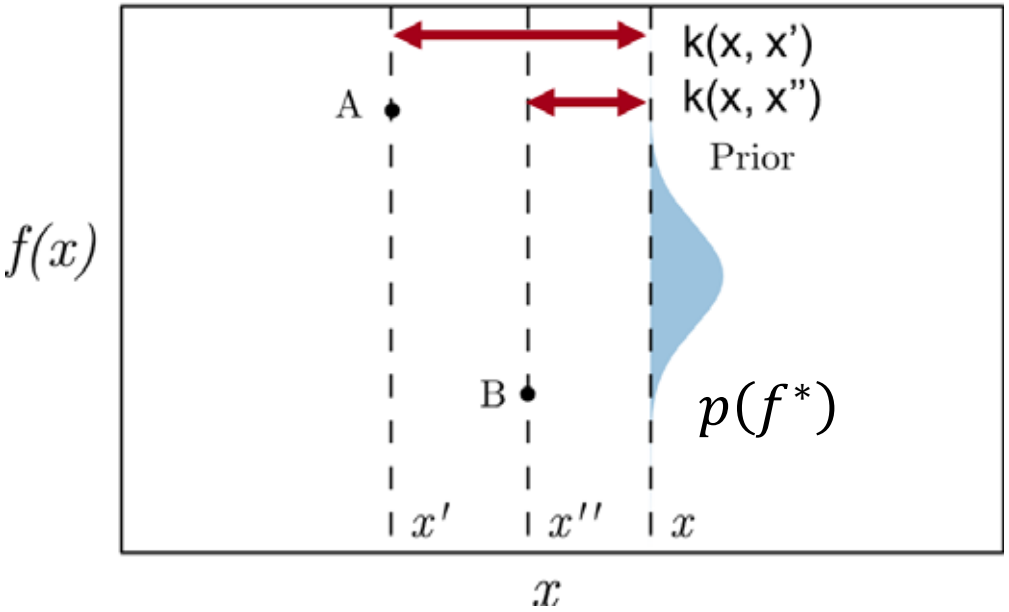


$$p(f_A, f_B, f^*) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} k(x', x') & k(x', x'') & k(x', x) \\ k(x', x'') & k(x'', x'') & k(x'', x) \\ k(x', x) & k(x'', x) & k(x, x) \end{pmatrix}$$

$$p(f^* | f_A, f_B) = \frac{p(f_A, f_B | f^*) p(f^*)}{p(f_A, f_B)}$$

Adding some math



$$p(f^* \mid f_A, f_B) = N(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)$$

$$\boldsymbol{\mu}^* = \boldsymbol{\mu} + K^* K^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$$
$$\boldsymbol{\sigma}^* = K^{**} - K^{*T} K^{-1} K^*$$

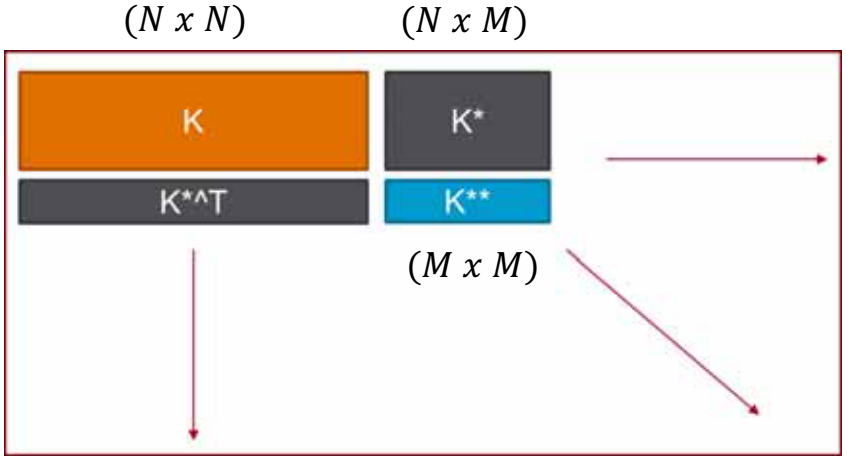
$$p(f^* \mid f_A, f_B) = \frac{p(f_A, f_B \mid f^*) p(f^*)}{p(f_A, f_B)}$$

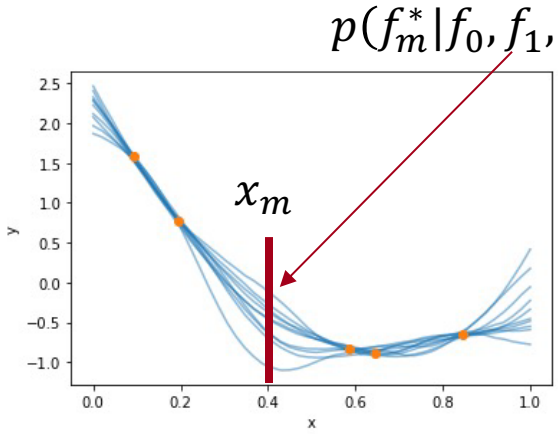
$K^{-1} \sim \mathcal{O}(N^3)$!

Making predictions with GP's

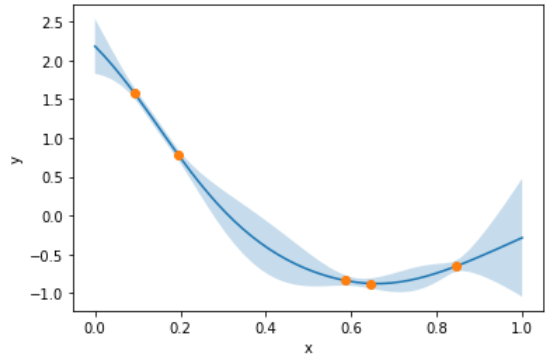
What about multiple predictions?

$$p(f_0^*, f_1^*, \ldots, f_M^* | f_0, f_1, \ldots, f_N) = N(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)$$





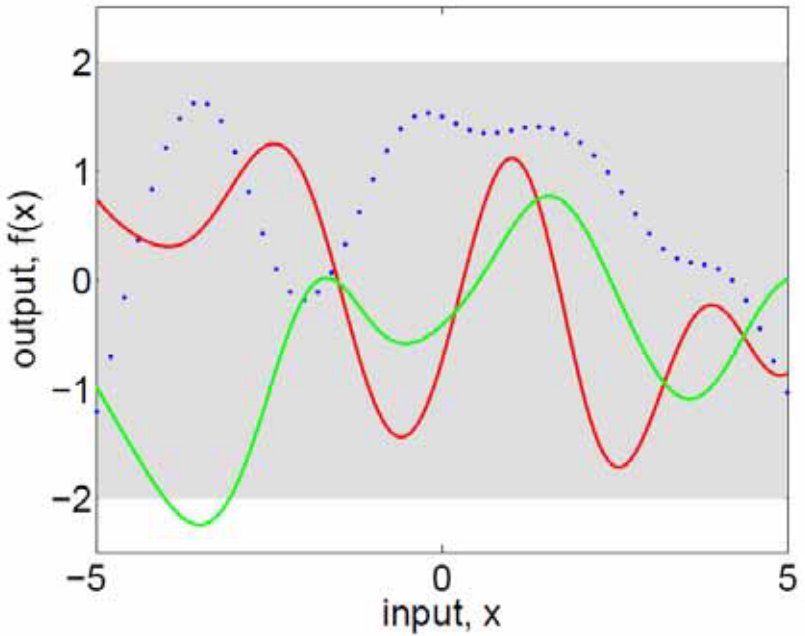
Samples



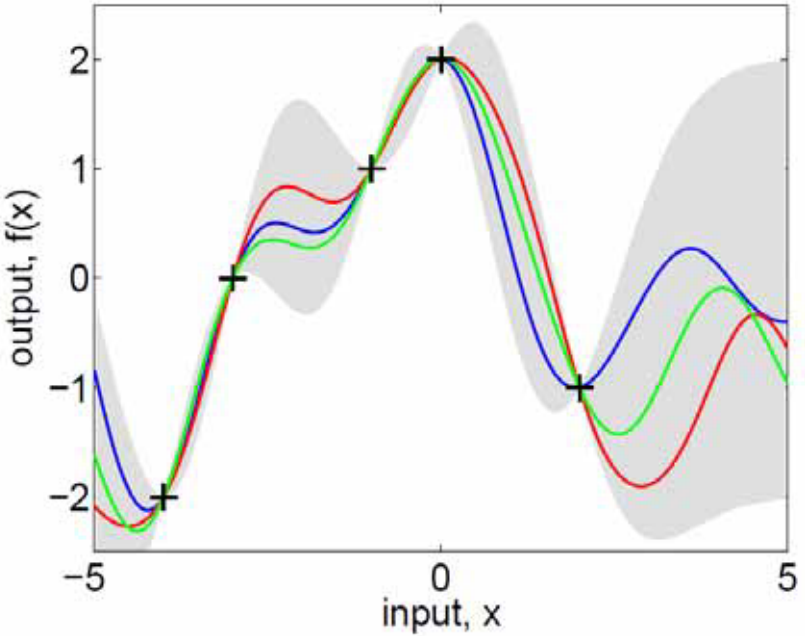
Distribution

Draw function samples? Sample from the joint posterior distribution at requested points

Another perspective



(a), prior



(b), posterior

Rasmussen and Williams. 2006

Fitting Gaussian Processes to Data

Need to determine the form and hyperparameters of the kernel

Each kernel has hyperparameters that control the overall function behavior.

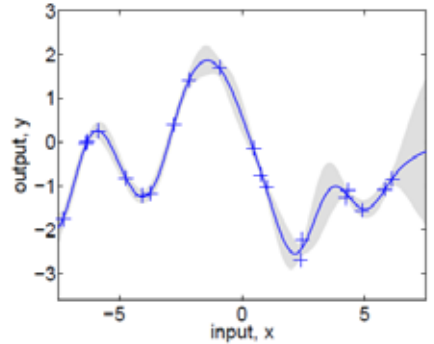
Radial Basis Function:

$$k(x, x') = \boldsymbol{\sigma_f^2} \exp\left(-\frac{1}{2\boldsymbol{l}^2}(x - x')^2\right) + \boldsymbol{\sigma_n^2}\delta_{xx'}$$

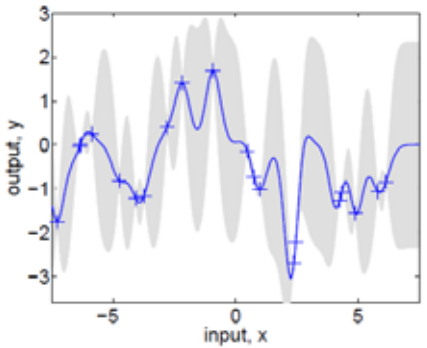
Kernel amplitude

Kernel length scale

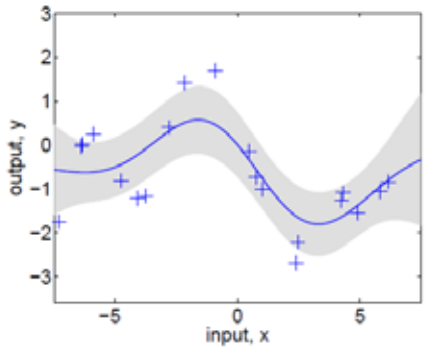
Noise



(a), $\ell = 1$



(b), $\ell = 0.3$



(c), $\ell = 3$

R&W

Fitting Gaussian Processes to Data

Need to determine the form and hyperparameters of the kernel

We fit kernel parameters to data by using Maximum Likelihood Estimation (MLE or MLE-II).

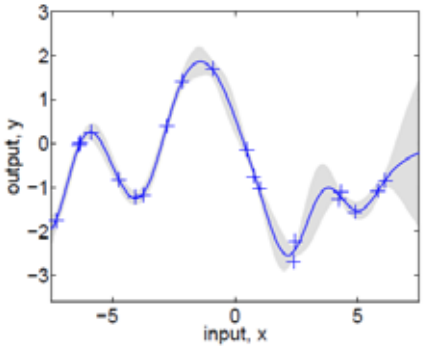
$$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{X})$$

With a Gaussian likelihood the log likelihood can be calculated analytically:

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{X}) = -\frac{1}{2} y^T K^{-1} y \; - \frac{1}{2} \log|K| - \frac{n}{2} \log 2\pi$$

Predictive accuracy

Model complexity



(a), $\ell = 1$

More Expressive Kernels

We can also encode low dimensional structure into the kernel.

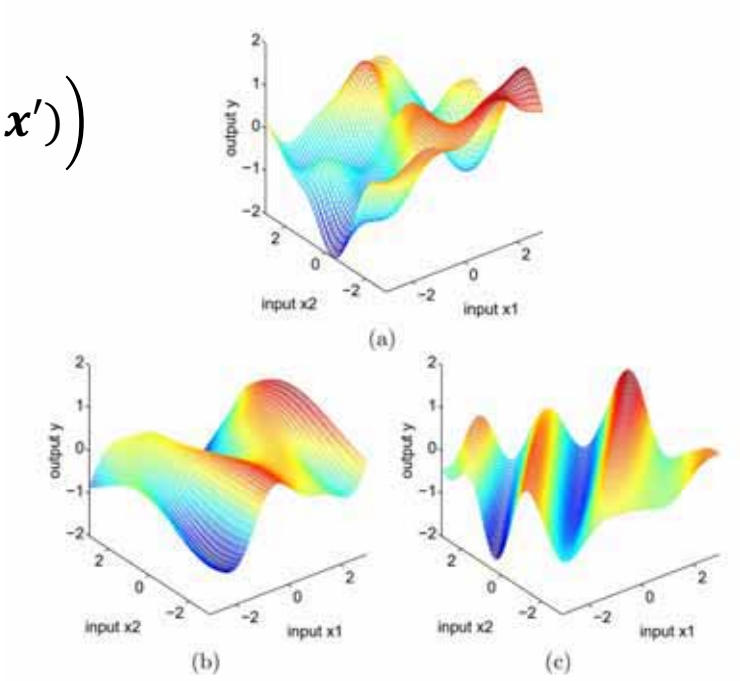
$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{\Sigma} (\boldsymbol{x} - \boldsymbol{x}')\right)$$

Automatic relevance determination

$$\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{l})^{-2}$$

Factor analysis distance

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \mathrm{diag}(\boldsymbol{l})^{-2}$$



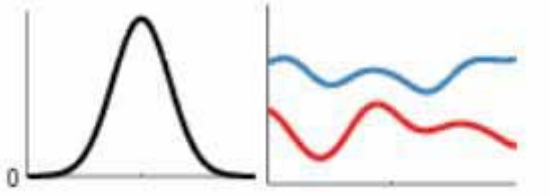
Rasmussen and Williams. 2006

Fitting Gaussian Processes to Data

Need to determine the form and hyperparameters of the kernel

Radial Basis Function:

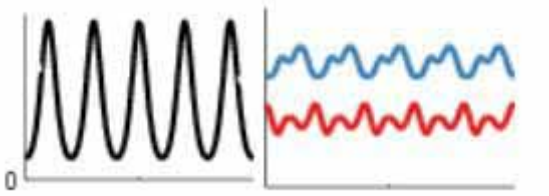
$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$$



Stationary

Periodic:

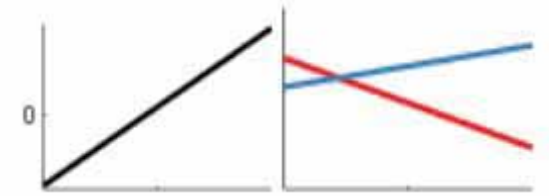
$$k_{\mathrm{Per}}(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)}{\ell^2}\right)$$



Stationary

Linear:

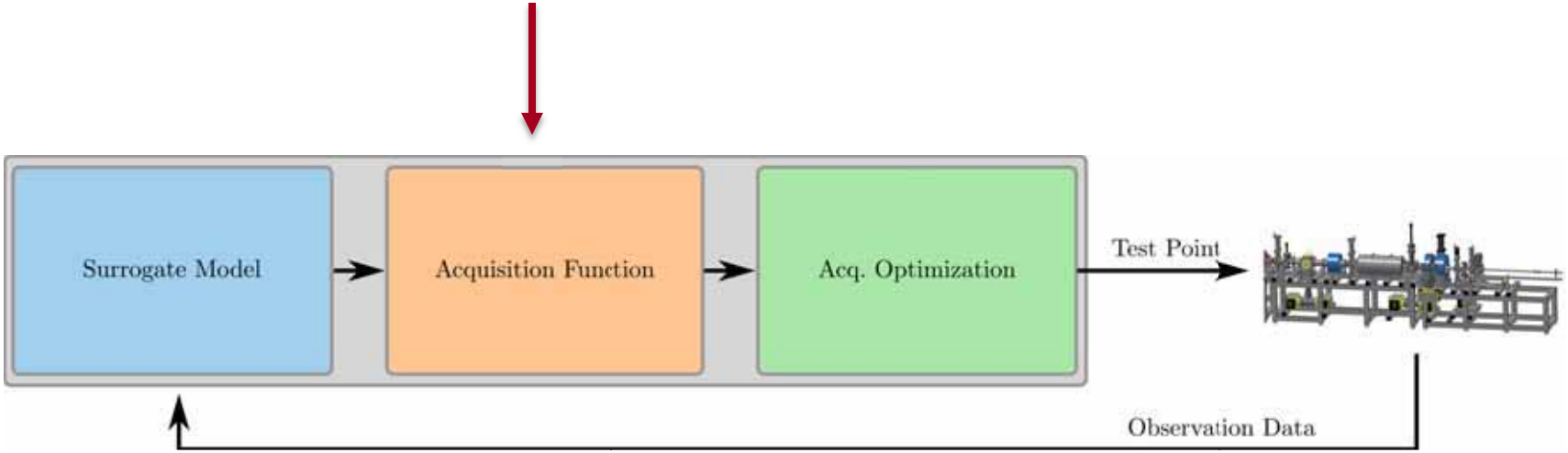
$$k_{\mathrm{Lin}}(x, x') = \sigma_b^2 + \sigma_v^2(x-c)(x'-c)$$



Non-stationary

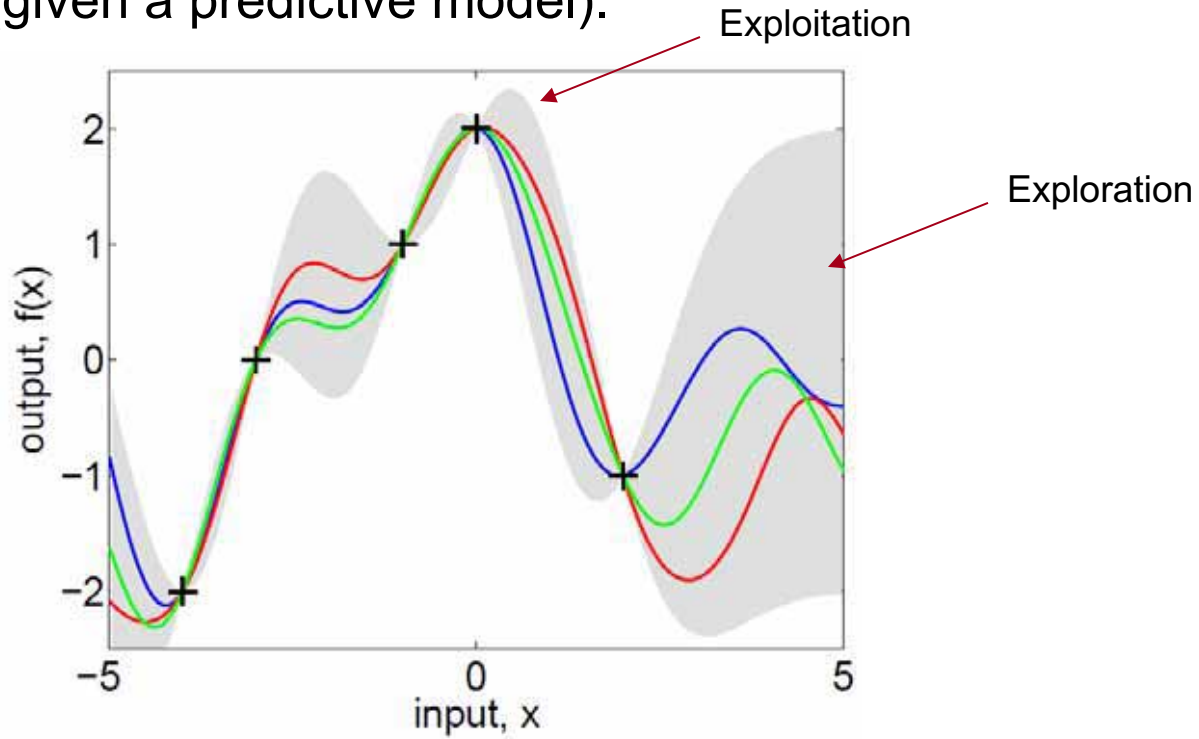
D. Duvenaud, https://www.cs.toronto.edu/~duvenaud/cookbook/

Model Based Optimization of Accelerators



The Acquisition Function

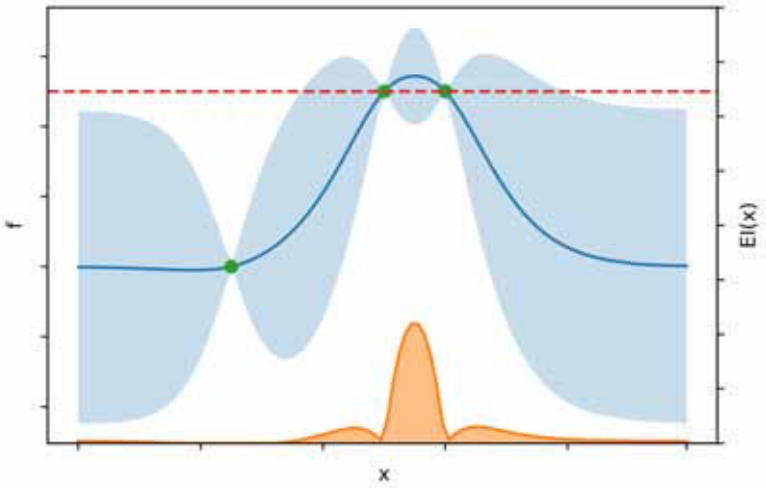
Define a function that characterizes the value of making a potential measurement (given a predictive model).



Single Objective Optimization

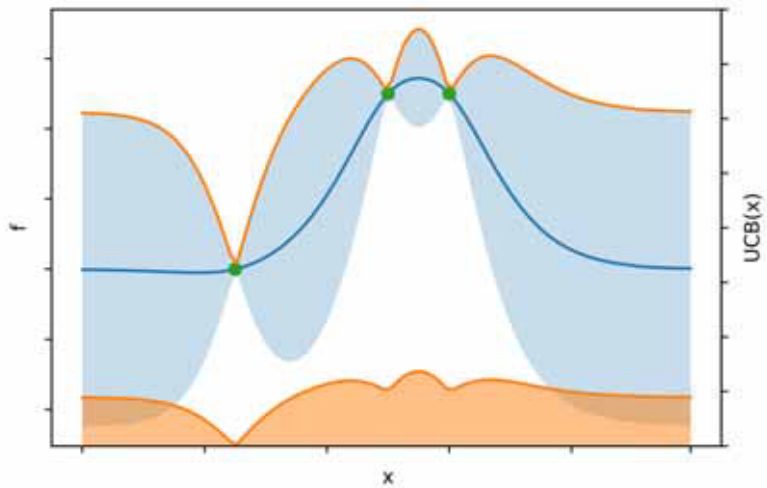
SLAC

Expected Improvement



$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}[\max(f(\boldsymbol{x}) - f^*)]$$

Upper Confidence Bound



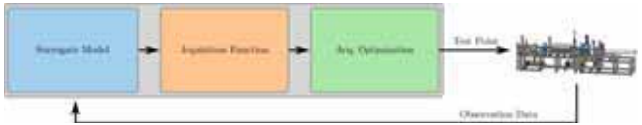
$$\mathrm{UCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \sqrt{\beta}\sigma(\boldsymbol{x})$$

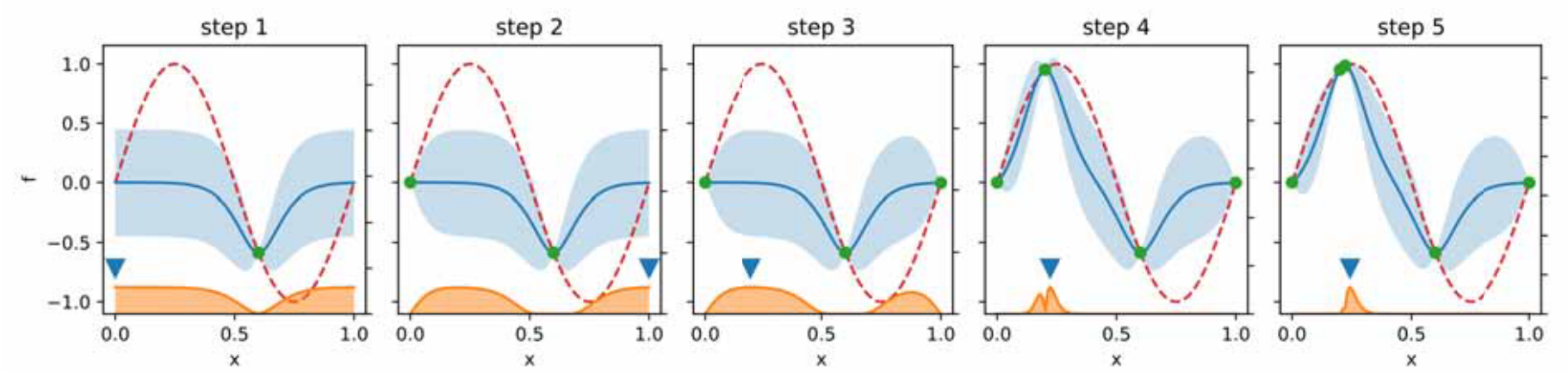
Note: most implementations of BO assume maximization

Single Objective Optimization

$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}[\max(f(\boldsymbol{x}) - f^*)]$$

$$\boldsymbol{x}_{t+1} = \mathrm{argmax}_{\boldsymbol{x}}\mathrm{EI}(\boldsymbol{x})$$



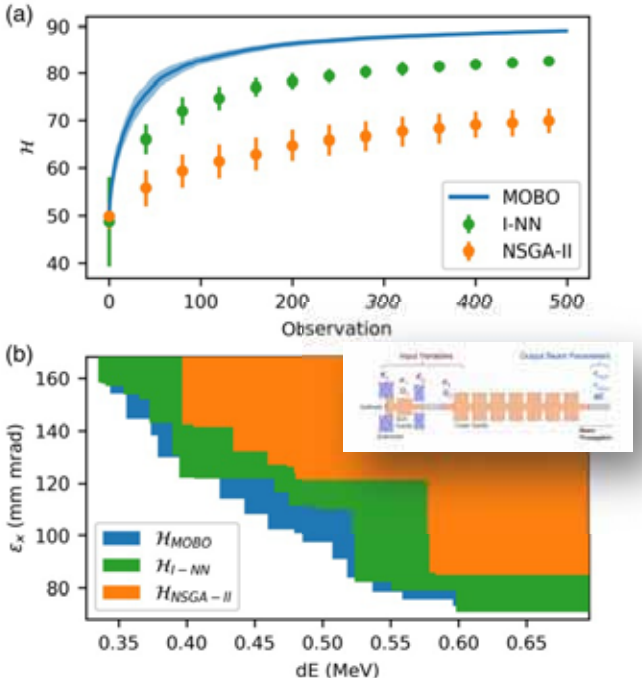


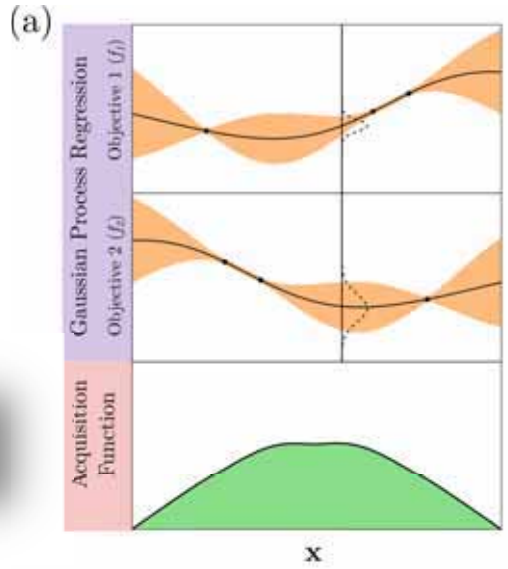
Some notes:

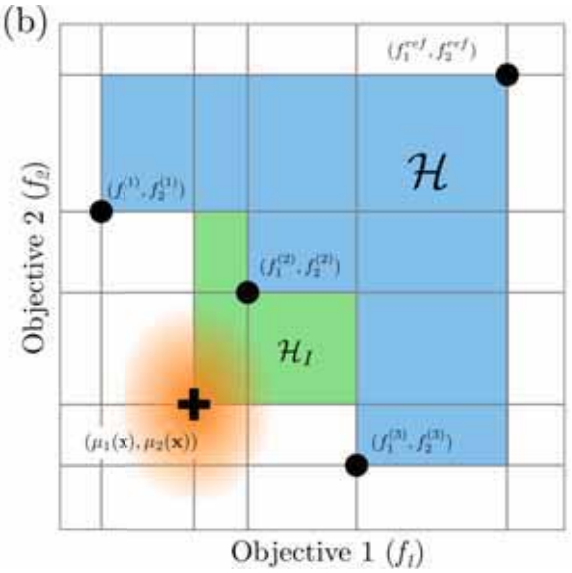
- The model accuracy improves in the region of interest!
- Initially the model uncertainty is maximized at the domain boundaries, BO likes to sample those
- Helpful if the acquisition function is differentiable → use gradient descent to optimize

Multi-Objective Optimization

Determine the optimal trade-off between objectives -> the Pareto front







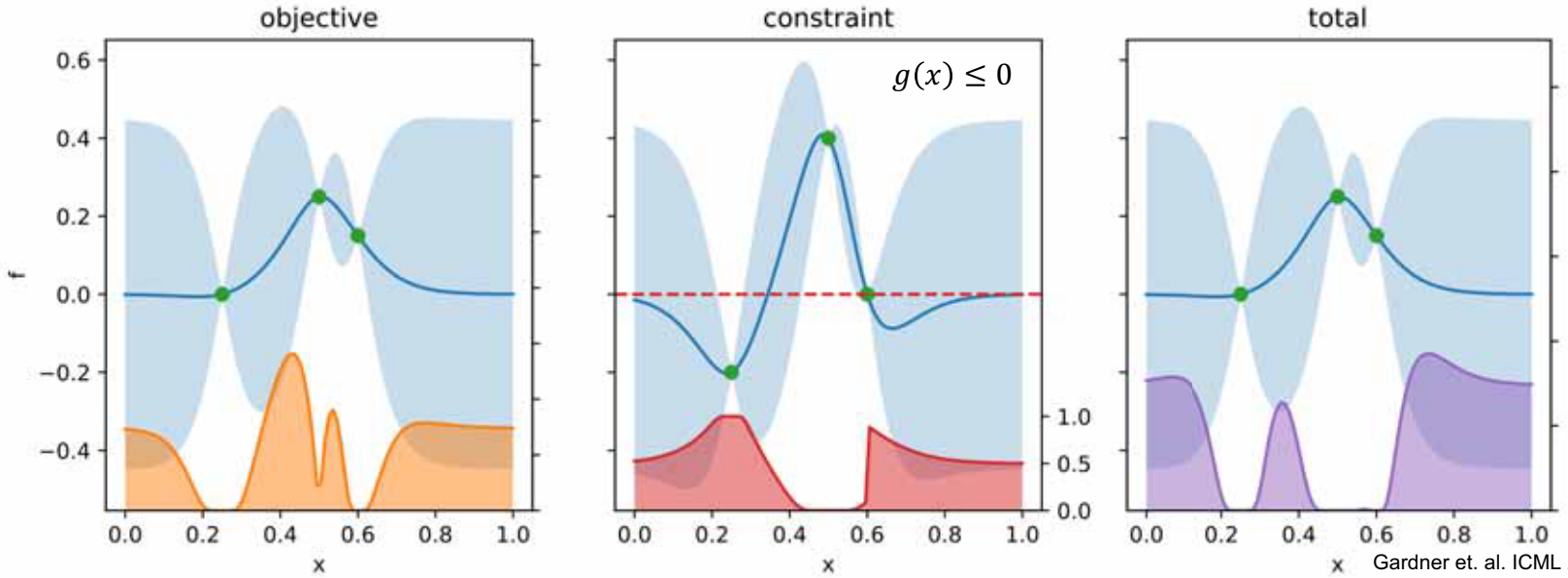
$$\alpha_{EHVI}(\boldsymbol{\mu},\boldsymbol{\sigma},\mathcal{P},\boldsymbol{r}) := \int_{\mathbb{R}^P} \mathrm{HVI}(\mathcal{P},\boldsymbol{y},\boldsymbol{r}) \cdot \boldsymbol{\xi}_{\mathbb{N},\boldsymbol{\sigma}}(\boldsymbol{y}) d\boldsymbol{y}$$

Roussel et. al. PRAB 2021

Incorporating Constraints

Weight the acquisition function by the probability that constraints are satisfied

$$\alpha(x) \rightarrow \alpha(x) \prod_i p[g_i(x) \leq h_i]$$ Warning: Heuristic / math abuse!

