Los Alamos
NATIONAL LABORATORY

Physics Constraint Machine Learning for Relativistic, Charged Particle Beams

Christopher Leon

March 8th, 2024

FLORIDA INTERNATIONAL UNIVERSITY

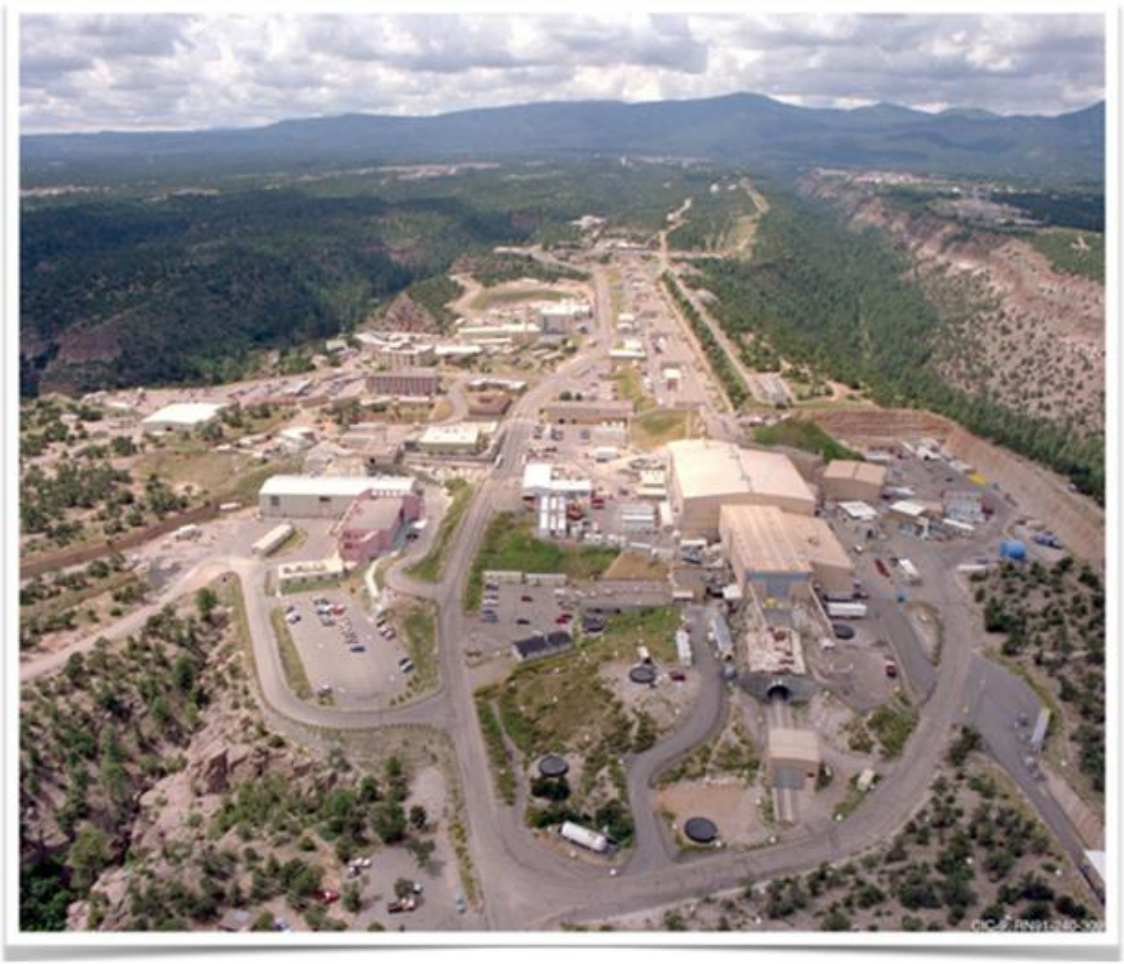
Outline

1. Intro and Background
2. EM Fields Generated by Relativistic Beams and Uncertainty Quantification
3. Coherent Synchrotron Radiation
4. Outlook and Conclusion

Intro and Background

Intro and Background

- LANSCE at LANL
- Running accelerator is expensive
- Beamtime precious
- Maintenance/diagnostics cuts down on experiments



LANSCE
Los Alamos Neutron Science Center

Intro and Background

Given physics of accelerators is well-understood, why ML?

1) **Complexity**

2) **Speed**

- Traditional simulations: slow
- Neural Networks: Slow to train, fast at runtime.

Large Hadron Collider breaks high-energy records

Large Hadron Collider produces first particle collisions and a round of applause from anxious scientists

Happy Cern scientists watch as the $10bn Large Hadron Collider fires up. Photograph: Anja Niedringhaus/AP

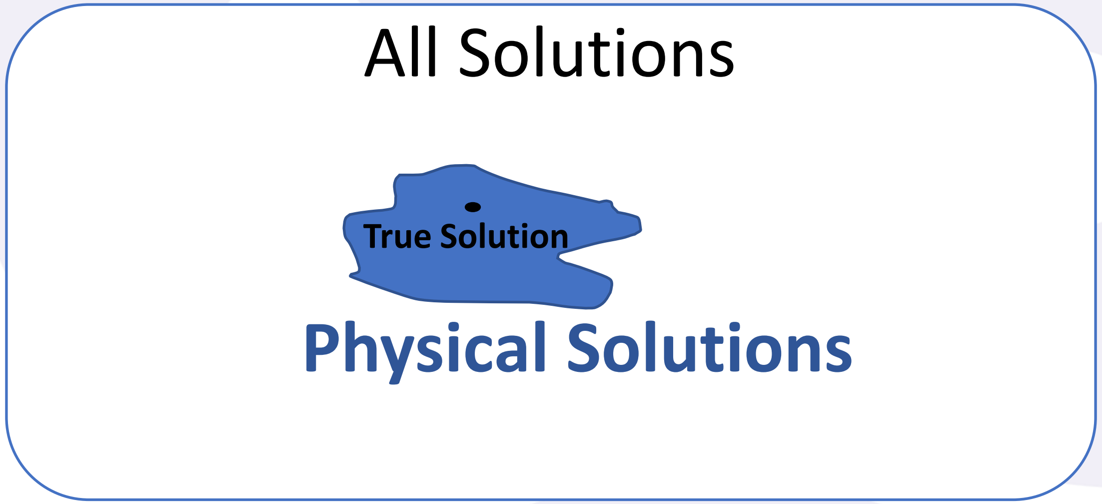
Staff working on the largest, most complex scientific instrument in the world joined in a standing ovation earlier today as the machine began its long search for new particles, forces and extra dimensions of space.

Goal

- Use **ML for fast predictions** for relativistic charged beam

Quantitative physics principles offer unique ways to incorporate into NNs

Created with DALL-E



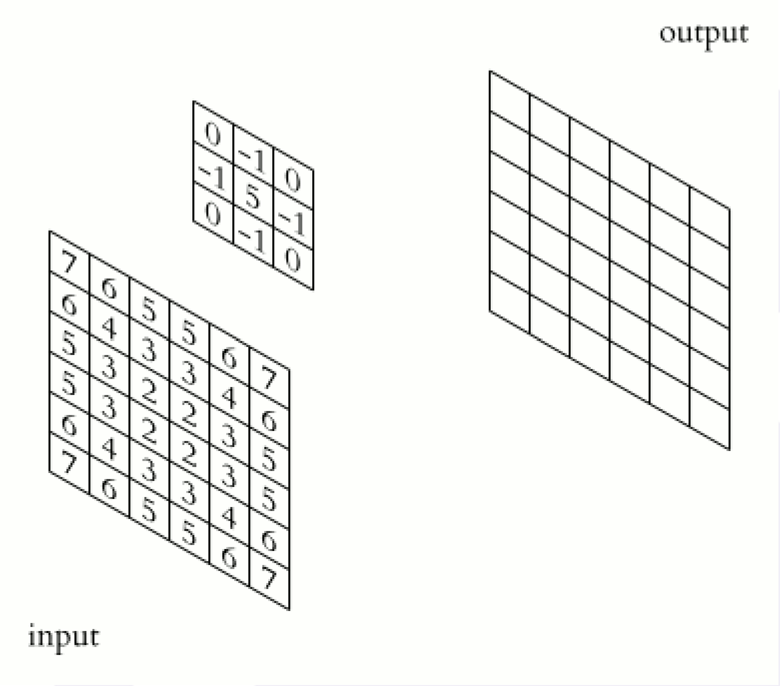
Tool: **Physics Constraint Convolutional Neural Networks**

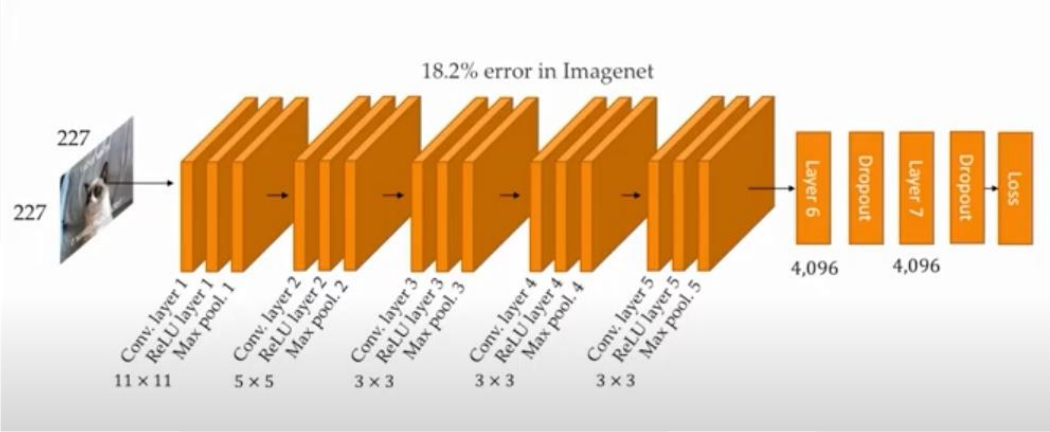
Convolutional Neural Network

- **Filters** run convolutions on input. NN itself finds useful filters

- Filter output produce **feature maps.** (very parallelizable)

- Output fed to non-linear functions. Process recursive.

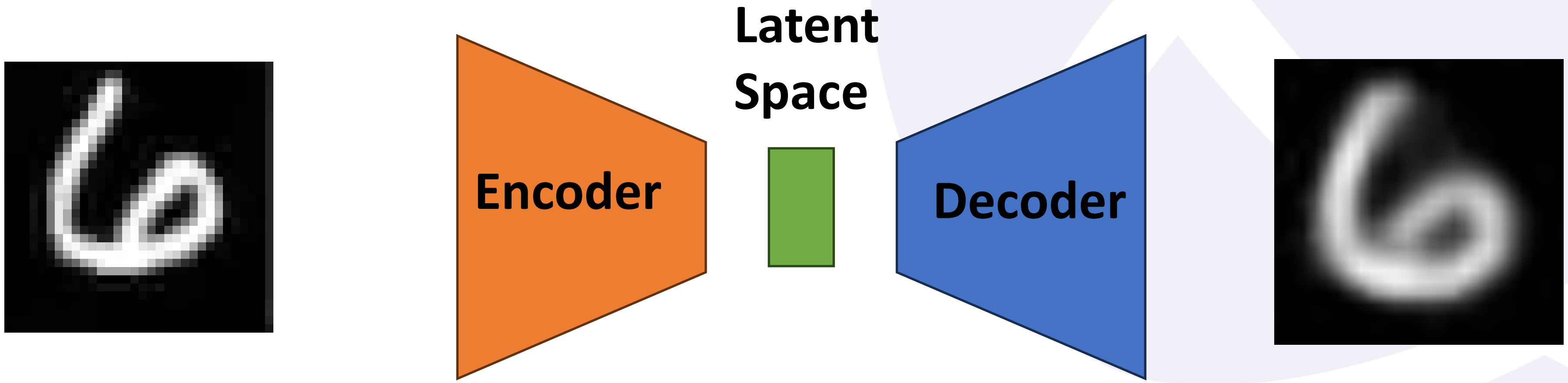




CNN Encoder-Decoder

- Example: MNIST database,
- $28 \times 28 = 784$ pixels
- Images in 784-dimensional space.

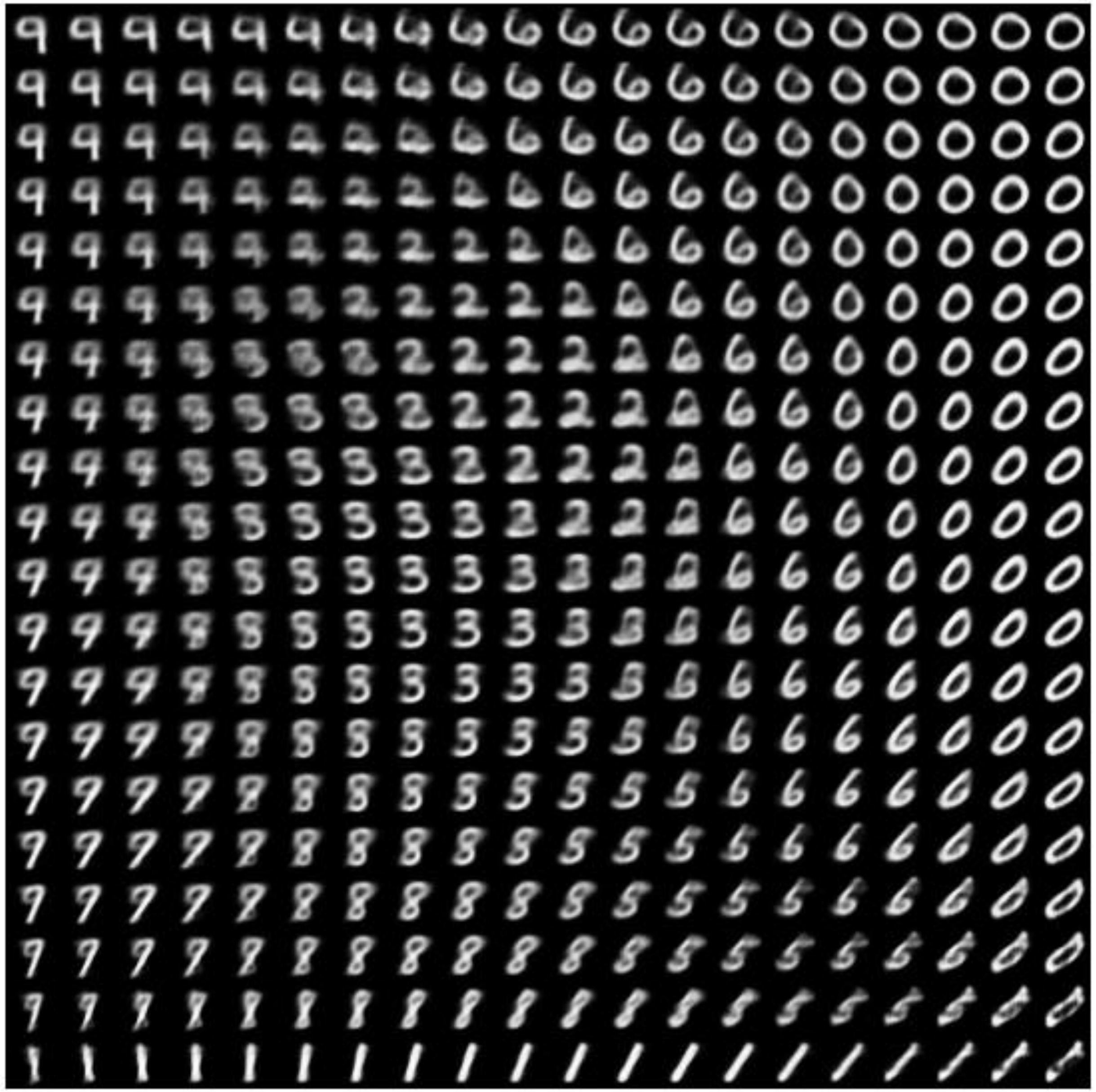
- Latent space: bottleneck



Latent Space

- Dimension reduction:
$$784 \rightarrow 2$$

- Extract most relevant features



TensorFlow: Variational Autoencoder Tutorial

Accelerator Example

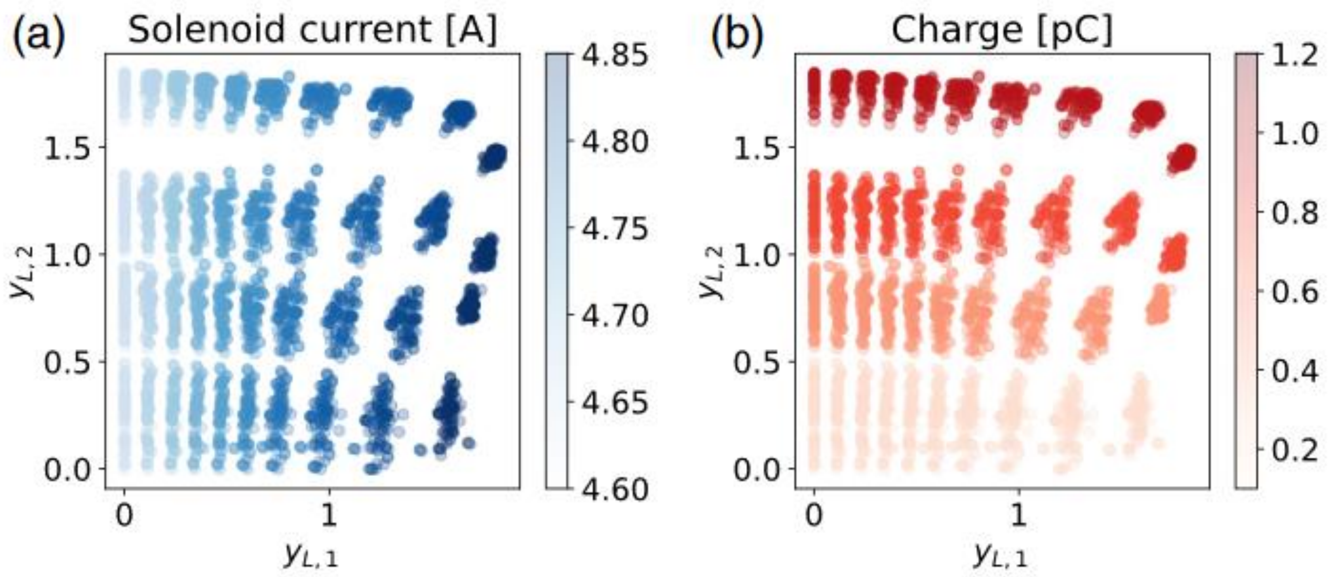
- *Input*: settings, initial projections of 6D phase space $(\boldsymbol{r}, \boldsymbol{p})$ distribution
- *Output*: phase space later

- 2706 dimensions
- Dimensions reduction:

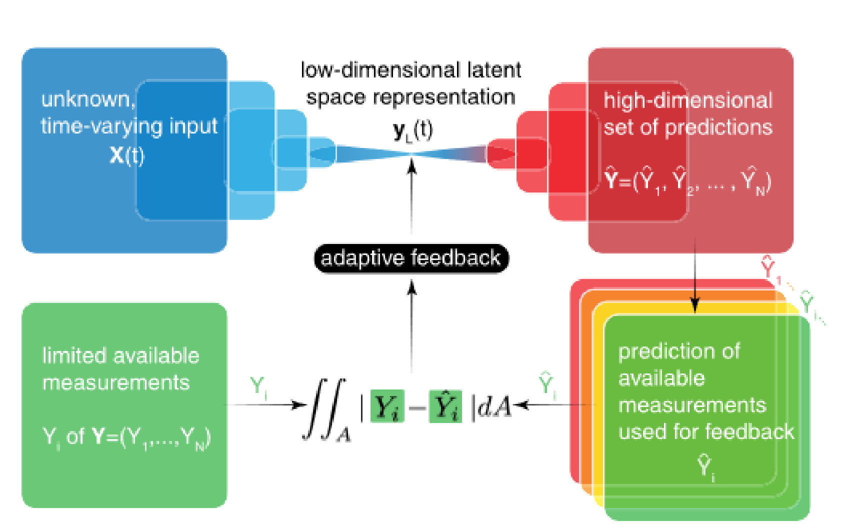
$$2706 \Rightarrow \quad 2$$

- Easier to work adjust parameters in latent space

Latent Space

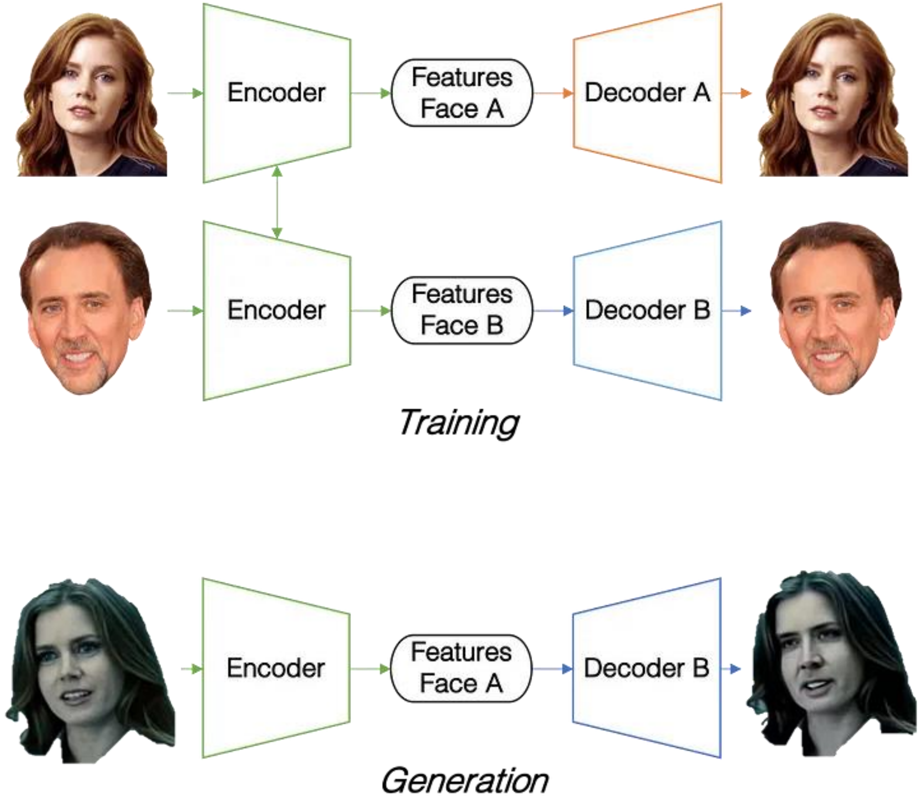


A Schienker et al. [Physical Review E](#) (2023)



CNN Encoder-Decoder

- CNN: Image → Image



[Overview of CNN-Based Deepfake Detection Methods](#)
Medium – Colin Tan

CNN's

- CNN: Image → Image

$$a_{ij} \to b_{ij}$$

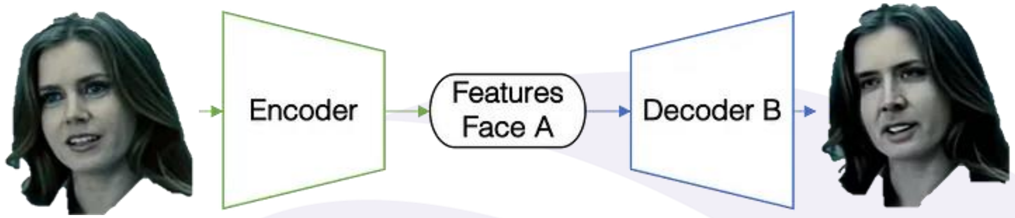
- Operator: Function → Function

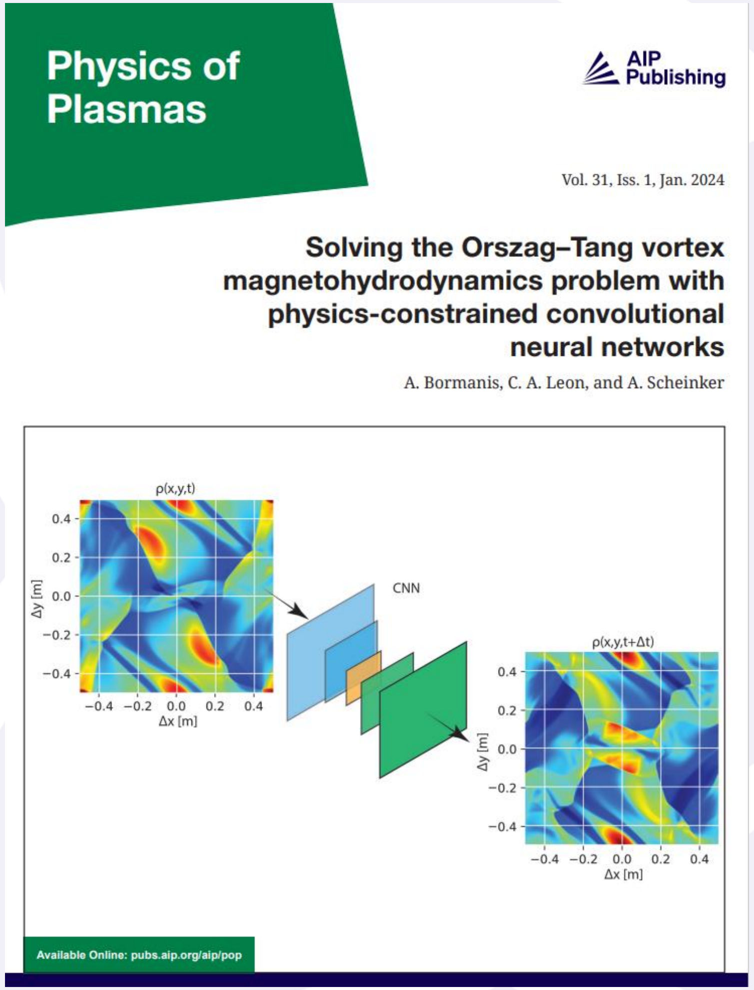
E.g.: $f(x, y) \to g(x, y)$

$$\partial_x\colon\ f(x, y) \to \partial_x f(x, y)$$

Evolution operator.

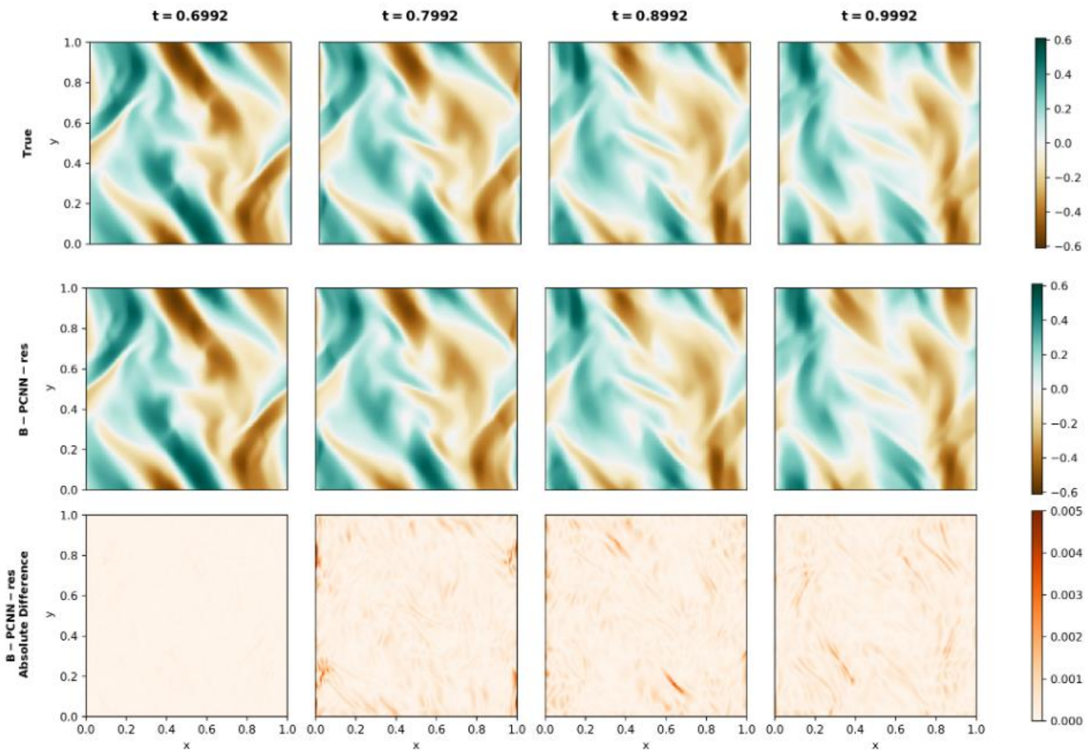
$$\rho_m(x, y, t) \to \rho_m(x, y, t + \Delta t)$$

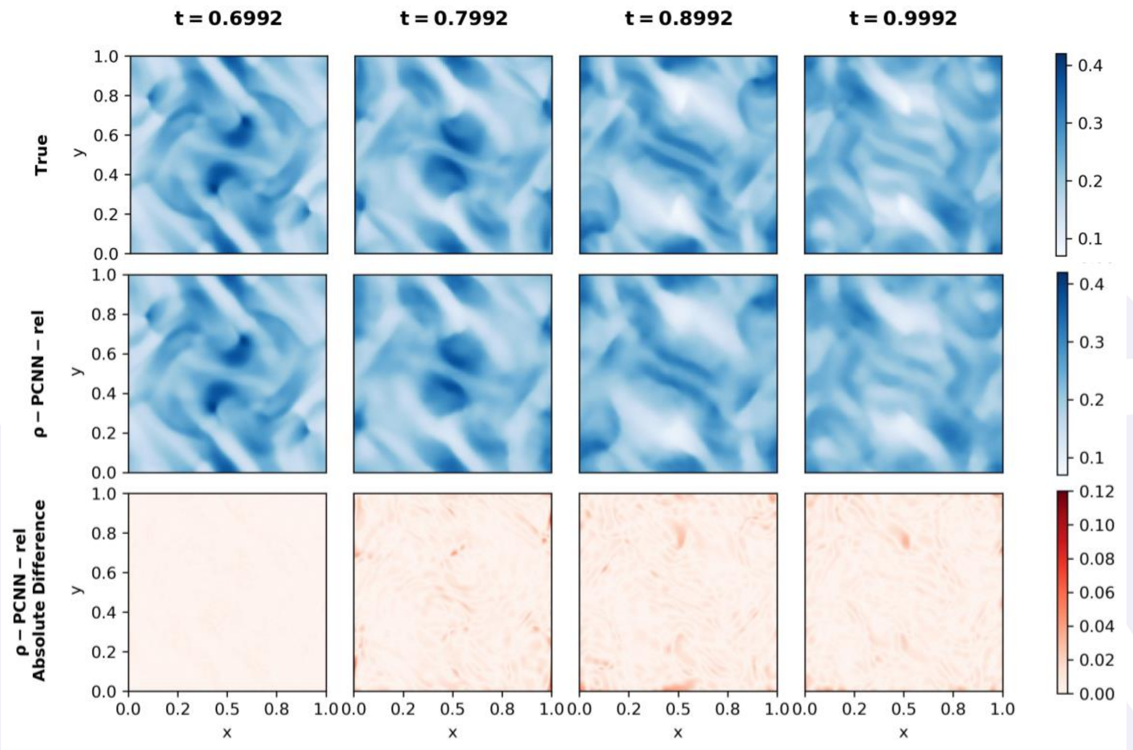




Magnetohydrodynamics (MHD)

- Forecast $\rho_m(x, y, t + \Delta t)$
- $\Delta t = 8 \times 10^{-4}$
- Test after $t = 0.75$





Bormanis, A., CL, and A. Scheinker *Physics of Plasmas* 31.1 (2024).

- Forecasting $B_y(x, y, t + \Delta t)$

EM Fields Generated by Relativistic Beams and UQ

How to Train NN's

- Have a loss function, L. E.g.,:

$$MSE = \sum_i \left(y_{data,i} - f_{\boldsymbol{\theta}}(x_i)\right)^2, \qquad \chi^2 = \sum_i \frac{\left(y_{data,i} - f_{\boldsymbol{\theta}}(x_i)\right)^2}{\sigma_i^2}$$

- Fit parameters, $\boldsymbol{\theta}$, by minimizing L on training data
- Test set withheld; see how model generalizes

Adding Physics Constraints

RESEARCH ARTICLE | APRIL 14 2023

Physics-constrained 3D convolutional neural networks for electrodynamics

Alexander Scheinker ; Reeju Pokharel

Check for updates

APL Mach. Learn. 1, 026109 (2023)
https://doi.org/10.1063/5.0132433

1. NN: $\boldsymbol{J} \rightarrow \boldsymbol{B}$

$$L = \int d^3\boldsymbol{r}\,\left|\boldsymbol{B}_{true} - \boldsymbol{B}_{pred}\right|^2$$

2. PINN: $\boldsymbol{J} \rightarrow \boldsymbol{B}$

$$L = w_B \int d^3\boldsymbol{r}\,\underbrace{\left|\boldsymbol{B}_{true} - \boldsymbol{B}_{pred}\right|^2}_{\text{Fit Data}} + w_\Delta \int d^3\boldsymbol{r}\,\underbrace{\left|\boldsymbol{\nabla}\cdot\boldsymbol{B}_{pred}\right|^2}_{\text{Be Divergenceless}}$$

Soft Constraint

3. PCNN: $\boldsymbol{J} \rightarrow \boldsymbol{A} \Rightarrow \boldsymbol{\nabla}\times\boldsymbol{A} = \boldsymbol{B}$

$$L = \int d^3\boldsymbol{r}\,\left|\boldsymbol{B}_{true} - \boldsymbol{\nabla}\times\boldsymbol{A}_{pred}\right|^2$$

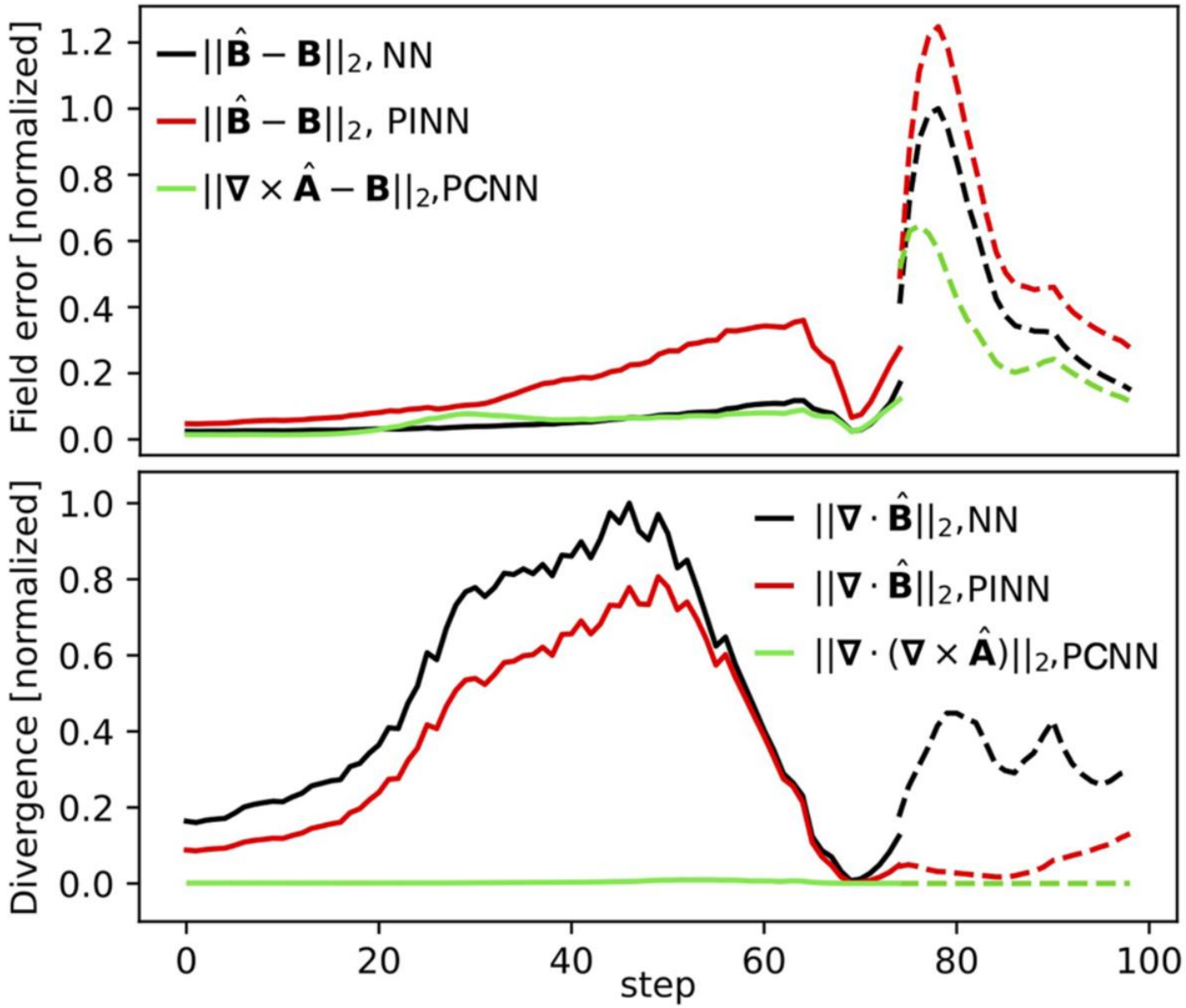
Hard Constraint

$$\nabla\cdot(\nabla\times\boldsymbol{A}) = 0$$

Predicting Magnetic Field

For simulation of e^- beam:

- Train: 0-74 time step
- Test: 75-99 time step



- PINN better than NN on $\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$, but higher error

- Hard physics constraints led to better generalization

Other Ways to Add Physics Constraints

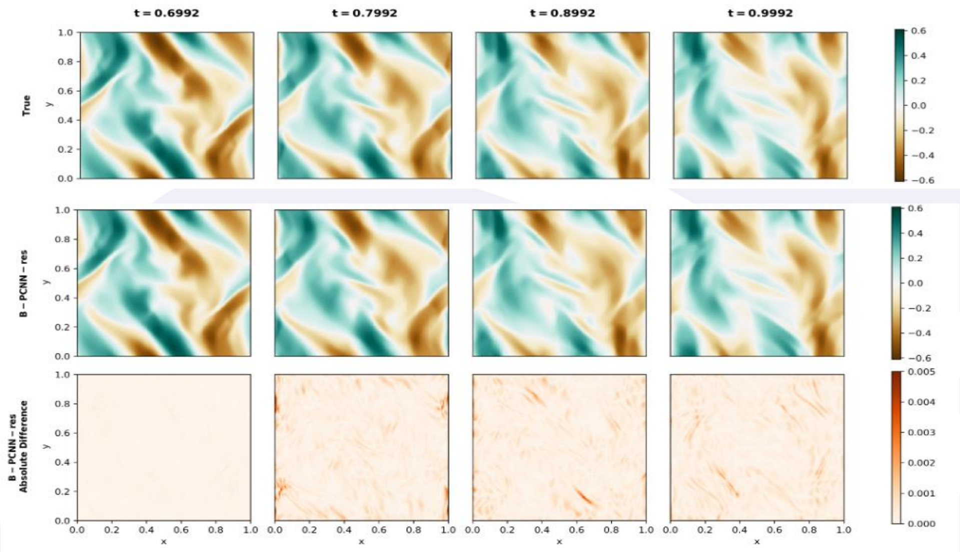
- Different ways to incorporate physics

- Research on architectures with other symmetries (e.g., rotation, Galilean)

Constraint	Implementation	Hard/Soft
Divergence-free	$\mathbf{B}(\mathbf{r}, t) = \nabla_\perp A(\mathbf{r}, t)$	Hard
Translation Equivariance	CNN Architecture	Hard
Non-negativity (e.g., $\rho \geq 0$)	Final Layer ReLU	Hard
Periodic BC's	Padding	Soft
Partial Differential Equation	Term in Cost Function	Soft

Bormanis, A., CL, and A. Scheinker
Physics of Plasmas 31.1 (2024).

MHD: Improvements via ML

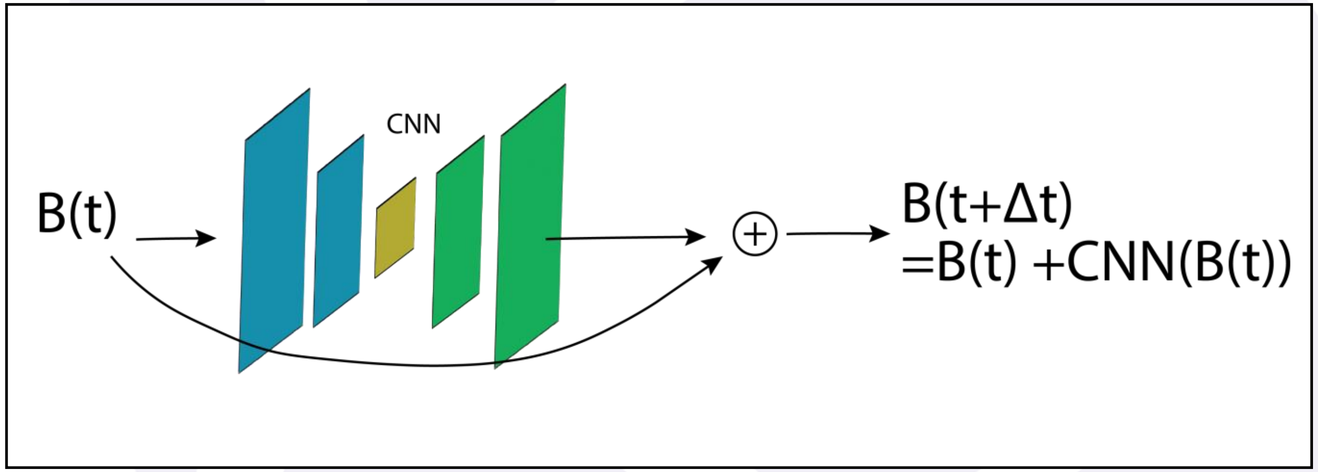


- **Data Augmentation**

Trained on 50 simulations

- **Res-Net Architecture**

$$\boldsymbol{B}(t+\Delta t) = \boldsymbol{B}(t) + CNN\big(\boldsymbol{B}(t)\big)$$



Decreased MSE by $\sim 10^4$

- Best results combined physics constraints and ML techniques

Goal

- Can we **increase the number of hard constraints** in EM?