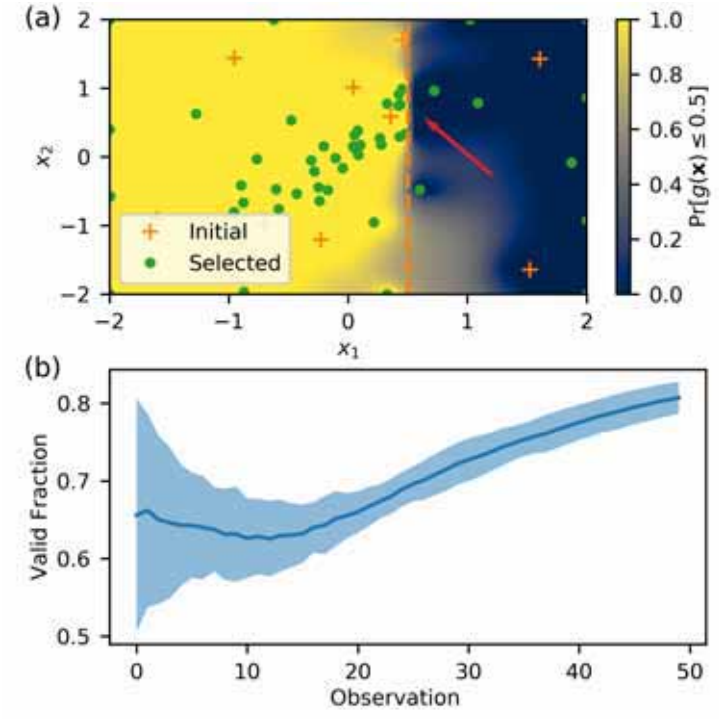


Gardner et. al. ICML 2014

Incorporating Constraints

Weight the acquisition function by the probability that constraints are satisfied

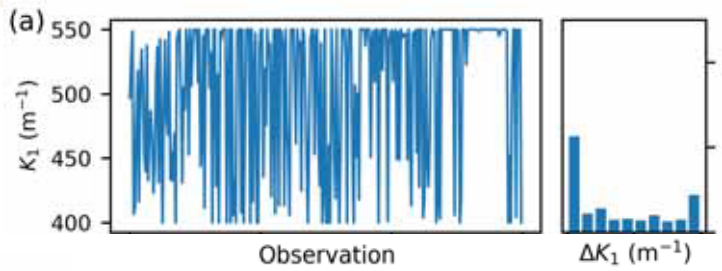
$$\alpha(x) \rightarrow \alpha(x) \prod_i p[g_i(x) \leq h_i]$$



Roussel et. al. PRAB 2021

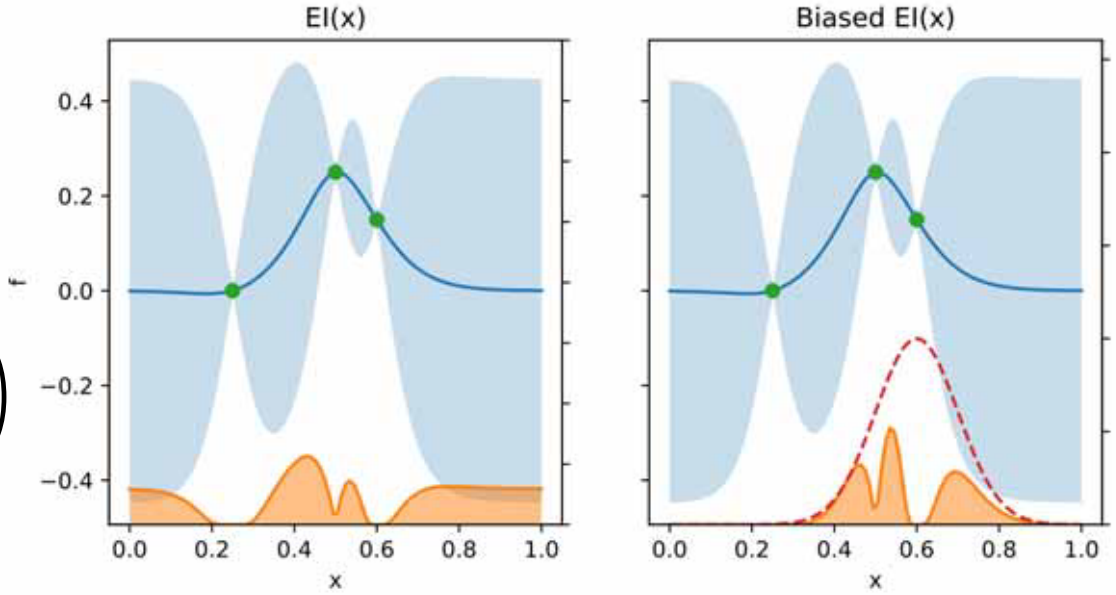
Proximal Biasing

Poor optimization behavior for experimental beamlines



Weight the acquisition function by travel distance → better than hard limits

$$\alpha(x) \rightarrow \alpha(x) \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right)$$

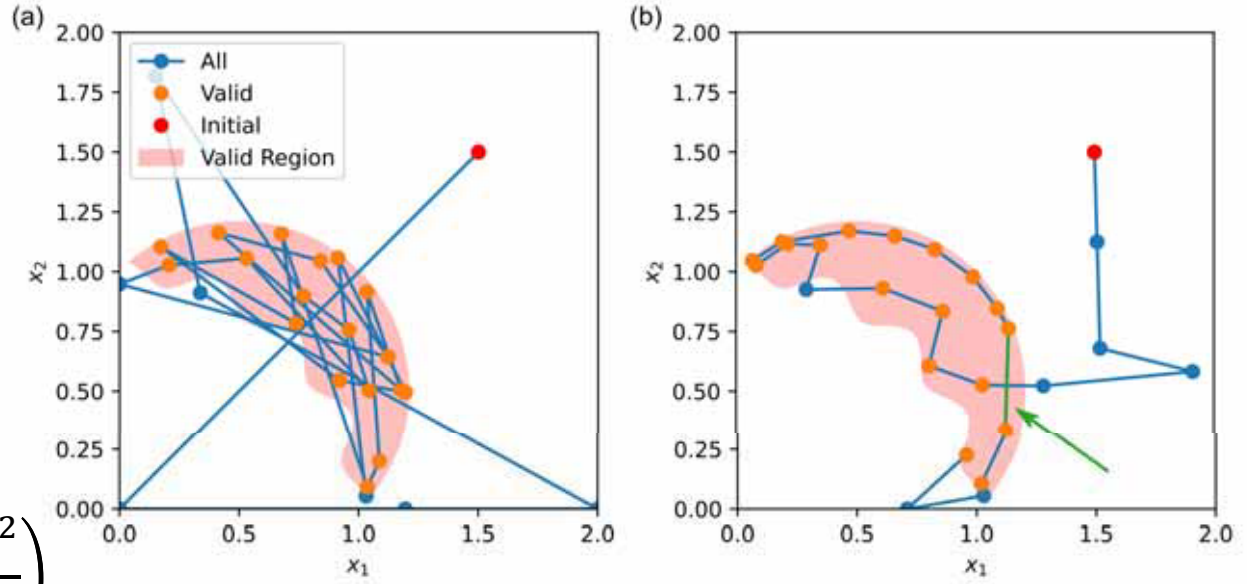


Proximal Biasing

Poor optimization behavior for experimental beamlines

Weight the acquisition function by travel distance → better than hard limits

$$\alpha(x) \rightarrow \alpha(x) \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right)$$

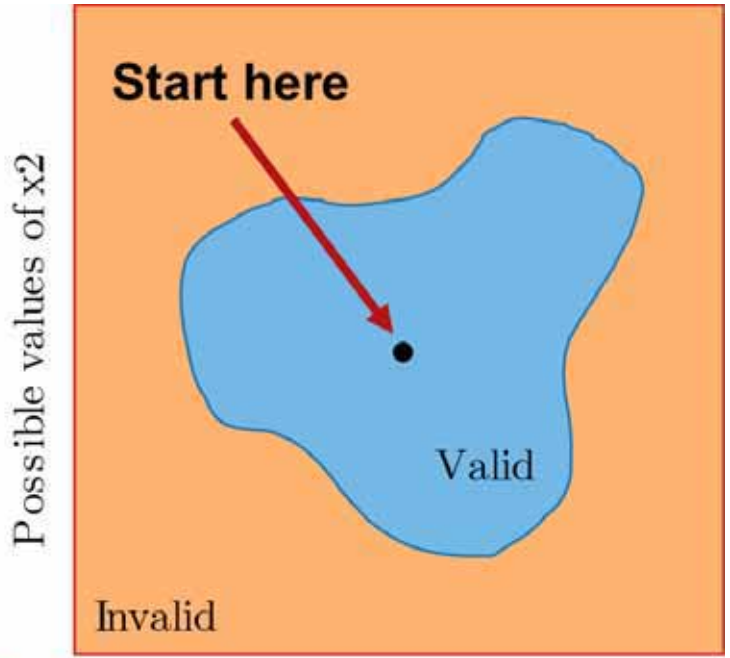


Roussel et. al. Nat. Comm. 2021

Case study: Bayesian Exploration

Autonomous characterization of novel systems.

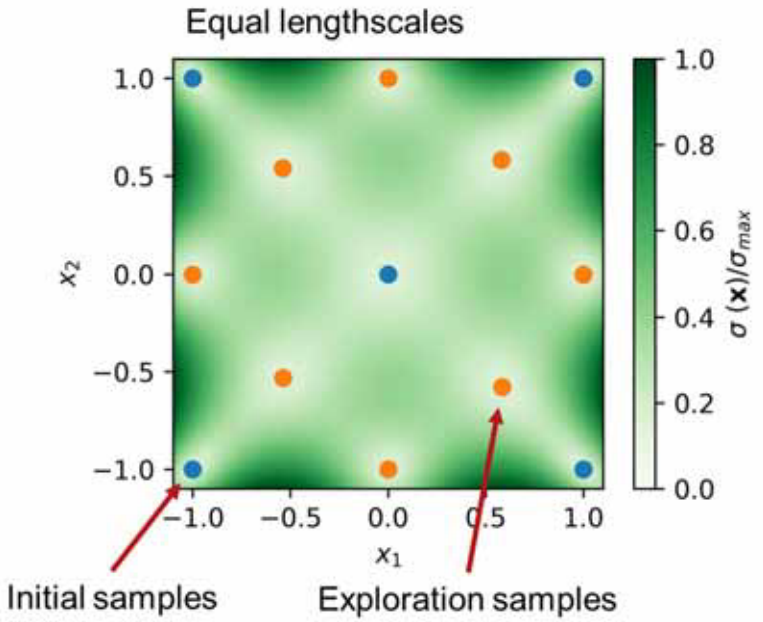
- Starts with a single valid observation AND **no prior information except for hardware limitations**
- Samples points in a quasi-uniform grid where the grid spacing is learned automatically based on the beam response
- Learns where the valid region is w/limited invalid observations
- Considers costs associated with changing parameters
- Natively applicable to multi-dimensional exploration
- General purpose that works in almost any case



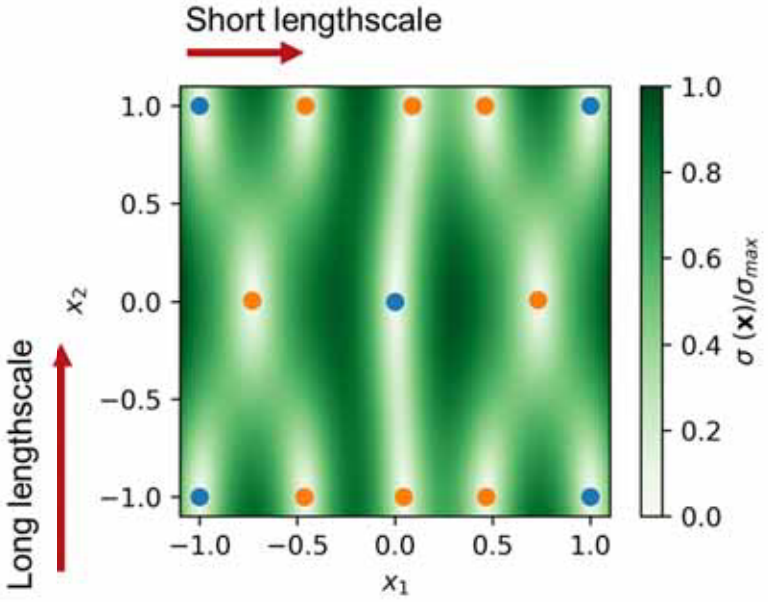
Click button → GO!

Uncertainty Sampling

If the function changes more rapidly along one axis, sample more points along that axis!



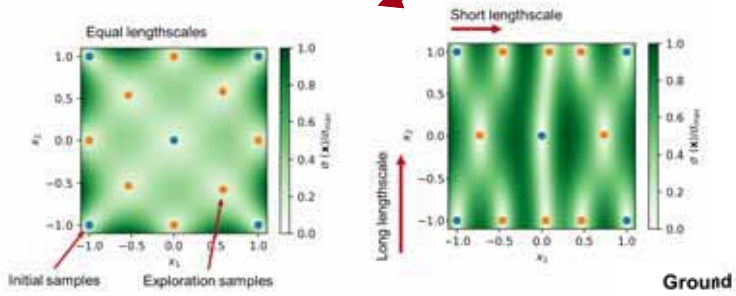
$$\alpha(\boldsymbol{x}) = \sigma(\boldsymbol{x})$$



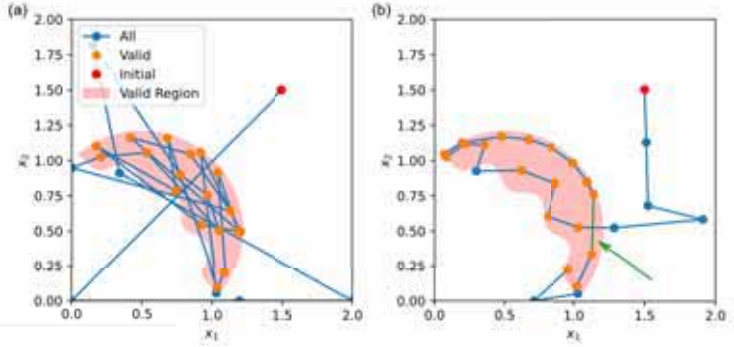
Bayesian Exploration

$$\alpha(\boldsymbol{x}) = \sigma(\boldsymbol{x}) \prod_{i=1}^{N} p(g_i(\boldsymbol{x}) \geq h_i)\, \Psi(\boldsymbol{x}, \boldsymbol{x_0})$$

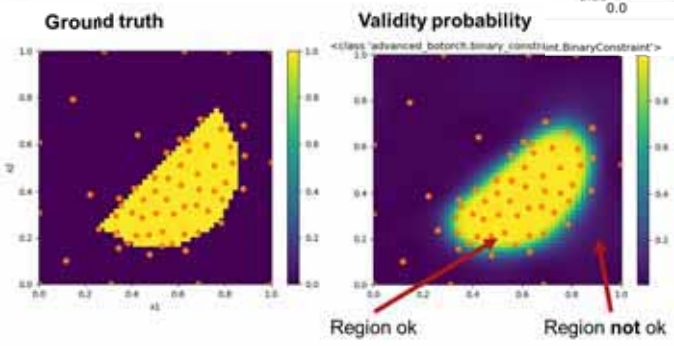
Adaptive sampling



Proximal biasing

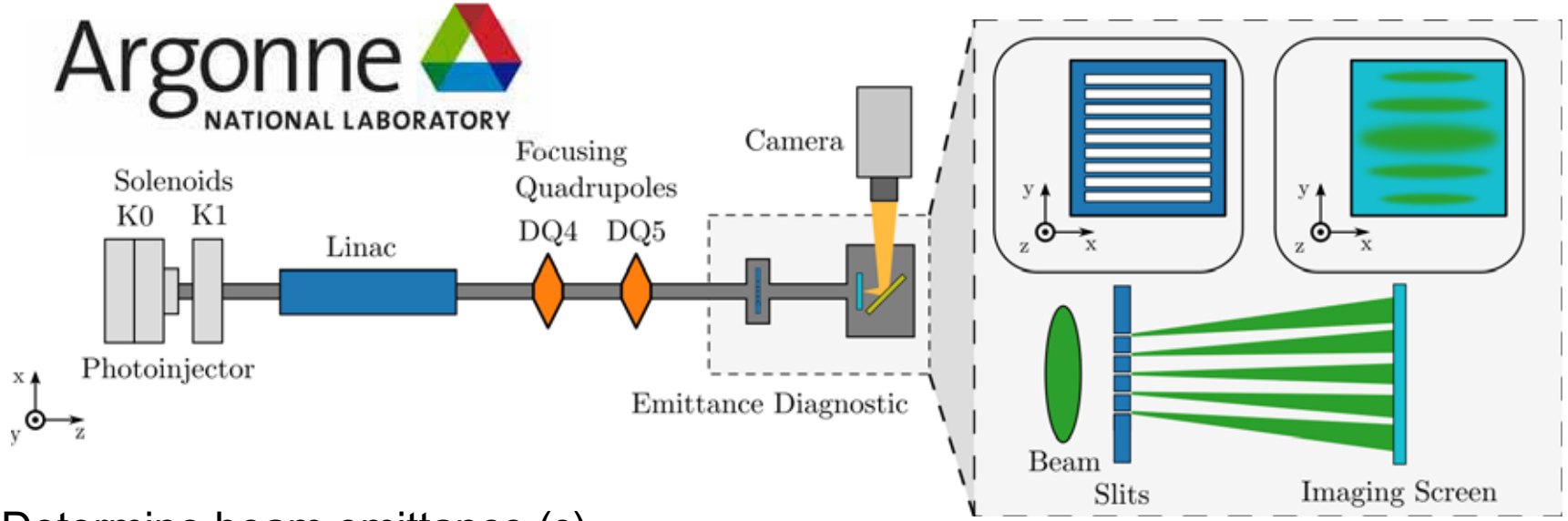


Unknown constraints



Roussel et. Al. *Nat. Comm.* **2021**

Characterizing Photoinjector Emittance at AWA

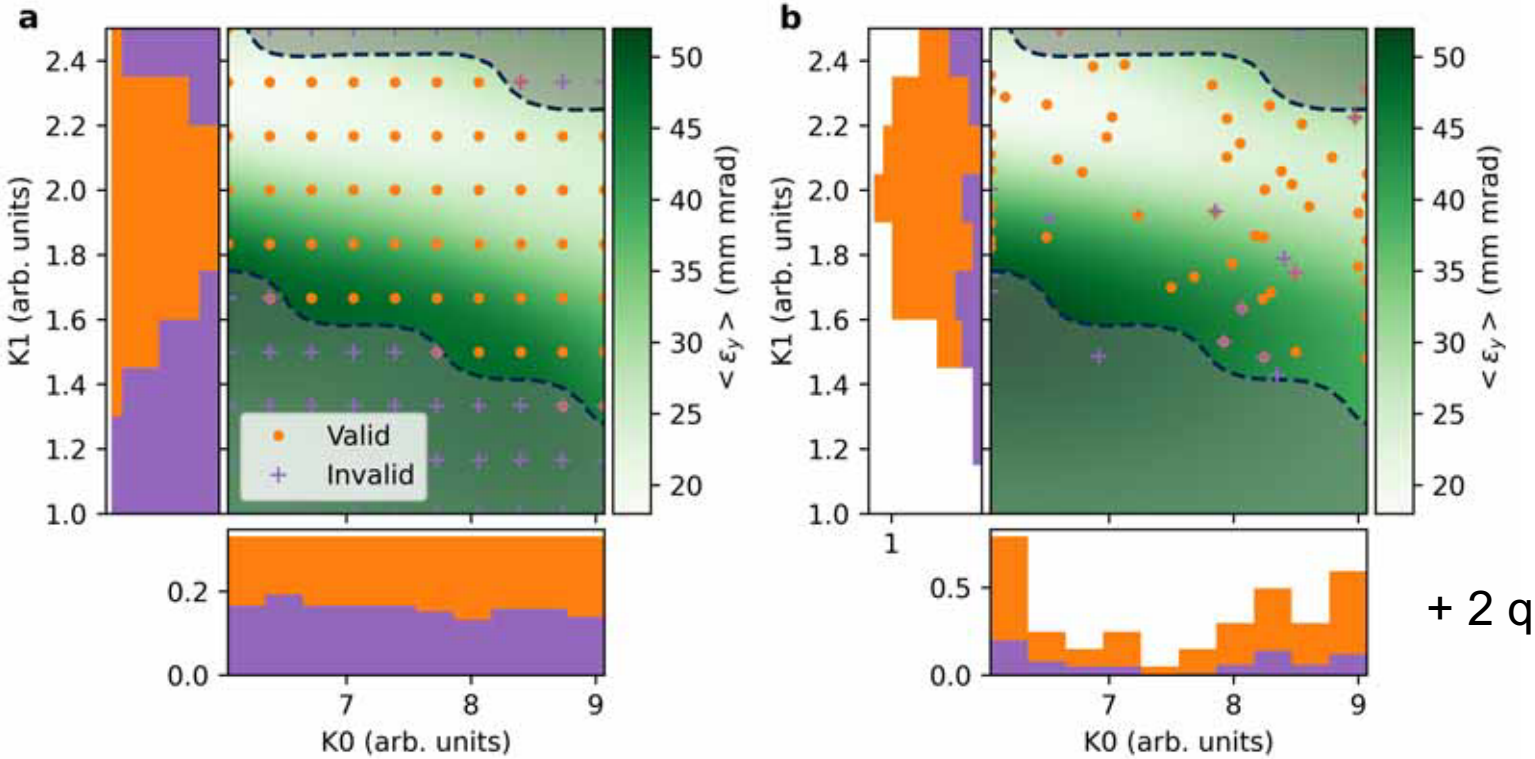


Determine beam emittance (ε) as a function of:

- 2 solenoids
- 2 quadrupoles

Roussel et. Al. *Nat. Comm.* **2021**

Characterizing Photoinjector Emittance at AWA



+ 2 quads!

Roussel et. Al. *Nat. Comm.* **2021**

Implementing Bayesian Optimization

Use Botorch

- Single/multi-objective Bayesian optimization (serial and parallel)
- Constraints
- Proximal biasing
- MC Acquisition function evaluation
- Flexible incorporation of pyTorch Modules
- GPU support

Xopt – Flexible Optimization in Python

SLAC

Flexible implementation of advanced optimization algorithms in python

- Requires only a single Python function to evaluate
- Available algorithms:
  - Bayesian exploration (*Nat. Comm.* **12**, 5612 (2021))
  - Multi-objective Bayesian optimization (*PRAB* **24**, 062801 (2021))
  - Time dependent BO
  - Continuous NSGA
- Serial and parallel optimization using python threading, MPI etc.
- Used on HPC systems (NERSC)
- Used for real-time control at AWA/SLAC

https://christophermayes.github.io/Xopt/

```
xopt:
    max_evaluations: 6400

generator:
    name: cnsga
    population_size: 64
    population_file: test.csv
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y2: MINIMIZE}
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x9: x1}
    constants: {a: dummy_constant}
```

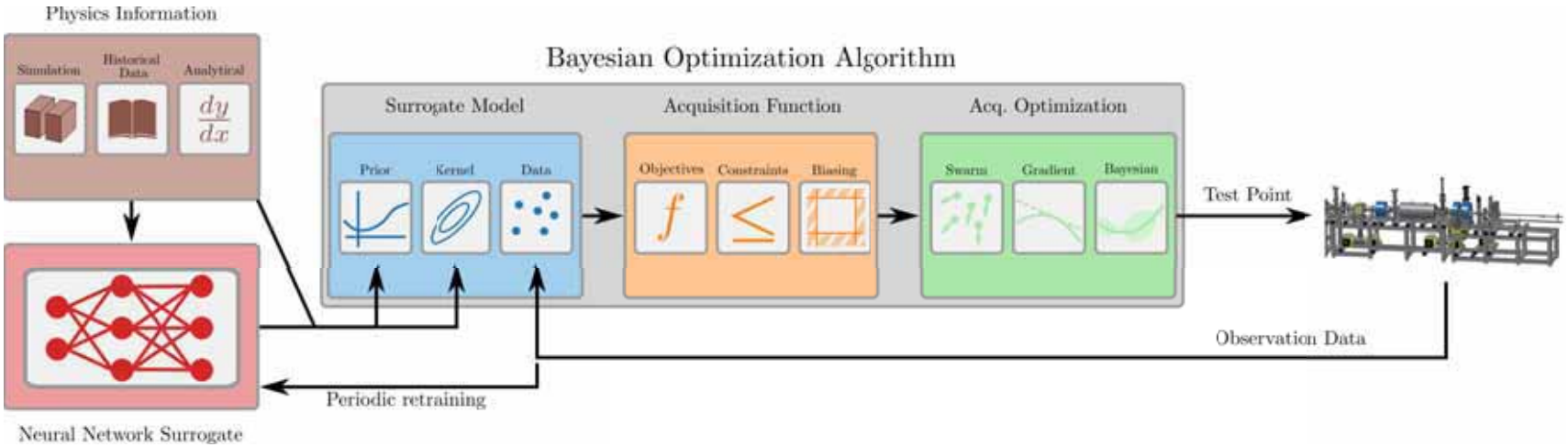
Xopt

conda install xopt –c conda-forge

Some Final Notes

- Always **normalize** your inputs to the range [0,1] and **standardize** your outputs to have a mean of 0 and variance of 1
- Consider if **training/inference costs** are worth it for your application
- Exploit **prior information**, generally, any info (approximations/guesses) is better than no info

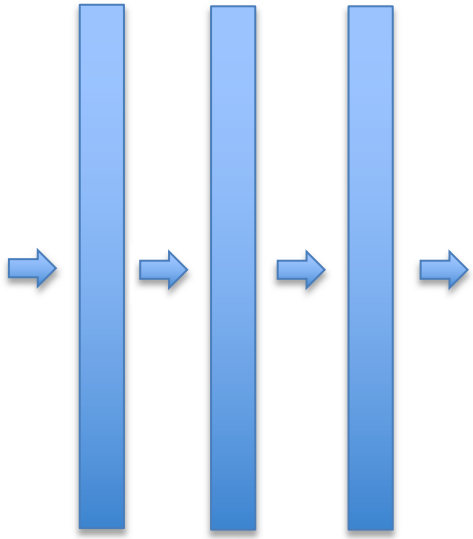
Future Directions



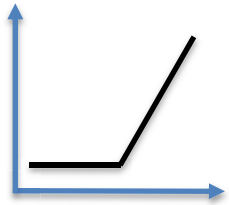
Goal: Include as much information into the GP **before** conducting optimization to beat $\mathcal{O}(N^3)$ scaling

- Take advantage of Bayes rule!
  - Priors on hyperparameters
  - Priors on function values

Part 2: Neural Networks



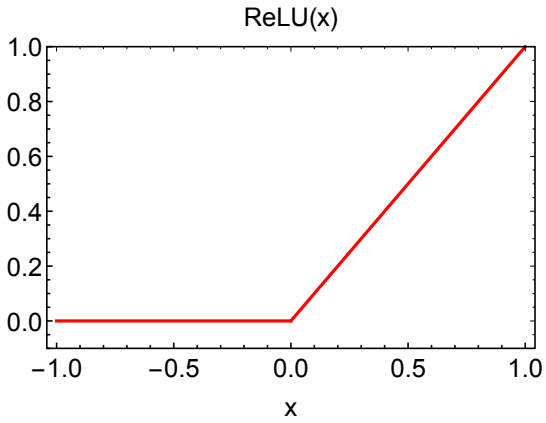
$$\begin{bmatrix} x_1^i \\ x_2^i \end{bmatrix} \begin{bmatrix} w_{11}^1 & w_{12}^1 \end{bmatrix} + b_1^1 = x_1^i w_{11}^1 + x_2^i w_{12}^1 + b_1^1$$

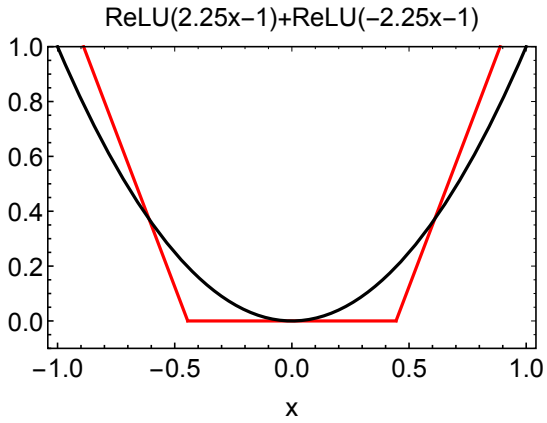


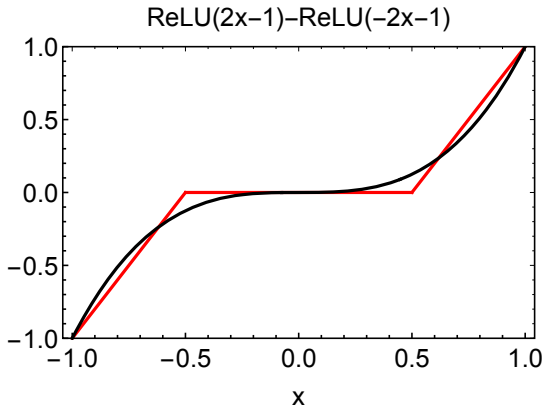
$$\text{ReLU}(x) = \max\{0, x\}$$

ReLU

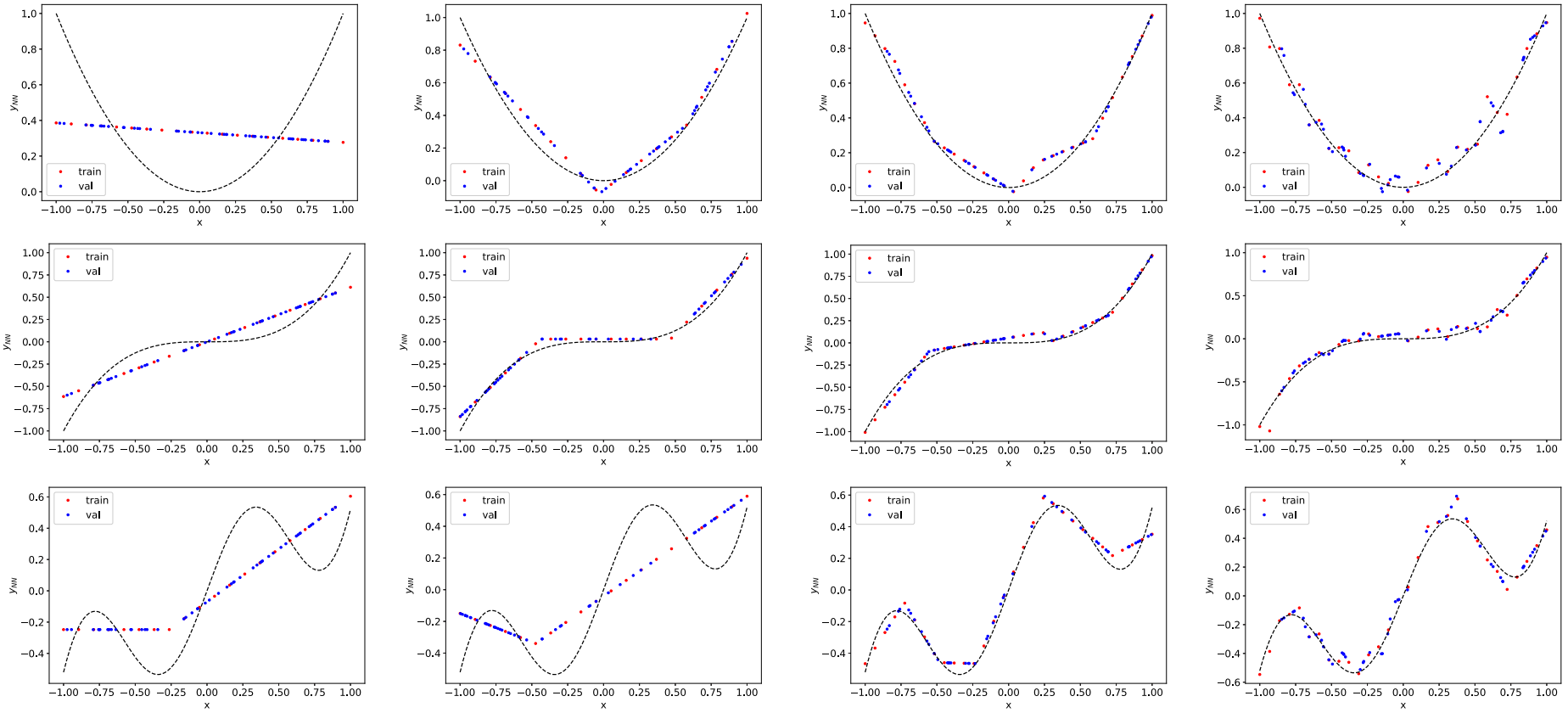
$$\text{ReLU}\left(x_1^i w_{11}^1 + x_2^i w_{12}^1 + b_1^1\right)$$