- Specifically: $\rho \;\&\; \boldsymbol{J} \;\; \Rightarrow \;\; \boldsymbol{E} \;\&\; \boldsymbol{B}$

Approach: Potentials in Fourier Space

Potentials

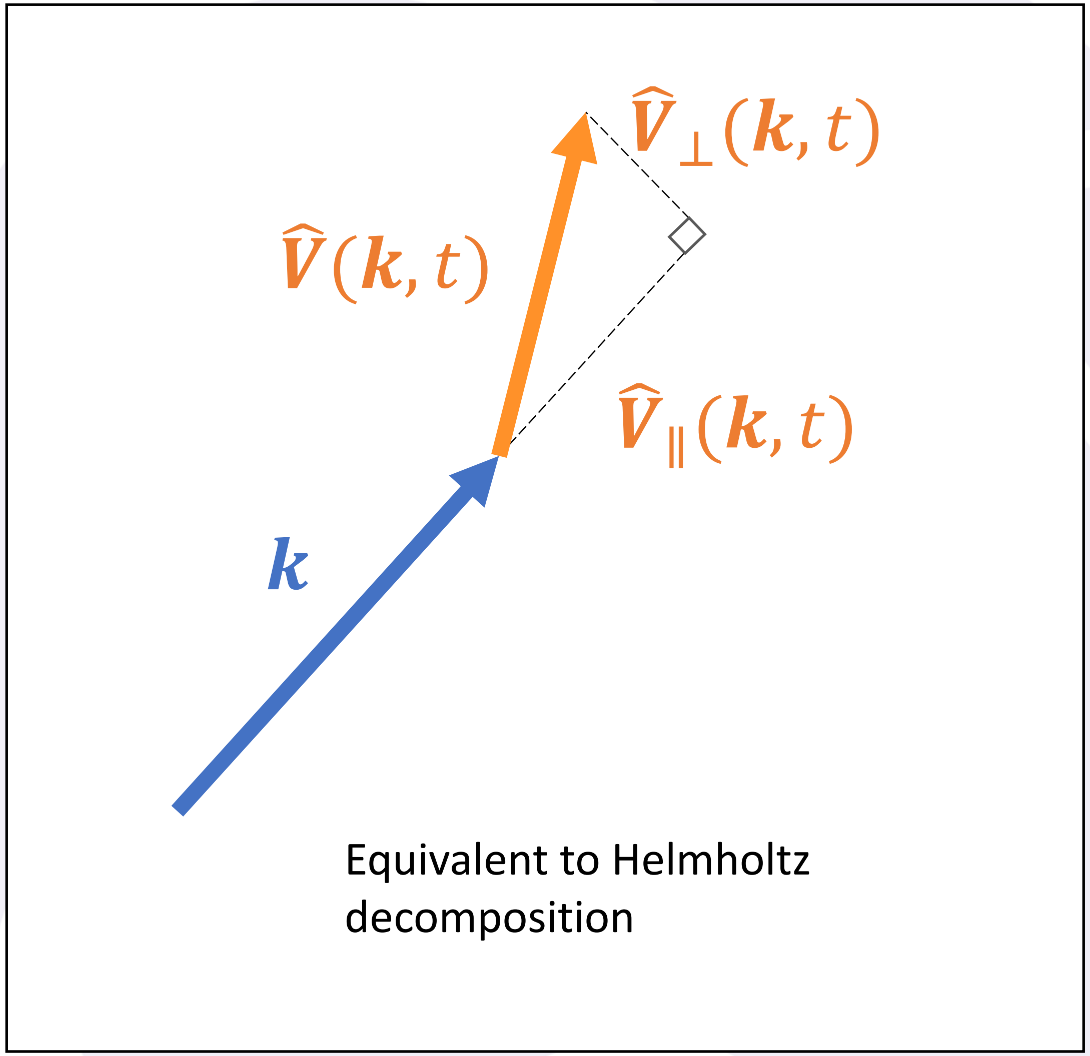
$$\boldsymbol{E} = -\boldsymbol{\nabla}\phi - \frac{\partial \boldsymbol{A}}{\partial t}$$

$$\boldsymbol{B} = \boldsymbol{\nabla} \times \boldsymbol{A}$$

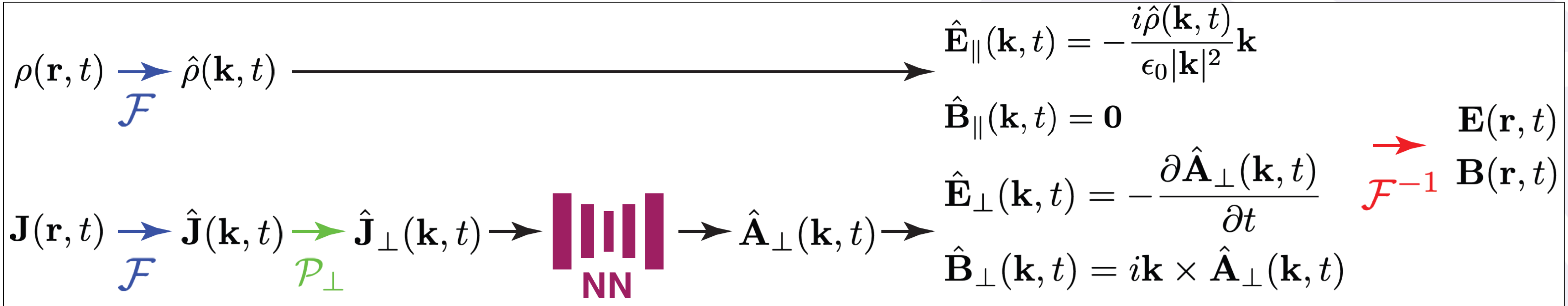
- 6 vs 4 fields

Fourier space

- Trade $\boldsymbol{\nabla}$ for $\mathrm{i}\boldsymbol{k}$
- FFT: $\mathcal{O}(N \log N) \Rightarrow$ fast with GPUs



FoHM-NO Framework



Built in hard physics constraints:

- Gauss's Law for **E**
- Faraday's Law
- Gauss's Law for **B**
- Longitudinal Ampere-Maxwell's Law

- Neural Network ⇒ **solution operator**: $\widehat{A}_\perp = \boldsymbol{O}\left[\widehat{J}_\perp\right]$

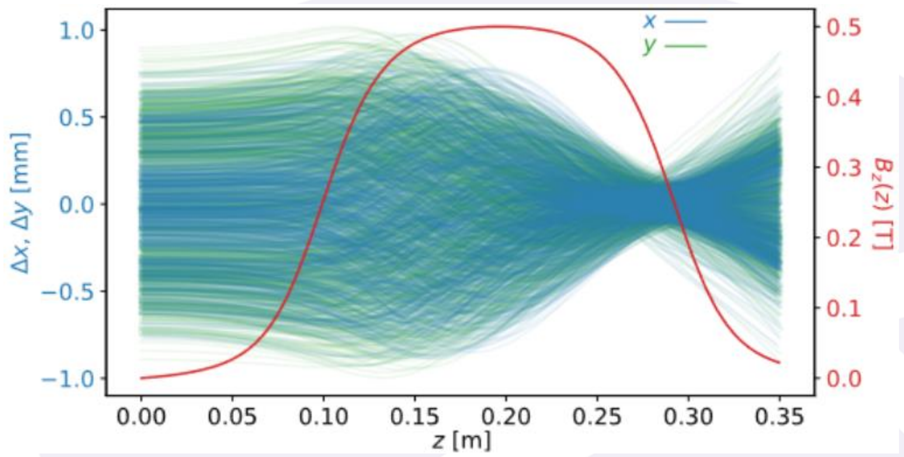
$$\frac{1}{c^2}\frac{\partial^2 \widehat{\boldsymbol{A}}_\perp}{\partial t^2} + |\boldsymbol{k}|^2 \widehat{\boldsymbol{A}}_\perp = \widehat{\boldsymbol{J}}_\perp$$

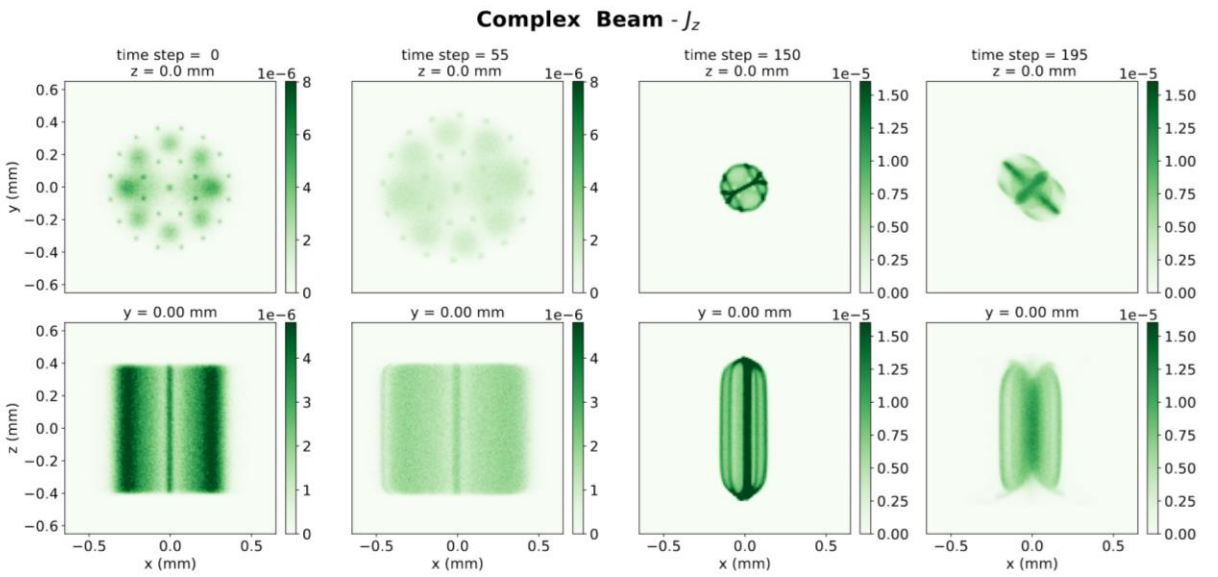
Data – Conventional Simulation

- Used General Particle Tracker (GPT)
- Simulation of electrons entering solenoid **B** field

Two Beams:

- **Complex Beam** (Training 85% + Validation 15%)
- **Gaussian Beam** (Test)





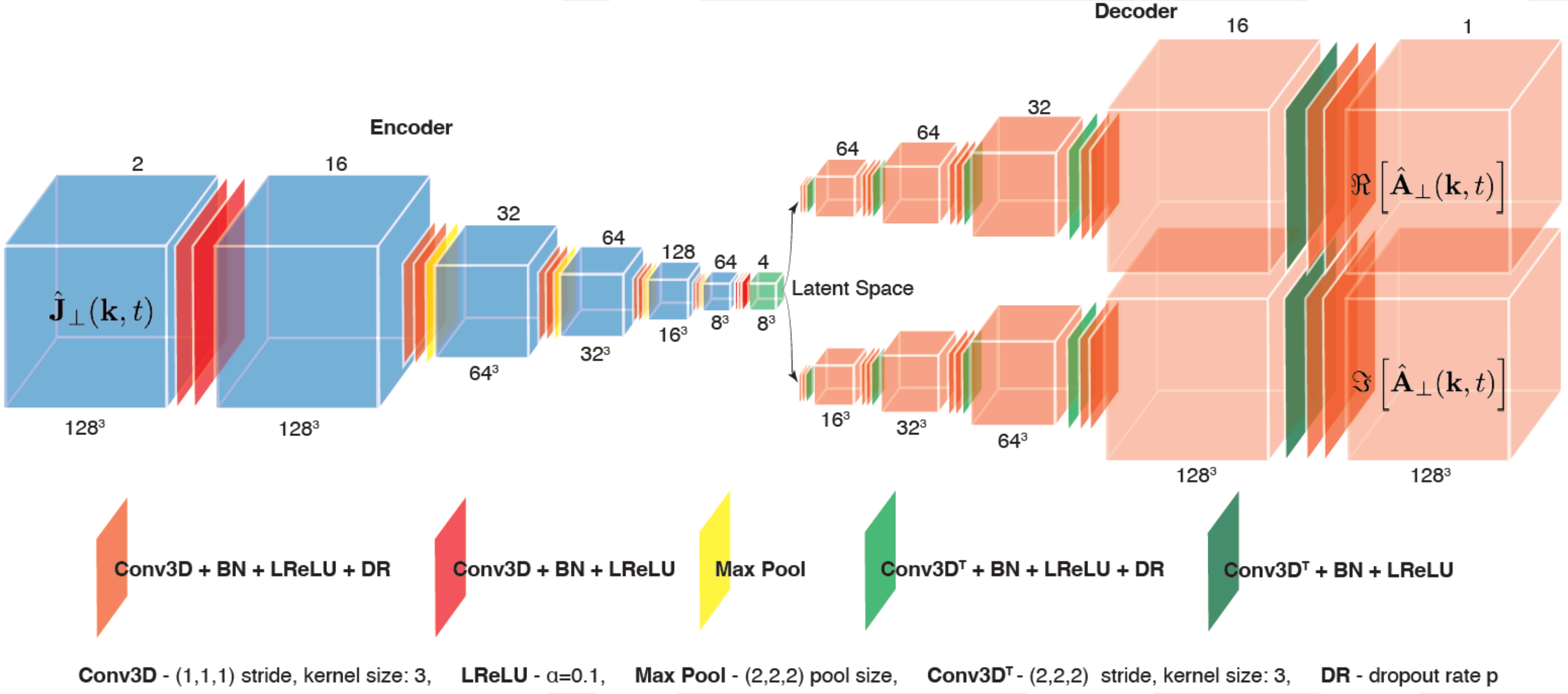
CL, Scheinker, A
Preprint: Research Square
https://doi.org/10.21203/rs.3.rs-3879834/v1

Model Architecture

Tried 3 variations of CNN encoder-decoder:

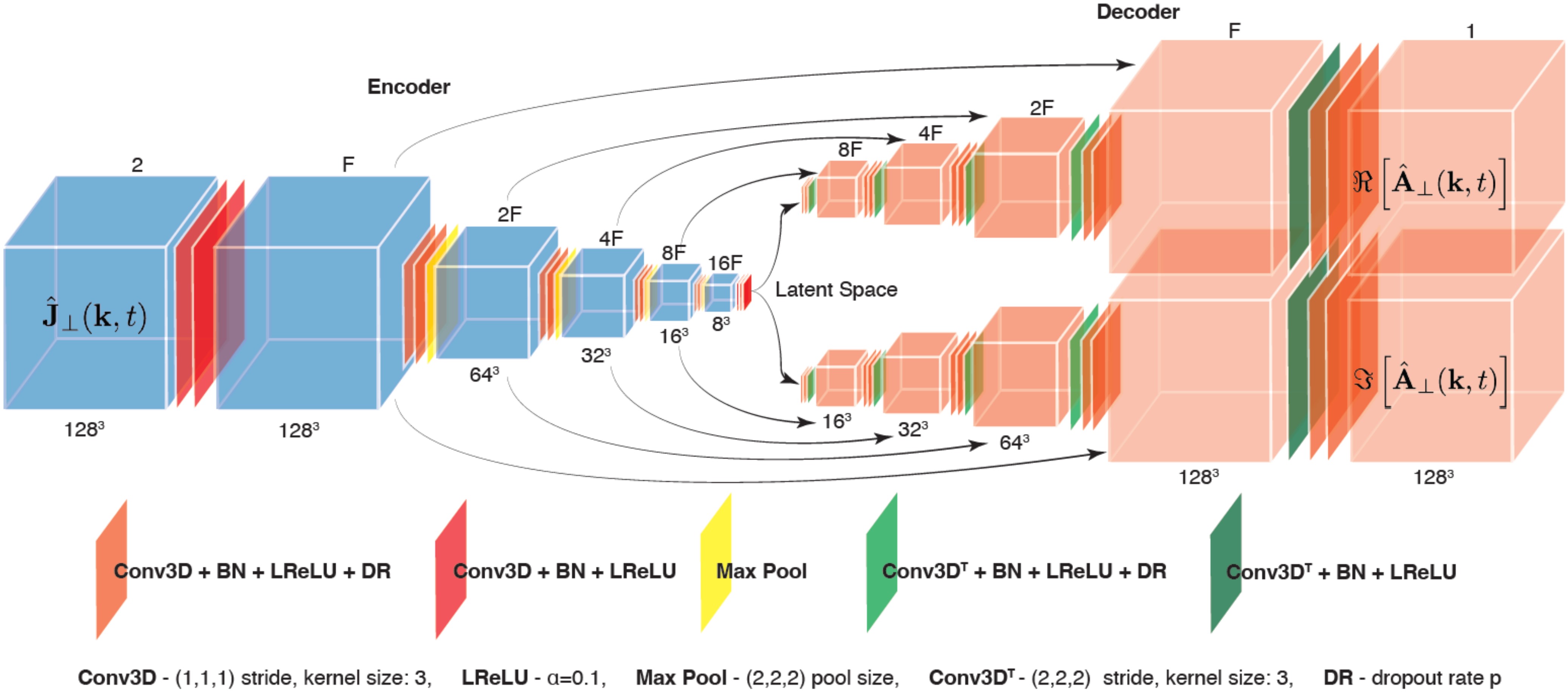
- CNN-small-4 (shown)
- CNN-small-6
- CNN-large