ReLU(x)
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0
x

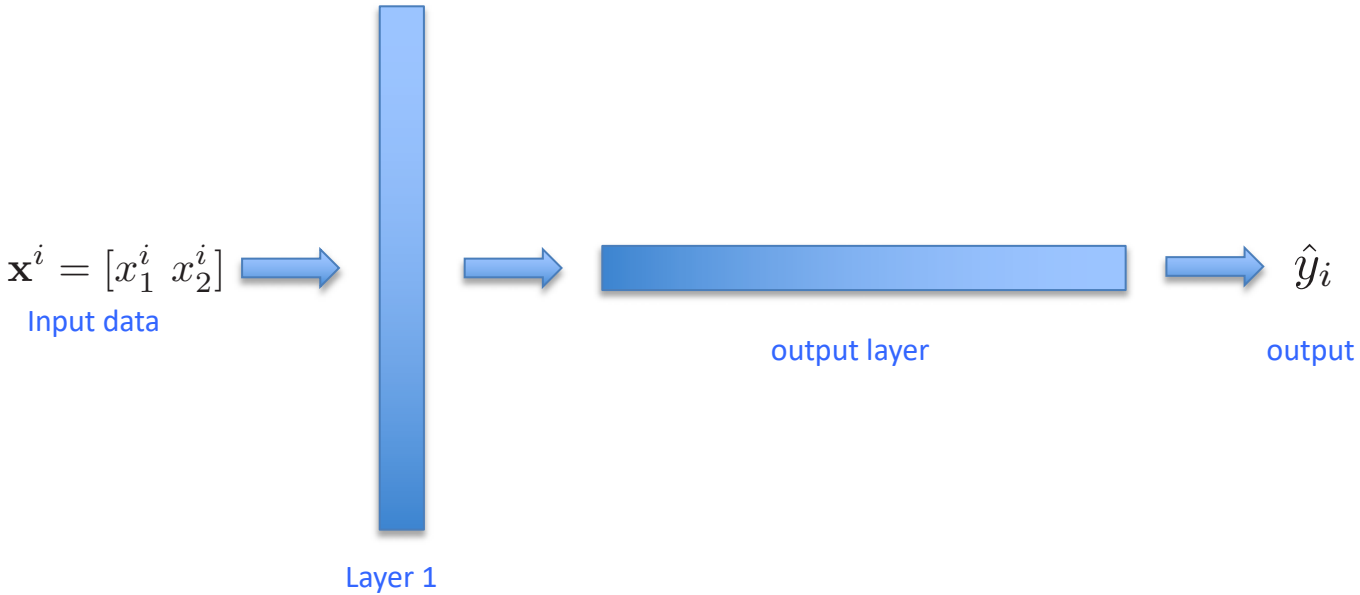

ReLU(2.25x−1)+ReLU(−2.25x−1)
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0
x


ReLU(2x−1)−ReLU(−2x−1)
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
x

n_1=1 n_1=2 n_1=10 n_1=100

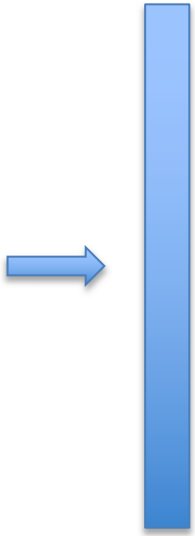


$$\mathbf{x}^i = [x_1^i \; x_2^i], \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \vdots \\ \mathbf{x}^N \end{bmatrix} = \begin{bmatrix} x_1^1 \; x_2^1 \\ x_1^2 \; x_2^2 \\ \vdots \\ x_1^N \; x_2^N \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$



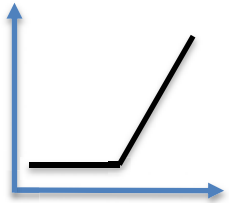
$$\mathbf{x}^i = [x_1^i \;\; x_2^i]$$

Input data



Layer 1

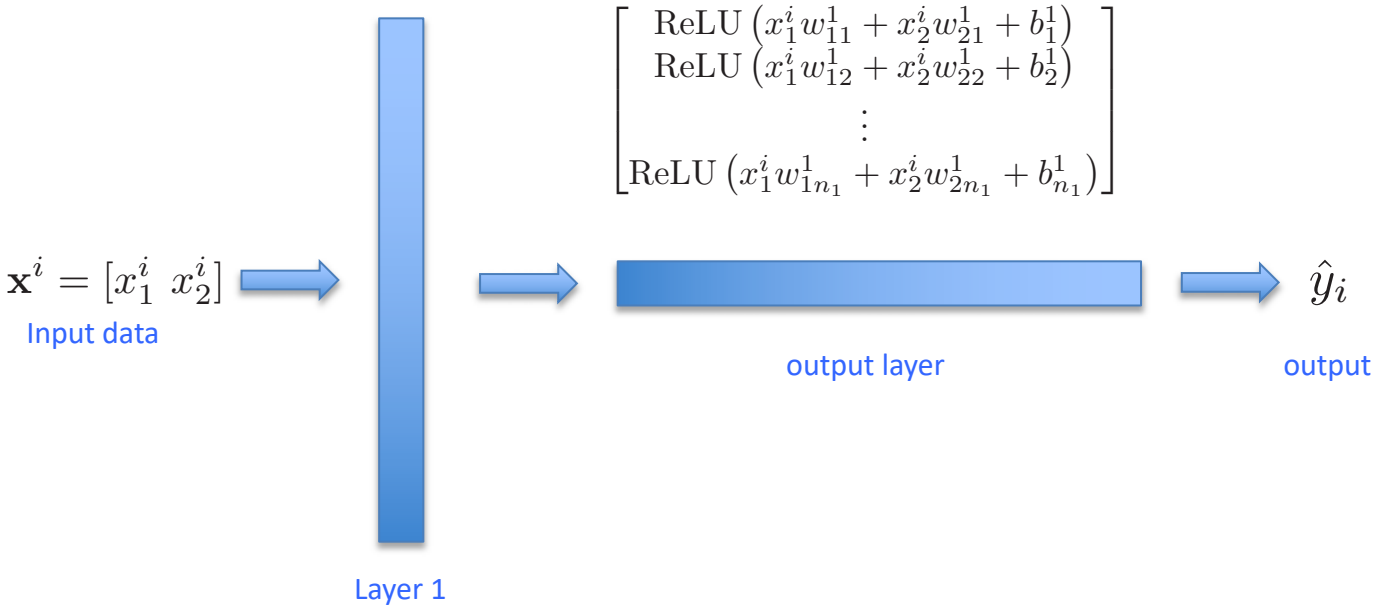
$$\begin{bmatrix} x_1^i \\ x_2^i \end{bmatrix} [w_{11}^1 \;\; w_{12}^1] + b_1^1 = x_1^i w_{11}^1 + x_2^i w_{12}^1 + b_1^1$$



ReLU

$$\text{ReLU}\left(x_1^i w_{11}^1 + x_2^i w_{12}^1 + b_1^1\right)$$

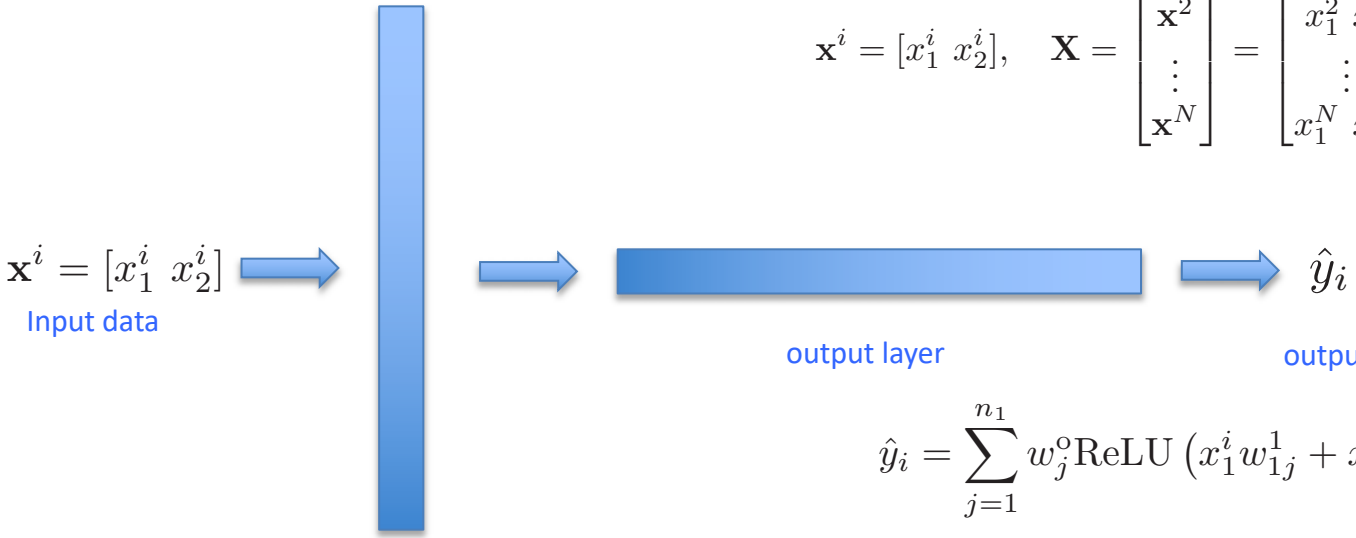
$$\begin{bmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \\ \vdots & \\ w_{1n_1}^1 & w_{2n_1}^1 \end{bmatrix} \begin{bmatrix} x_1^i \\ x_2^i \end{bmatrix} + \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_{n_1}^1 \end{bmatrix} = \begin{bmatrix} x_1^i w_{11}^1 + x_2^i w_{21}^1 + b_1^1 \\ x_1^i w_{12}^1 + x_2^i w_{22}^1 + b_2^1 \\ \vdots \\ x_1^i w_{1n_1}^1 + x_2^i w_{2n_1}^1 + b_{n_1}^1 \end{bmatrix} \Longrightarrow \begin{bmatrix} \text{ReLU}\left(x_1^i w_{11}^1 + x_2^i w_{21}^1 + b_1^1\right) \\ \text{ReLU}\left(x_1^i w_{12}^1 + x_2^i w_{22}^1 + b_2^1\right) \\ \vdots \\ \text{ReLU}\left(x_1^i w_{1n_1}^1 + x_2^i w_{2n_1}^1 + b_{n_1}^1\right) \end{bmatrix}$$



$$\begin{bmatrix} w_1^o \; w_2^o \; \dots \; w_{n_1}^o \end{bmatrix} \begin{bmatrix} \text{ReLU}\left(x_1^i w_{11}^1 + x_2^i w_{21}^1 + b_1^1\right) \\ \text{ReLU}\left(x_1^i w_{12}^1 + x_2^i w_{22}^1 + b_2^1\right) \\ \vdots \\ \text{ReLU}\left(x_1^i w_{1n_1}^1 + x_2^i w_{2n_1}^1 + b_{n_1}^1\right) \end{bmatrix} + b^o$$

$$\hat{y}_i = \sum_{j=1}^{n_1} w_j^o \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right) + b^o$$

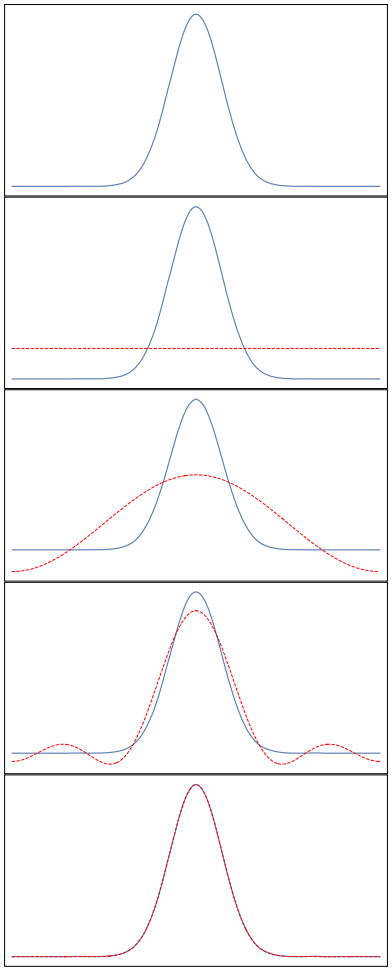
$$\mathbf{x}^i = [x_1^i \ x_2^i], \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \vdots \\ \mathbf{x}^N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 \\ x_1^2 & x_2^2 \\ & \vdots \\ x_1^N & x_2^N \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$



$$\hat{y}_i = \sum_{j=1}^{n_1} w_j^{\mathrm{o}} \mathrm{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right) + b^{\mathrm{o}}$$

$$C = \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2, \quad p \Longrightarrow p - \delta \nabla_p C$$

$$\nabla_{w_{kj}^1} C = 2 \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right) \frac{\partial \hat{y}_i}{\partial w_{kj}^1} = 2 w_j^{\mathrm{o}} \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right) \begin{cases} x_k^i, & \text{if} \quad \mathrm{ReLU}\left(\sum_{k=1}^{2} x_k^i w_{kj}^1 + b_j^1\right) > 0 \\ 0, & \text{if} \quad \mathrm{ReLU}\left(\sum_{k=1}^{2} x_k^i w_{kj}^1 + b_j^1\right) < 0 \end{cases}$$

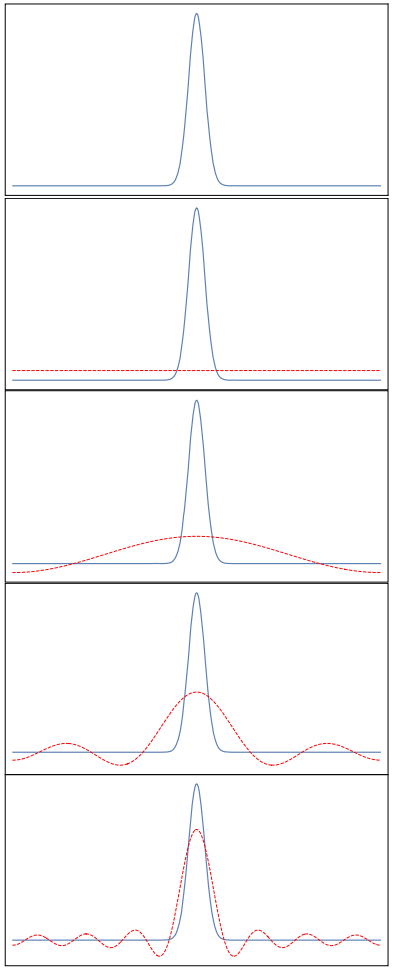


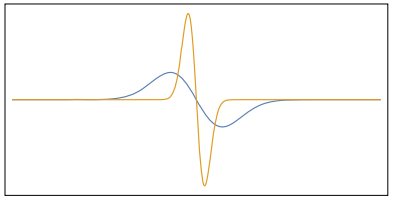
N=1

N=2

N=4

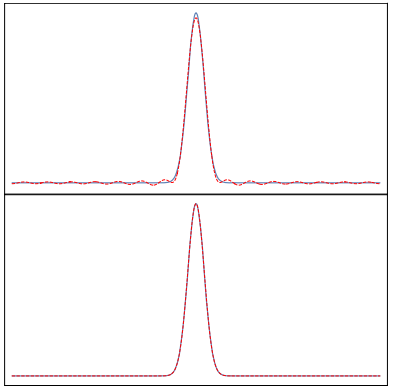
N=8





N=16

N=32



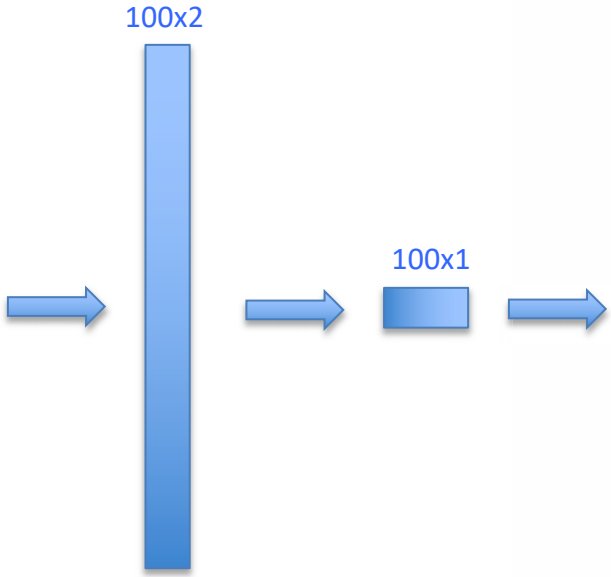
$$f(x) \approx a_0 + \sum_{n=1}^{N} \left[a_n \cos\left(\frac{2\pi n x}{L}\right) + b_n \sin\left(\frac{2\pi n x}{L}\right) \right]$$

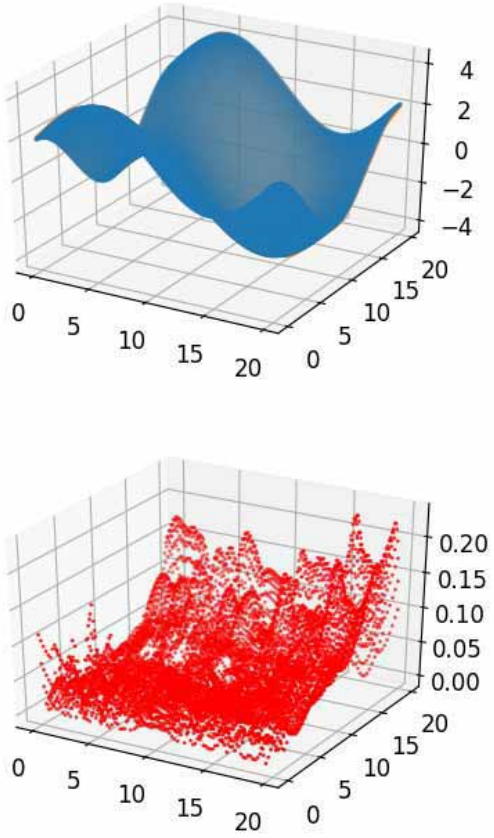
$$a_0 = \frac{1}{L}\int_0^L f(x)dx$$

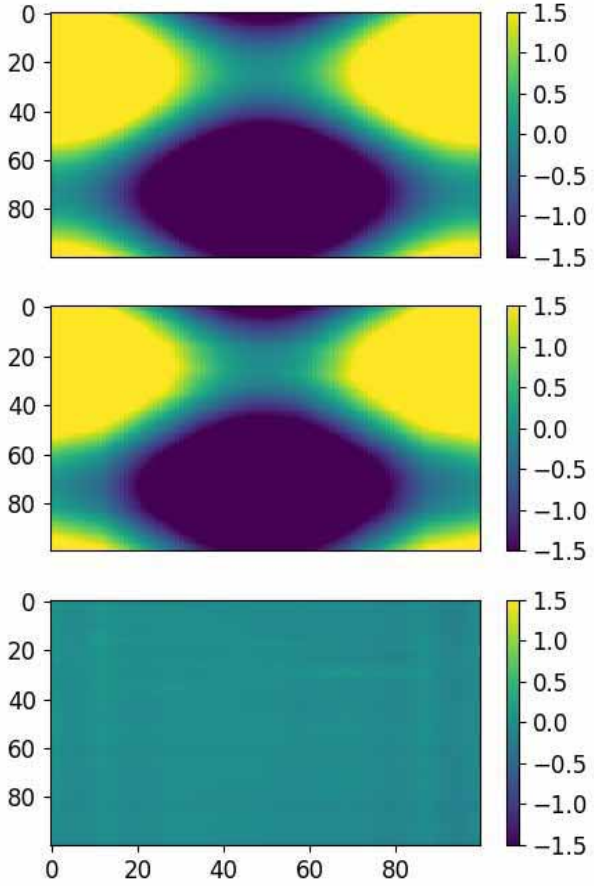
$$a_n = \frac{2}{L}\int_0^L f(x)\cos\left(\frac{2\pi n x}{L}\right)dx$$

$$b_n = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{2\pi n x}{L}\right)dx$$

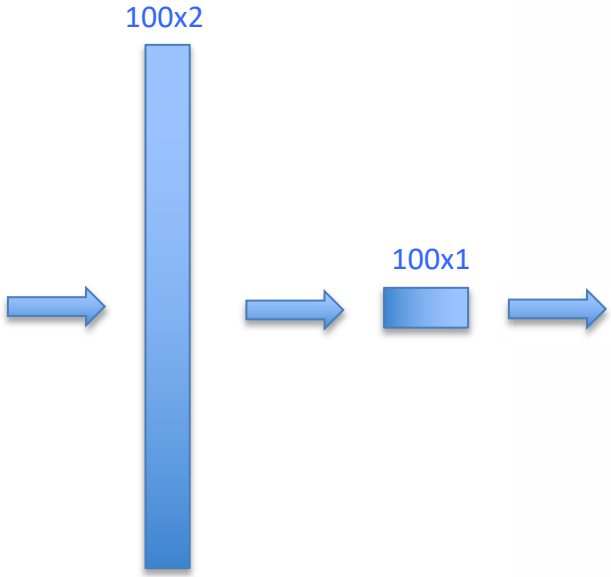
$$f(x, y) = 2\left[\sin(0.6x) + \cos(0.6y)\right]$$

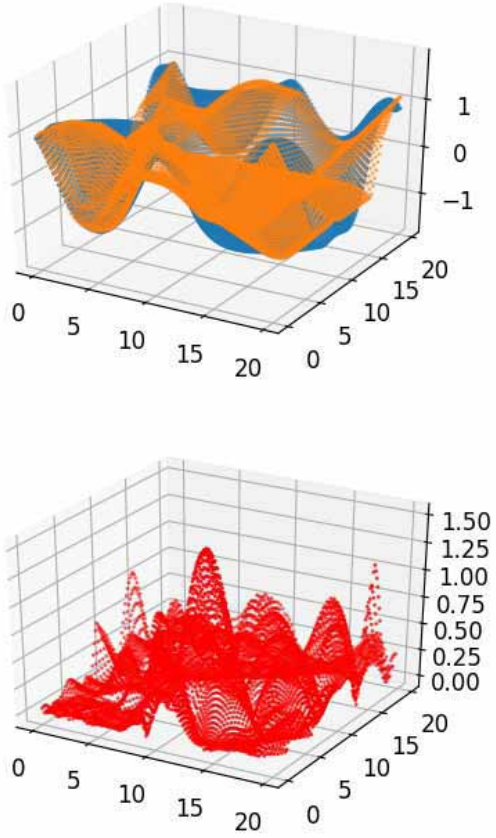


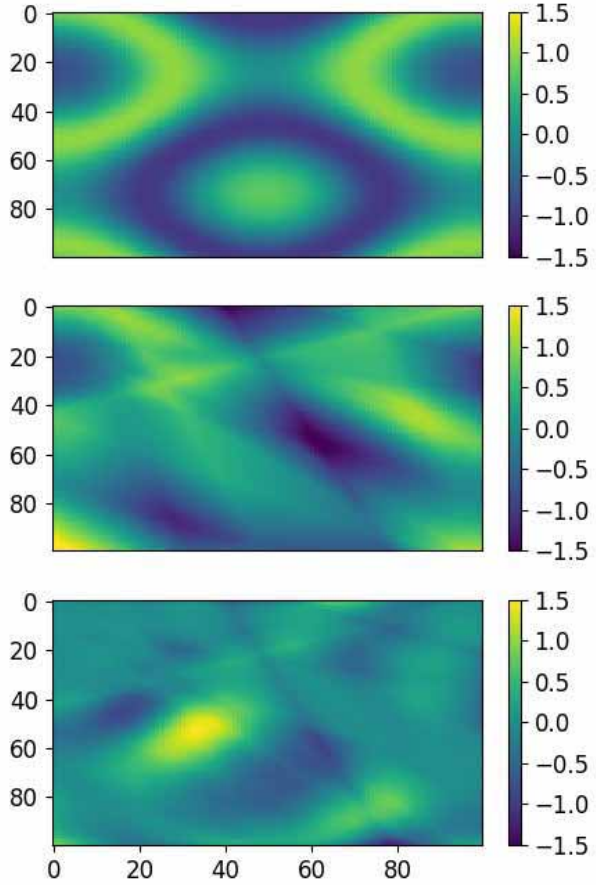




$$f(x, y) = \sin\left(2\left[\sin(0.6x) + \cos(0.6y)\right]\right)$$

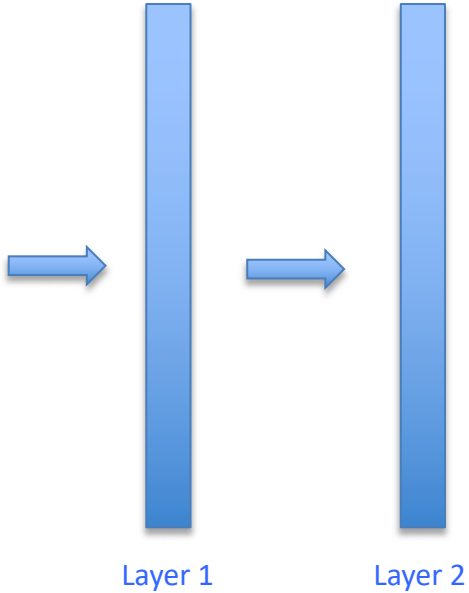






$$\mathbf{x}^i = [x_1^i \;\; x_2^i]$$

Input data



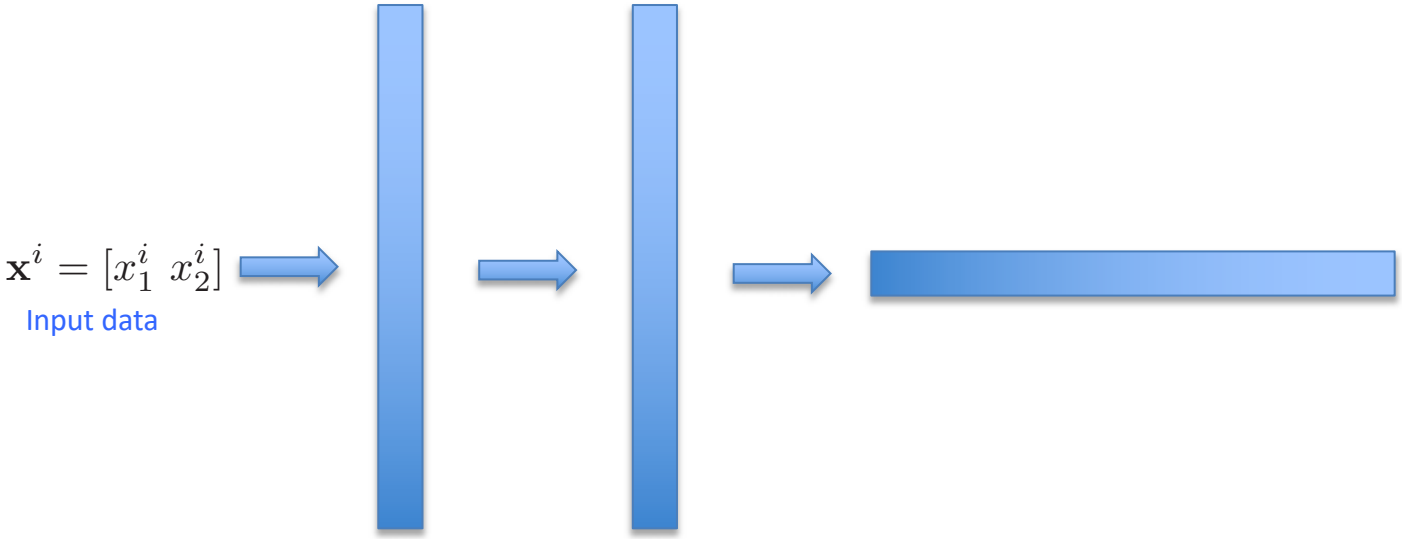
Layer 1 Layer 2

$$\begin{bmatrix} \text{ReLU}\left(x_1^i w_{11}^1 + x_2^i w_{21}^1 + b_1^1\right) \\ \text{ReLU}\left(x_1^i w_{12}^1 + x_2^i w_{22}^1 + b_2^1\right) \\ \vdots \\ \text{ReLU}\left(x_1^i w_{1n_1}^1 + x_2^i w_{2n_1}^1 + b_{n_1}^1\right) \end{bmatrix}$$

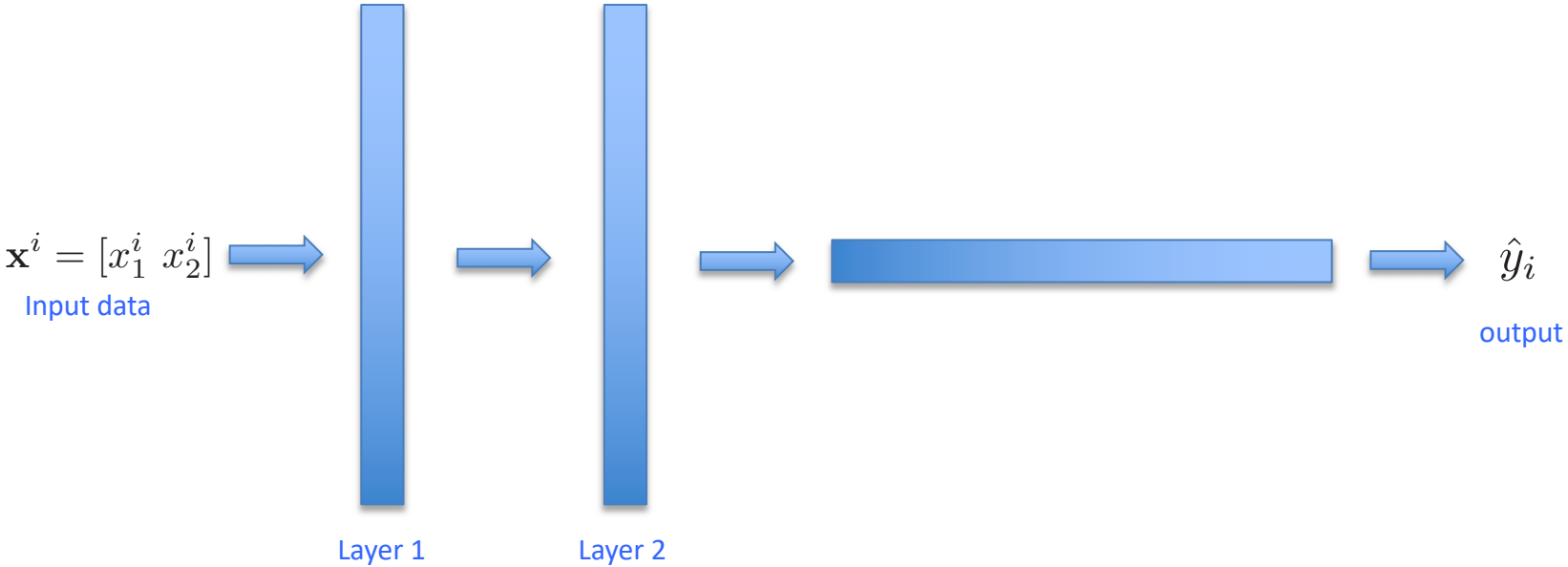
$$\begin{bmatrix} w_{11}^2 & w_{12}^2 & \dots & w_{1n_1}^2 \\ w_{21}^2 & w_{22}^2 & \dots & w_{2n_1}^2 \\ & & \vdots & \\ w_{n_2 1}^2 & w_{n_2 2}^2 & \dots & w_{n_2 n_1}^2 \end{bmatrix} \begin{bmatrix} \text{ReLU}\left(x_1^i w_{11}^1 + x_2^i w_{21}^1 + b_1^1\right) \\ \text{ReLU}\left(x_1^i w_{12}^1 + x_2^i w_{22}^1 + b_2^1\right) \\ \vdots \\ \text{ReLU}\left(x_1^i w_{1n_1}^1 + x_2^i w_{2n_1}^1 + b_{n_1}^1\right) \end{bmatrix} + \begin{bmatrix} b_1^2 \\ b_2^2 \\ \vdots \\ b_{n_2}^2 \end{bmatrix}$$

$$\begin{bmatrix} b_1^2 + \sum_{j=1}^{n_1} w_{1j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right) \\ b_2^2 + \sum_{j=1}^{n_1} w_{2j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i x_{2j}^1 + b_j^1\right) \\ \vdots \\ b_{n_2}^2 + \sum_{j=1}^{n_1} w_{n_2 j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right) \end{bmatrix}$$

$$\begin{bmatrix} \text{ReLU}\left(b_1^2 + \sum_{j=1}^{n_1} w_{1j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right)\right) \\ \text{ReLU}\left(b_2^2 + \sum_{j=1}^{n_1} w_{2j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i x_{2j}^1 + b_j^1\right)\right) \\ \vdots \\ \text{ReLU}\left(b_{n_2}^2 + \sum_{j=1}^{n_1} w_{n_2 j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right)\right) \end{bmatrix}$$

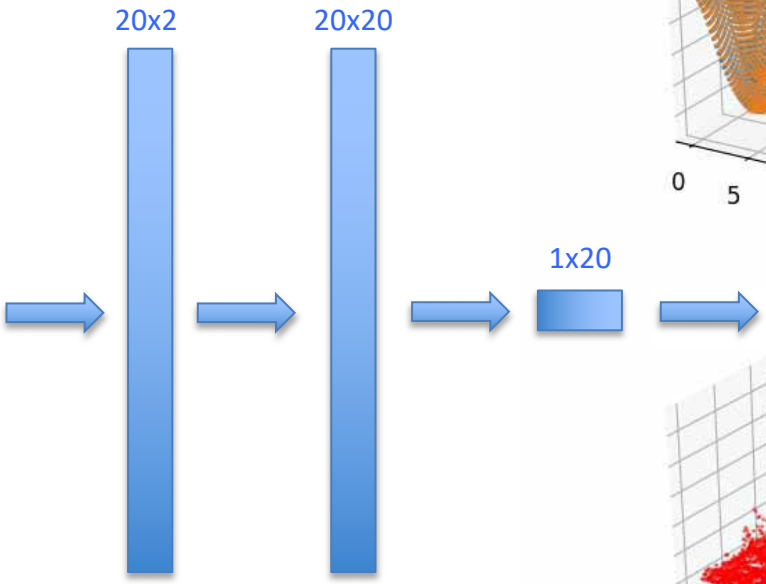


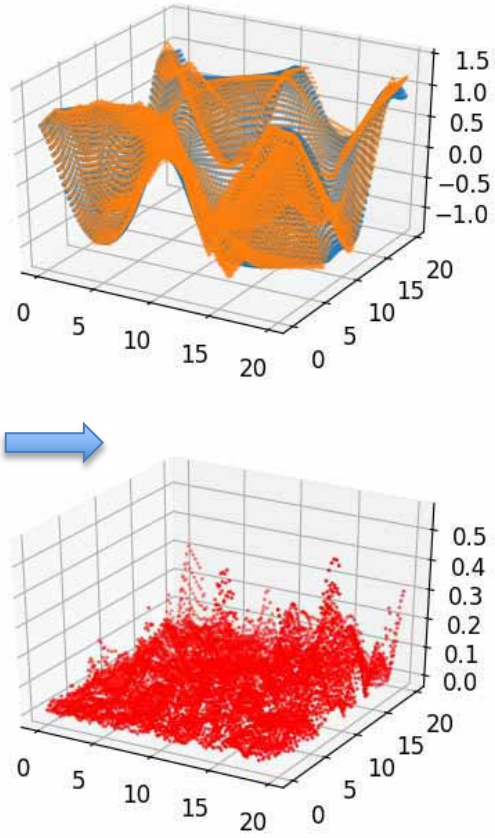
$$[w_1^o \; w_2^o \; \dots \; w_{n_2}^o] \begin{bmatrix} \text{ReLU}\left(b_1^2 + \sum_{j=1}^{n_1} w_{1j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right)\right) \\ \text{ReLU}\left(b_2^2 + \sum_{j=1}^{n_1} w_{2j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i x_{2j}^1 + b_j^1\right)\right) \\ \vdots \\ \text{ReLU}\left(b_{n_2}^2 + \sum_{j=1}^{n_1} w_{n_2 j}^2 \text{ReLU}\left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1\right)\right) \end{bmatrix} + b^o$$