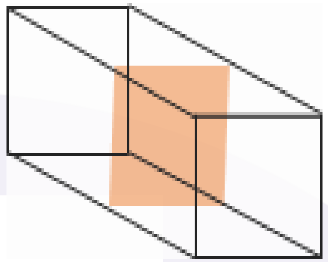
- All failed on finer details



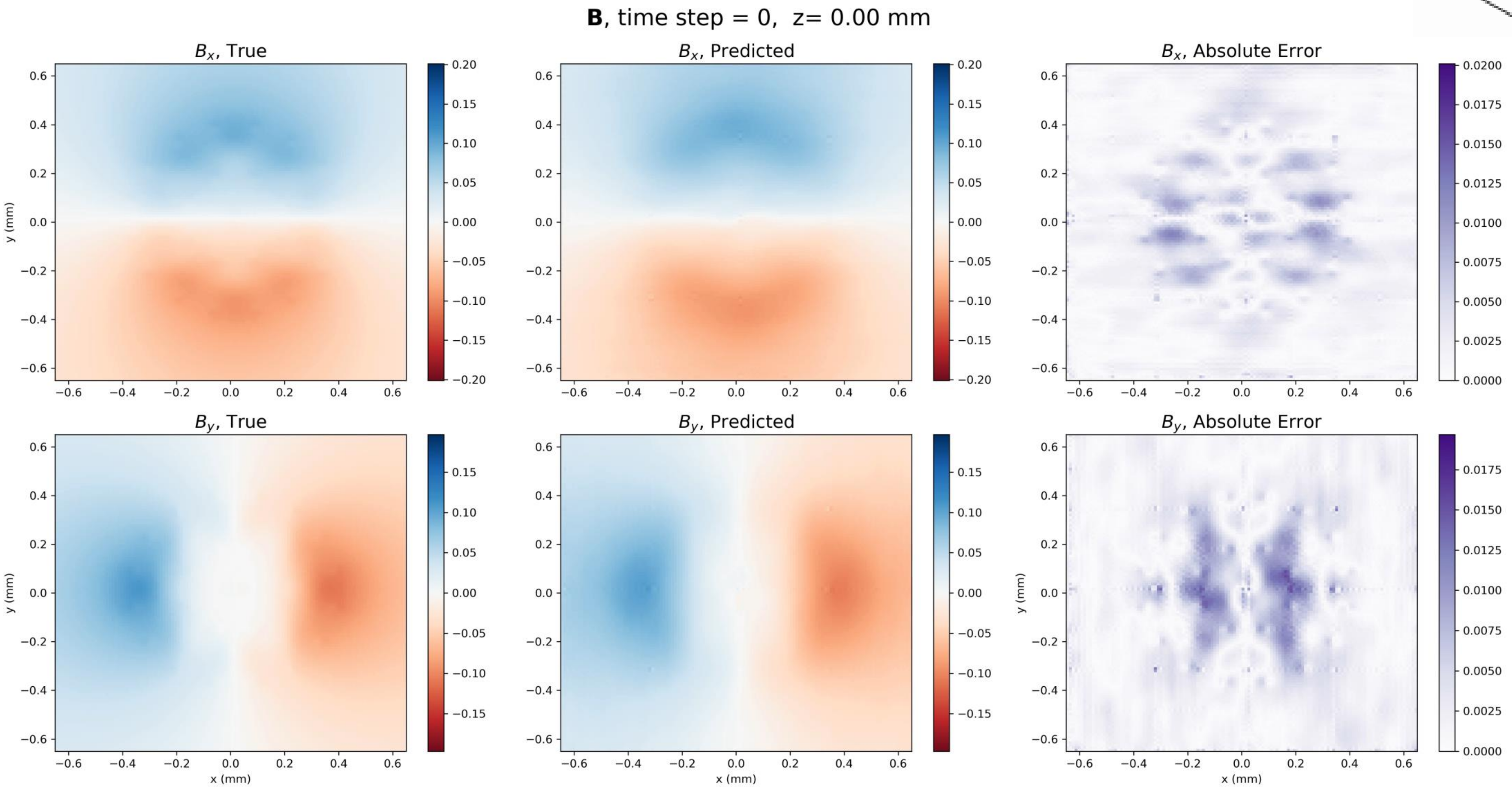
U-Net

- Add skipped connections⇒Concatenation, $F = 16$

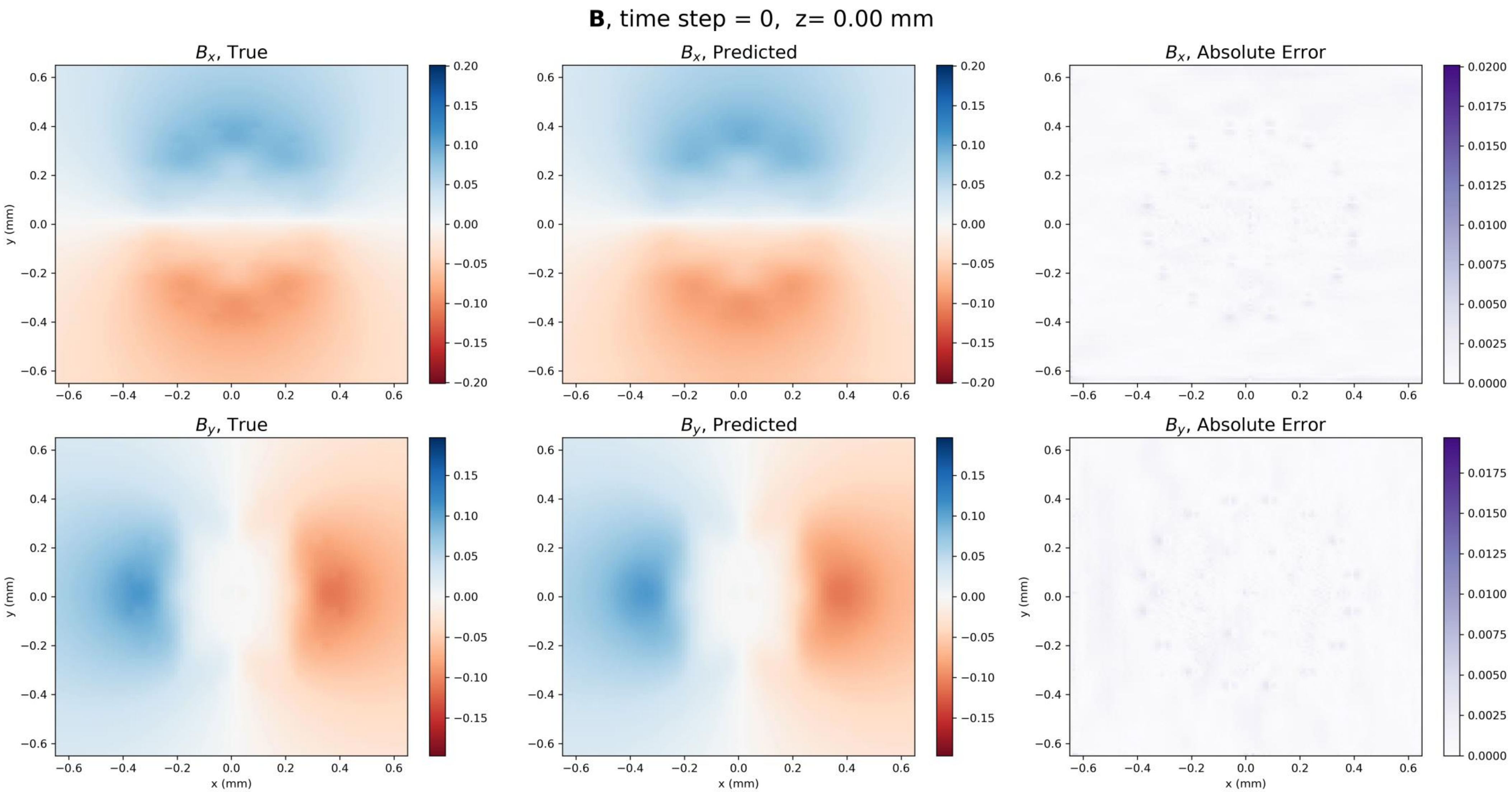




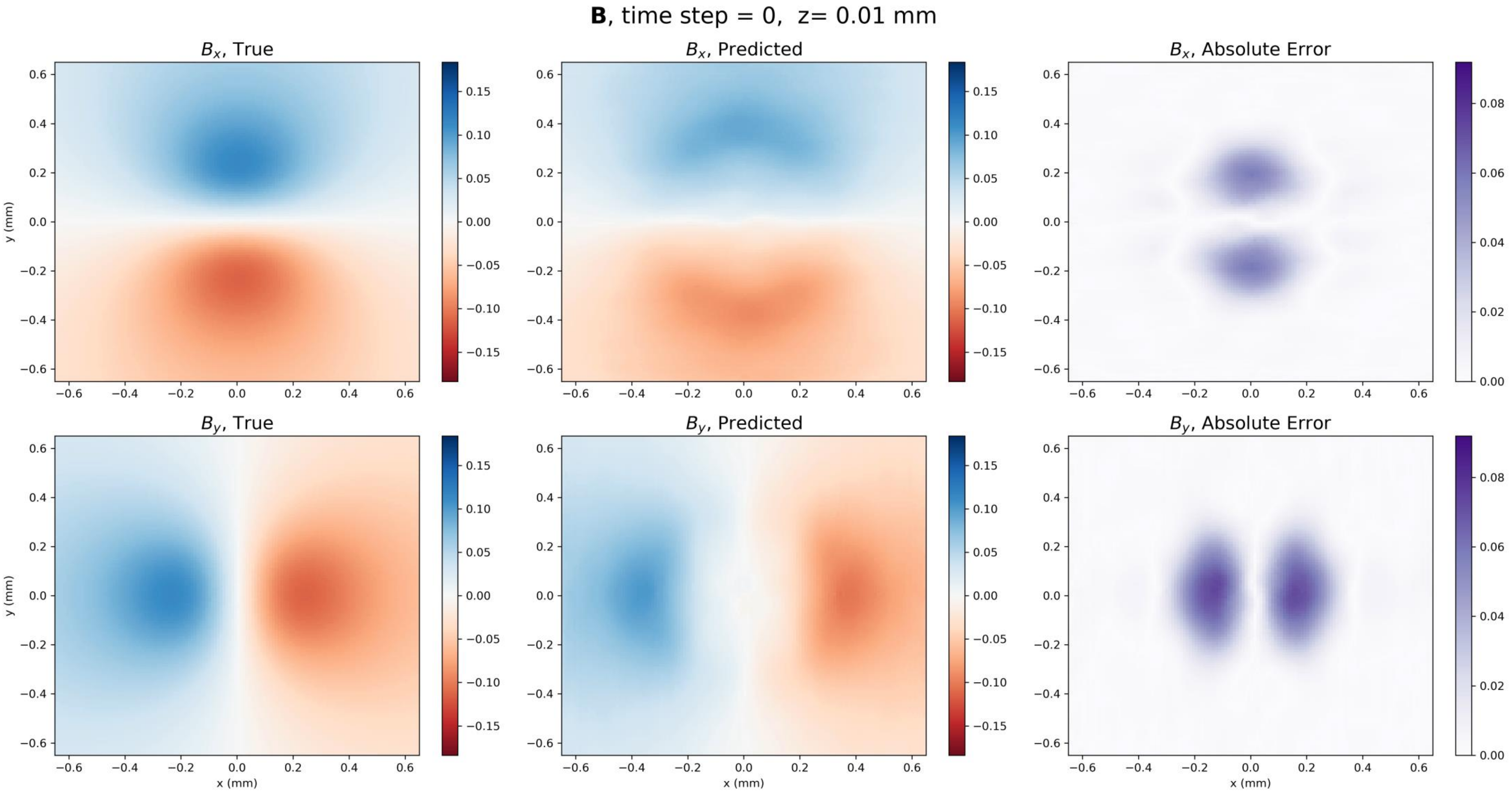
CNN-small-4: Complex Beam



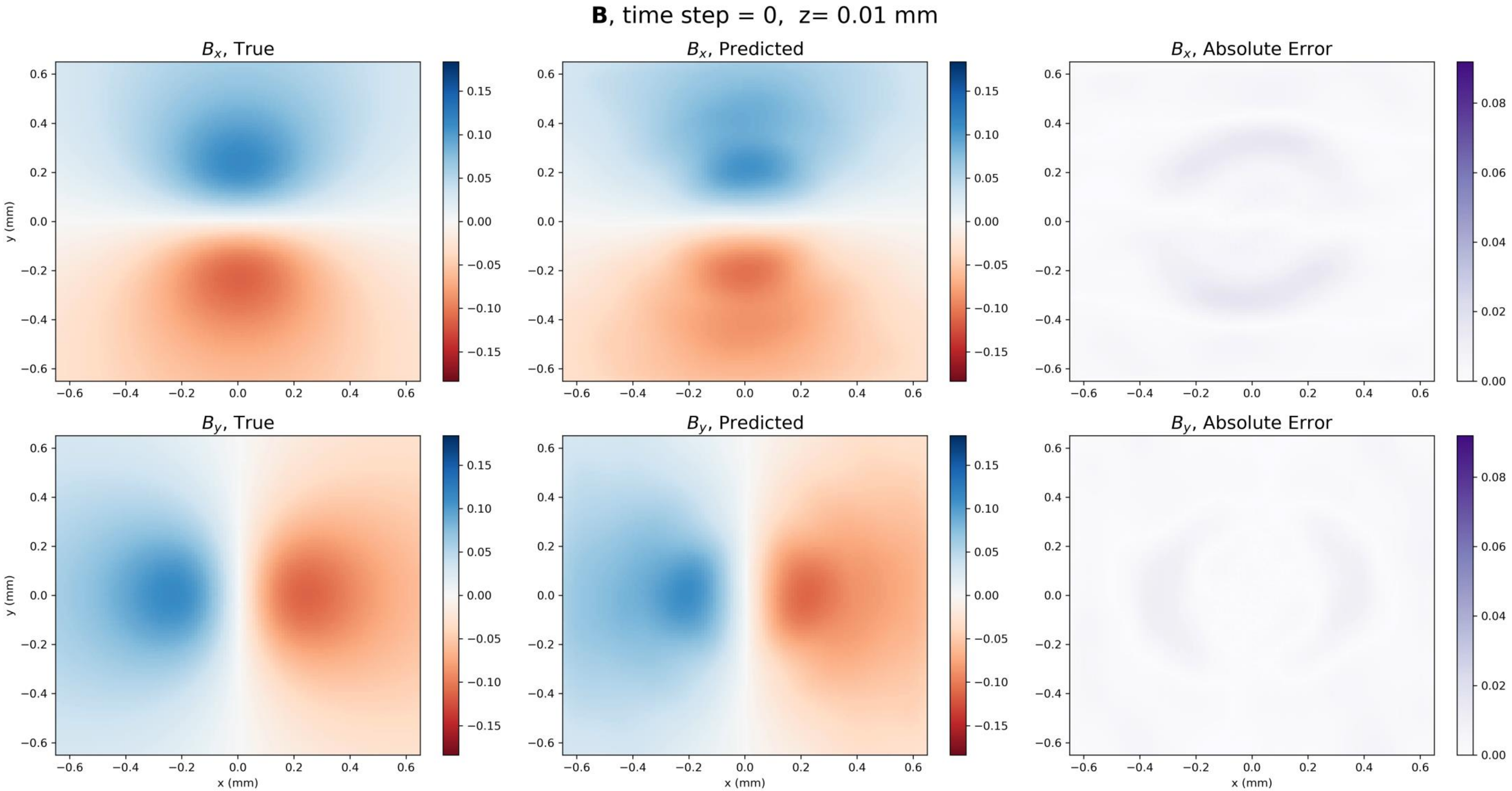
U-Net: Complex Beam



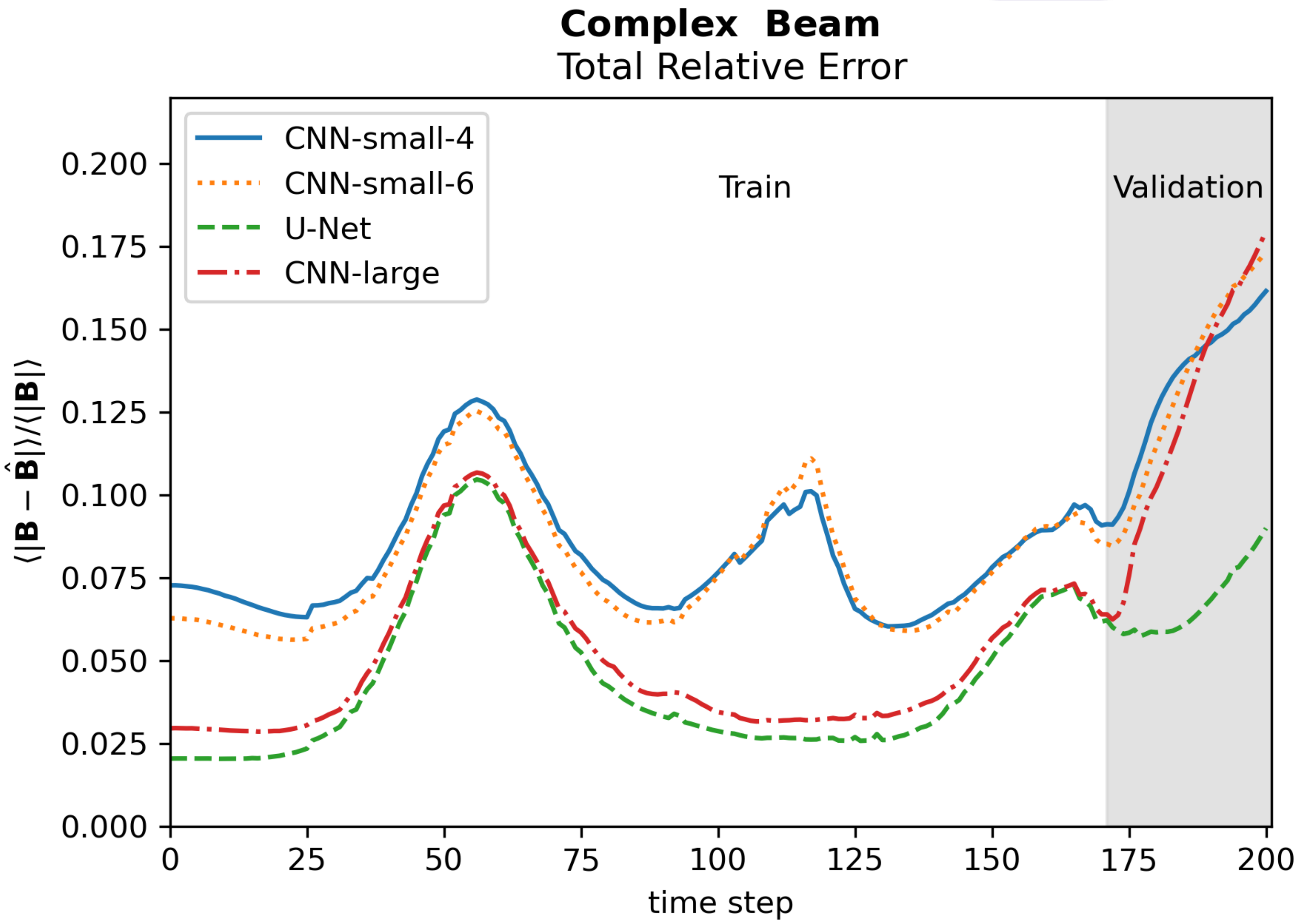
CNN-small-4: Gaussian Beam



U-Net: Gaussian Beam



Results – Four Bunch Beam

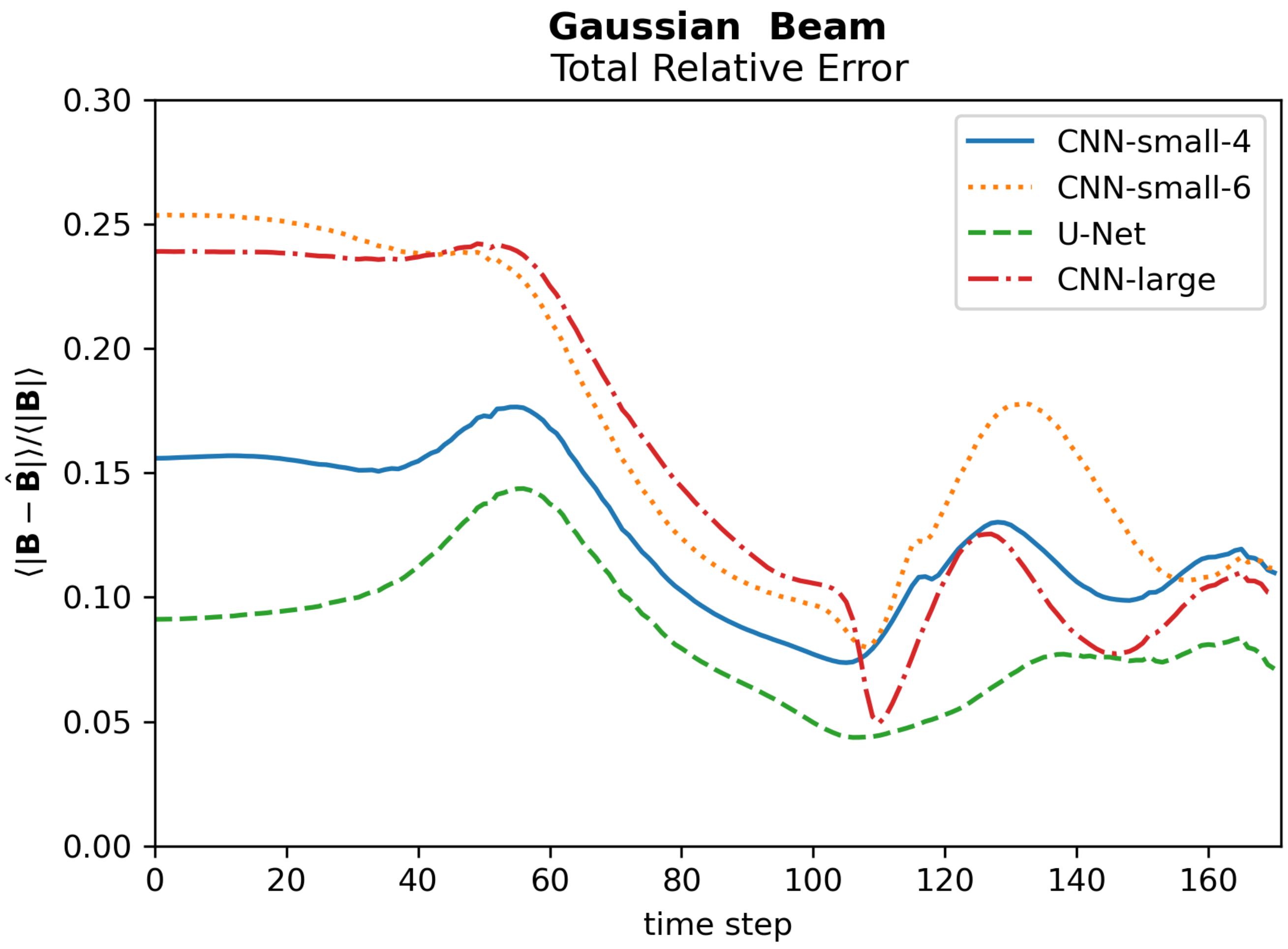


Results – Gaussian Beam

- U-Net does better on all datasets

Model	Training Error (%)	Validation Error (%)	Test Error (%)
CNN-small-4	8.2	13.3	12.6
CNN-small-6	7.9	13.5	17.0
U-Net	**4.5**	**6.7**	**8.6**
CNN-large	5.0	12.4	15.9

- Gaussian Beam: GPT simulation: ~ **1 day**
- NN: ~ **seconds**



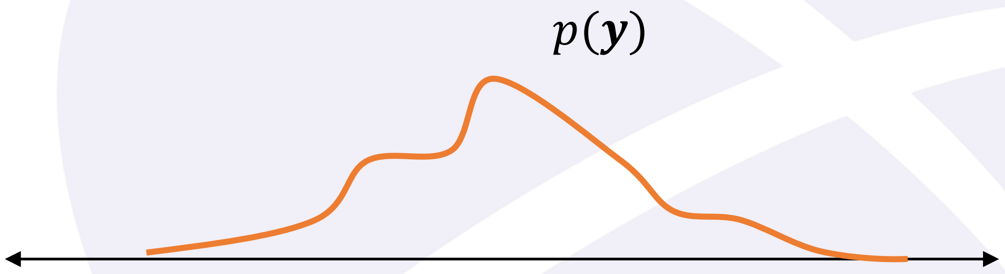
CL, Scheinker, A

Preprint: Research Square

https://doi.org/10.21203/rs.3.rs-3879834/v1

Uncertainty Quantification

- NNs: lack interpretability
- UQ becomes even more important
- Simulation: model uncertainties
- Due to large number of parameters, many methods (e.g, MCMC) not practical



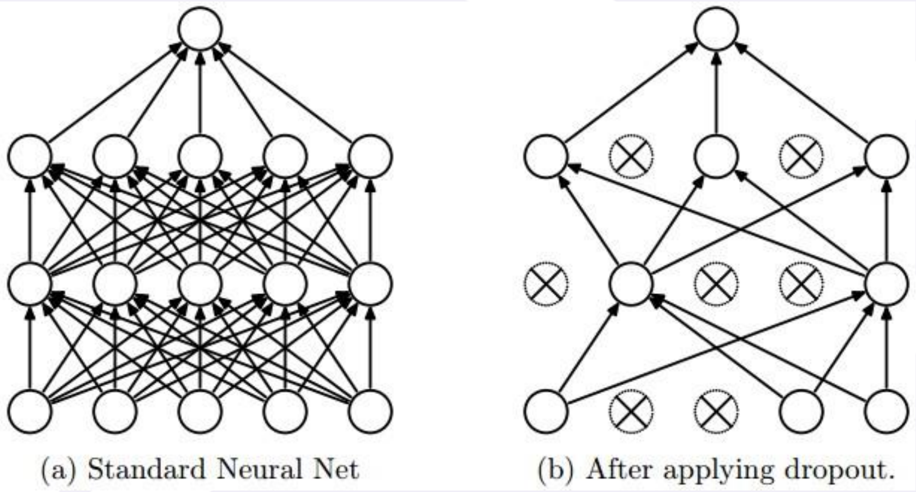
$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{x})p(\boldsymbol{\theta})d\boldsymbol{\theta}$$

Dropout

- **Regular Dropout**: during training randomly “turn off” neurons

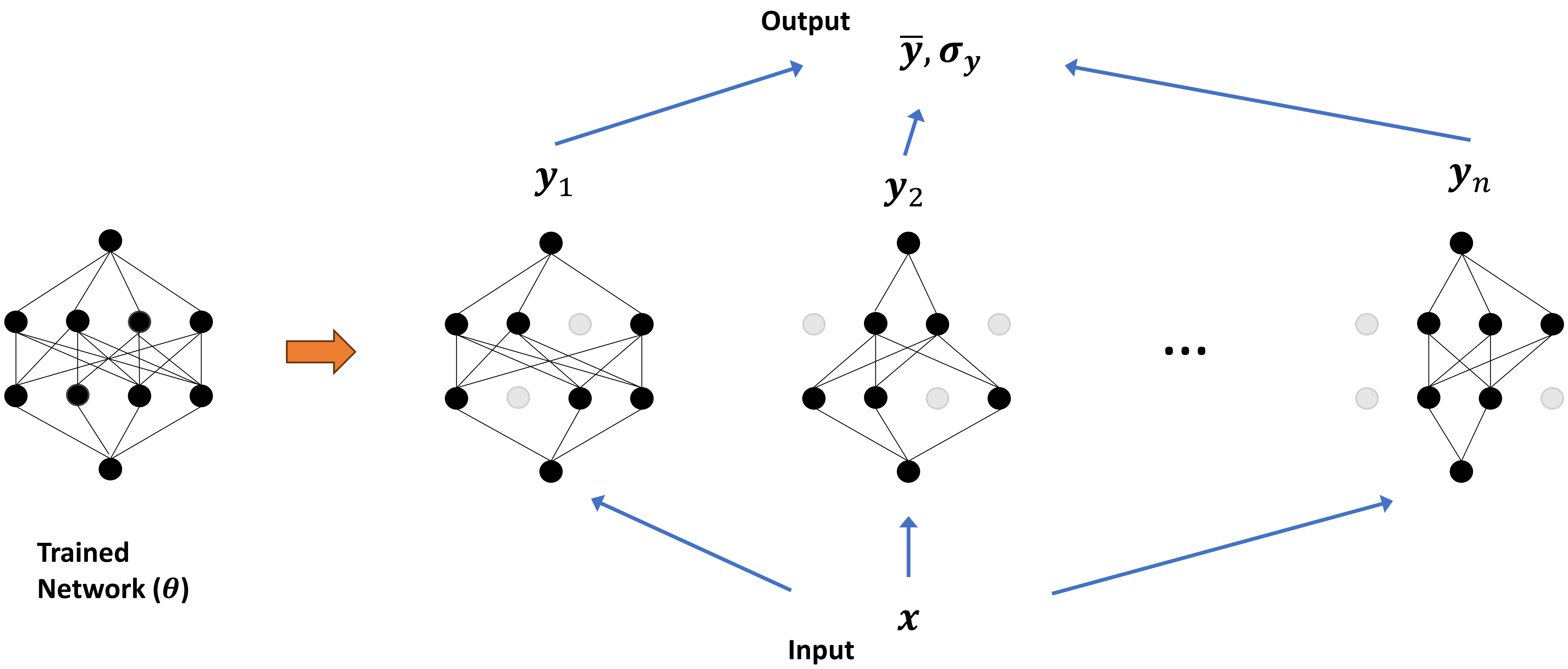
- **Monte Carlo Dropout**: dropout at test time for UQ

- Theoretical interpretation: Approximate Bayesian Inference of Deep Gaussian Processes (Gal and Zoubin, 2016)



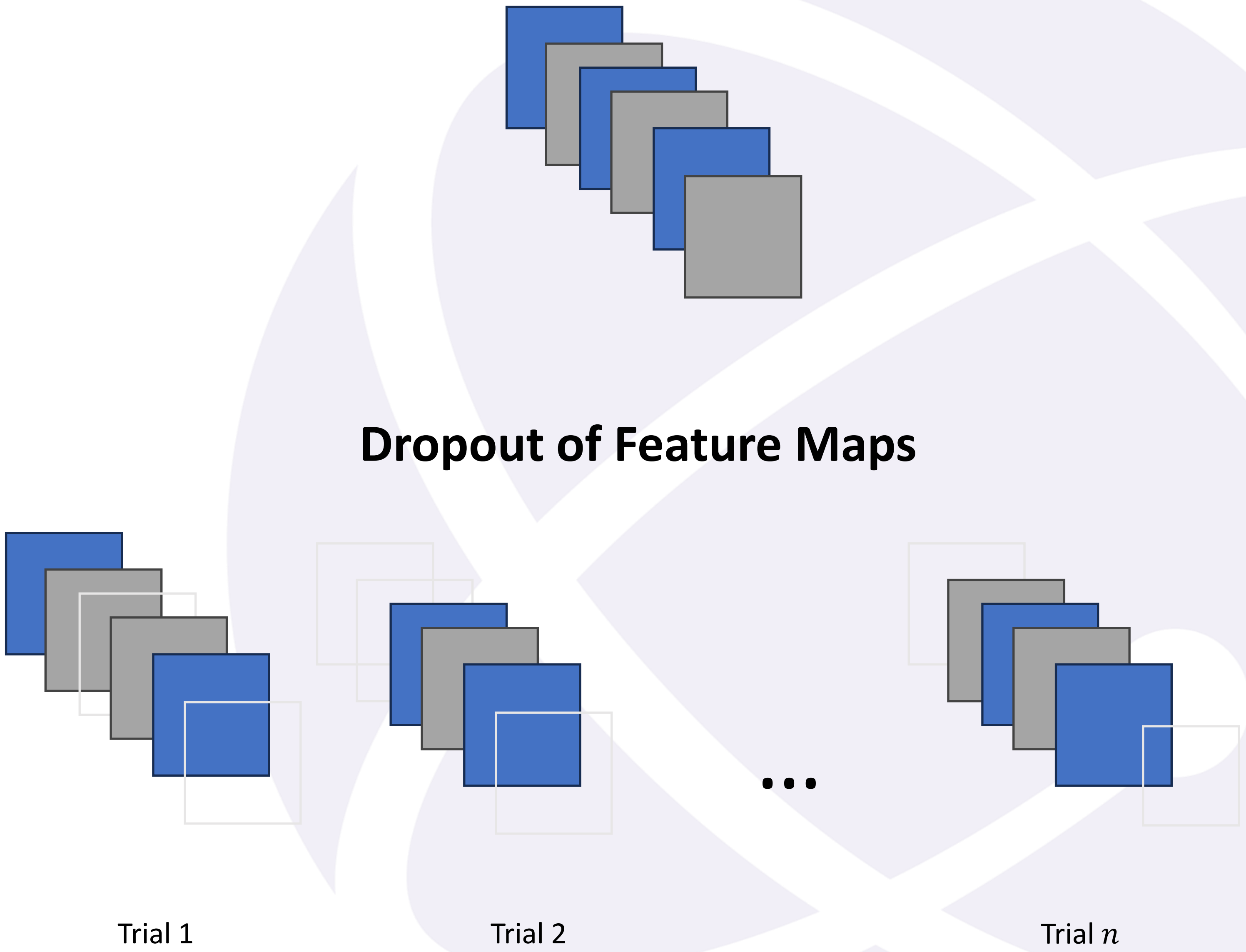
Srivastava, Hinton, et al. (2014)

Monte Carlo Dropout



Spatial Dropout

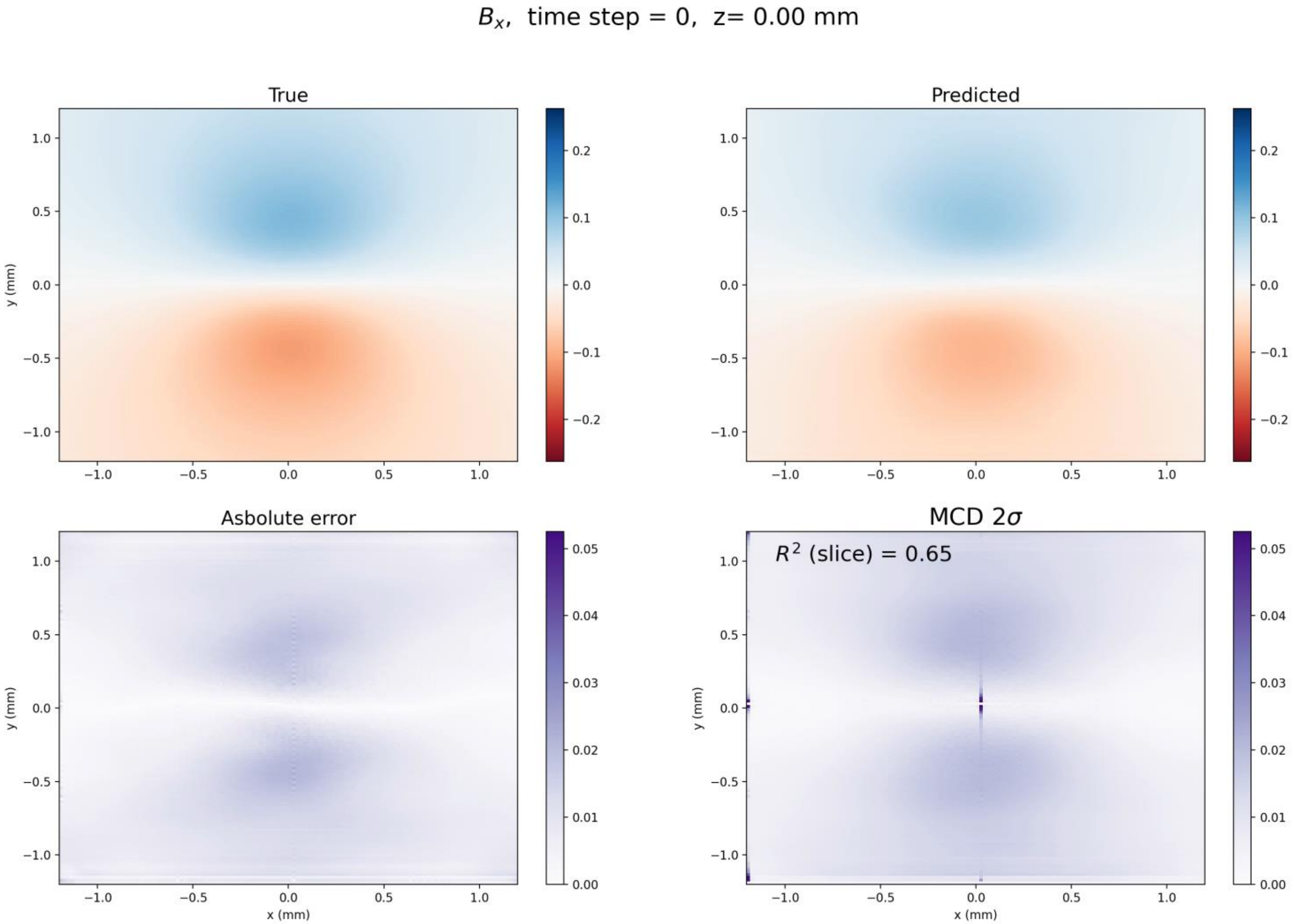
- From Tompson, Goroshin, LeCun, and Bregler (2015)
- Dropout for CNN's: **drop feature maps** instead of neurons
- Promote independence of feature maps



Preliminary Results

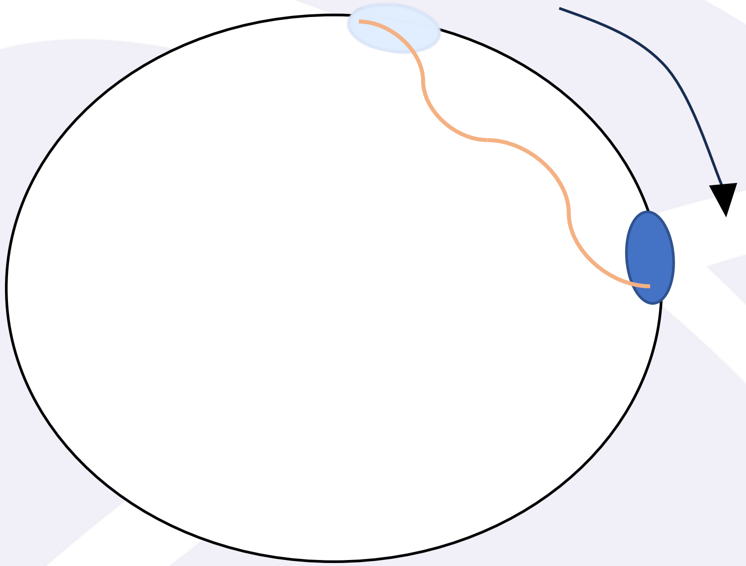
- Again, trained on Complex Beam

- With 10% DO and 10% SPO

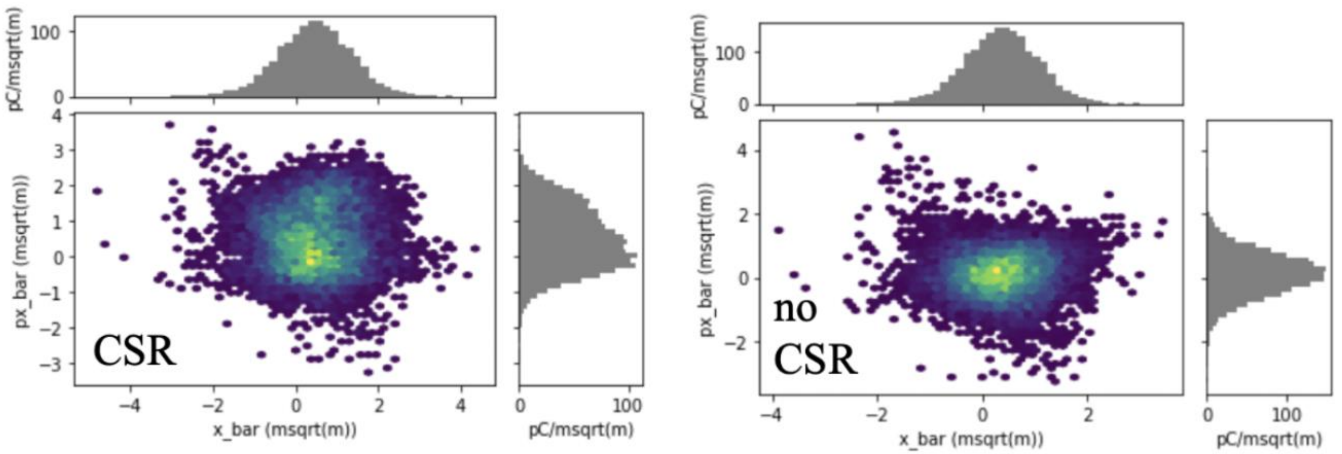


Coherent Synchrotron Radiation

Coherent Synchrotron Radiation



- Radiation emitted by particles (e^-'s) moving in circular orbit
- Accelerator: e^- storage ring, bunch compressor

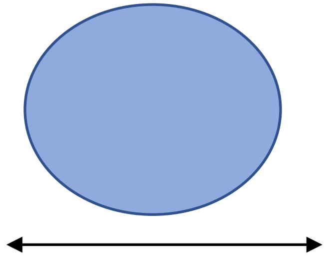


- Distorts bunch, increases spread
- Computationally expensive

arXiv: [2203.07542](https://arxiv.org/abs/2203.07542)

Coherence

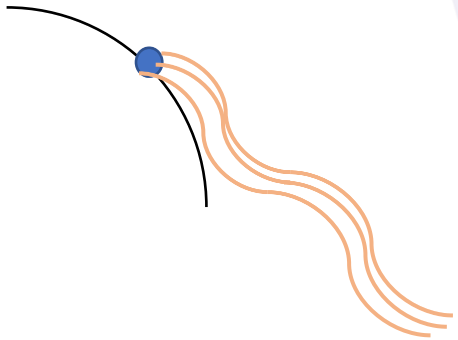
- Bunch



$$spread \ = \sigma$$

- For emissions $\lambda \gg \sigma$:

  Emissions in phase $\Rightarrow$ adds coherently



Bunch looks like point

$$\boldsymbol{I \propto N_e^2}$$

Liénard-Wiechert (LW) Fields

$$\boldsymbol{E}(\boldsymbol{r},t) = \frac{e}{4\pi\epsilon_0}\left(\underbrace{\frac{\boldsymbol{n}-\boldsymbol{\beta}'}{\gamma^2(1-\boldsymbol{n}\cdot\boldsymbol{\beta}')^3\,\rho^2}}_{\text{Coulomb field}} + \underbrace{\frac{\boldsymbol{n}\times\left((\boldsymbol{n}-\boldsymbol{\beta}')\times\dot{\boldsymbol{\beta}}'\right)}{c(1-\boldsymbol{n}\cdot\boldsymbol{\beta}')^3\rho}}_{\text{Radiation field}}\right)\Bigg|_{ret}$$

$$\boldsymbol{B} = \boldsymbol{n}\times\boldsymbol{E}$$

Coulomb field
"Space charge" (SC)

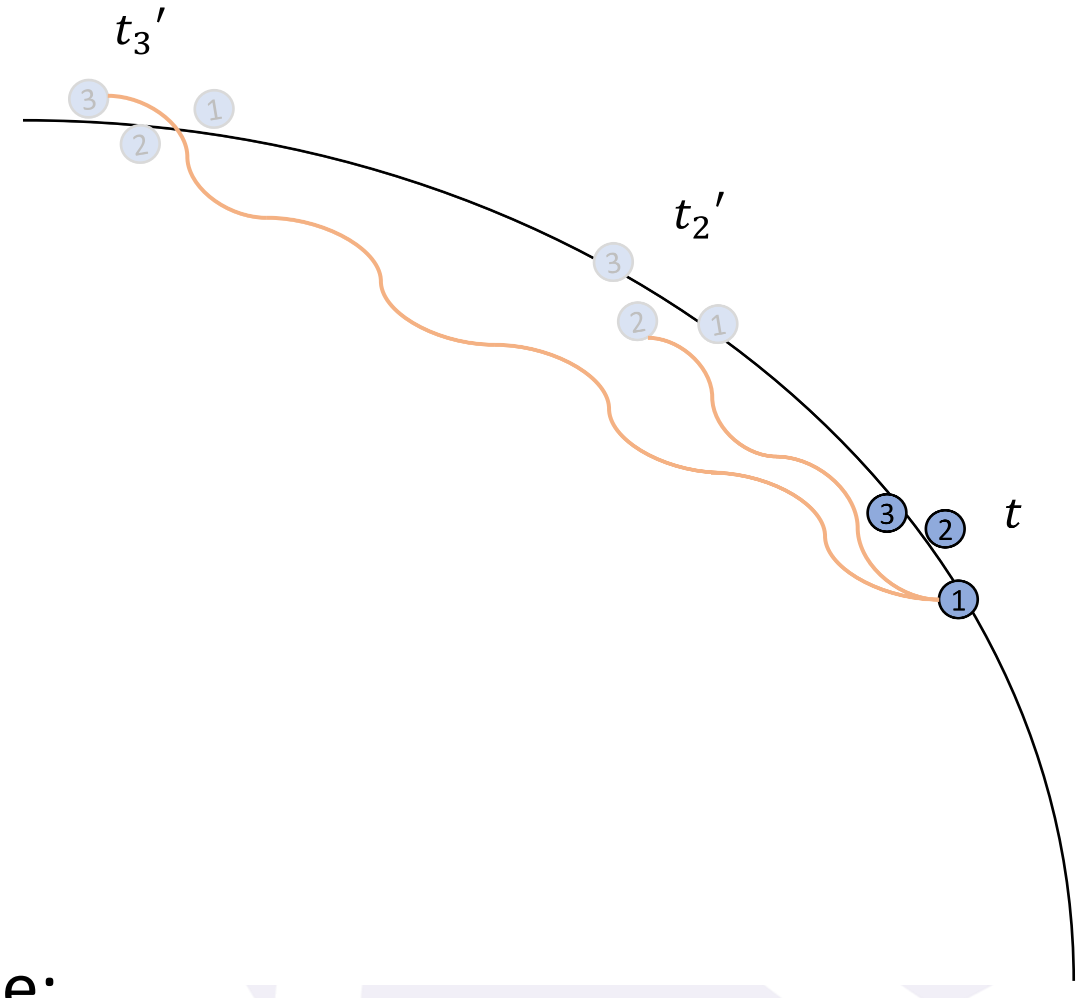
Radiation field
(⇒ Synchrotron radiation)

- $\boldsymbol{\beta}'$ = **v**/c, $\gamma = \frac{1}{\sqrt{1-v^2/c^2}}$
- Need to evaluate at "retarded time"

Synchrotron Radiation

- e^- bunch in circular motion

- At time t, particle 1 feels emissions of particle 2 at times $t_2{}'$ and 3 at $t_3{}'$



- Use LW fields and Lorentz Force:

$$\boldsymbol{F} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})$$

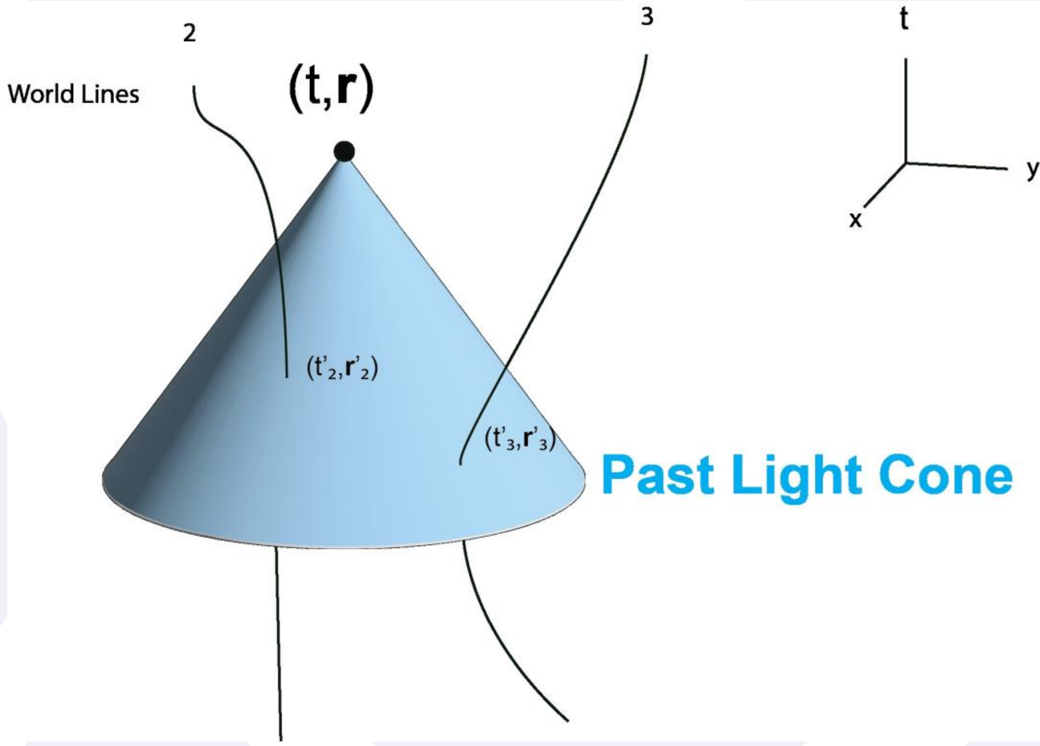
Wakefield

- Force felt by a particle at (t, **r**) gotten by integrating for e^- along past light cone, $\triangle$. E.g.:

$$\boldsymbol{W} = \int_{\triangle} \frac{dq(t',\boldsymbol{r}')}{e} \boldsymbol{F}(t',\boldsymbol{r}-\boldsymbol{r}')\Big|_{ret}$$

$$= \int_{\triangle} d^3\boldsymbol{r}' \big(N_b\, \lambda(t',\boldsymbol{r}')\big) \boldsymbol{F}(t',\boldsymbol{r}-\boldsymbol{r}')\Big|_{ret}$$