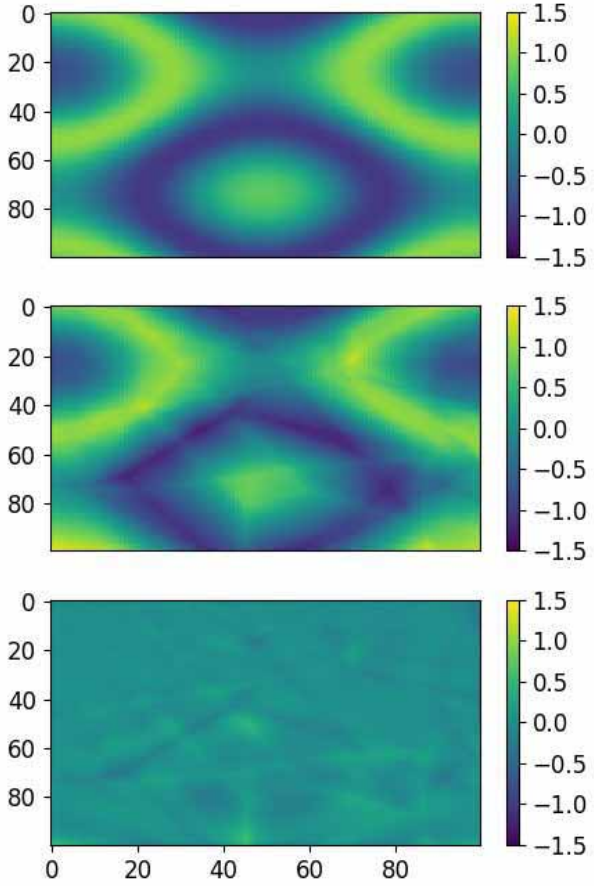


$$\hat{y}_i = \sum_{k=1}^{n_2} \left[w_k^{\mathrm{o}} \mathrm{ReLU} \left(b_k^2 + \sum_{j=1}^{n_1} w_{kj}^2 \mathrm{ReLU} \left(x_1^i w_{1j}^1 + x_2^i w_{2j}^1 + b_j^1 \right) \right) \right] + b^{\mathrm{o}}$$

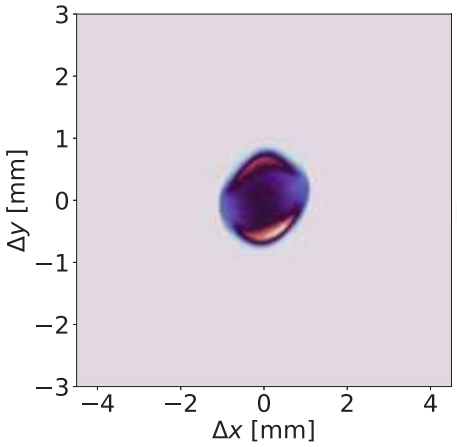
$$f(x, y) = \sin\left(2\left[\sin(0.6x) + \cos(0.6y)\right]\right)$$

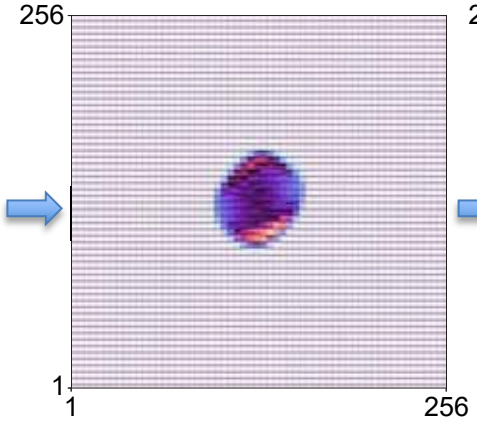


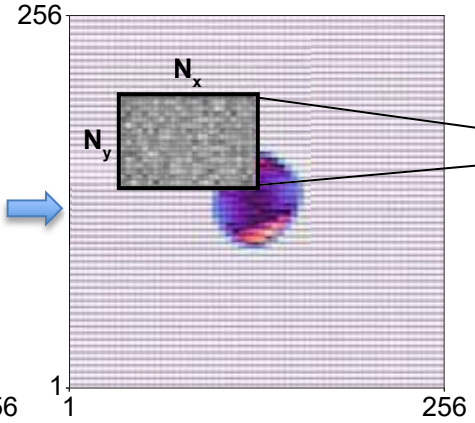




Convolutional Neural Networks





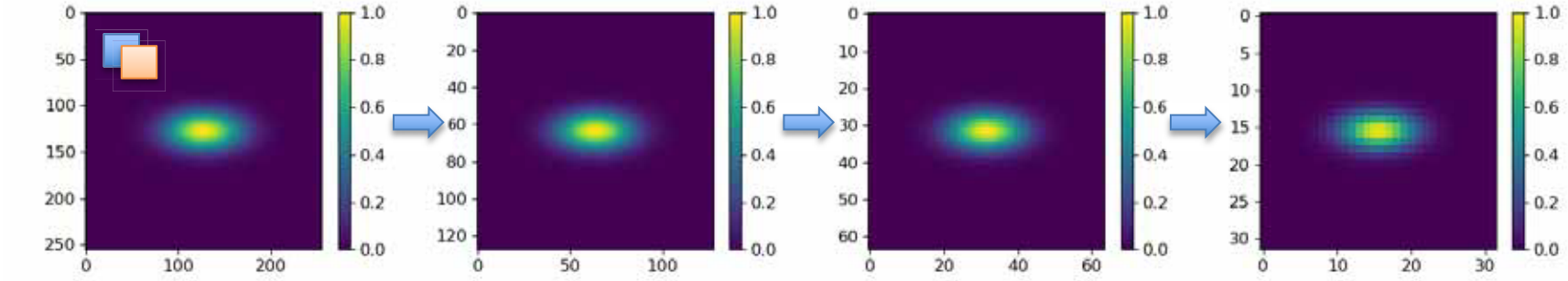


$$\sum_{-\frac{N_x}{2}}^{\frac{N_x}{2}} \sum_{-\frac{N_y}{2}}^{\frac{N_y}{2}} A_{lm} I_n(i+l, j+m)$$

A =

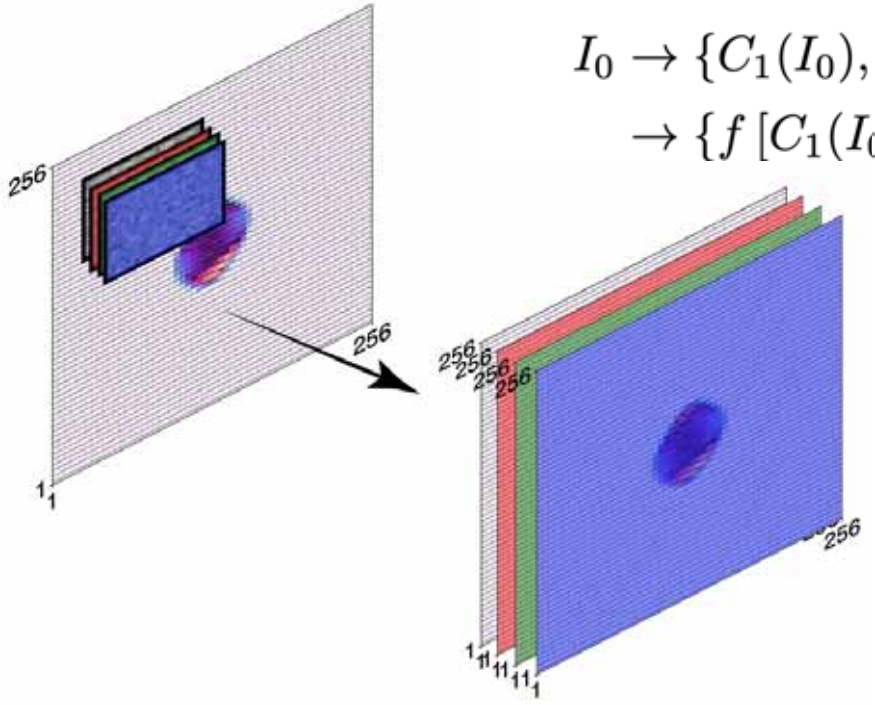
0	0	0
0	1	0
0	0	0

$$I_1 = f\left(b_0 + \sum_l \sum_m A_{lm} I_0(i+l, j+m)\right)$$



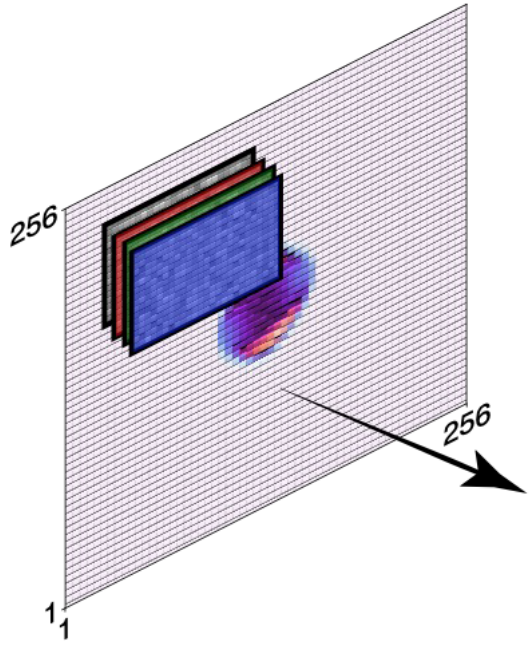
$$I_0 \in \mathbb{R}^{n_x \times n_y}$$

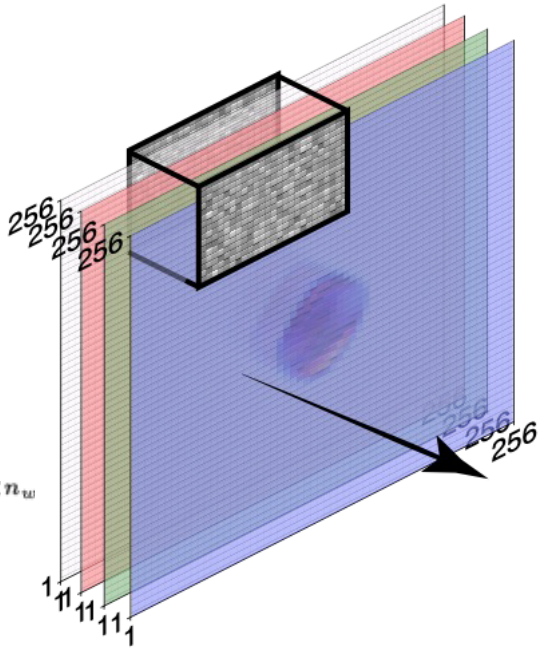
$$C_k(I_0) = b_{0k} + \sum_{l,m} A_{lmk} I_0(i+l, j+m)$$

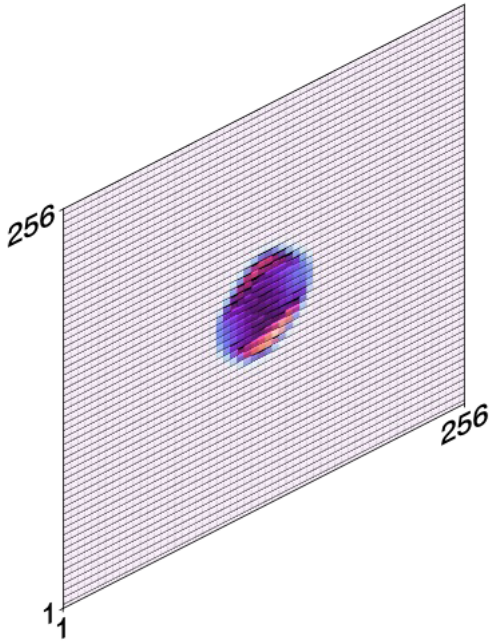


$$I_0 \to \{C_1(I_0), C_2(I_0), \ldots, C_{n_w}(I_0)\} \in \mathbb{R}^{n_x \times n_y \times n_w}$$
$$\to \{f\left[C_1(I_0)\right], f\left[C_2(I_0)\right], \ldots, f\left[C_{n_w}(I_0)\right]\} \in \mathbb{R}^{n_x \times n_y \times n_w}$$

$$I_1 = b_1 + \sum_{k=1}^{n_w} w_k f\left[C_k(I_0)\right]$$



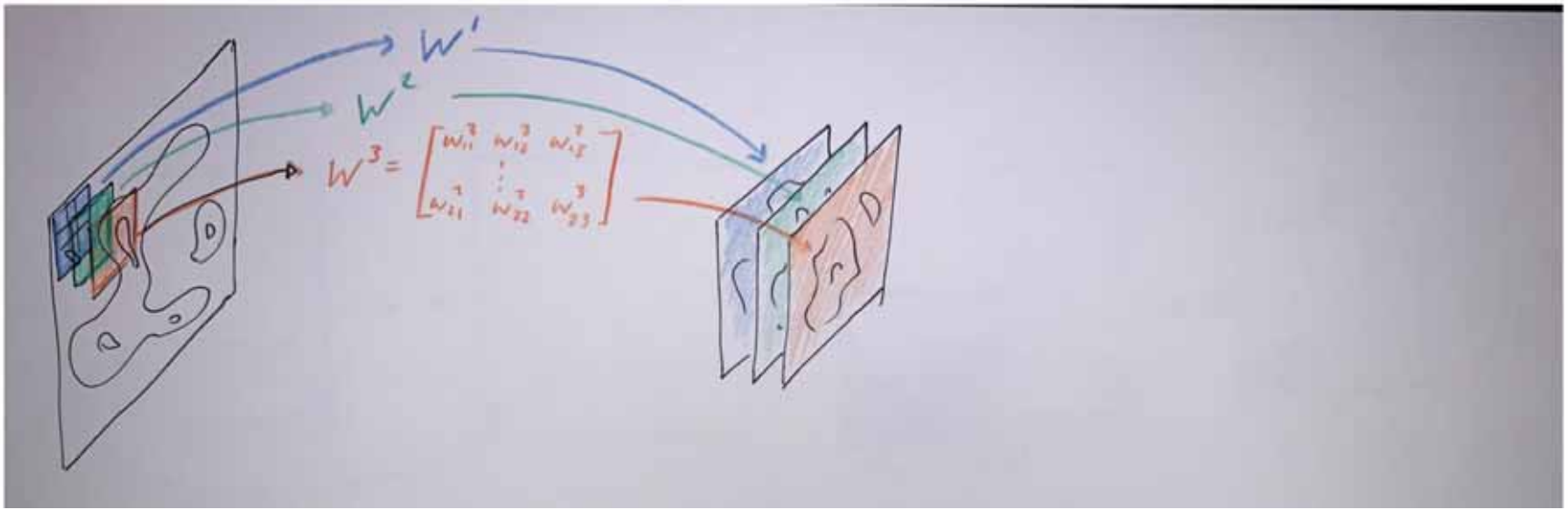


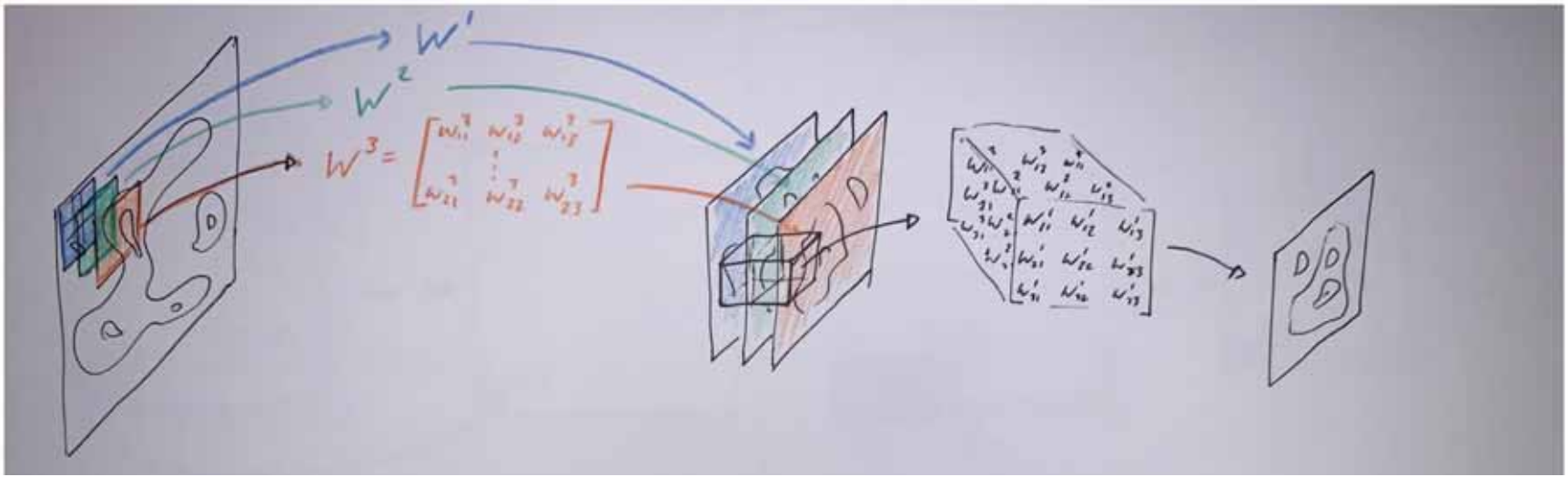


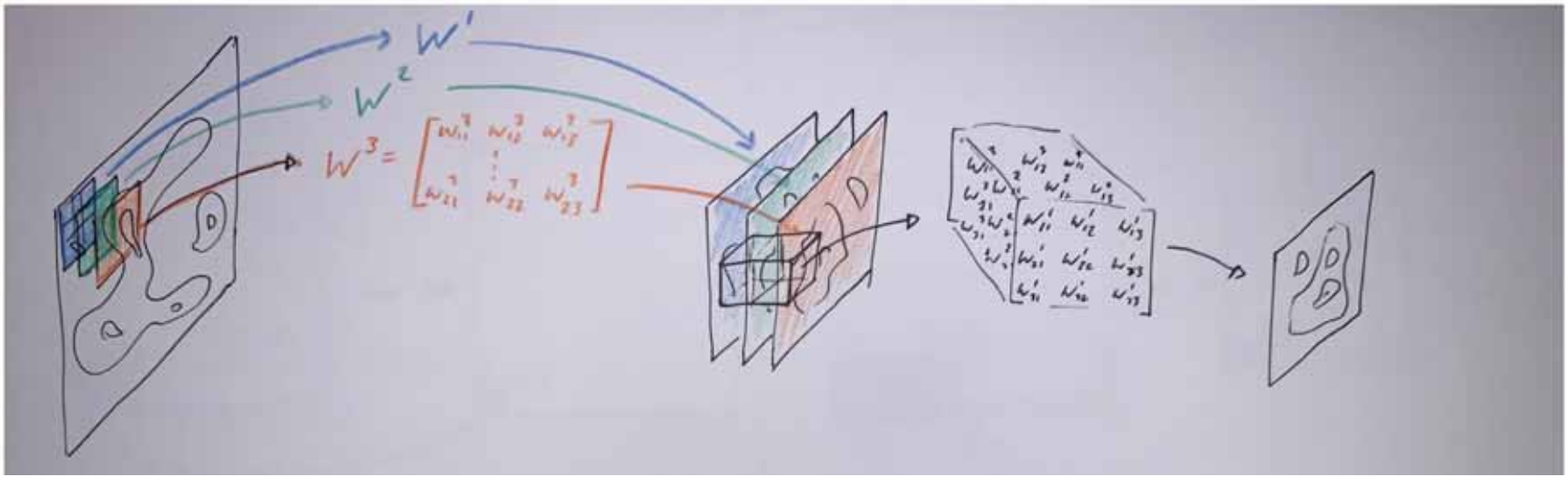
$$S_0 = \{f\left[C_1(I_0)\right], f\left[C_2(I_0)\right], \ldots, f\left[C_{n_w}(I_0)\right]\} \in \mathbb{R}^{n_x \times n_y \times n_w}$$
$$= \left\{S^0_{ijk}\right\}$$

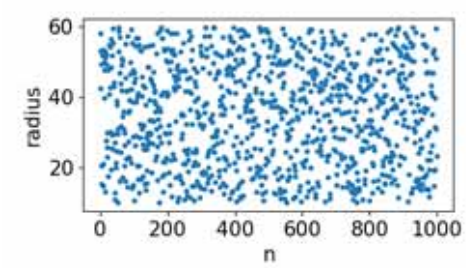
$$W, \quad W_{lmn} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$$

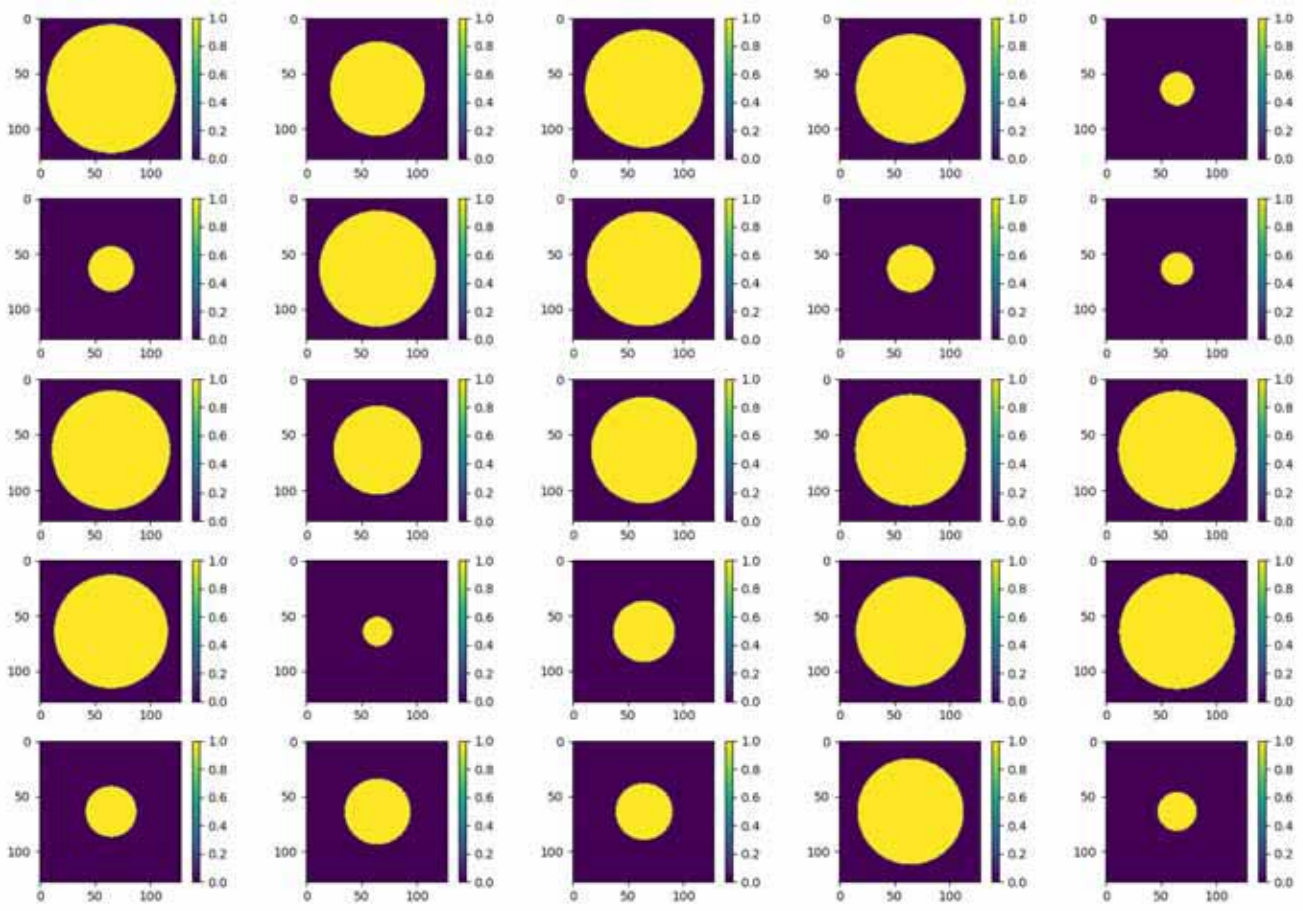
$$S^1_{ijk} = S^0 \star W = \sum_{ijk} S^0_{i+l,j+m,k+n} W_{lmn}$$

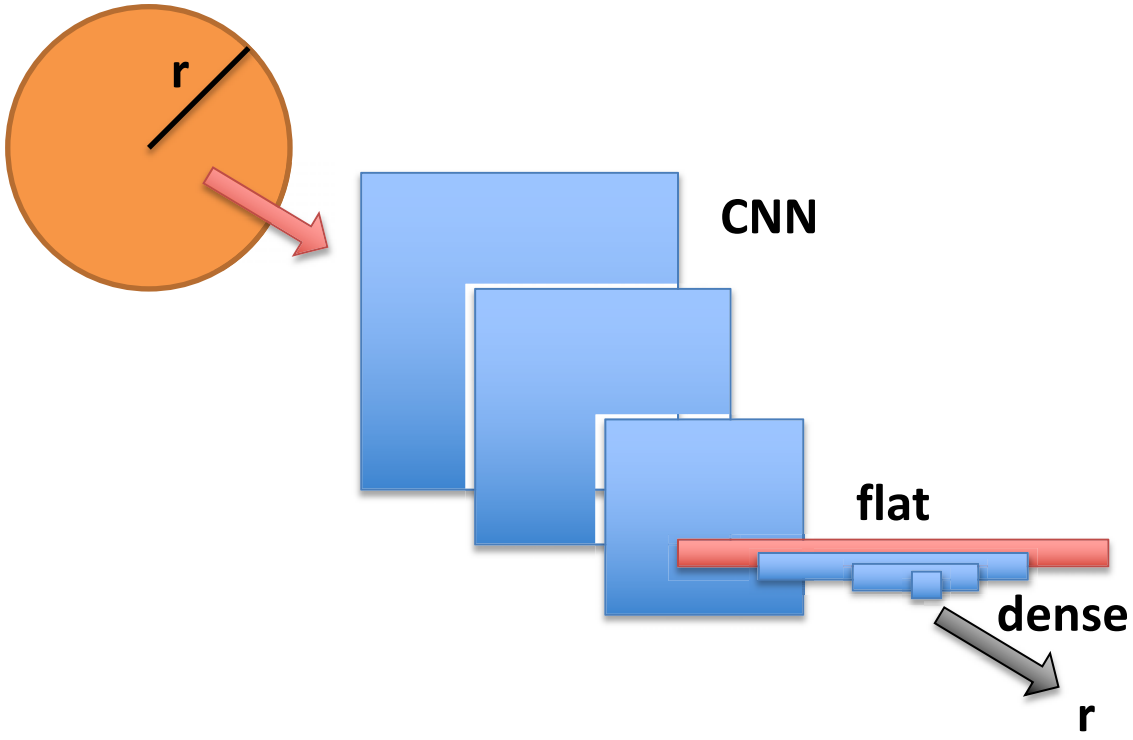

W^1
W^2
$W^3 = \begin{bmatrix} w_{11}^3 & w_{12}^3 & w_{13}^3 \\ \vdots & & \\ w_{31}^3 & w_{32}^3 & w_{33}^3 \end{bmatrix}$