- $N_e \sim 10^{10}, \sim N_e^2$ interactions

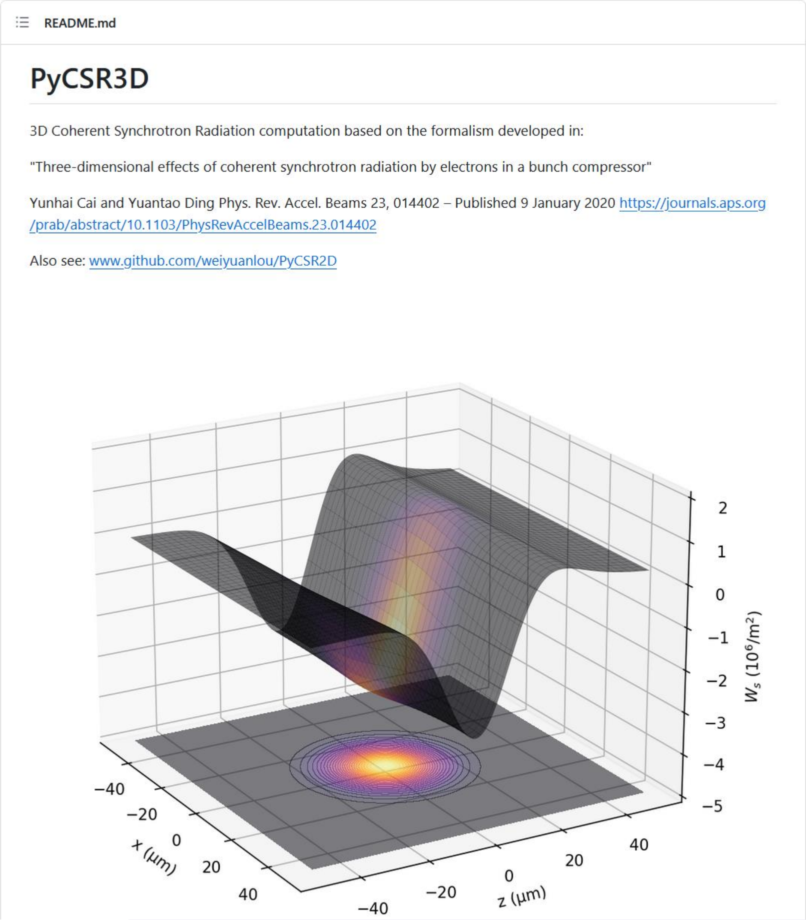


PyCSR3D

- Python package to calculate 3D CSR wakefields
- Trained NN's on data from PyCSR3D

PyCSR3D:

- Physics formalism based on Cai & Ding (2020)
  - Lienard Wiechart Fields
  - Snyder-Courant Theory
  - Hamiltonian in Frenet-Seret Co-ord.'s
- Paraxial + 4th order approx. for retarded time condition
- Assumes transience and shielding negligible
- Uses Integrated Green Function (IGF) method



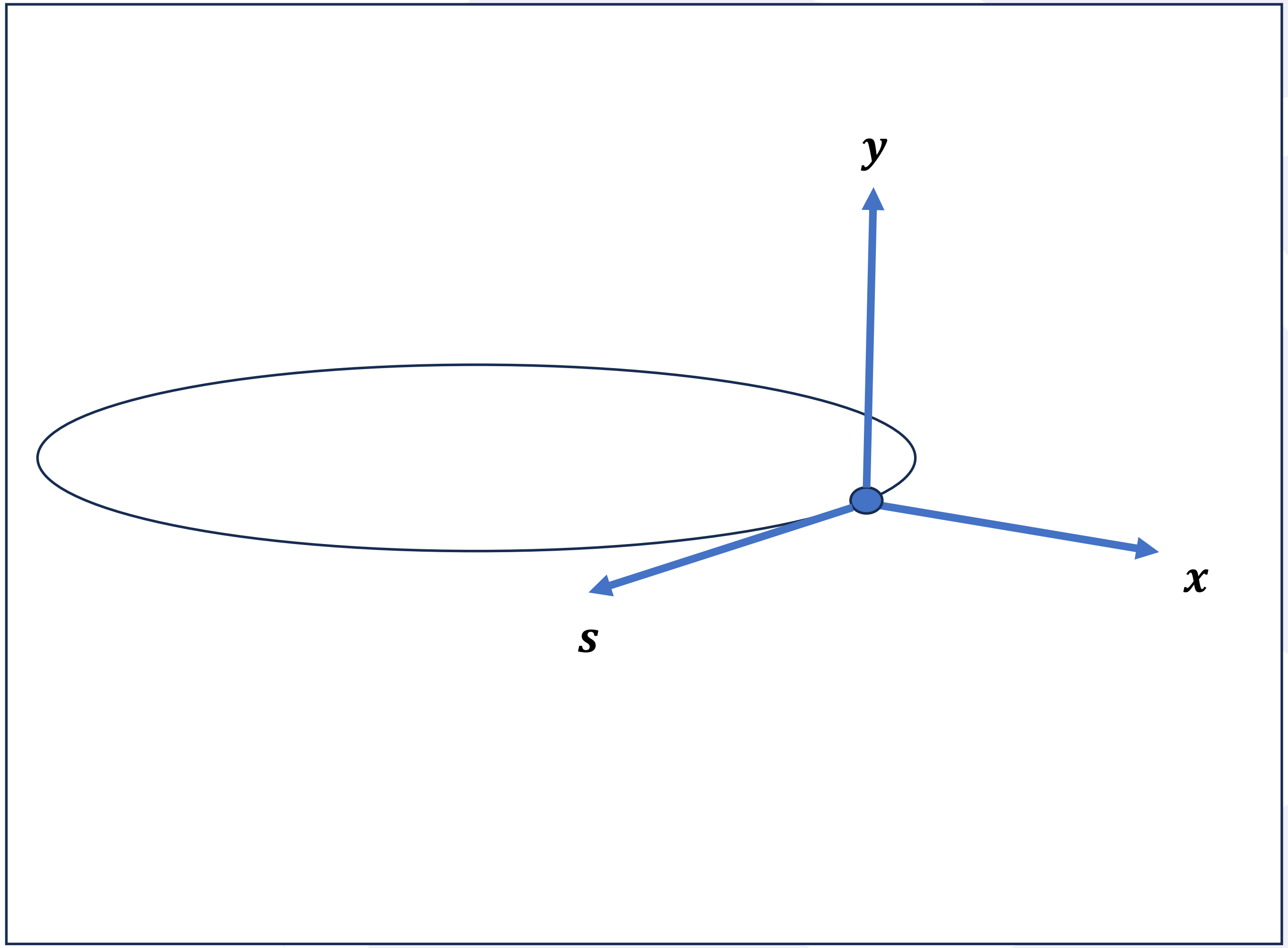
Available: https://github.com/ChristopherMayes/PyCSR3D

Paper: *Journal of Instrumentation* 16.10 (2021): P10010.

Frenet-Seret Co-ordinates for Circular Motion

Circle in horizontal plane

- Longitudinal: s
- Horizontal: x
- Vertical: y



Models - Baseline

- σ-NN Model (Gaussian):

$$\sigma_s, \sigma_x, \sigma_y \rightarrow \boldsymbol{W}(\boldsymbol{r})$$

- λ-NN Model (U-Net):

$$\lambda(\boldsymbol{r}) \rightarrow \boldsymbol{W}(\boldsymbol{r})$$

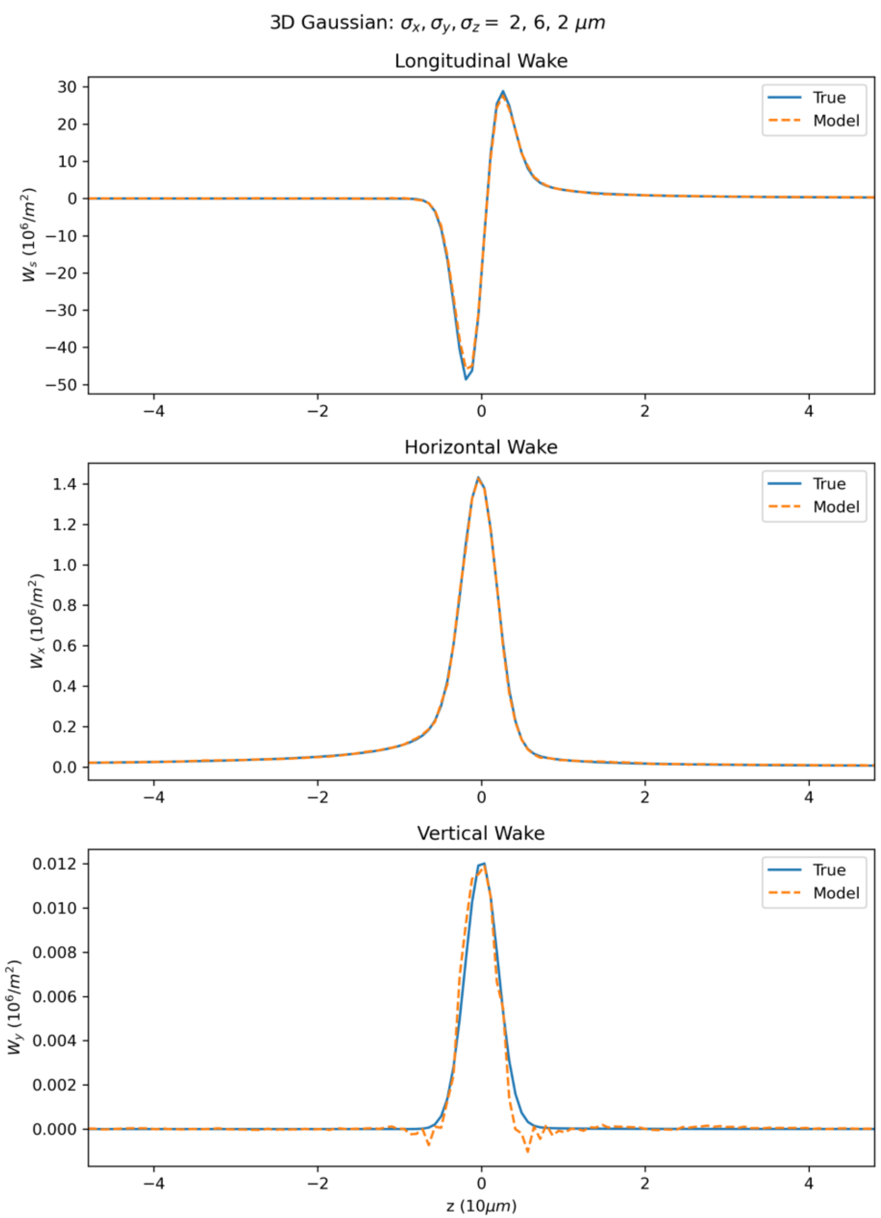
Dataset: 216 Wakefields generated by Gaussians with:

$$2 \leq \sigma_s, \sigma_x, \sigma_y \leq 12 \ \mu m$$

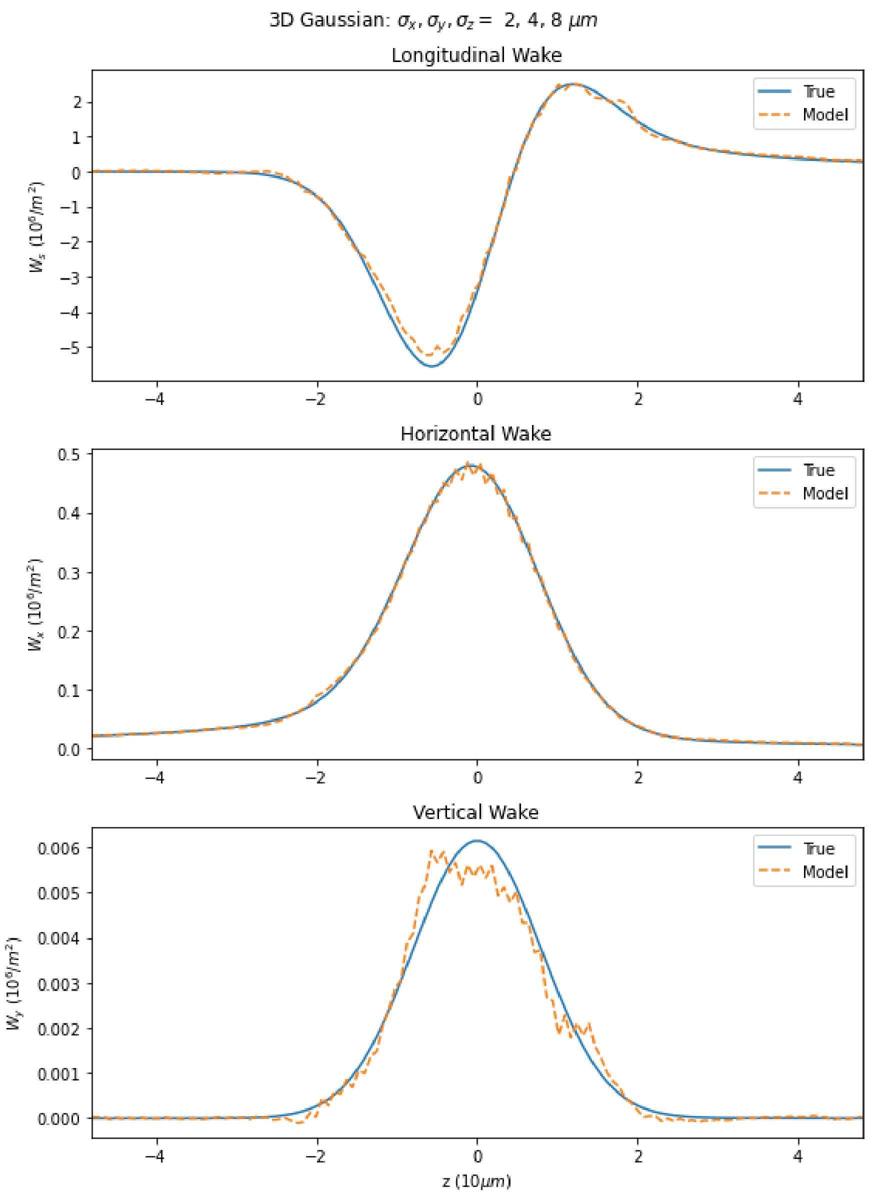
- Train/Test split: 85/15

σ-NN

Train



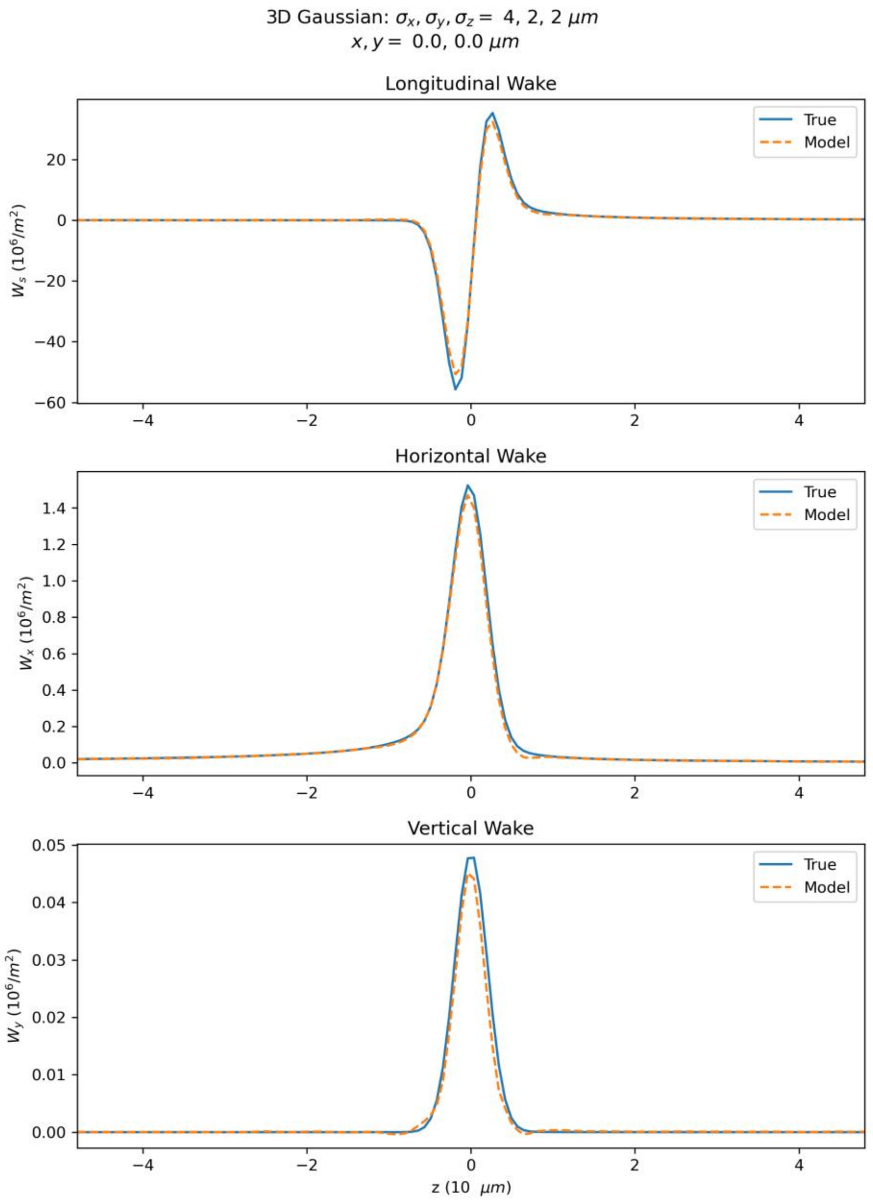
Test

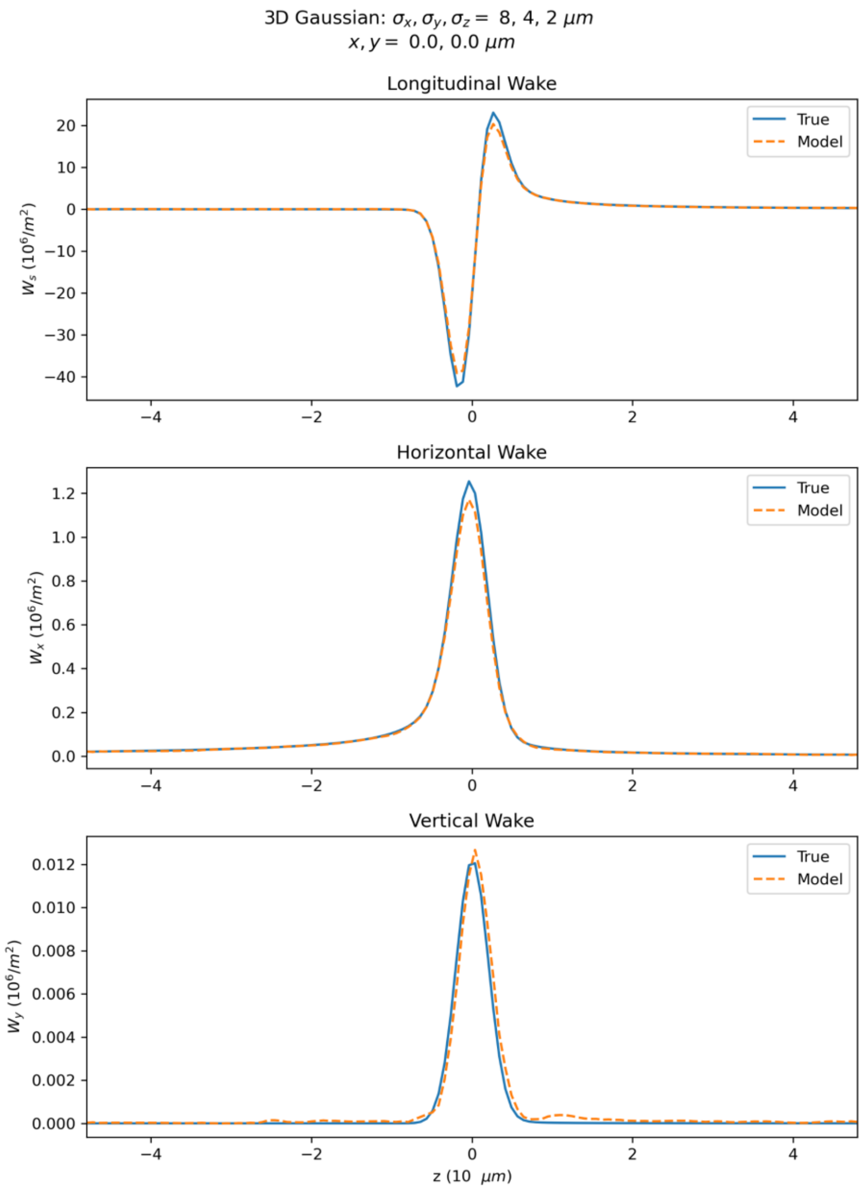


Train

Test

λ-NN

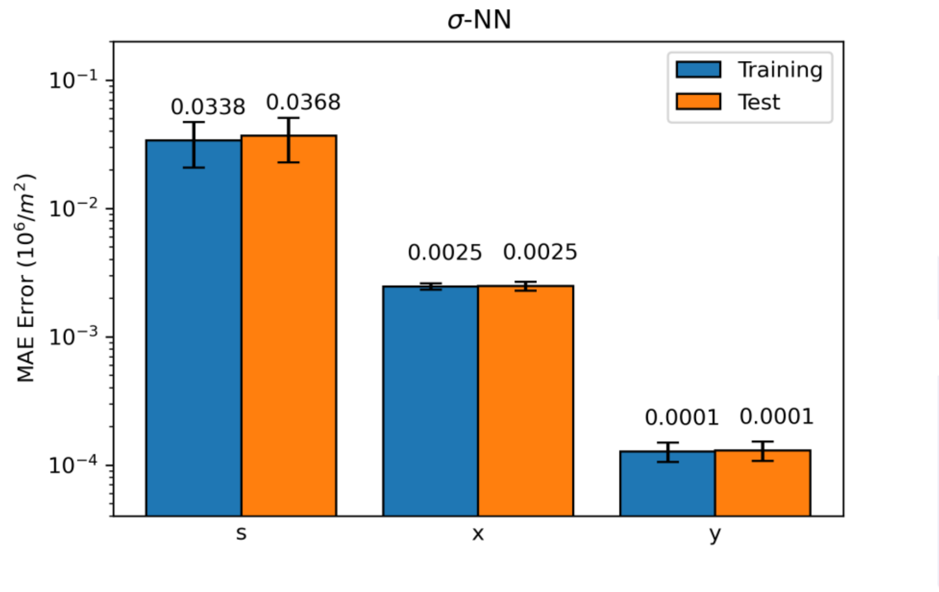


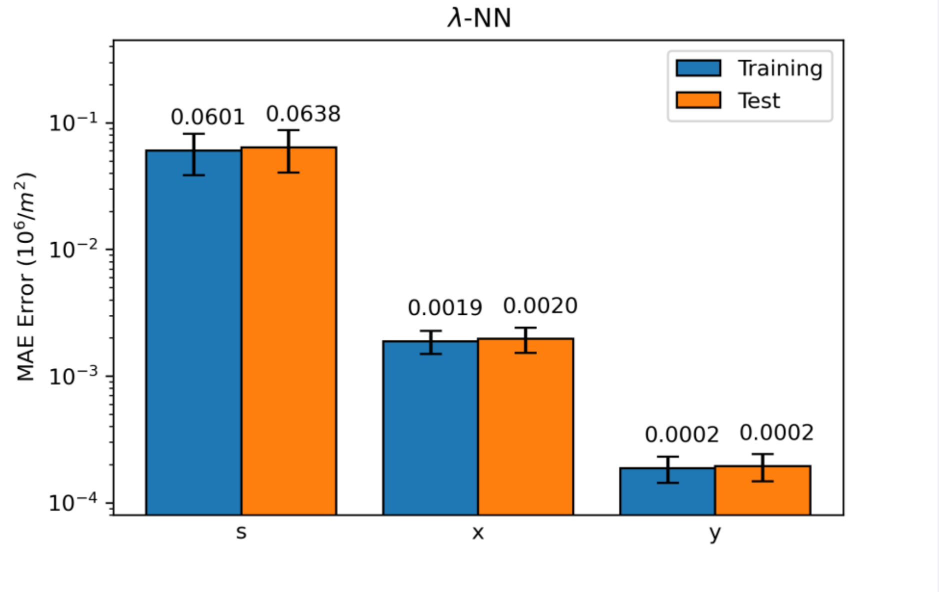


Training/Test
Mean Absolute Error

- Using standard deviation for scale,

Model	Set	s	x	y
σ-NN	Training	1.3%	2.8%	2.8%
	Test	1.4%	2.9%	2.9%
λ-NN	Training	2.3%	2.1%	4.1%
	Test	2.4%	2.2%	4.2%





Time

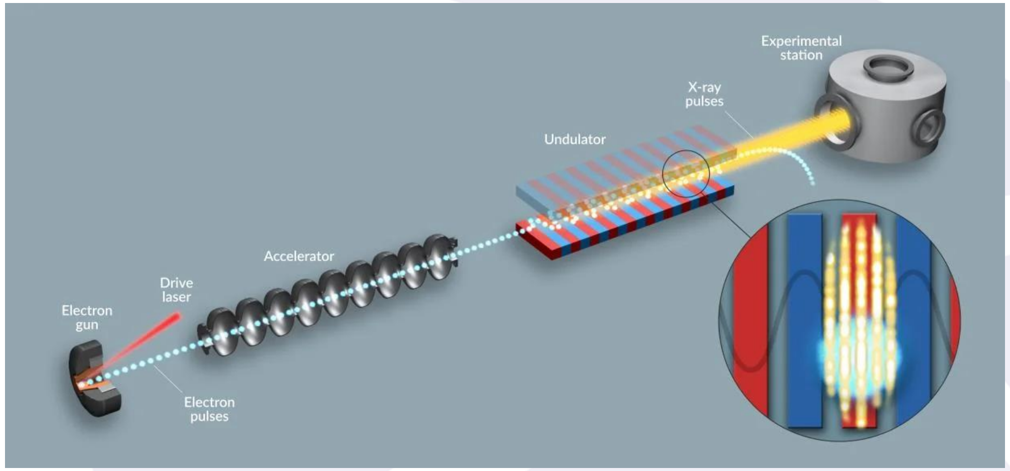
- Generate $\boldsymbol{W}$ on (128, 128, 128) grid.