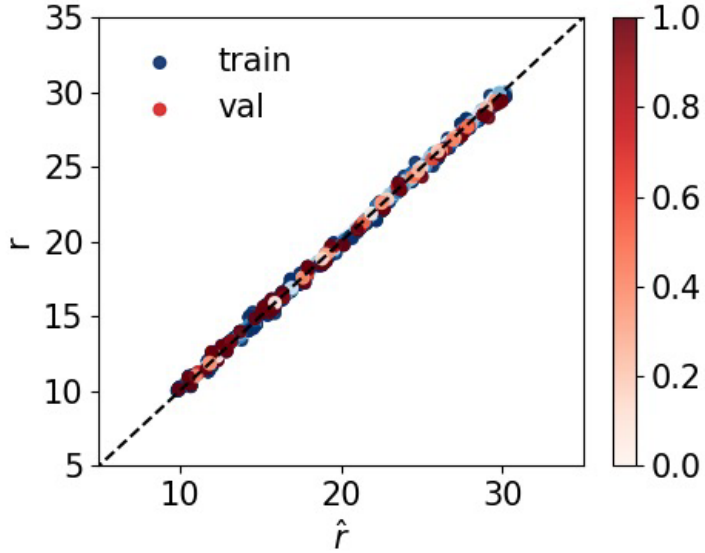

W¹
W²
W³ =

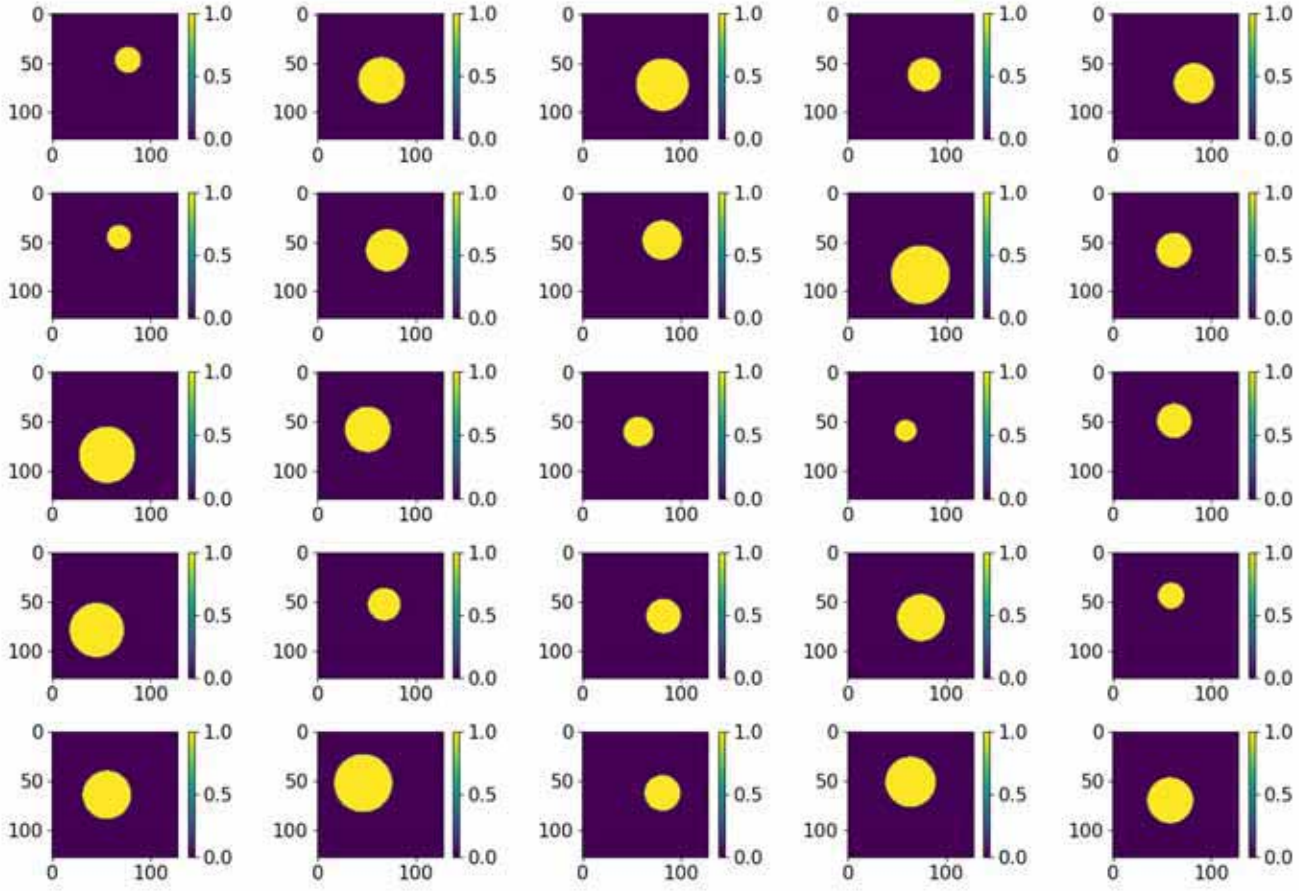

W¹
W²
W³ =

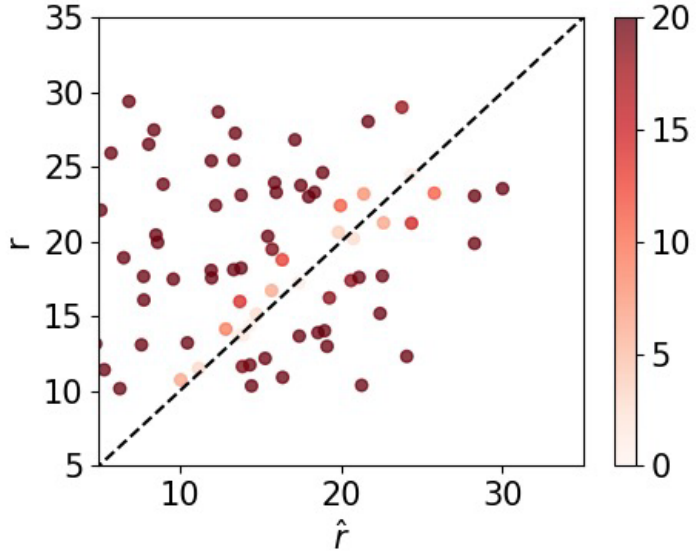

radius
n
60
40
20
0
200
400
600
800
1000

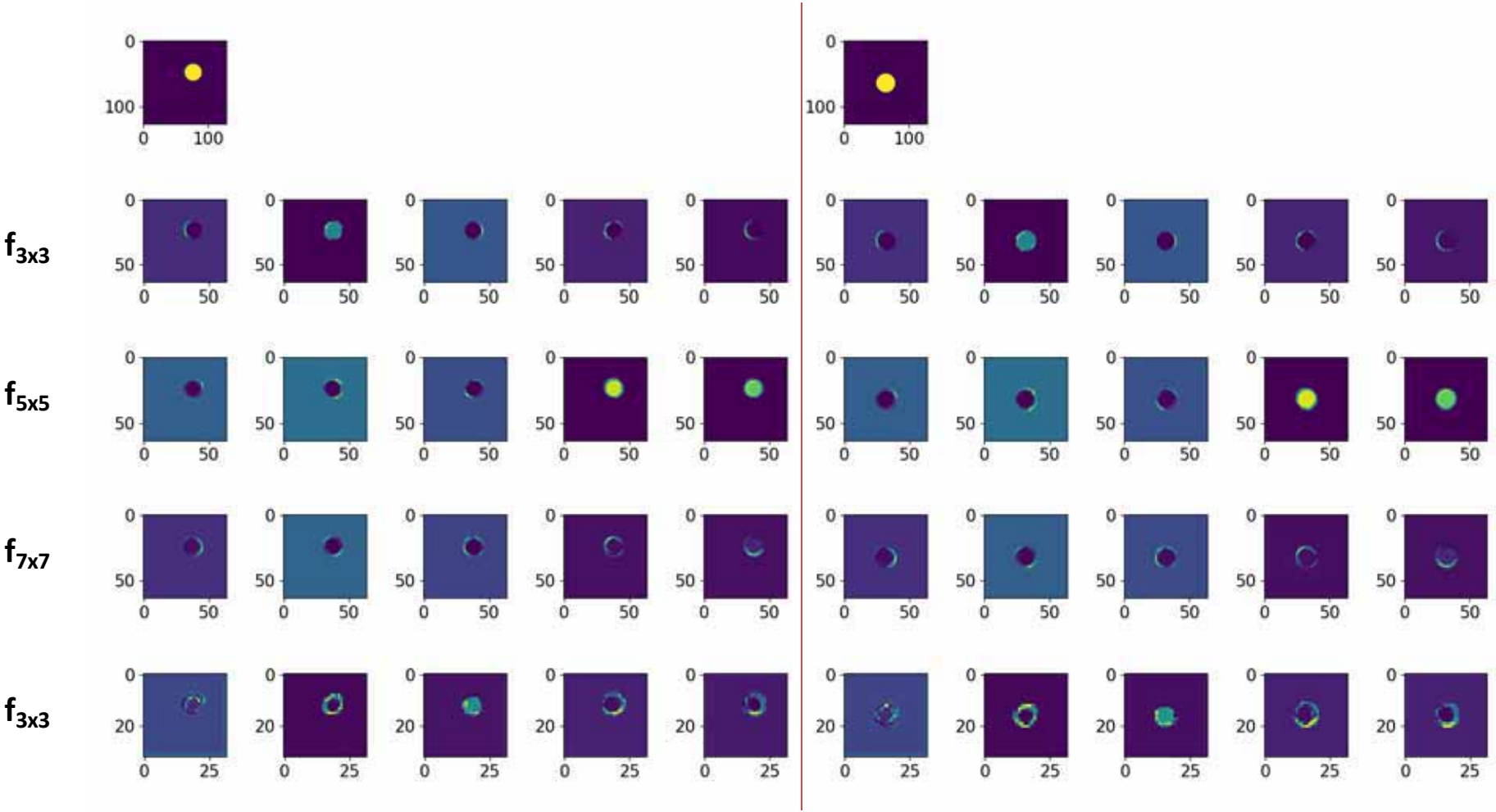


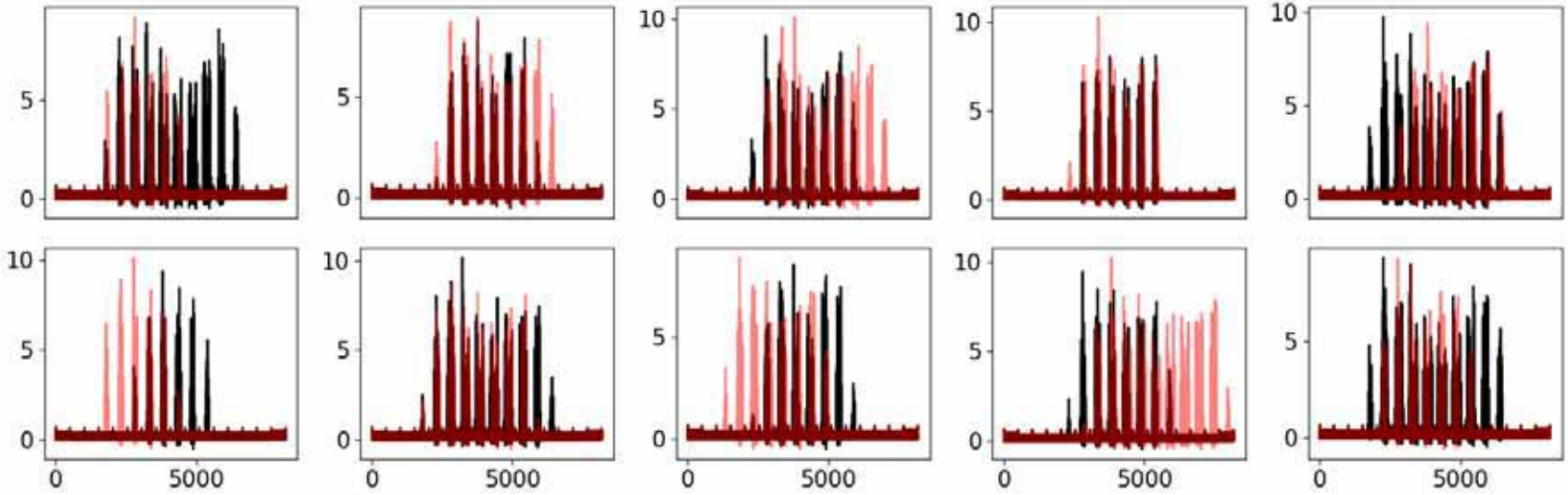

r
CNN
flat
dense
r

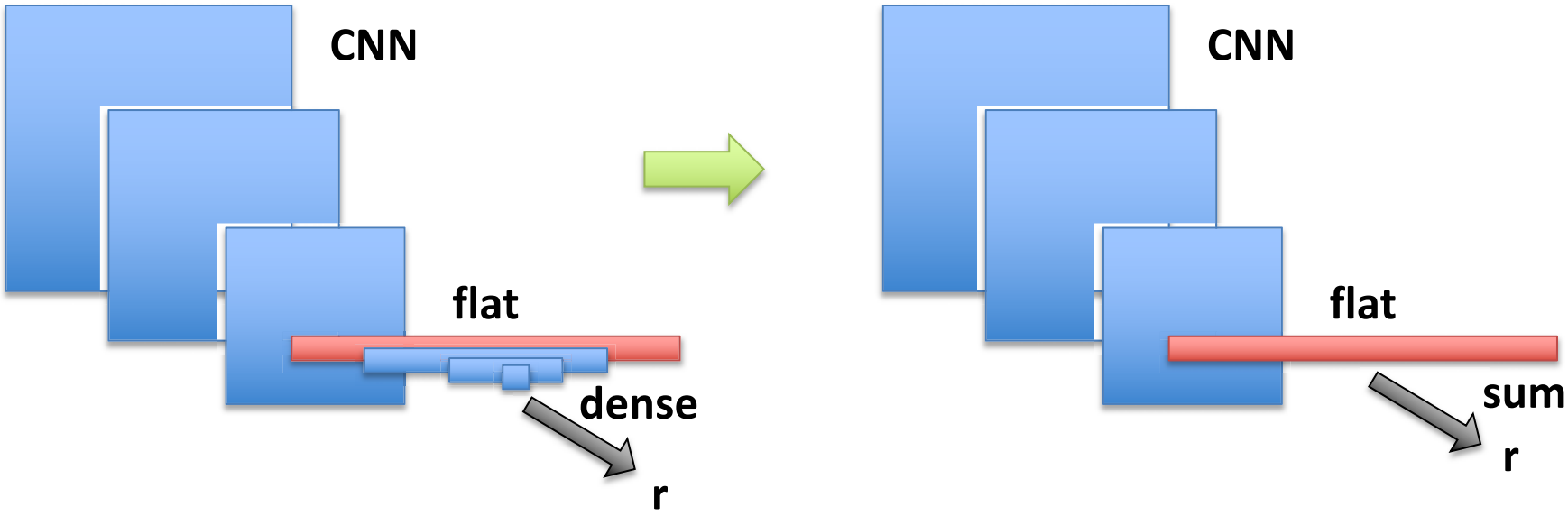

train
val
35
30
25
20
15
10
5
r
10
20
30
$\hat{r}$
1.0
0.8
0.6
0.4
0.2
0.0

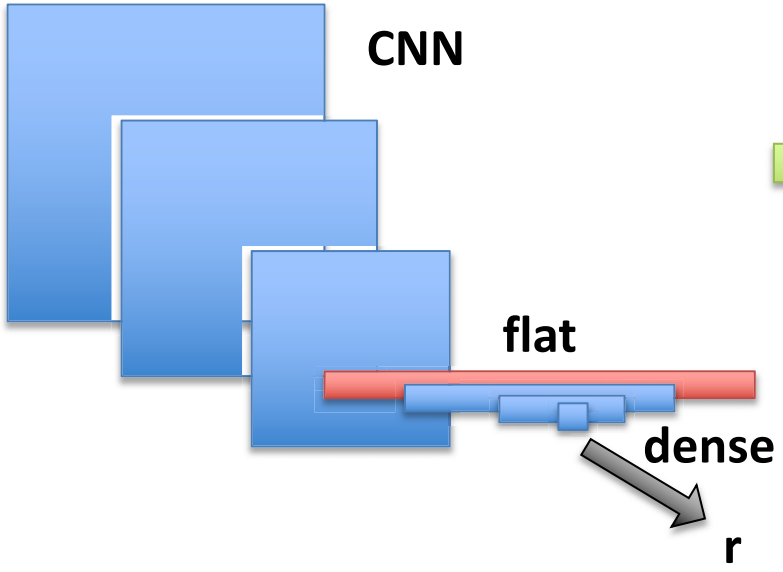

0
50
100
0
100
1.0
0.5
0.0

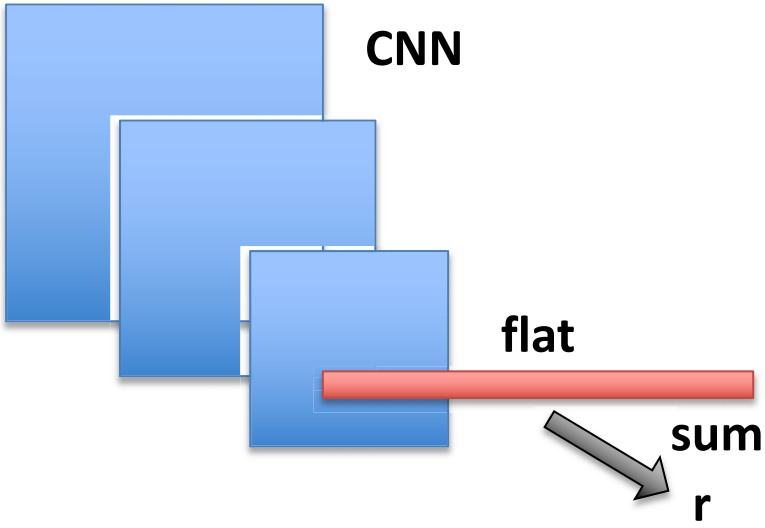

35
30
25
20
15
10
5
r
10
20
30
$\hat{r}$
20
15
10
5
0

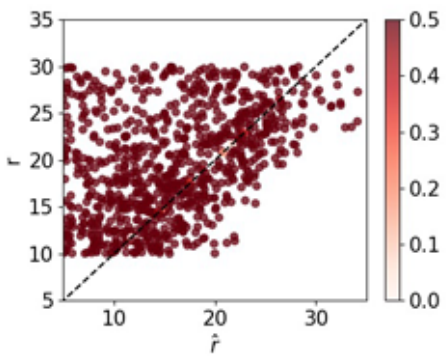

f_{3x3}
f_{5x5}
f_{7x7}
f_{3x3}

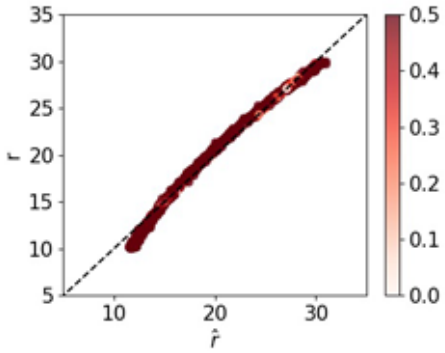

10
5
0
0
5000

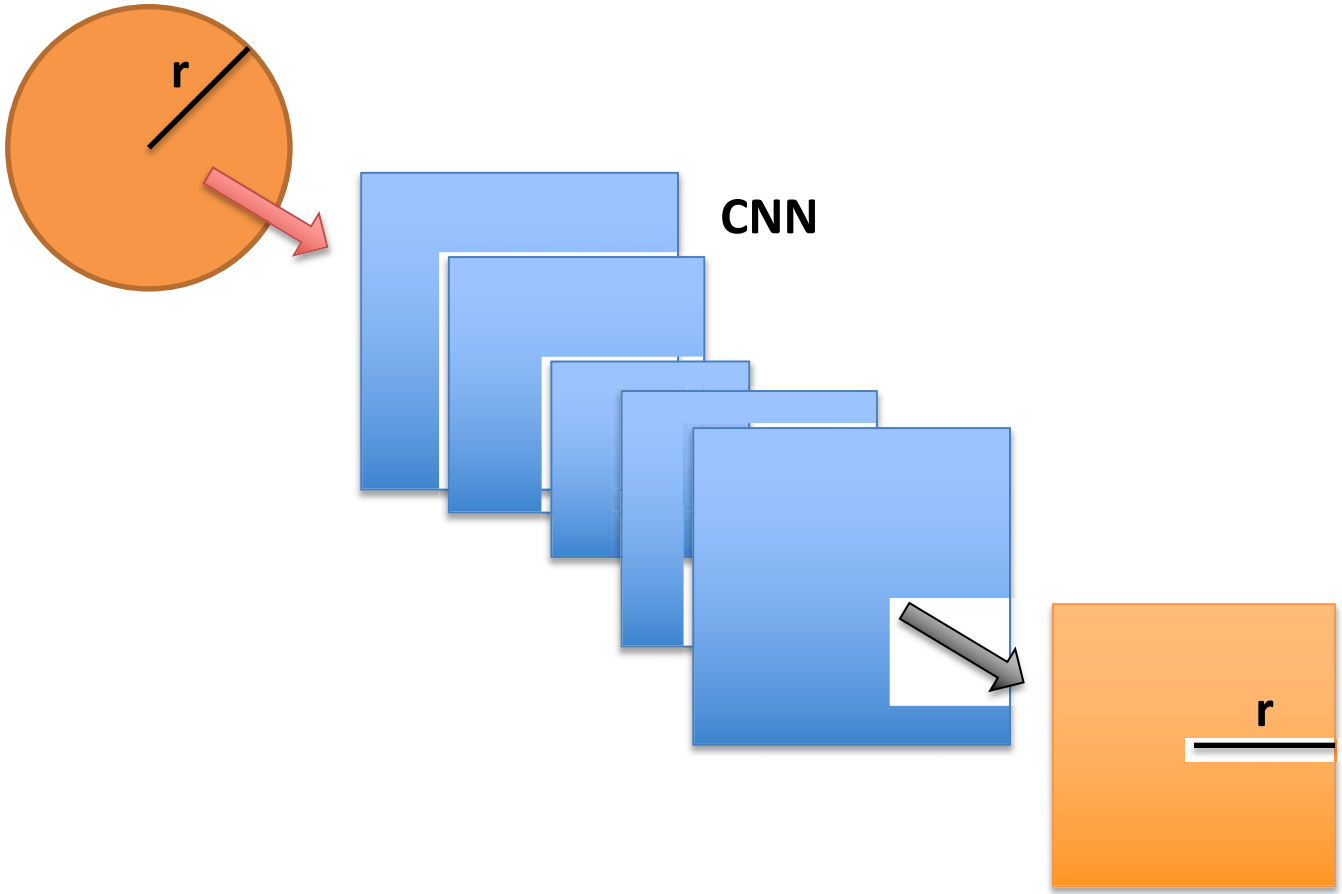

CNN
flat
dense
r
CNN
flat
sum
r

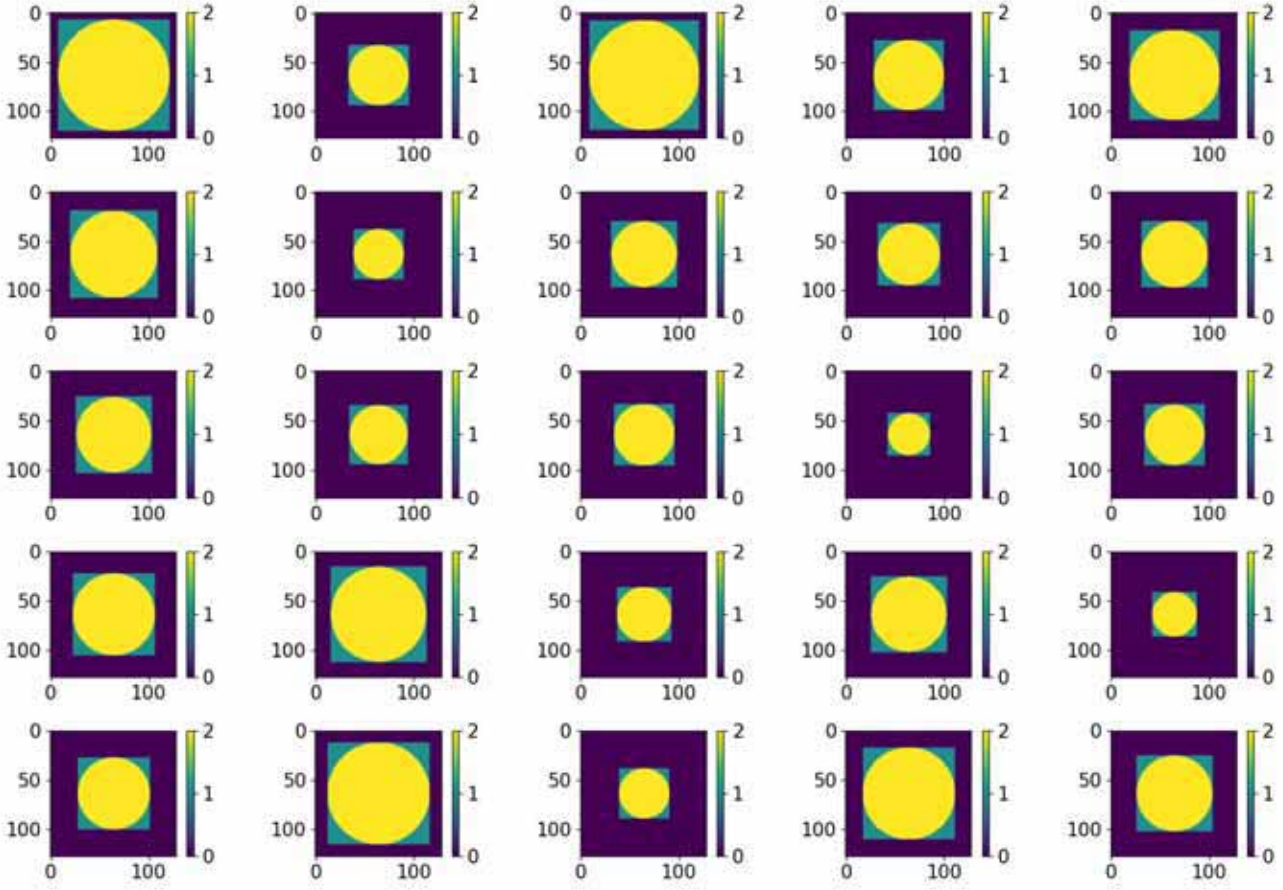

CNN
flat
dense
r

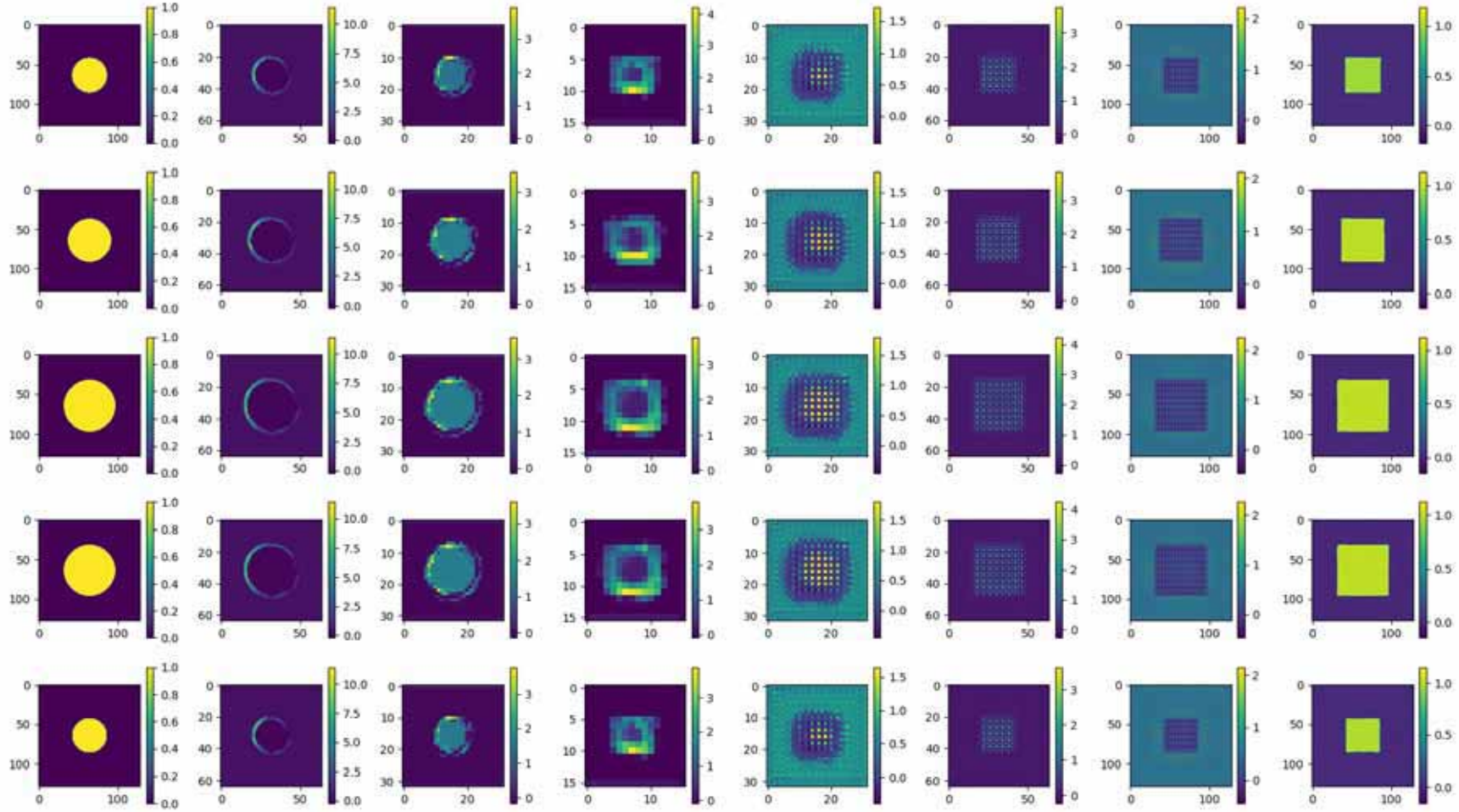

CNN
flat
sum
r

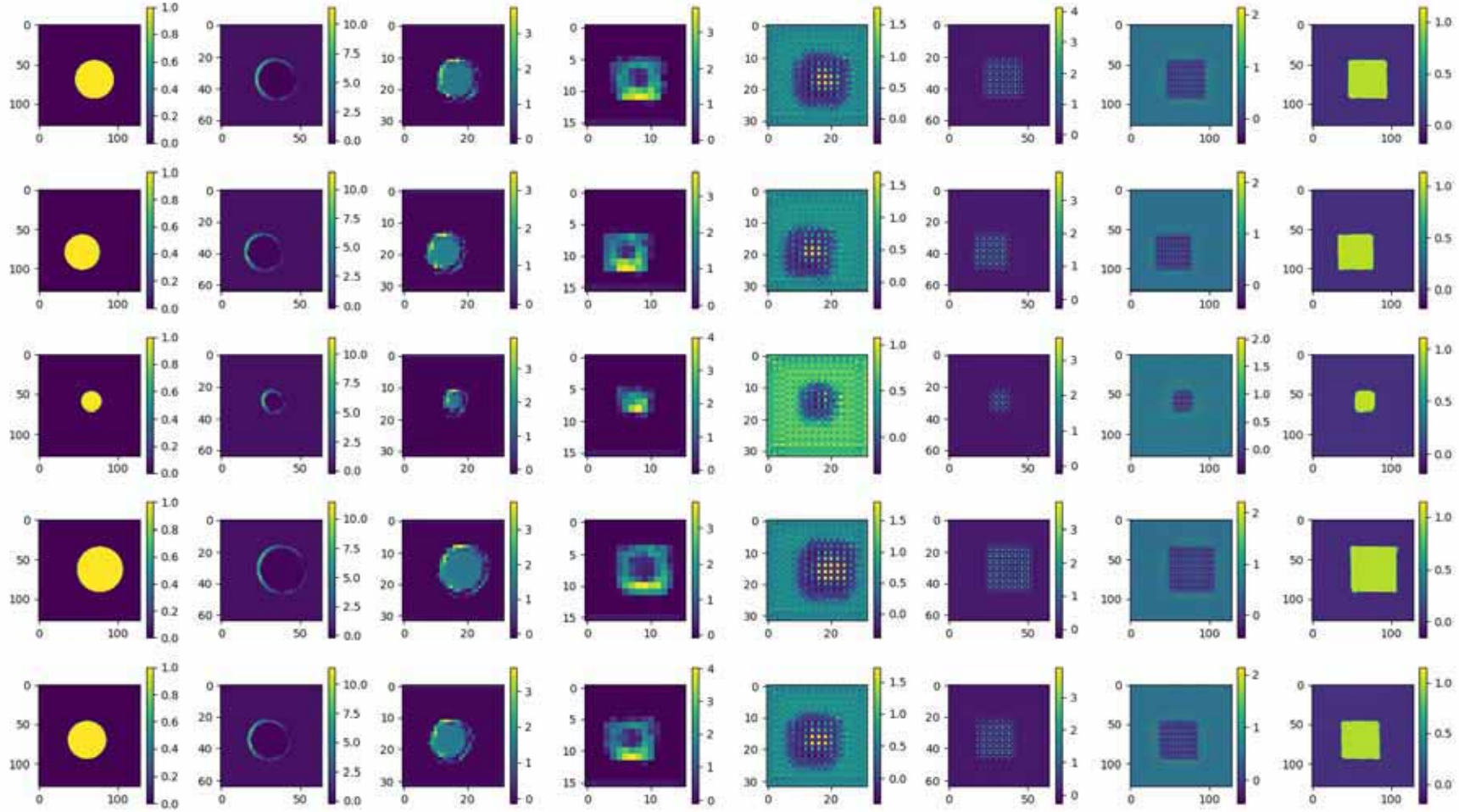

35
30
25
20
15
10
5
r
10
20
30
r̂
0.5
0.4
0.3
0.2
0.1
0.0

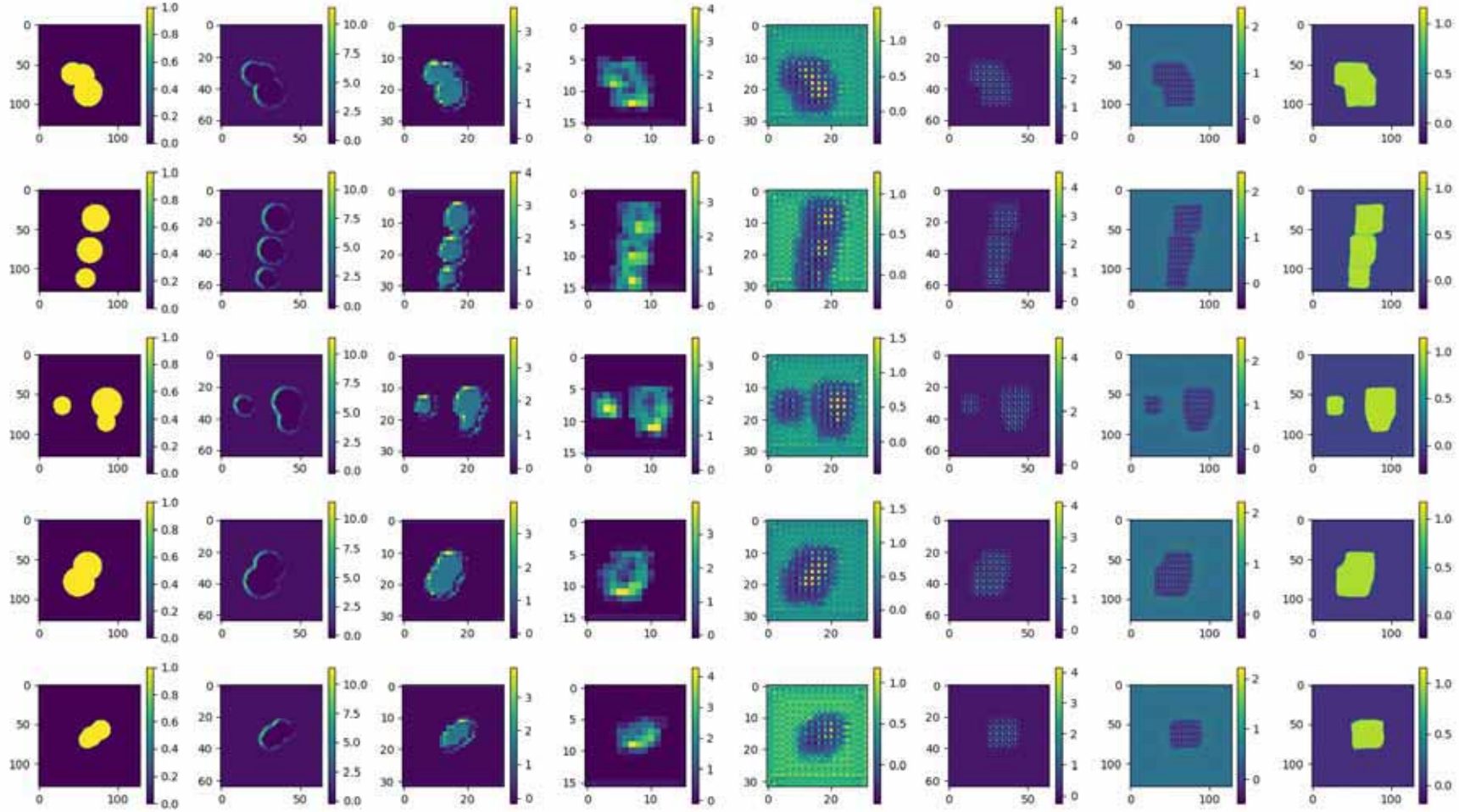

35
30
25
20
15
10
5
r
10
20
30
r̂
0.5
0.4
0.3
0.2
0.1
0.0

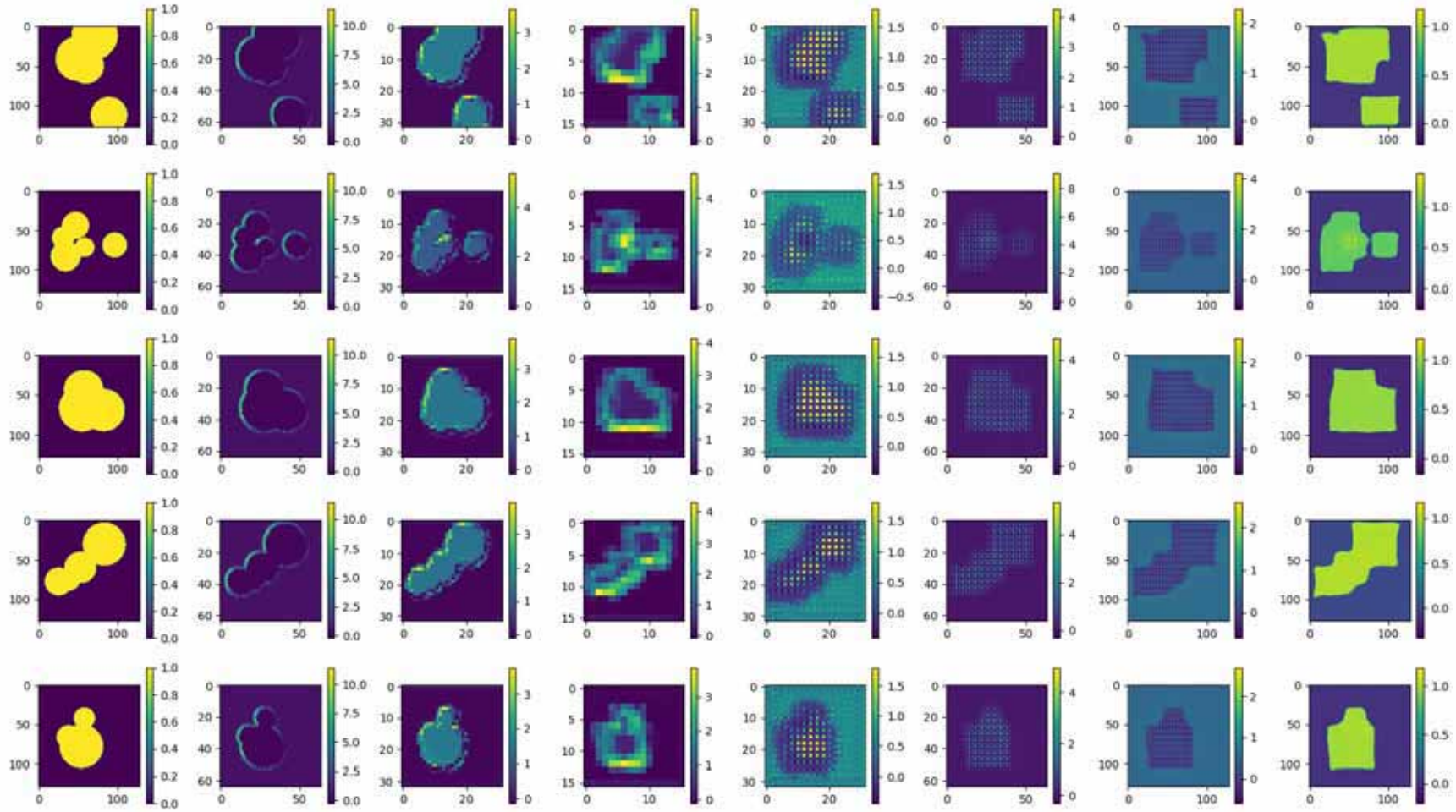

r
CNN
r

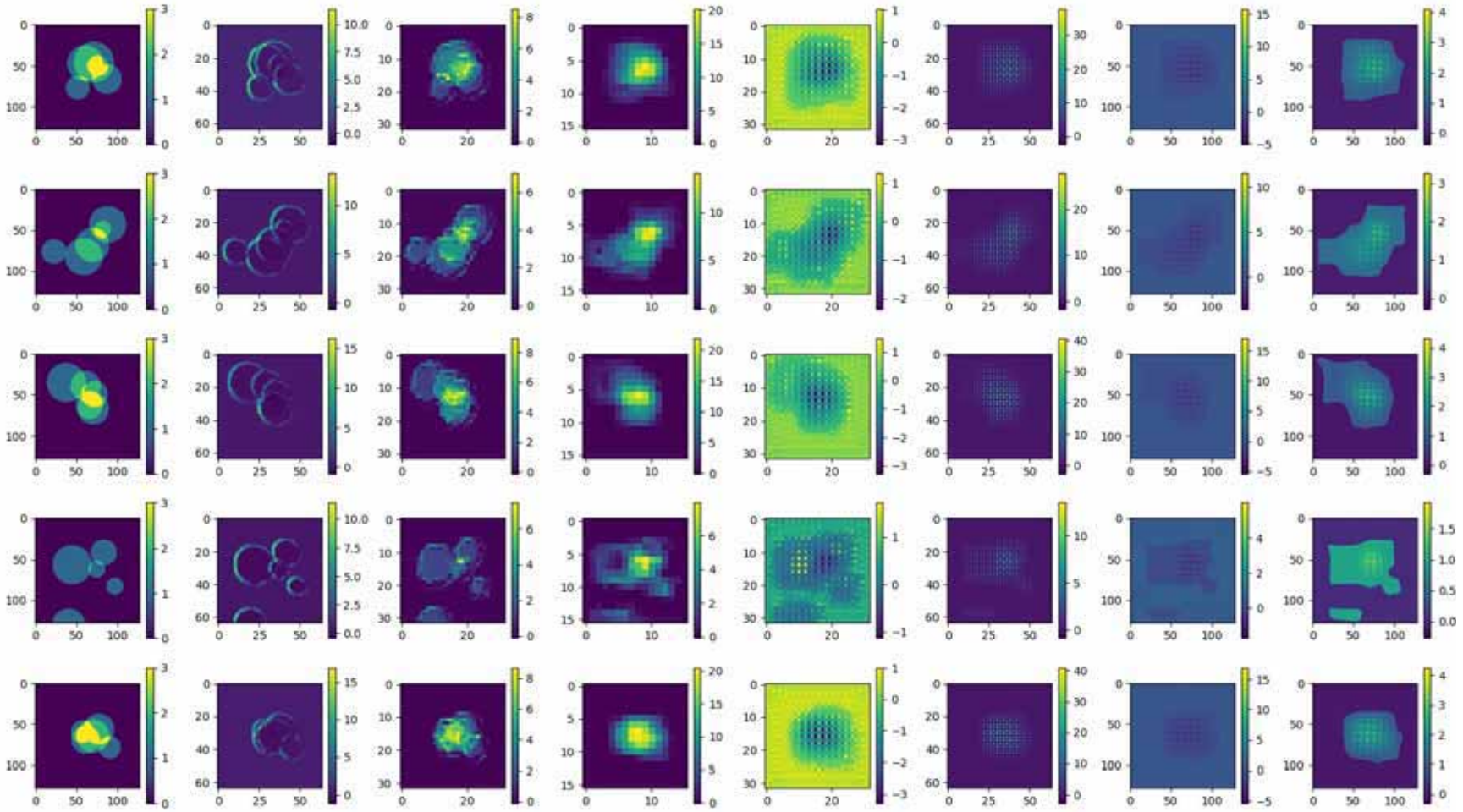

0
50
100
0
100
2
1
0

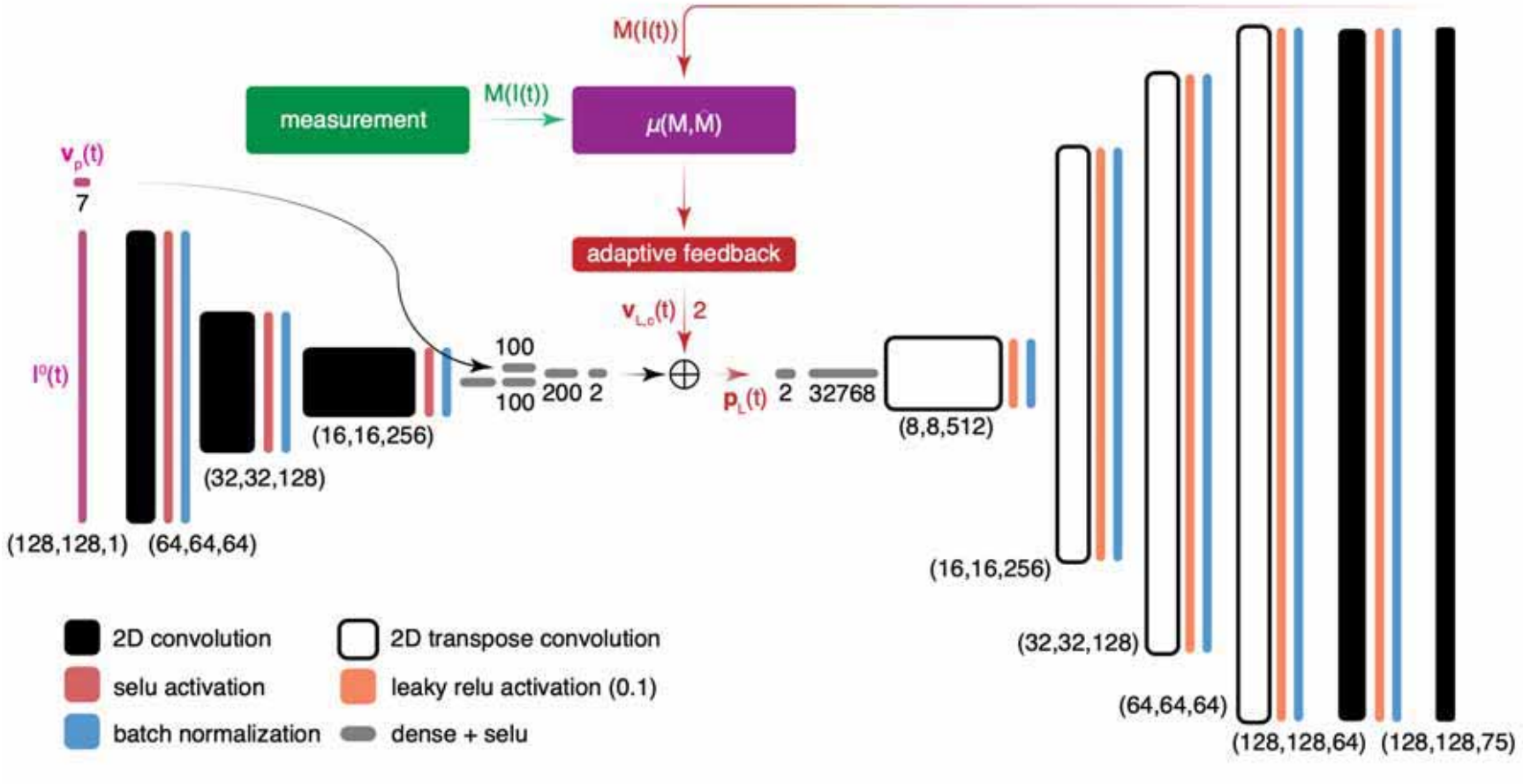











M(I(t))
measurement
M(I(t))
μ(M,M)
adaptive feedback
v_p(t)
7
I^o(t)
(128,128,1)
(64,64,64)
(32,32,128)
(16,16,256)
100
100
200
2
v_L,o(t)
2
p_L(t)
2
32768
(8,8,512)
(16,16,256)
(32,32,128)
(64,64,64)
(128,128,64)
(128,128,75)
2D convolution
2D transpose convolution
selu activation
leaky relu activation (0.1)
batch normalization
dense + selu

```
# Regularization for network weights
l2w = 1e-7

# Input image size
nx = 512
ny = 512

# Image input
image_input = Input(shape = (nx,ny,1))
# Accelerator parameter input

im_CNN_11 = Conv2D(32, kernel_size=3, strides=(1,1), padding='same', kernel_regularizer=l2_reg(l2w))(image_input)
im_CNN_11 = BatchNormalization()(im_CNN_11)
im_CNN_11 = layers.LeakyReLU(alpha=0.1)(im_CNN_11)
# 512x512x32
im_CNN_11 = MaxPool2D(pool_size=(2,2))(im_CNN_11)
# 256x256x32

im_CNN_12 = Conv2D(32, kernel_size=5, strides=(1,1), padding='same', kernel_regularizer=l2_reg(l2w))(image_input)
im_CNN_12 = BatchNormalization()(im_CNN_12)
im_CNN_12 = layers.LeakyReLU(alpha=0.1)(im_CNN_12)
# 512x512x32
im_CNN_12 = MaxPool2D(pool_size=(2,2))(im_CNN_12)
# 256x256x32


im_CNN_13 = Conv2D(32, kernel_size=7, strides=(1,1), padding='same', kernel_regularizer=l2_reg(l2w))(image_input)
im_CNN_13 = BatchNormalization()(im_CNN_13)
im_CNN_13 = layers.LeakyReLU(alpha=0.1)(im_CNN_13)
# 512x512x32
im_CNN_13 = MaxPool2D(pool_size=(2,2))(im_CNN_13)
# 256x256x32

im_CNN_123 = concatenate([im_CNN_11,im_CNN_12,im_CNN_13])
# 256x256x96

im_CNN_2 = Conv2D(32, kernel_size=3, strides=(1,1), padding='same', kernel_regularizer=l2_reg(l2w))(im_CNN_123)
im_CNN_2 = BatchNormalization()(im_CNN_2)
im_CNN_2 = layers.LeakyReLU(alpha=0.1)(im_CNN_2)
# 256x256x32

im_CNN_2 = MaxPool2D(pool_size=(2,2))(im_CNN_2)
# 128x128x32

im_flat = flatten()(im_CNN_2)
```


```
% conda install tensorflow
```

Model-Independent Adaptive Feedback

A. Scheinker and D. Scheinker, "Extremum Seeking with Discontinuous Dithers," *Automatica*, vol. 69, pp. 250-257, 2016.