- $\sim$ 1000x faster

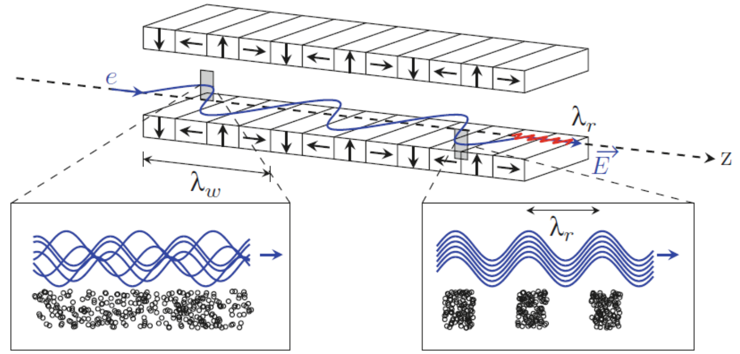
	Time (Average $\pm$ STD, 7 trials)
PyCSR3D	60.0 ± 0.4 s
σ-NN	46.9 ± 4.4 ms
λ-NN	59.4 ± 11.1 ms

Going Forward

- Other UQ techniques
- CSR NN: add physics constraints, make more general $\lambda(\boldsymbol{r})$, use on buncher
- X-ray Free Electron Laser
- Getting beam distribution from LANSCE (simulation + real data)



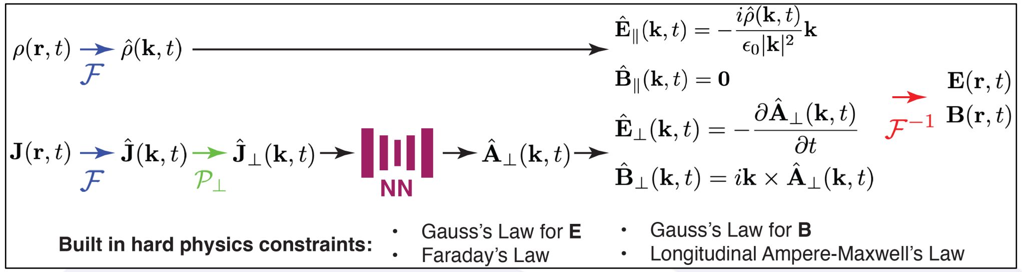
SLAC: What is an X-Ray free electron laser?

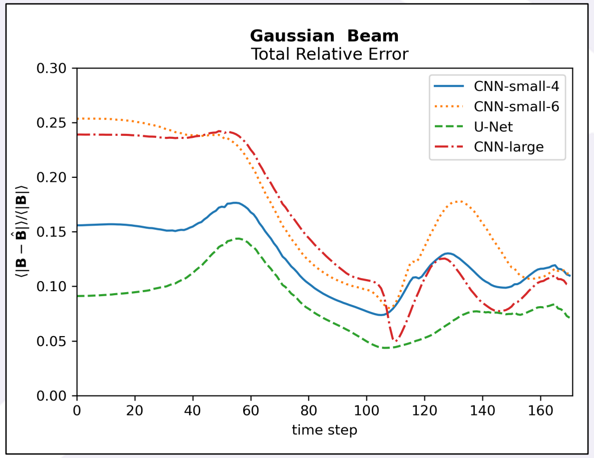


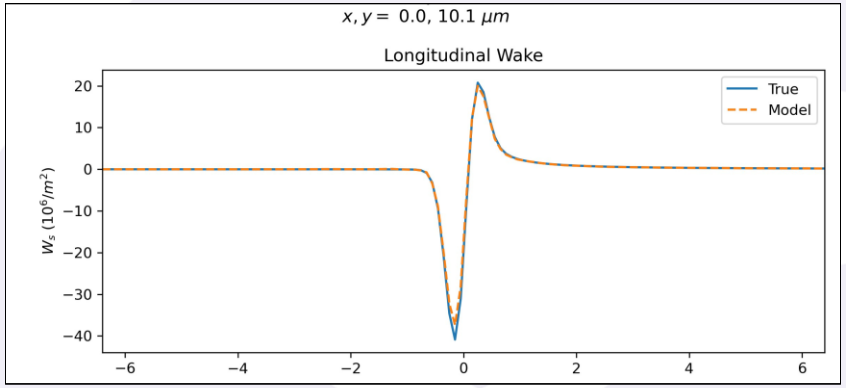
Jaeschke, Eberhard J., et al.
Synchrotron light sources and free-electron lasers

Conclusion

- Developed novel **FoHM-NO** method
- U-Net: best performance
- Simulation ~ day, U-Net: ~ seconds
- Used **MCD for UQ**
- NN: Accurate $\boldsymbol{W(r)}$ with **speedups of ~ 1000**







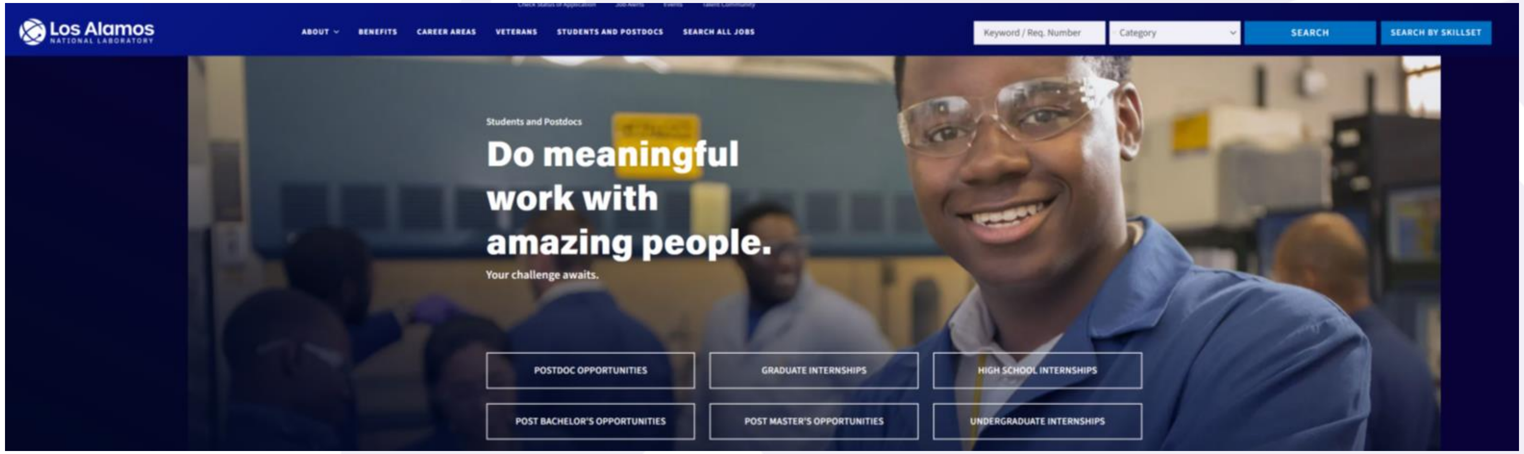
Funded by LANL-LDRD

Los Alamos National Lab

Internships:

- Undergraduate
- Graduate

Postdocs



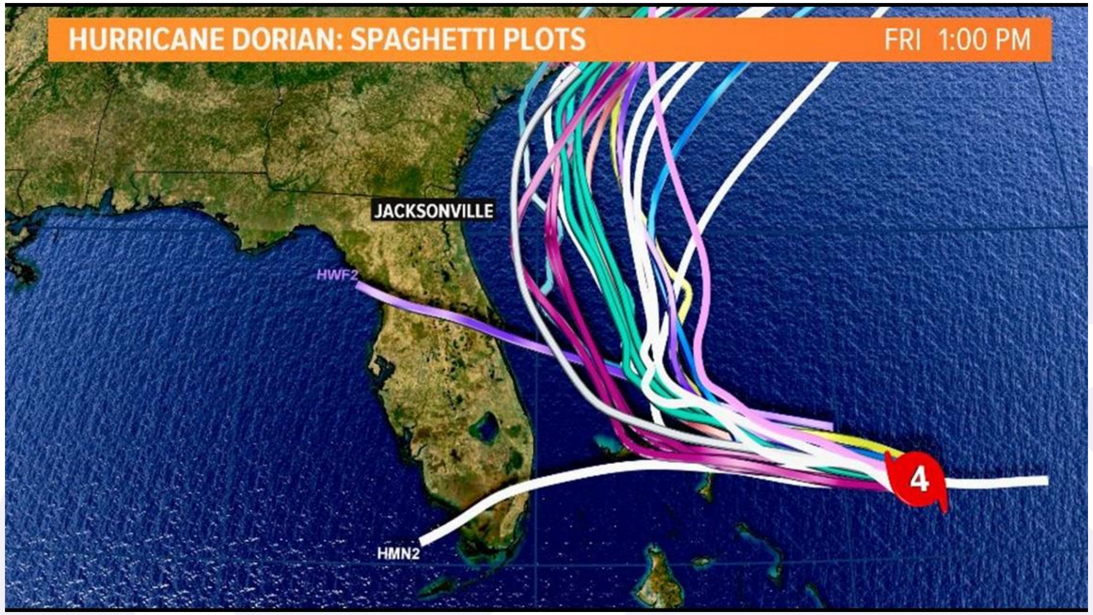
https://lanl.jobs/creative/students

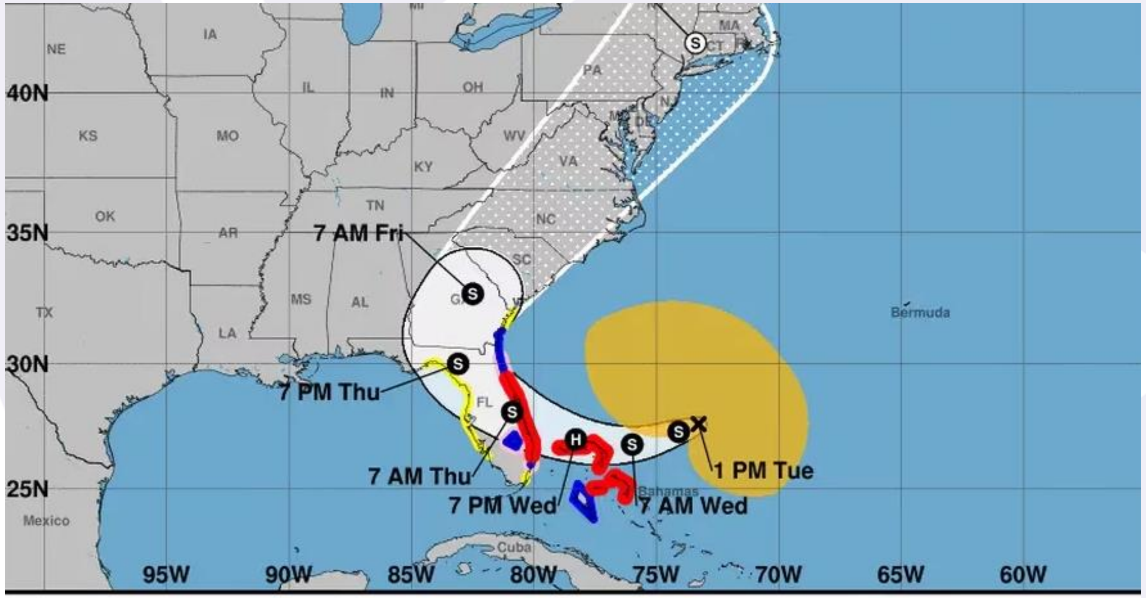
Los Alamos

Questions?

Backup Slide: weather

- Conventional simulations slow
- Limits incorporation of new data
- Also limits uncertainty quantification (UQ)

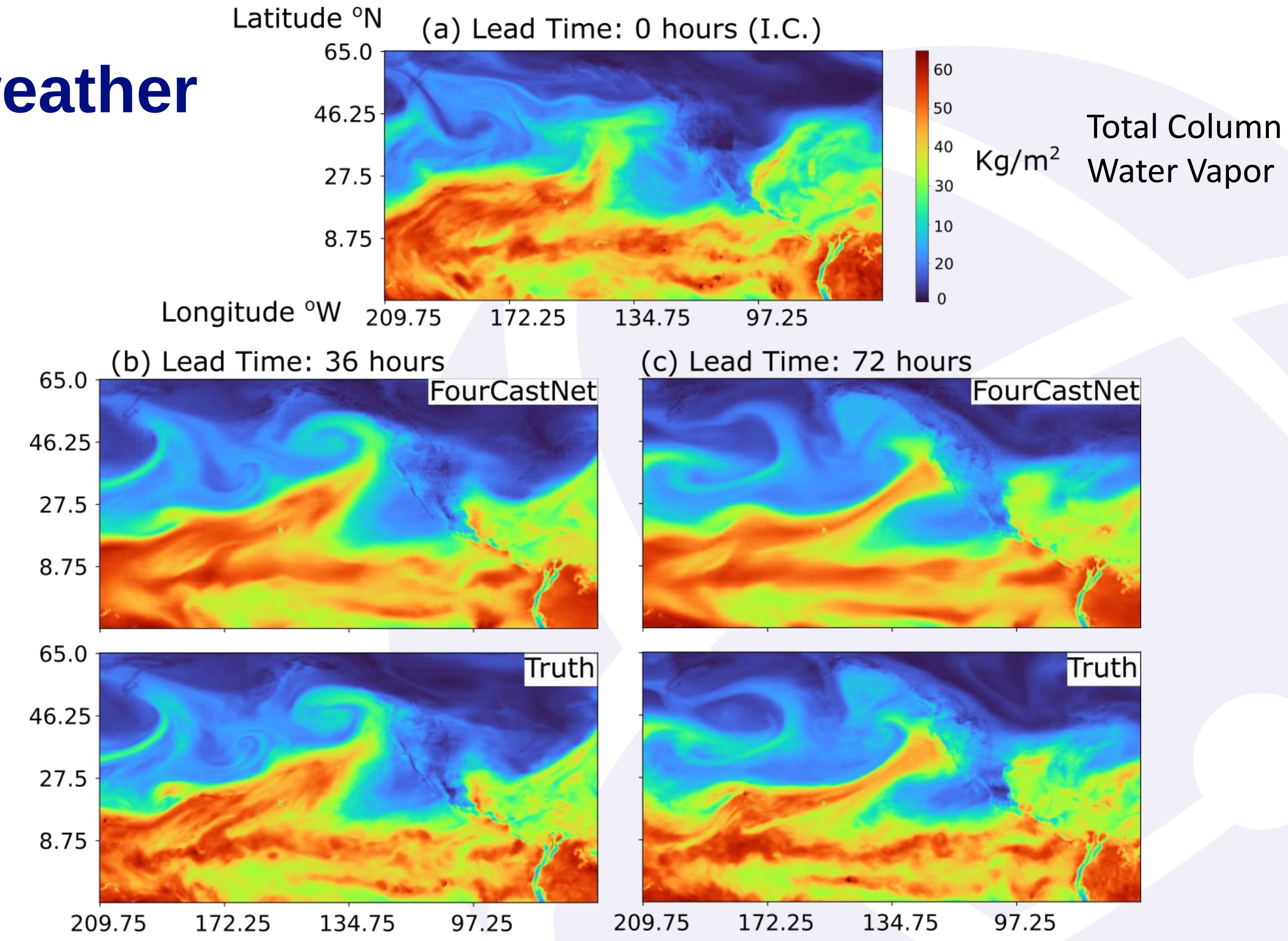




Backup Slide: weather

- Nvidia's FourCastNet

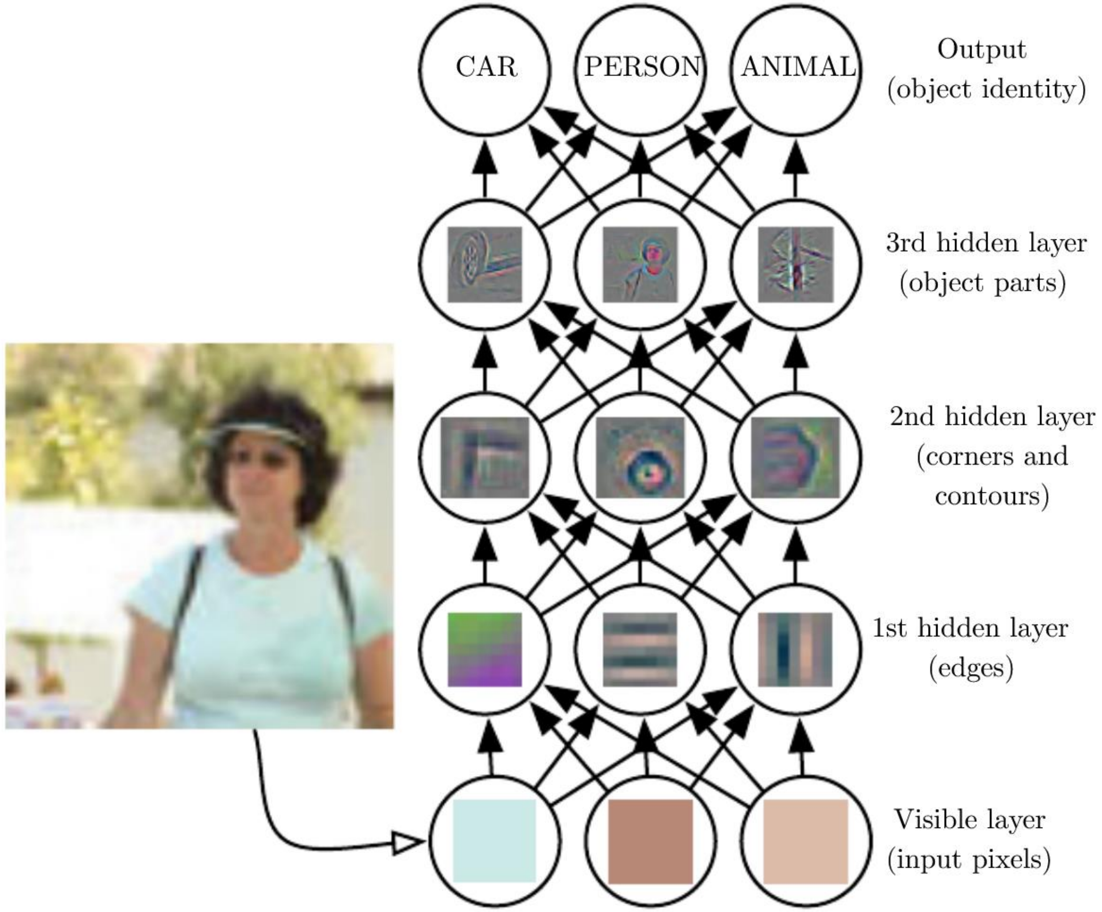
- Runs **50,000 times faster** than conventional simulations



FourCastNet , arXiv: [2202.11214](2202.11214)

Backup Slide: Abstraction Through Depth

- Build complexity from ground up
- First layers: lower-level features
- Later layers: high level features
- **Machine learns filters by itself**

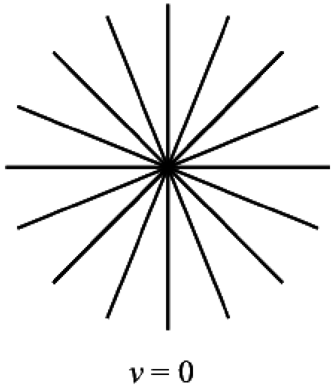


Deep Learning
Goodfellow, Bengio and Courville (2016)