A. Scheinker and D. Scheinker, "Extremum Seeking for Stabilization of Systems not Affine in Control," 2017.

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{p}, t) = \begin{bmatrix} f_1(x_1, \ldots, x_n, p_1, \ldots, p_m, t) \\ \vdots \\ f_n(x_1, \ldots, x_n, p_1, \ldots, p_m, t) \end{bmatrix}$$

$$y = V(\mathbf{x}, t) + n(t)$$

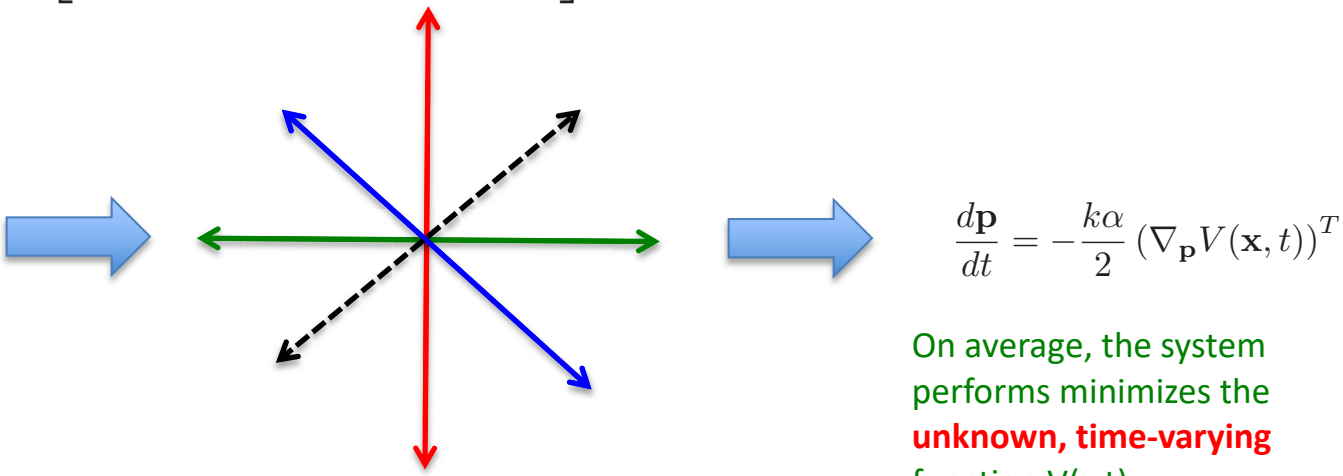
$$\frac{dp_1}{dt} = \sqrt{\alpha\omega_1}\cos(\omega_1 t + ky)$$

$$\frac{dp_2}{dt} = \sqrt{\alpha\omega_2}\cos(\omega_2 t + ky)$$

$$\frac{dp_3}{dt} = \sqrt{\alpha\omega_3}\cos(\omega_3 t + ky)$$

$$\vdots$$

$$\frac{dp_m}{dt} = \sqrt{\alpha\omega_m}\cos(\omega_m t + ky)$$



$$\frac{d\mathbf{p}}{dt} = -\frac{k\alpha}{2}\left(\nabla_{\mathbf{p}} V(\mathbf{x}, t)\right)^T$$

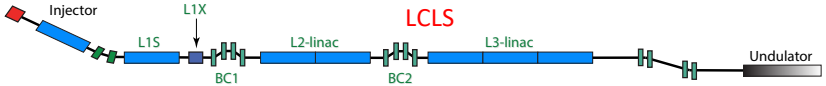
On average, the system performs minimizes the **unknown, time-varying** function V(x,t)

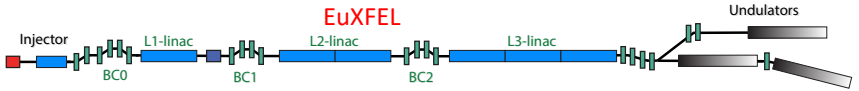
Allows simultaneous tuning of ALL parameters in parallel.

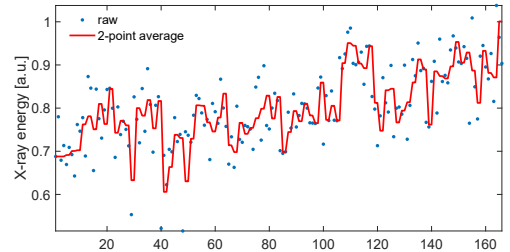
$$\omega_i = \omega r_i, \;\; r_i \neq r_j \implies \text{for any } t > 0$$

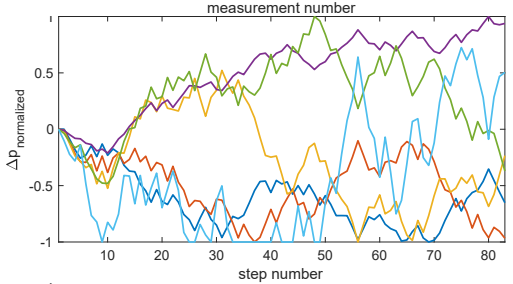
$$\lim_{\omega\to\infty} \langle \cos(\omega_i t), \cos(\omega_j t) \rangle = \lim_{\omega\to\infty} \int_0^t \cos(\omega_i \tau)\cos(\omega_j \tau) d\tau = 0$$

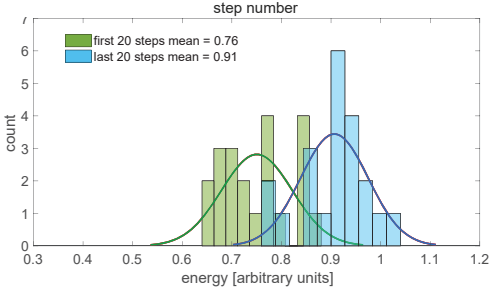
Model-independent Adaptive Feedback for pulse energy maximization at **LCLS** & **EuXFEL**

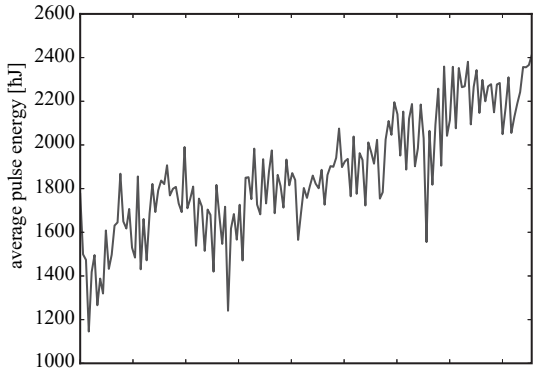


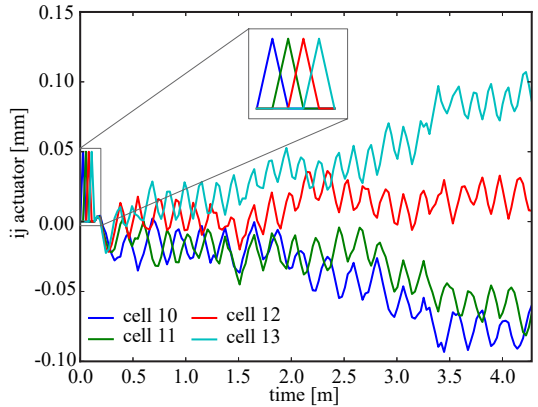


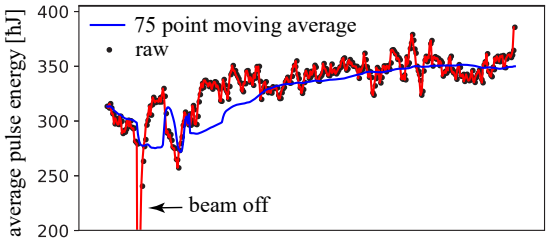


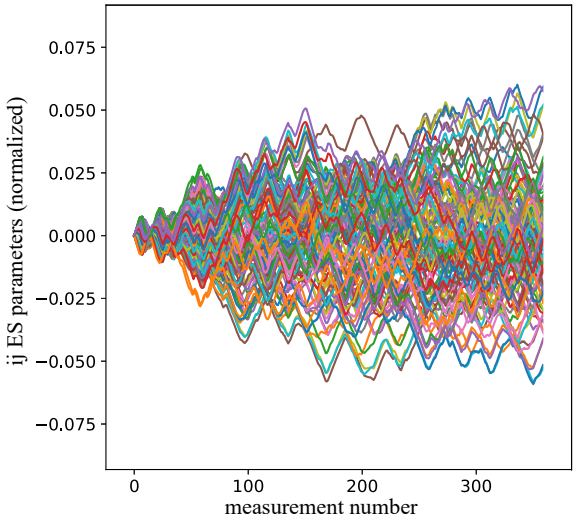








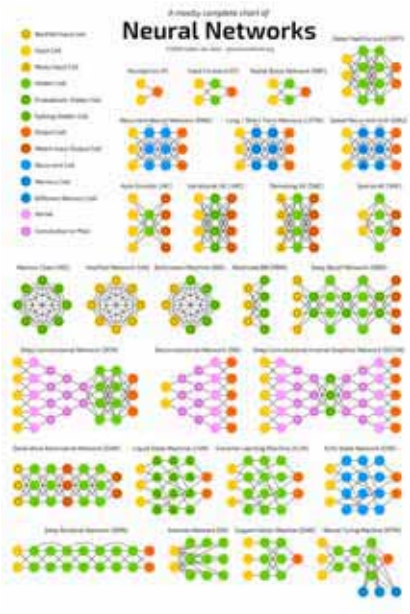


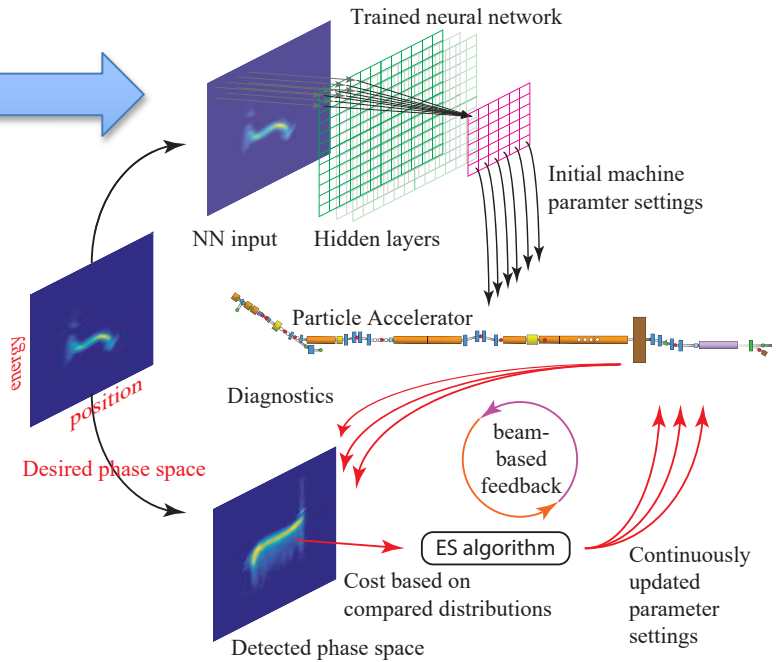


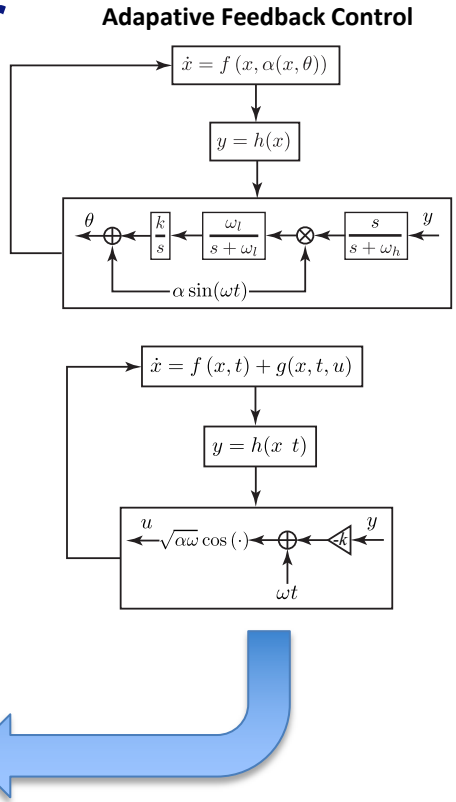
A. Scheinker, et al. “Model-independent tuning for maximizing free electron laser pulse energy.” Physical Review Accelerators and Beams, 22.8, 082802, 2019. https://doi.org/10.1103/PhysRevAccelBeams.22.082802

Adaptive Machine Learning for Time-Varying Systems

Adaptive Machine Learning for Time Varying Systems

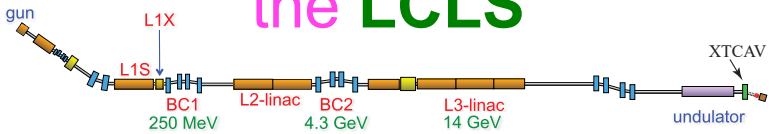




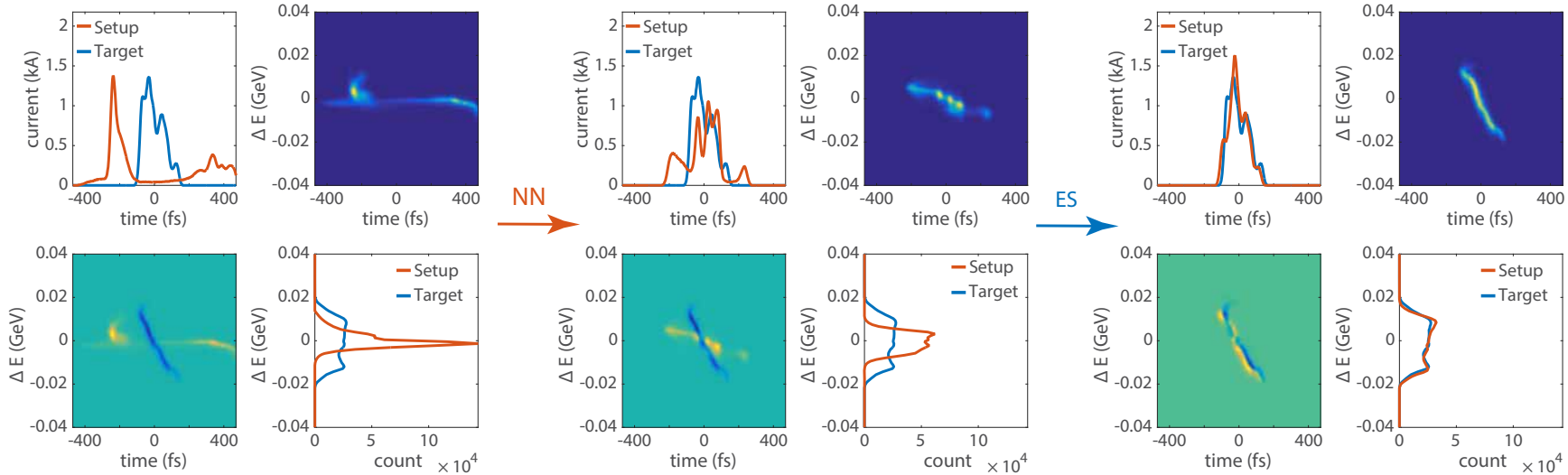


A. Scheinker, et al. “Demonstration of model-independent control of the longitudinal phase space of electron beams in the Linac-coherent light source with Femtosecond resolution.” Physical Review Letters, 121.4, 044801, 2018. https://doi.org/10.1103/PhysRevLett.121.044801

Adaptive ML for automatic longitudinal phase space control at the **LCLS**



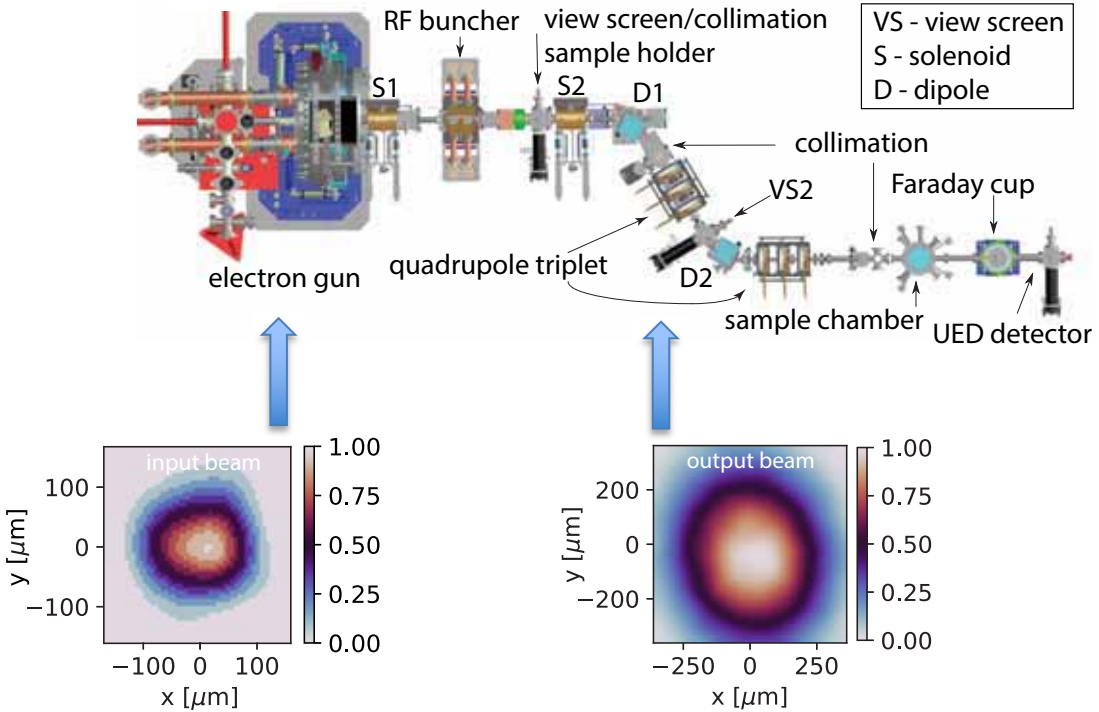
$$C = \int_{-\Delta L}^{\Delta L} \int_{-\Delta E}^{\Delta E} |\hat{\rho}(z, E) - \rho(z, E)| \, dE dz$$



A. Scheinker, et al. “Demonstration of model-independent control of the longitudinal phase space of electron beams in the Linac-coherent light source with Femtosecond resolution.” Physical Review Letters, 121.4, 044801, 2018. https://doi.org/10.1103/PhysRevLett.121.044801

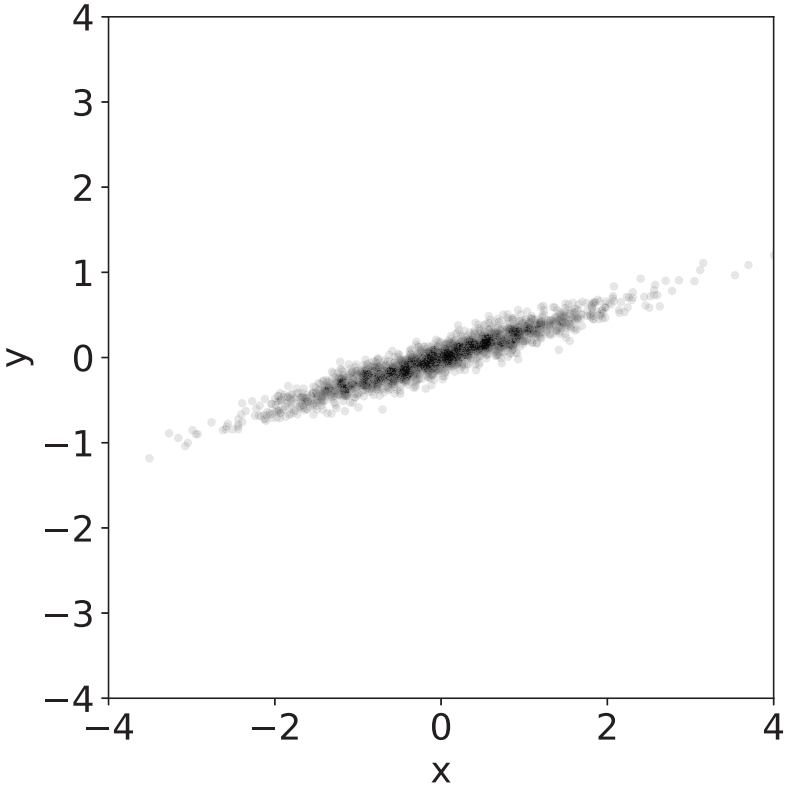
Adaptive ML for Inverse Mapping of Downstream Beam Measurements to Time-Varying Input Beam Distributions

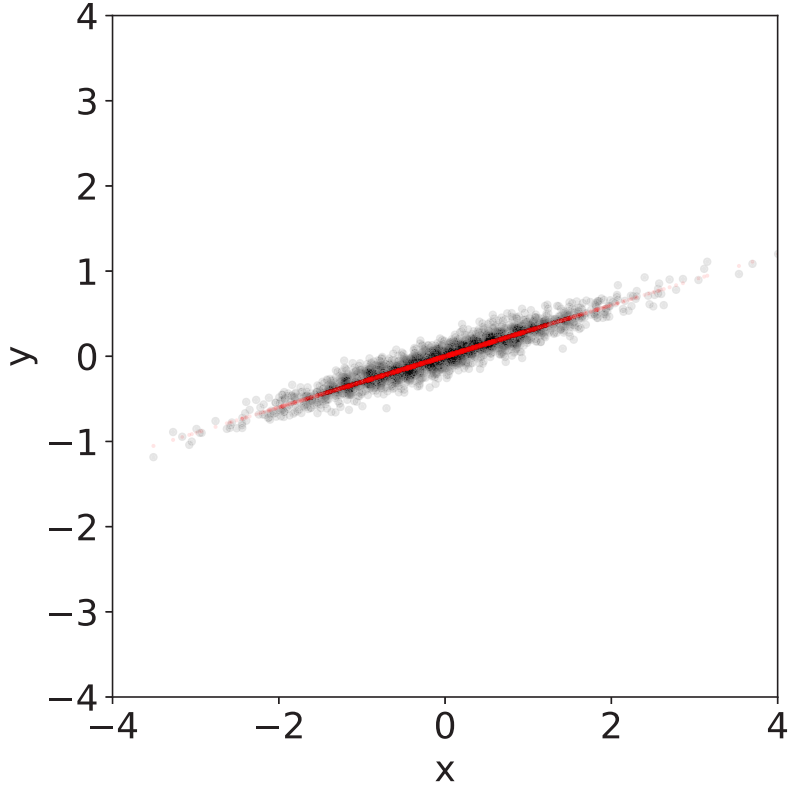
HiRES Ultrafast Electron Diffraction (UED) at LBNL



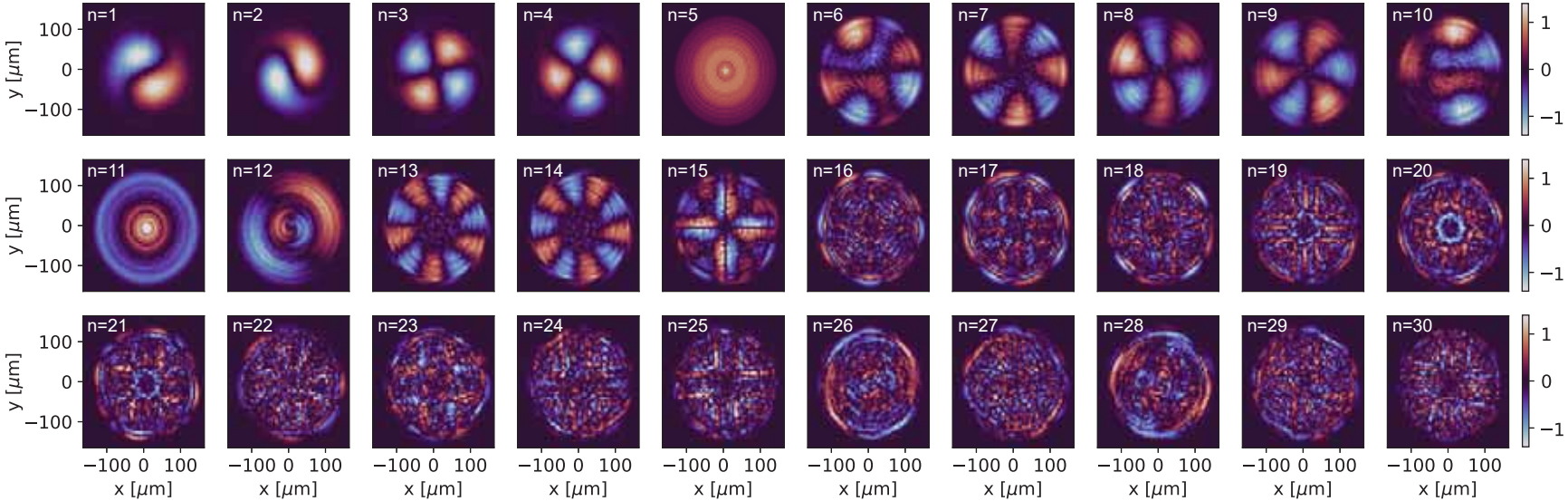
Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.

Principal Component Analysis (PCA)





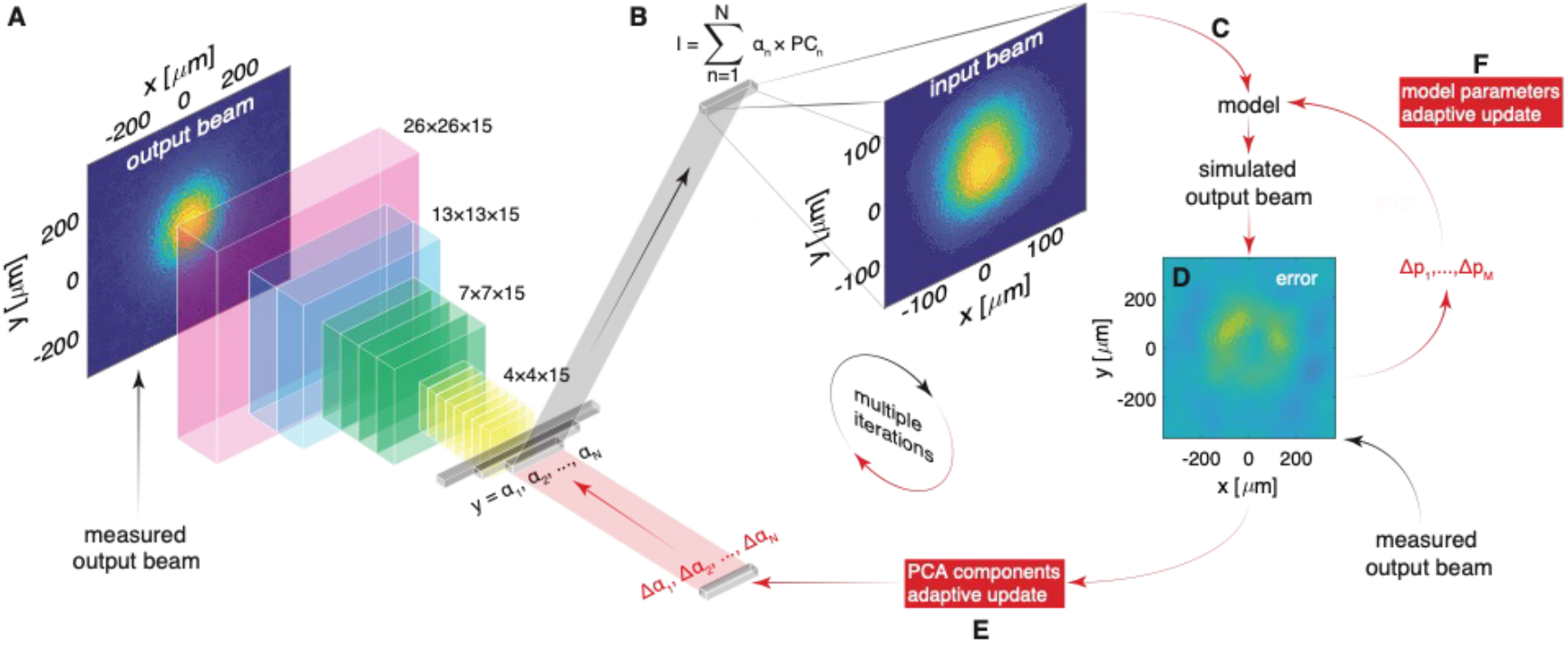
PCA component basis for electron beam.



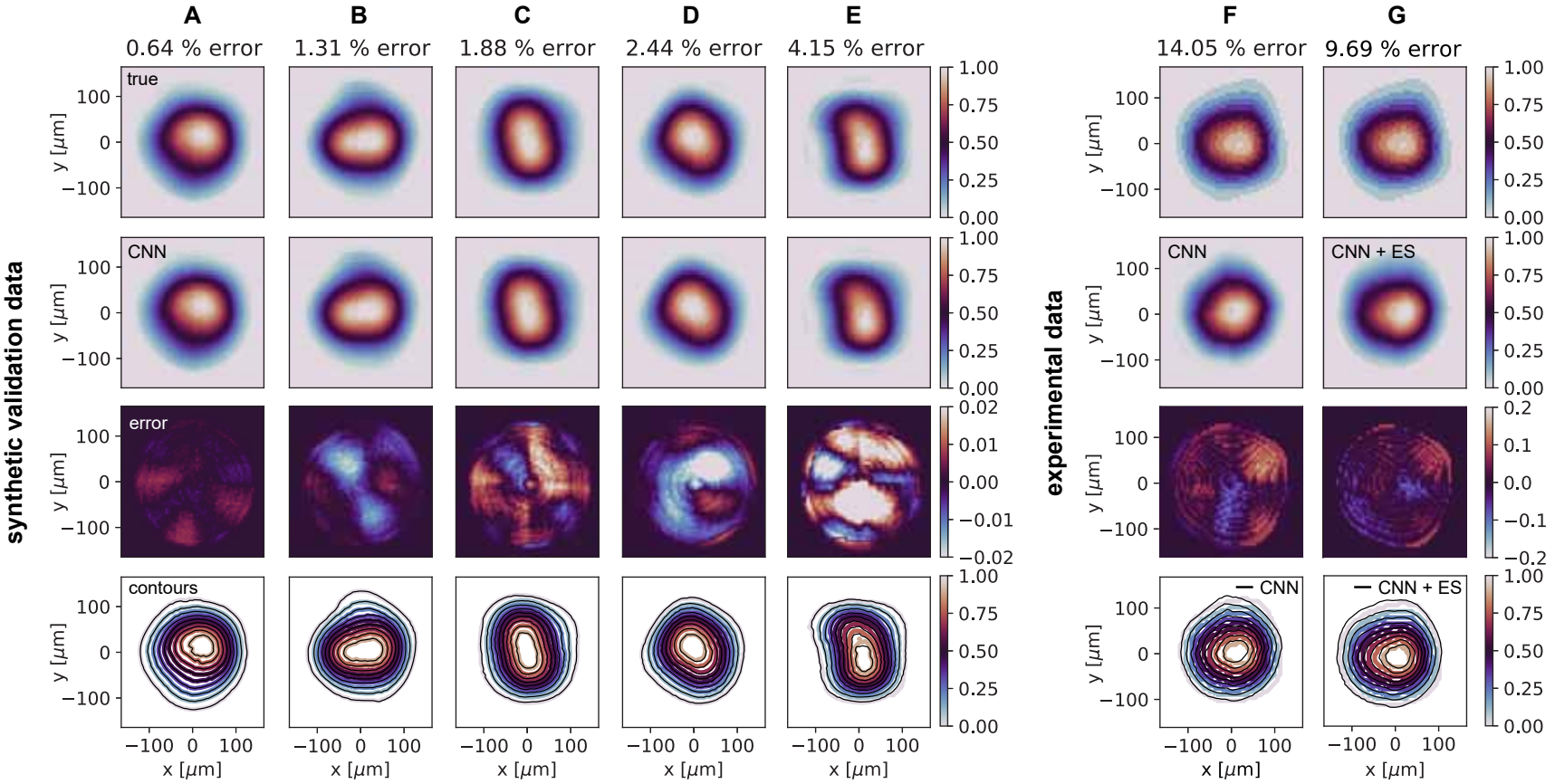
$$I_{i,N_{pca}} = \sum_{n=1}^{N_{pca}} \alpha_{i,n} \times \mathrm{PC}_n.$$

Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.

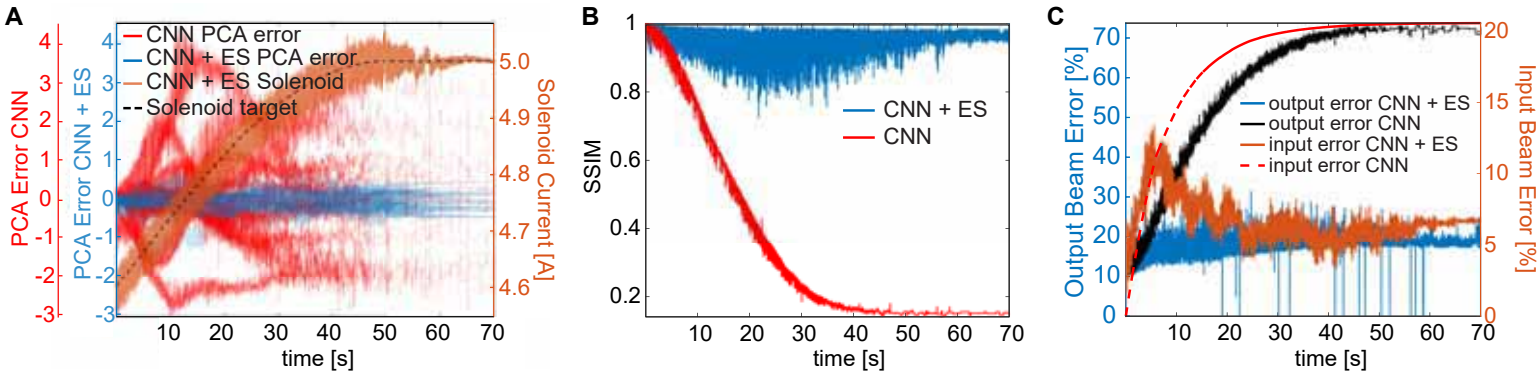
AML for adaptive inverse physics models



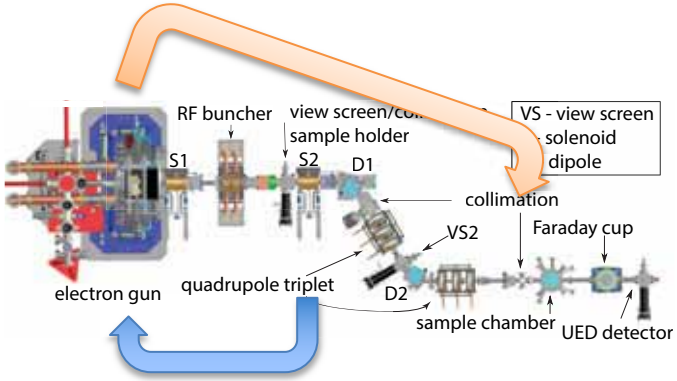
Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.

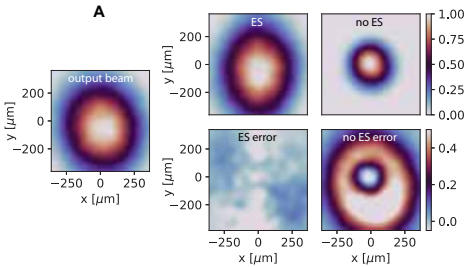


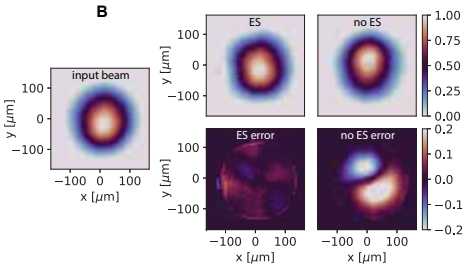
Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.

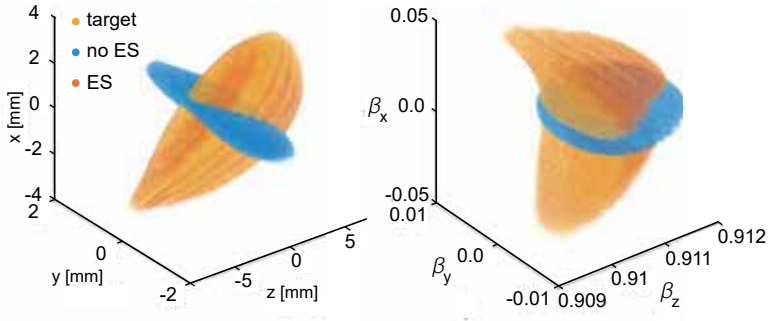


Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.

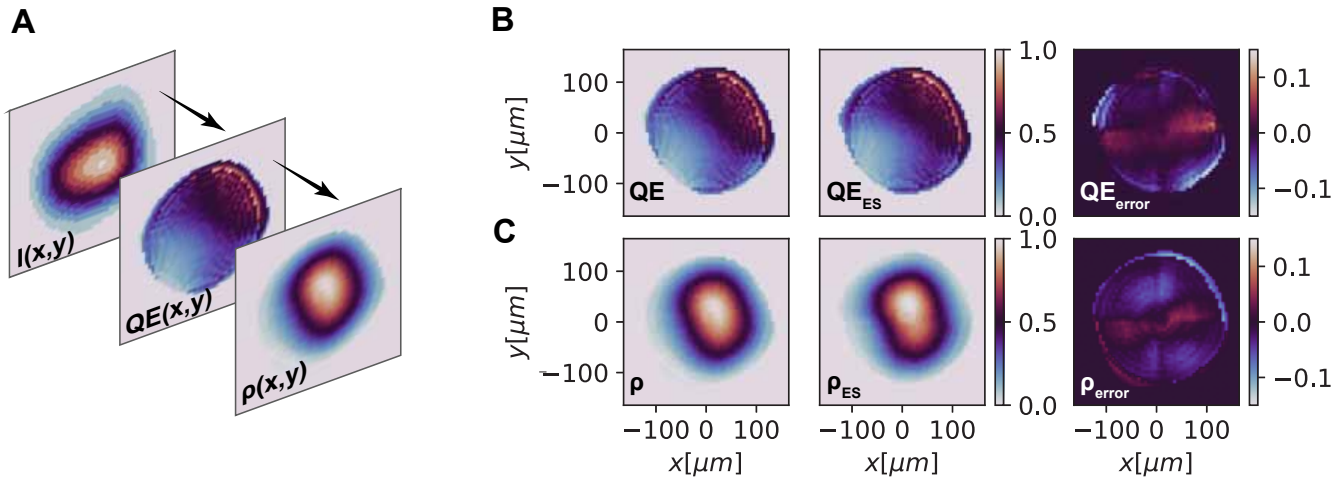






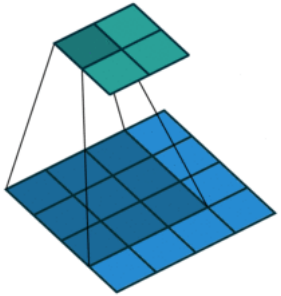


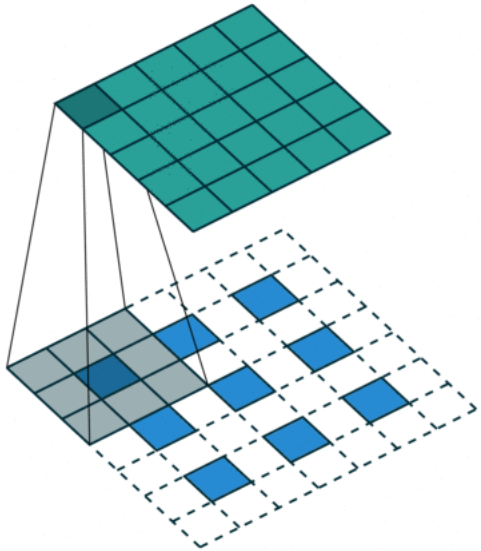
Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.



Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.

Encoder-Decoder Convolutional Neural Networks



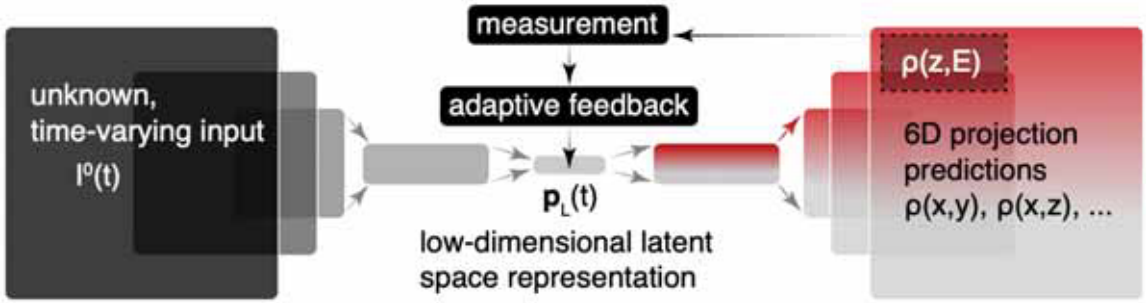


$$I^1_{i_1,j_1} = b^1 + \sum_{n=1}^{N_f} w_n \times f_n \left(b^0_n + \sum_{i=-1}^{1} \sum_{j=-1}^{1} F_{0,ij,n} \times I^0_{i_0+i,j_0+j} \right)$$

CNN animations: Vincent Dumoulin, Francesco Visin

Adaptive Machine Learning (AML) for Time-Varying Systems – Adaptively Tuning the Latent Space

General approach for any complex time-varying system



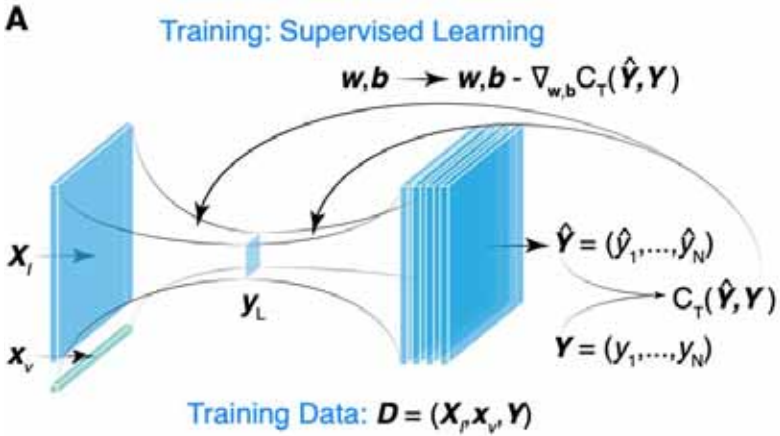
A. Scheinker. "Adaptive machine learning for time-varying systems: low dimensional latent space tuning." *Journal of Instrumentation* 16.10 (2021): P10008.

A. Scheinker, F. Cropp, S. Paiagua, & D. Filippetto. “An adaptive approach to machine learning for compact particle accelerators.” *Scientific Reports* **11,** 19187, 2021.

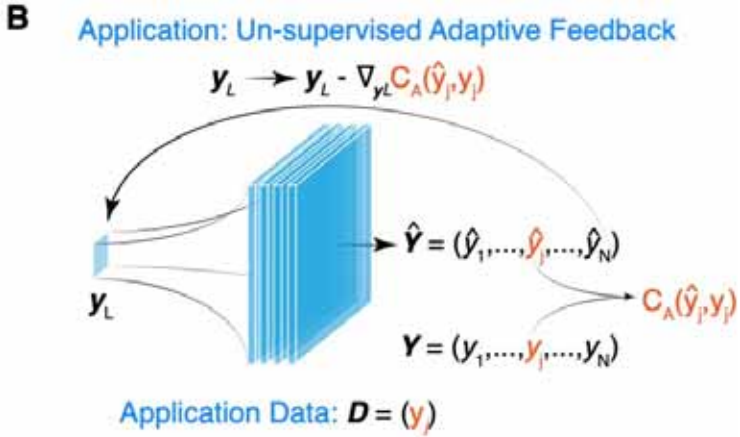
Physics-Informed Adaptive ML for 6D phase space diagnostics. Observational biases introduced directly through data that embody the underlying physics to learn functions that reflect the physical structure of the data. Encoder-decoder CNN for nonlinear data compression: Low-dimensional latent space tuning.

$$(x, y, z, x', y', E)$$

$$(x, y), (x, z), (x, x'), (x, y'), (x, E)$$
$$(x', y), (x', z), (x', y'), (x', E)$$
$$(y, z), (y, y'), (y, E)$$
$$(y', z), (y', E)$$
$$(z, E)$$

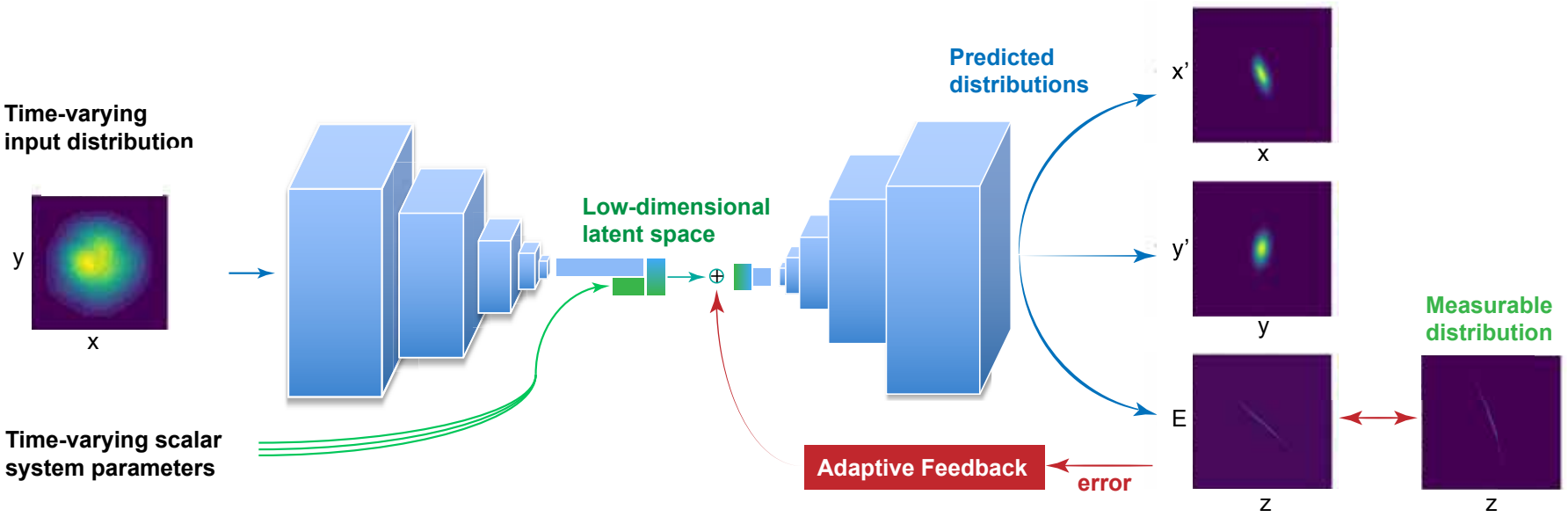


A. Scheinker. "Adaptive machine learning for time-varying systems: low dimensional latent space tuning." *Journal of Instrumentation* 16.10 (2021): P10008.



A. Scheinker, F. Cropp, S. Paiagua, & D. Filippetto. “An adaptive approach to machine learning for compact particle accelerators.” *Scientific Reports* **11,** 19187, 2021.

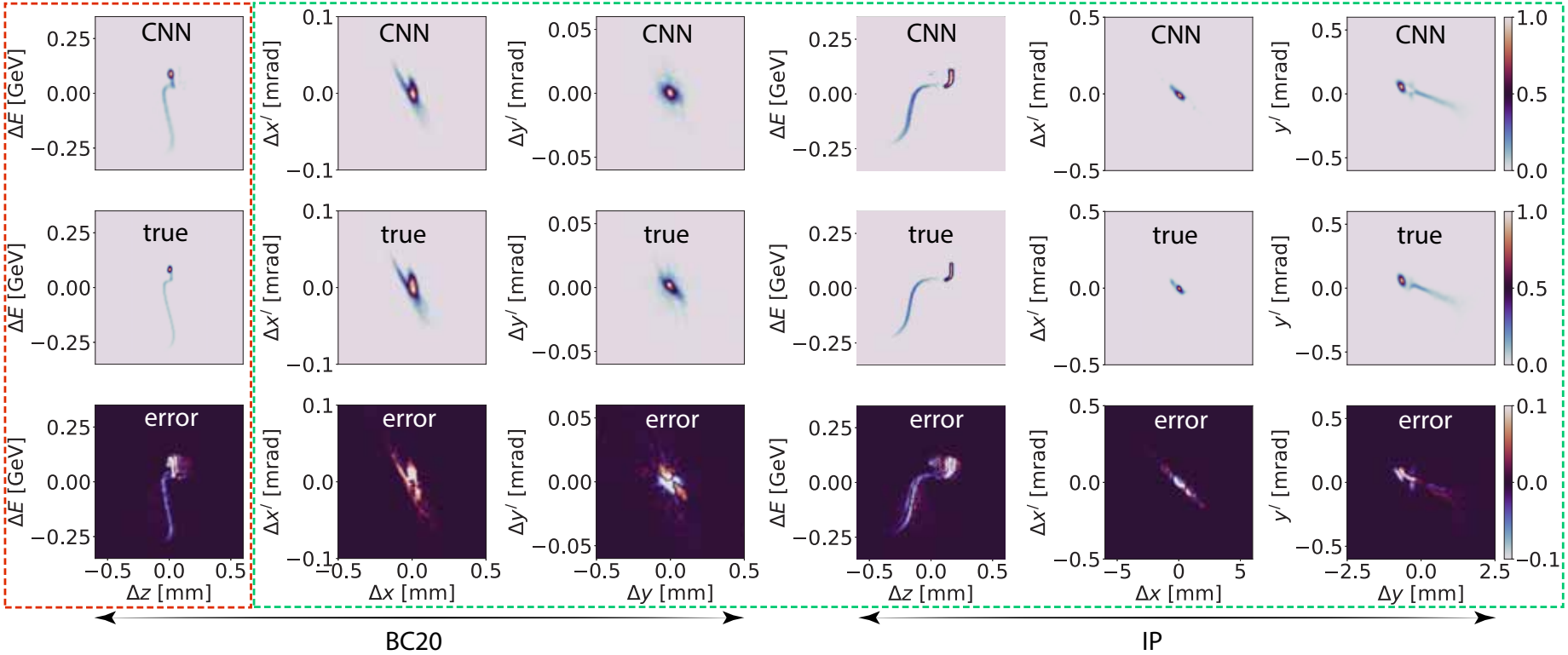
Adaptive Machine Learning (AML) for Time-Varying Systems – Adaptively Tuning the Latent Space

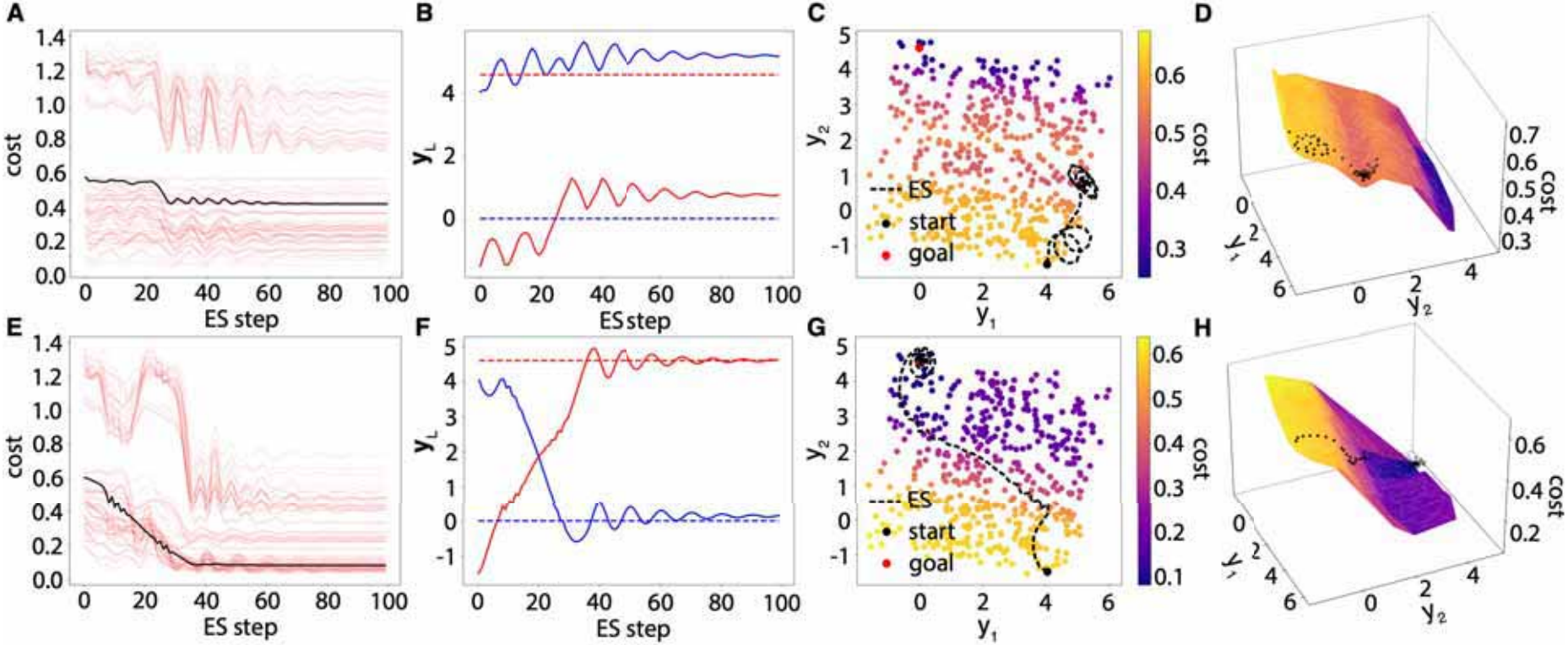


Scheinker, A., Cropp, F., Paiagua, S., & Filippetto, D. (2021). Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators. *arXiv preprint arXiv:2102.10510*.

RF Gun
L0
L1
BC11
L2
BC14
L3
BC20
Final Focus and Experiment
Interaction Point (IP)
Δy [mm]
Δx' [mrad]
Δz [mm]
Δy' [mrad]
ΔE
ΔE [GeV]
Δx [mm]
Δx' [mrad]
Δy [mm]
Δy' [mrad]
Δz [mm]
(x,y,q,Φx,Φy,Vx,Vy)

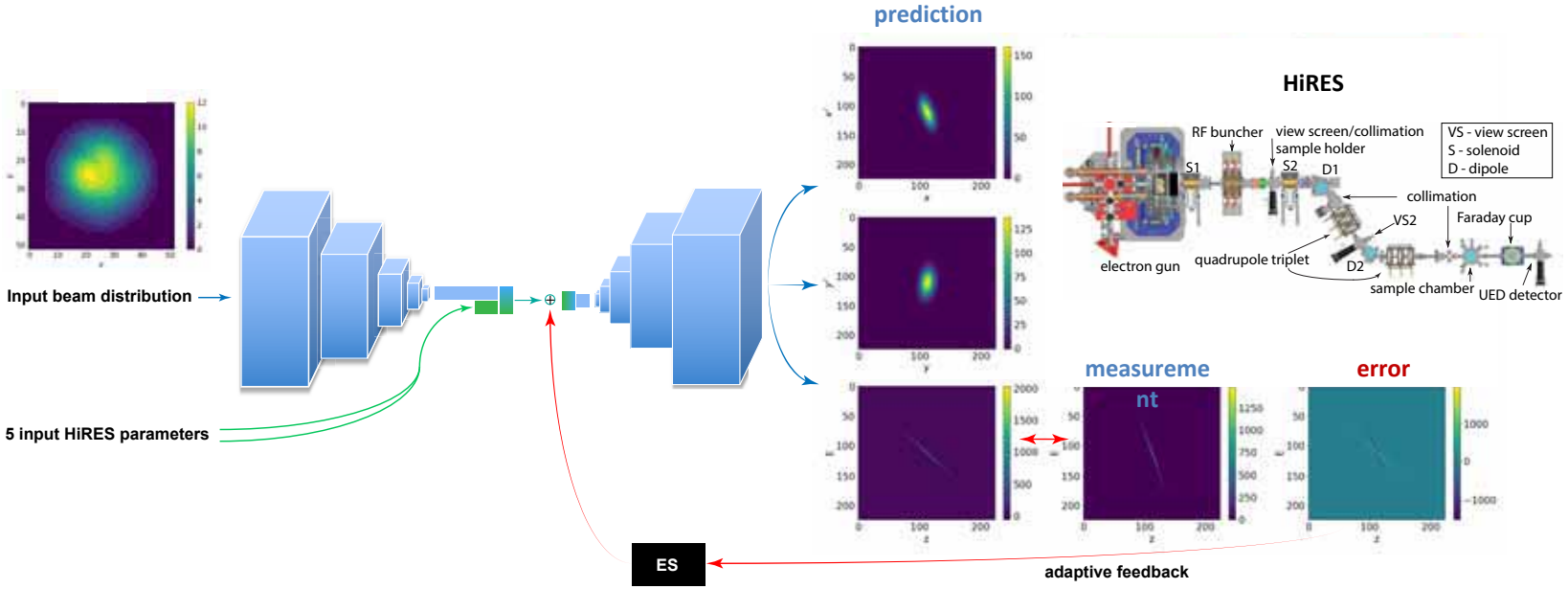
Adaptive Latent Space Tuning

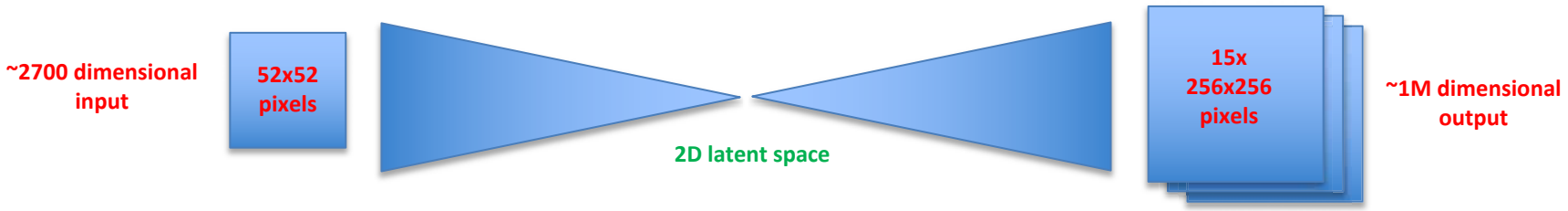



A
B
C
D
E
F
G
H
cost
ES step
y_L
y_1
y_2
ES
start
goal

AML Development at HiRES – Compact Ultra-fast Electron Diffraction (UED)

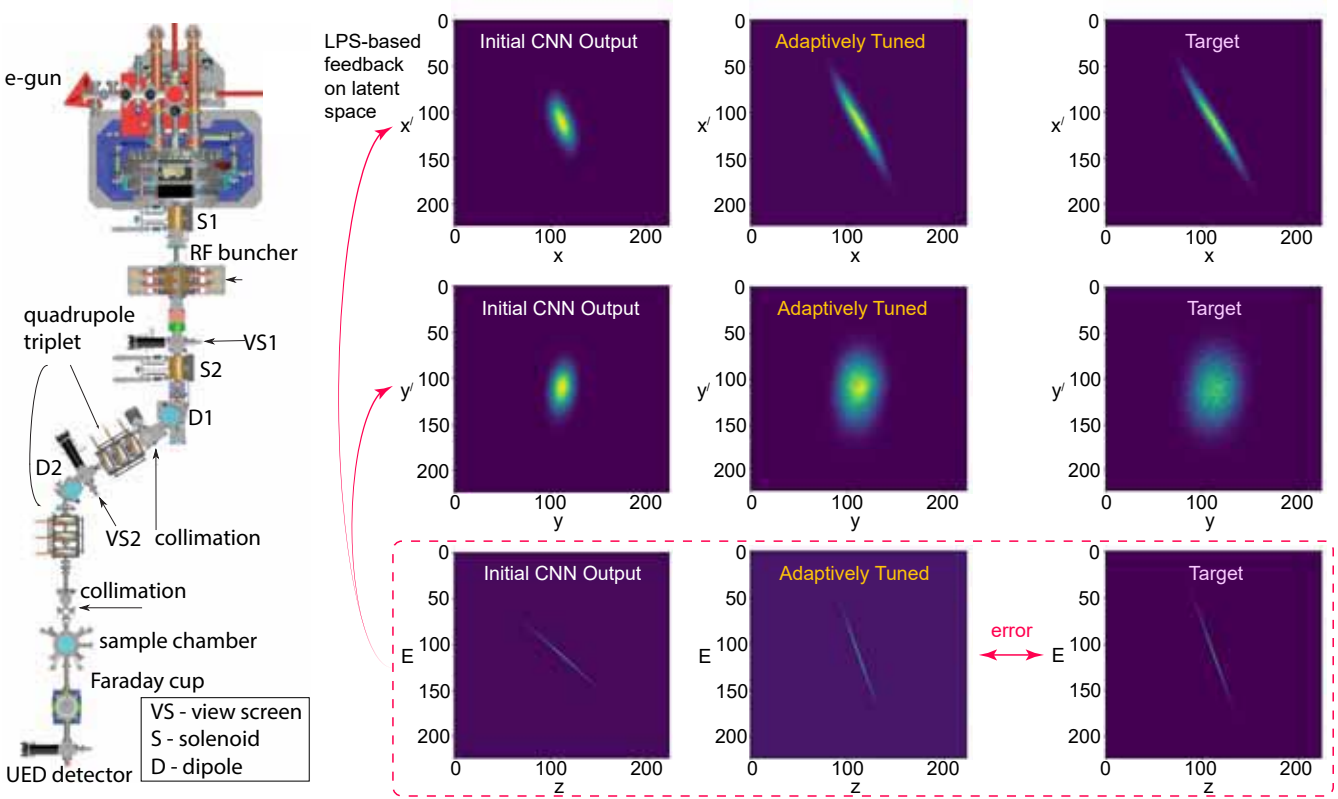
Longitudinal phase space measurements used to guide adaptive feedback within 2D latent space to predict all 2D projections of a beam's 6D phase space.





Adaptive ML-Based Diagnostics @ HiRES

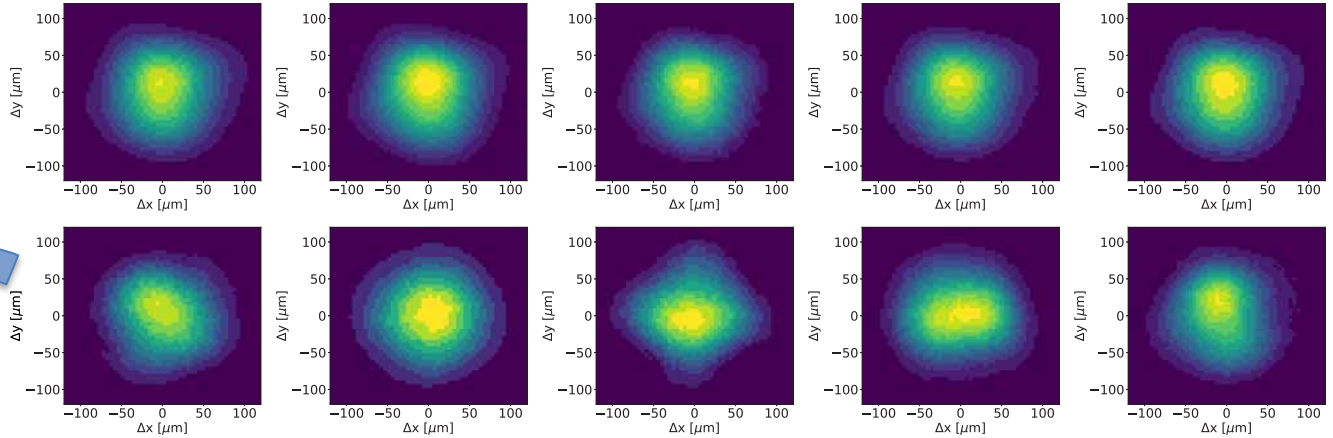
Work with: Eric Cropp and Daniele Filippetto

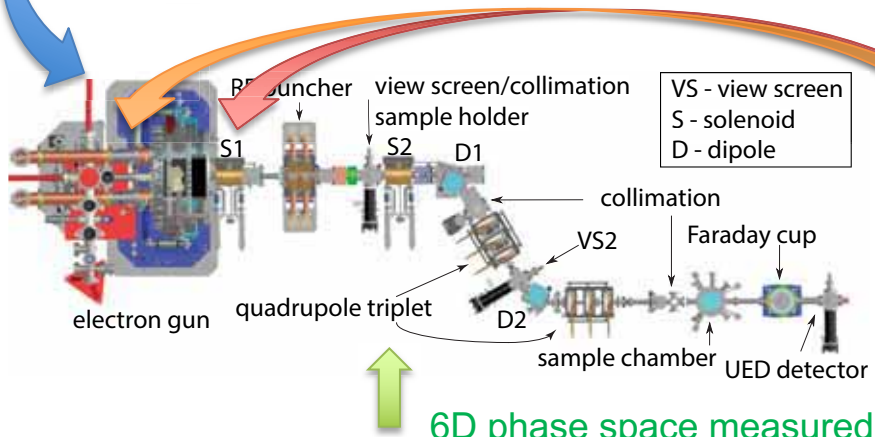


A. Scheinker. "Adaptive machine learning for time-varying systems: low dimensional latent space tuning." *Journal of Instrumentation* 16.10 (2021): P10008.
https://doi.org/10.1088/1748-0221/16/10/P10008

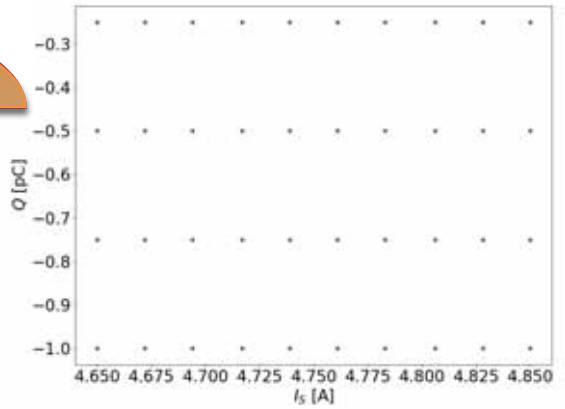
HiRES Numerical Study – AML for 6D Phase Space

Trained with Measured and Synthetic Input Images



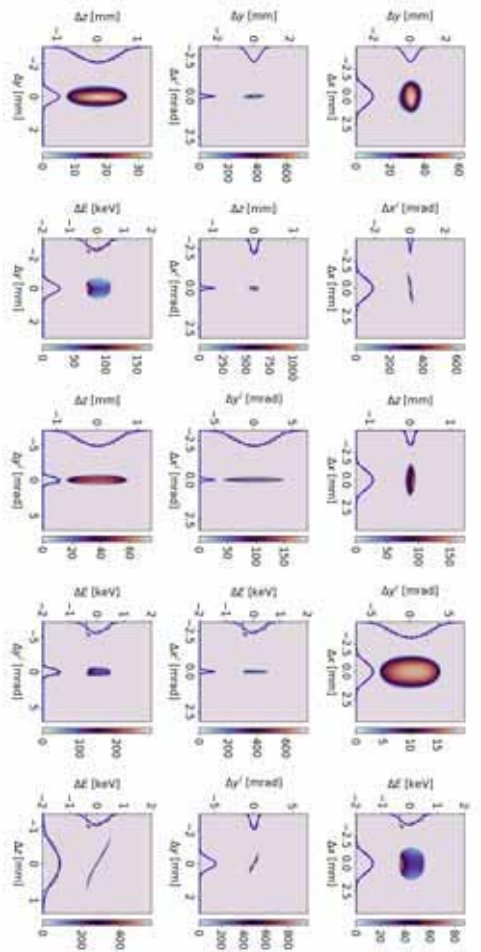


6D phase space measured

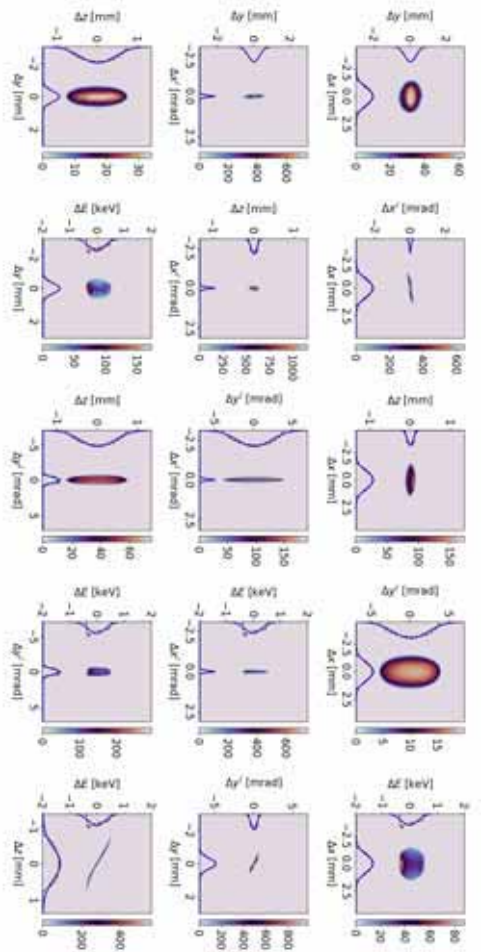


Each image simulated over a grid of 4 bunch charges and 10 solenoid currents

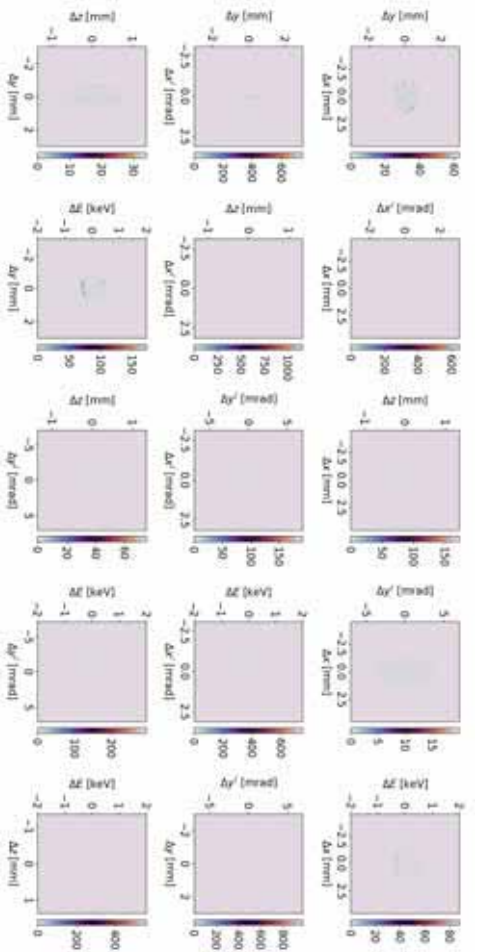
True



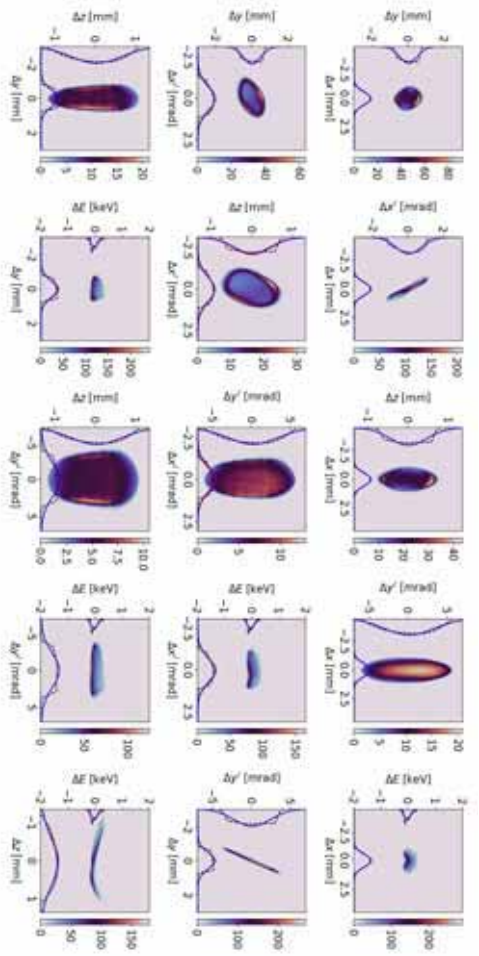
CNN



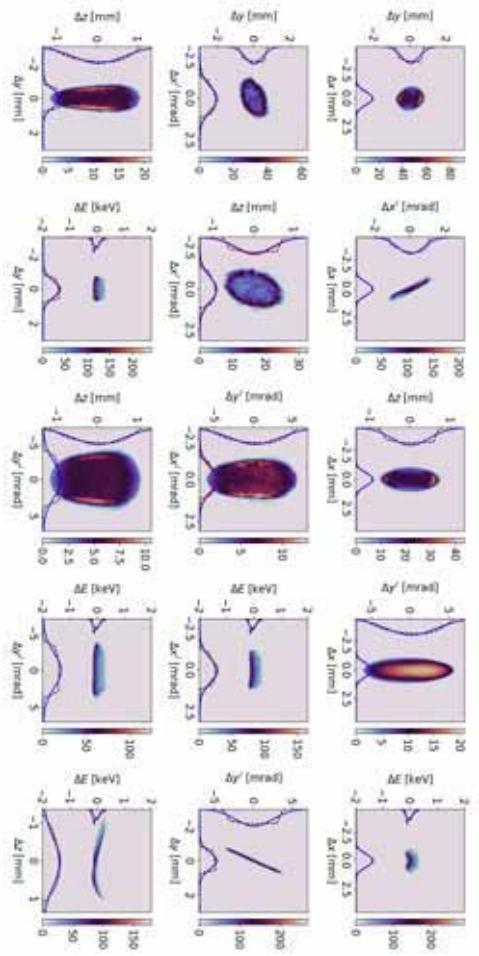
Difference



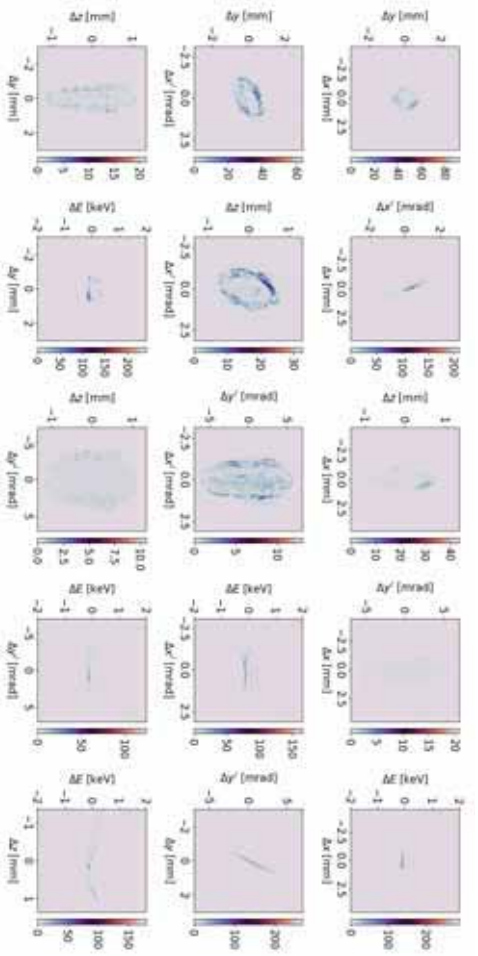
True



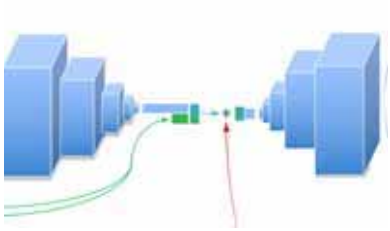
CNN

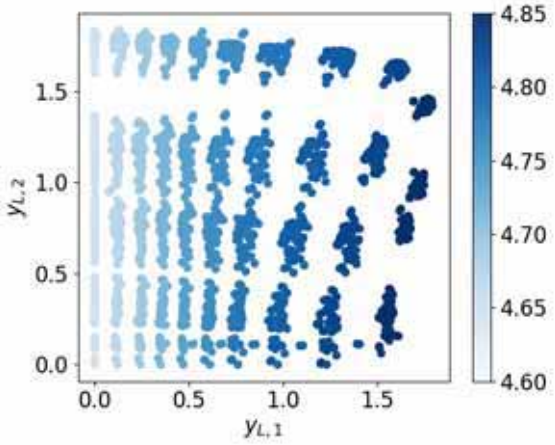


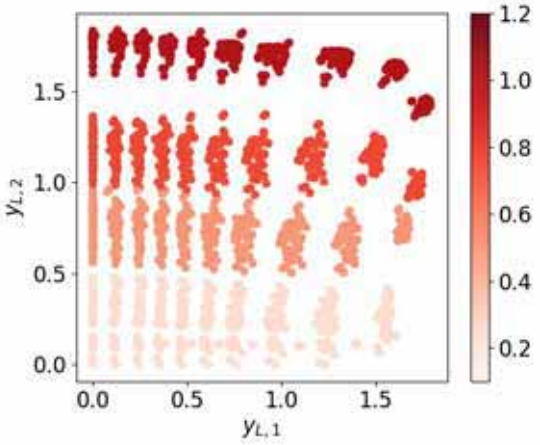
Difference

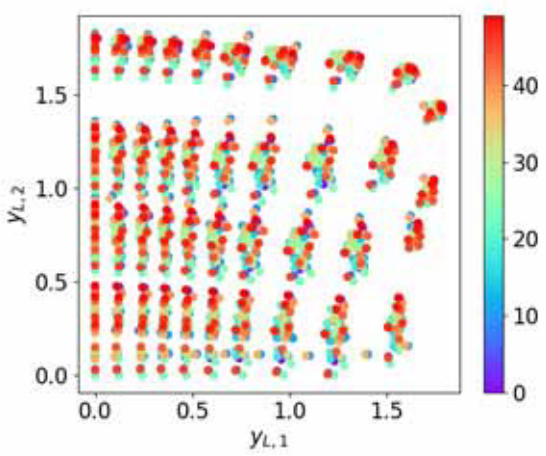


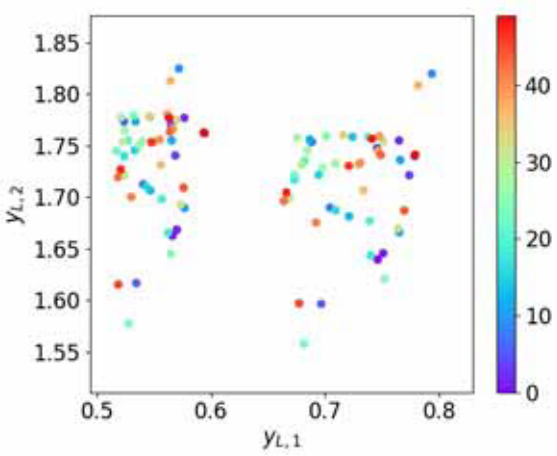
HiRES – Latent space representation

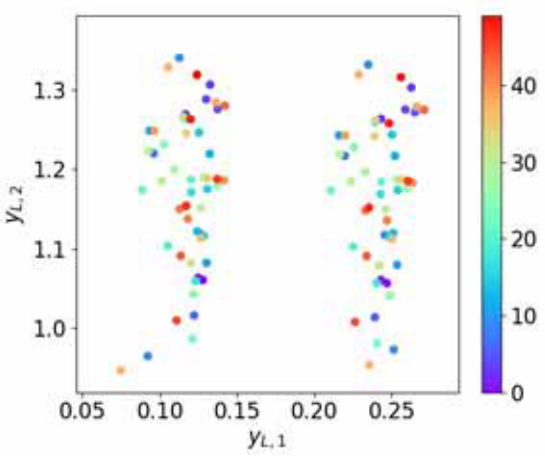




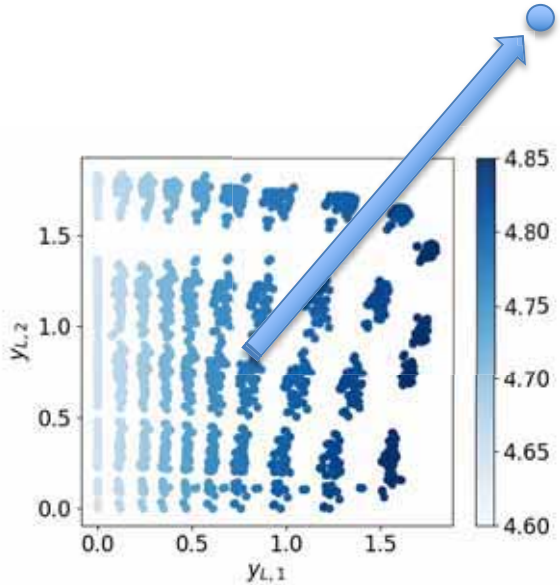




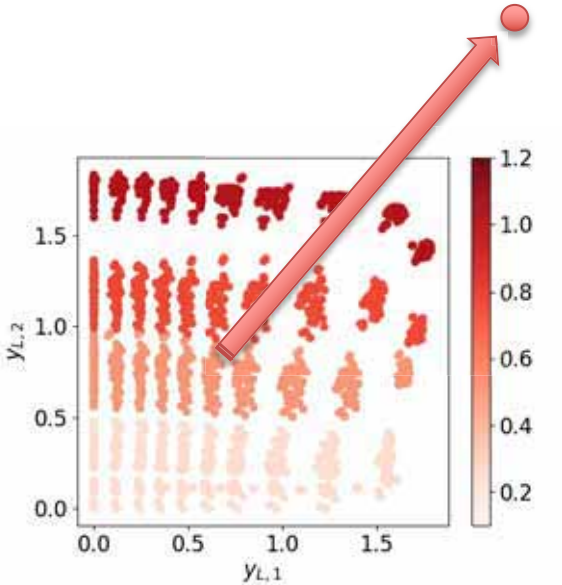




Robustness test: Moving far beyond the span of the training data to a unseen input beam distribution, higher solenoid strength, and larger charge.



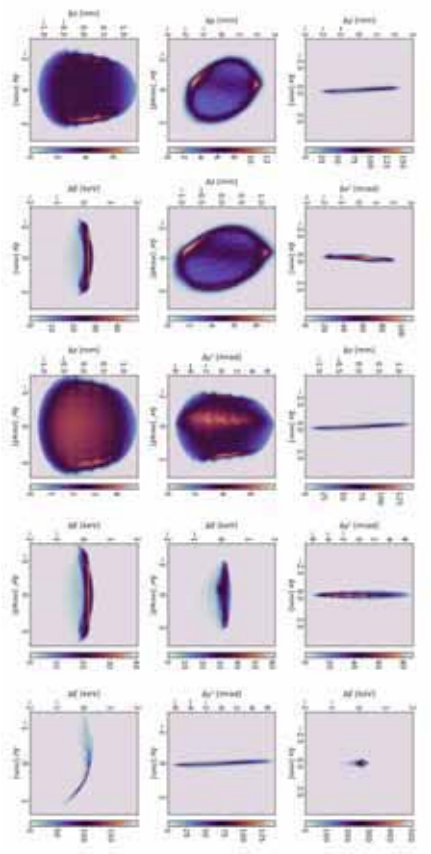
Colored by solenoid strength [A]



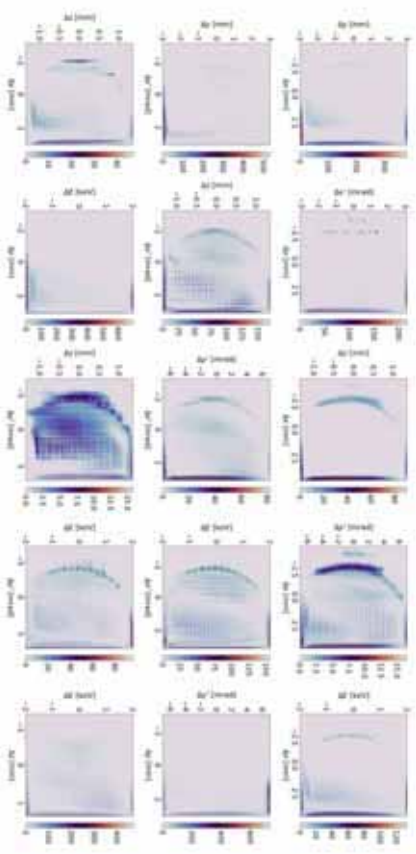
Colored by charge [pC]

6D phase space projections for beam and parameters far outside of training set.

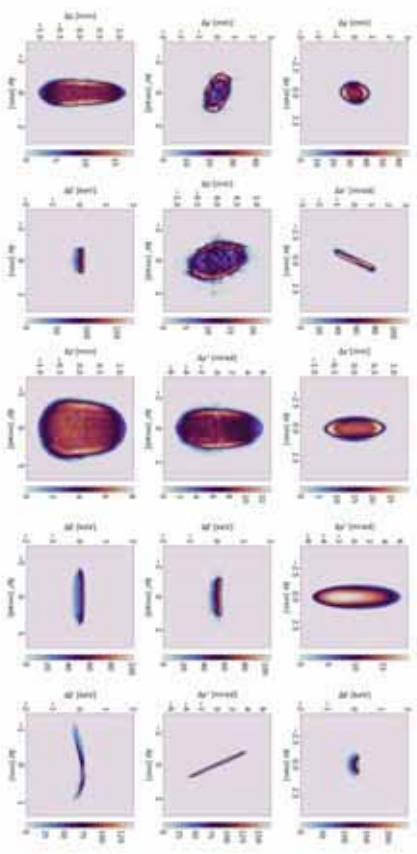
True



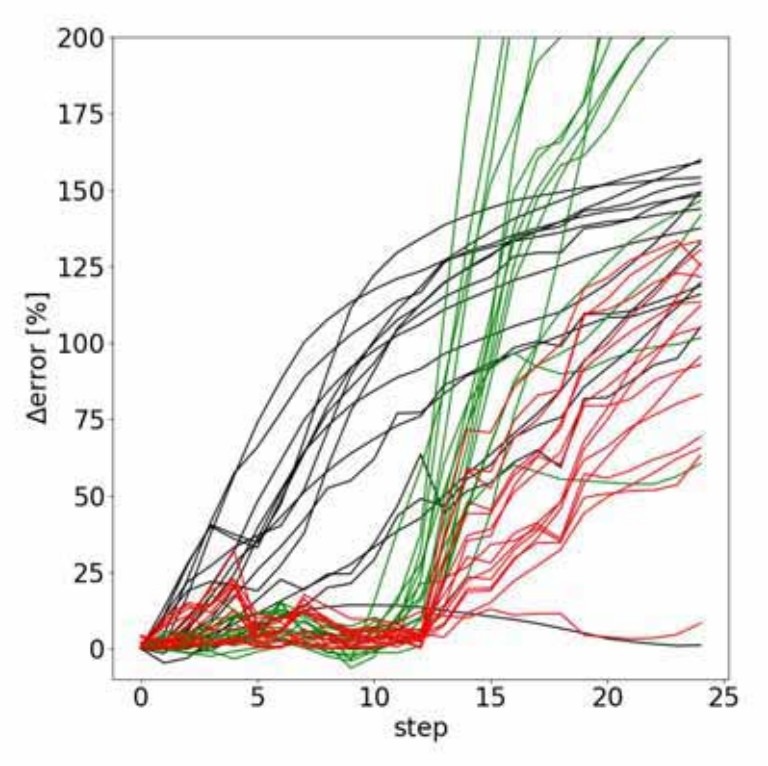
CNN using known beam distribution, solenoid, and charge as inputs



AML with adaptive feedback and unknown beam distribution, solenoid, and charge.



Showing the errors [%] of 15 projections of the 6D phase space as the input beam distribution, solenoid current and bunch charge leave the span of the training set .



Change: % difference of the 15 projections relative to initial input and parameter settings as the beam changes.

CNN: % difference of the 15 projections if the input beam and parameter settings are known. The error remains small within the span of the training set and then the CNN catastrophically fails as the training set is left behind (it is actually worse than doing nothing), as expected.

AML: % error of the 15 projections if the input beam and parameter settings are unknown, but adaptive ML is used for active feedback based on (z,E) measurements, resulting in higher accuracy tracking and no catastrophic failure with this robust approach.