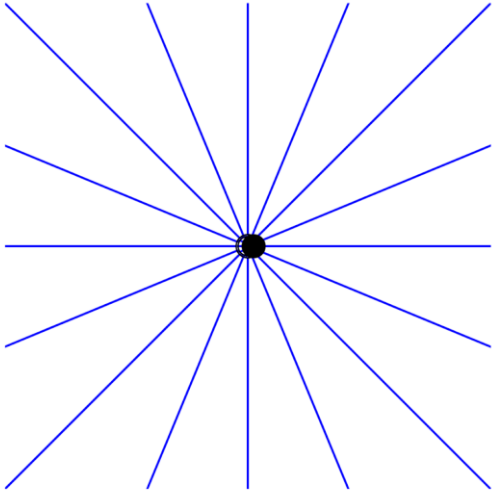
Backup Slide: EM field for a Particle

- Coloumb's law (at rest)

$$\boldsymbol{E} = \frac{e}{4\pi\epsilon_0}\frac{\hat{r}}{\rho^2}$$



- Jolt, then uniform motion:



Backup Slide: Solution Operator

Example: Poisson's equation

$$\nabla^2 \phi(\boldsymbol{r}) = \frac{\rho(\boldsymbol{r})}{\epsilon_0}$$

Solution:

$$\boxed{\phi(\boldsymbol{r}) = \frac{1}{4\pi\epsilon_0} \int d^3\boldsymbol{r}' G(\boldsymbol{r} - \boldsymbol{r}')\rho(\boldsymbol{r}')}$$

$$G(\boldsymbol{r} - \boldsymbol{r}') = \frac{1}{|\boldsymbol{r} - \boldsymbol{r}'|}$$

Solution operator

$$\rho \to \phi$$

Backup Slide: Universal Operator Approximation Theorem

(Informal) A neural network can approximate any operator (Chen and Chen, 1995)

- Note: architecture is not necessarily a CNN encoder-decoder

- Operator can offer family of solutions to PDE's

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 6, NO. 4, JULY 1995 911

Universal Approximation to Nonlinear Operators by Neural Networks with Arbitrary Activation Functions and Its Application to Dynamical Systems

Tianping Chen and Hong Chen

Abstract—**The purpose of this paper is to investigate neural network capability systematically. The main results are: 1) every Tauber–Wiener function is qualified as an activation function in the hidden layer of a three-layered neural network, 2) for a continuous function in $S'(R^1)$ to be a Tauber–Wiener function, the necessary and sufficient condition is that it is not a polynomial, 3) the capability of approximating nonlinear functionals defined on some compact set of a Banach space and nonlinear operators has been shown, which implies that 4) we show the possibility by neural computation to approximate the output as a whole (not at a fixed point) of a dynamical system, thus identifying the system.**

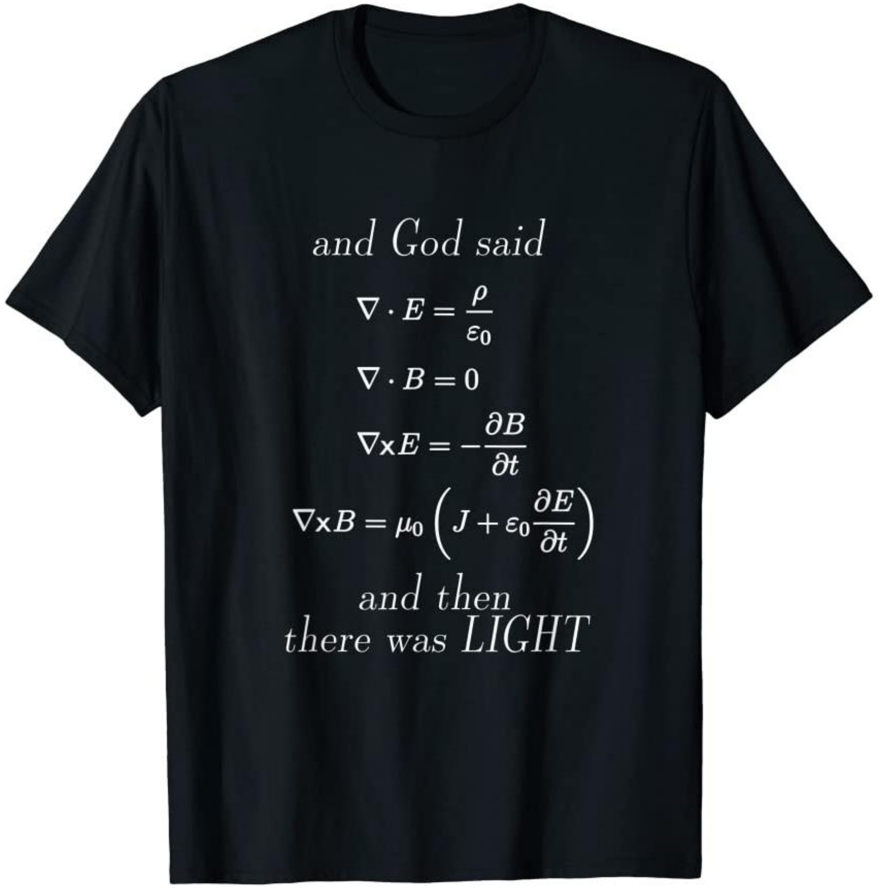
Mhaskar and Micchelli [11] showed that under some restriction on the amplitude of a continuous function near infinity, any nonpolynomial function is qualified to be an activation function.

It is clear that all the aforementioned works are concerned with approximation to a continuous function defined on a compact set in R^n (a space of finite dimensions). In engineering problems such as computing the output of dynamic systems or designing neural system identifiers, however, we often encounter the problem of approximating nonlinear functionals

FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators

arXiv: 2202.11214

Backup Slide: Maxwell in Fourier Space



$$\widehat{\boldsymbol{F}}(\boldsymbol{k}) = \frac{1}{(2\pi)^{3/2}} \int d^3r \, e^{-i\boldsymbol{k}\cdot\boldsymbol{r}} \boldsymbol{F}(\boldsymbol{r})$$

$\Longrightarrow^{*}$

$$i\,\boldsymbol{k} \cdot \widehat{\boldsymbol{E}} = \frac{\hat{\rho}}{\epsilon_0}$$

$$i\,\boldsymbol{k} \cdot \widehat{\boldsymbol{B}} = 0$$

$$i\,\boldsymbol{k} \times \widehat{\boldsymbol{E}} = -\frac{\partial \widehat{\boldsymbol{B}}}{\partial t}$$

$$i\,\boldsymbol{k} \times \widehat{\boldsymbol{B}} = \mu_0 \left(\hat{\boldsymbol{J}} + \epsilon_0 \frac{\partial \widehat{\boldsymbol{E}}}{\partial t} \right)$$

* Need to treat $\boldsymbol{k} = \boldsymbol{0}$ case separately

Backup Slide: Decomposition in Fourier Space

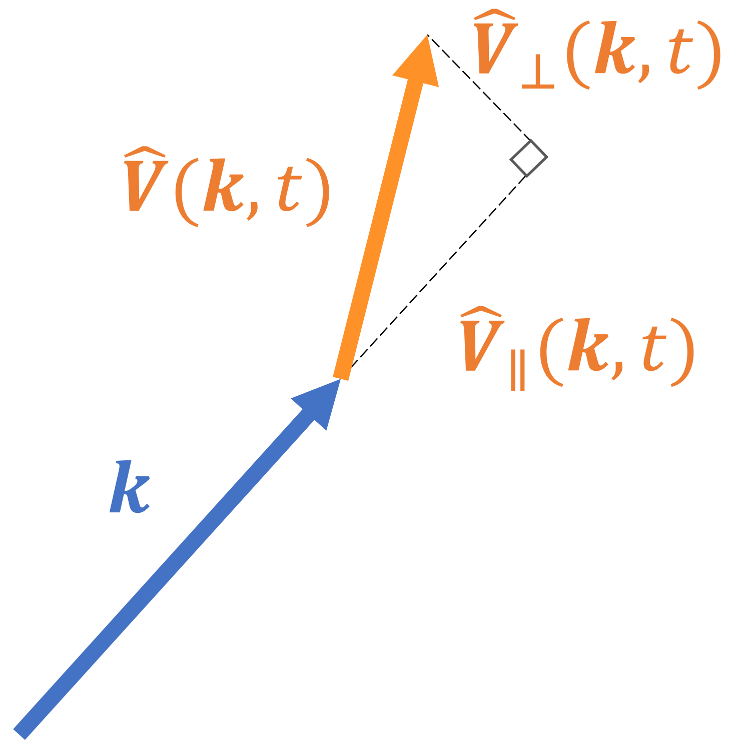
- Gauss's Laws:

$$\widehat{\boldsymbol{E}}_{\parallel} = \frac{-i\,\hat{\rho}}{\epsilon_0 |\boldsymbol{k}|^2}\boldsymbol{k} \qquad \widehat{\boldsymbol{B}}_{\parallel} = \boldsymbol{0}$$

- Use $A^{\mu} = (\phi/c\,, \boldsymbol{A})$ for transverse components:

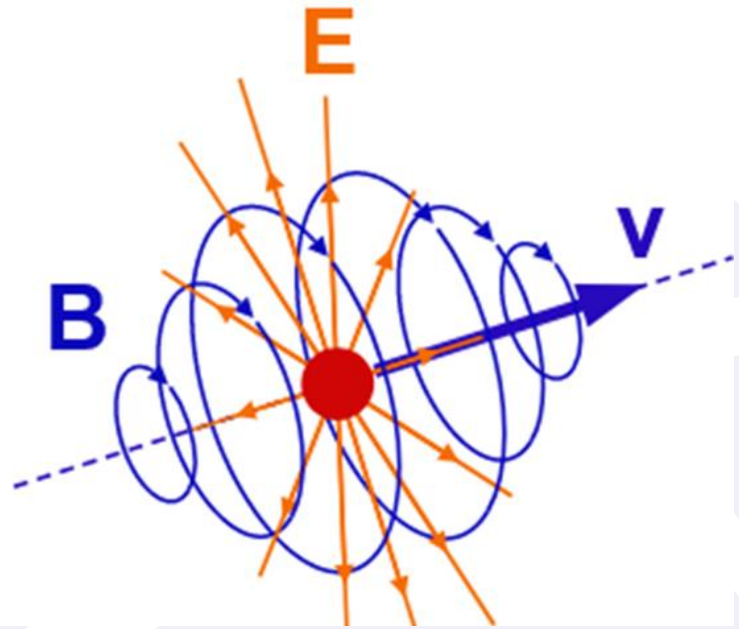
$$\widehat{\boldsymbol{E}}_{\perp} = -\frac{\partial \widehat{\boldsymbol{A}}_{\perp}}{\partial t} \qquad \widehat{\boldsymbol{B}}_{\perp} = i\boldsymbol{k} \times \widehat{\boldsymbol{A}}_{\perp}$$

- $\widehat{\boldsymbol{A}}_{\perp} = \boldsymbol{O}\left[\hat{J}_{\perp}\right]$. Model $\boldsymbol{O}$.
- **FoHM-NO**: Fourier-Helmholtz-Maxwell Neural Operator



Equivalent to Helmholtz decomposition

Backup Slide: Electron Beam in $\boldsymbol{B_{ext}}$



- Approximation:

$$|J_z| \gg |J_x|, |J_y| \quad \Rightarrow \quad |B_x||B_y| \gg |B_z|$$

- Model:

$$Re(\hat{J}_{\perp,z}), Im(\hat{J}_{\perp,z}) \rightarrow NN \ \rightarrow Re(\hat{A}_{\perp,z}), Im(\hat{A}_{\perp,z})$$

- Then get x and y Fourier magnetic components from:

$$\widehat{\boldsymbol{B}} = \mathrm{i}\, \mathbf{k} \times (\hat{A}_{\perp,z}\, \hat{\boldsymbol{z}})$$

Backup Slide: Gauge Transformations

$$\hat{\phi} \rightarrow \hat{\phi} + \frac{\partial f}{\partial t} \qquad \widehat{\boldsymbol{A}} \rightarrow \widehat{\boldsymbol{A}} + \mathrm{i}\mathbf{k}f$$

for some f

- But:

$$\widehat{\boldsymbol{A}}_{\perp} \rightarrow \widehat{\boldsymbol{A}}_{\perp}$$

- $\widehat{\boldsymbol{A}}_{\perp}$: need to be careful changing reference frames
- Related to photon having 2 physical traverse polarization d.o.f.

Backup Slide: Fitting and Evaluation Metric

- Loss function:

$$L = \sum_i \int d^3\boldsymbol{k} \left| \widehat{\boldsymbol{B}}(\boldsymbol{k}, t_i) - \widehat{\boldsymbol{B}}_{pred}(\boldsymbol{k}, t_i) \right|^2$$

By Parseval's theorem equivalent to:

$$L = \sum_i \int d^3\boldsymbol{r} \left| \boldsymbol{B}(\boldsymbol{r}, t_i) - \boldsymbol{B}_{pred}(\boldsymbol{r}, t_i) \right|^2$$

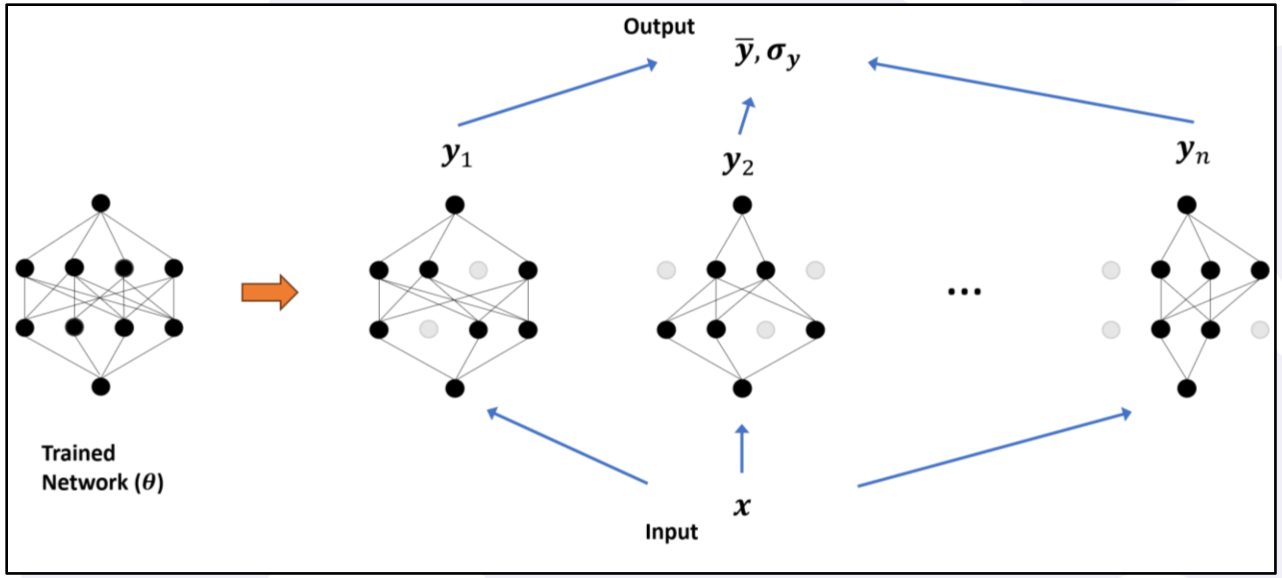
- Evaluation metric, Ave. Relative Error:

$$\epsilon = \frac{\langle |\boldsymbol{B} - \boldsymbol{B}_{pred}| \rangle}{\langle |\boldsymbol{B}| \rangle} = \frac{\int d^3\boldsymbol{r} |\boldsymbol{B} - \boldsymbol{B}_{pred}|}{\int d^3\boldsymbol{r}\, |\boldsymbol{B}|}$$

Backup Slide: MCD

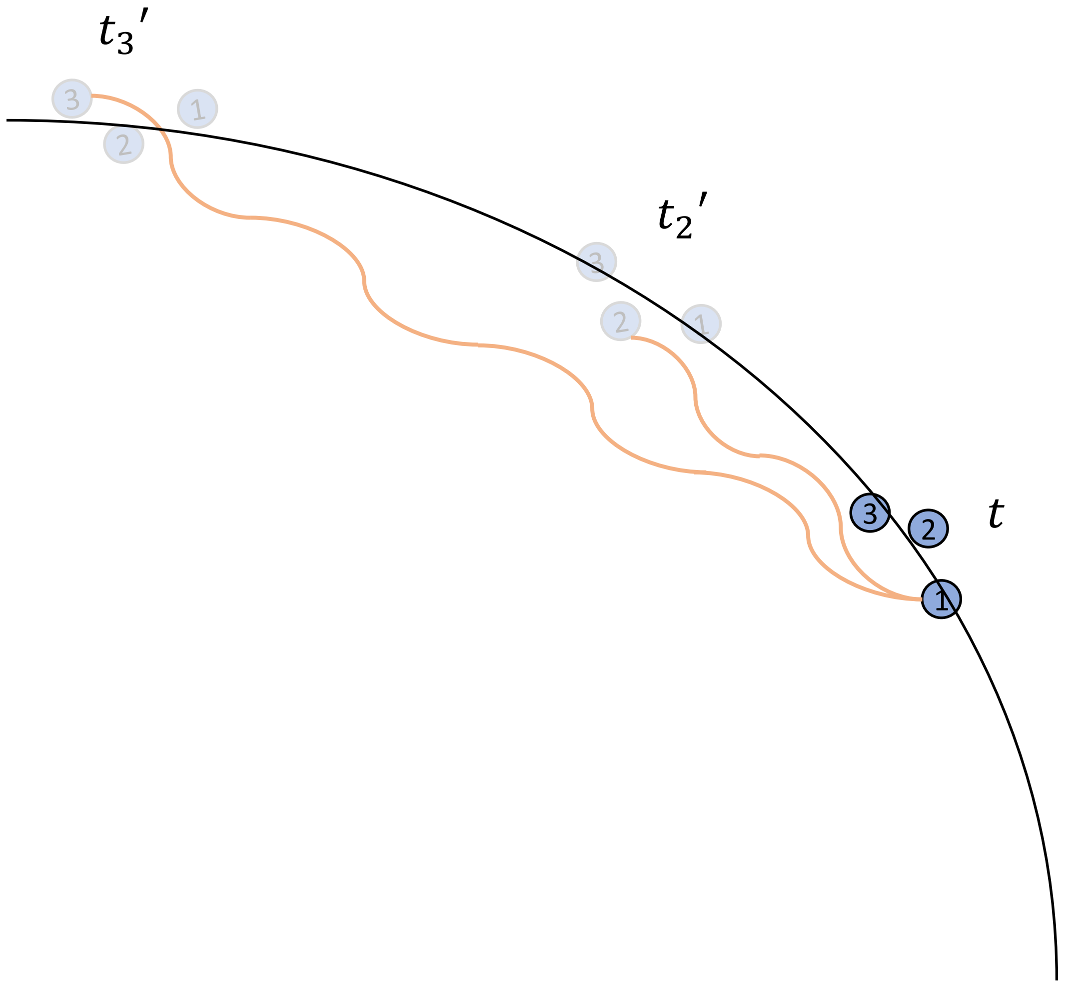
- Computationally efficient
- Easy to implement
- Only train one network at once

- Con: less variety than independently trained ensemble.



Backup Slide: CSR - Computationally Expensive

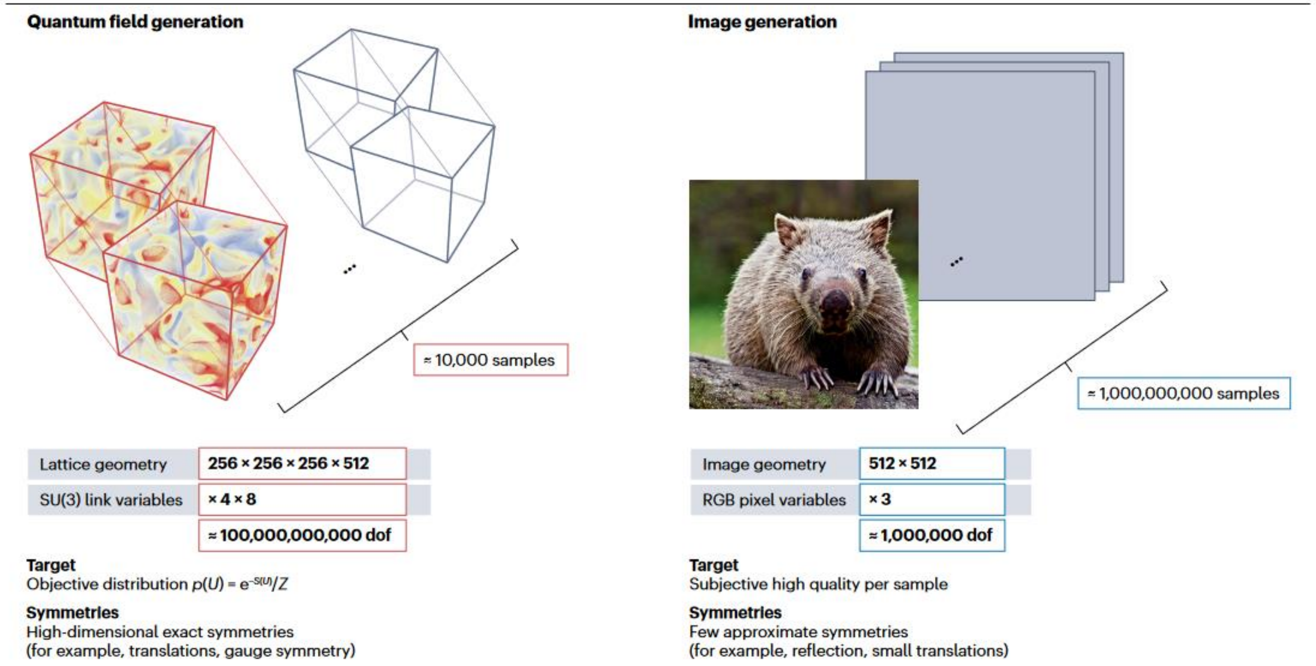
- $N_e \sim 10^{10}$
- $\sim N_e^2$ interactions + saving distribution at earlier times
- 1D approx. has been used, can slow down by factor ~ 10
- Can CNN’s help with speed up?



Backup: Other Cases

Lattice Gauge Theory

- MC Sampling



Kranmer et al. Nature Reviews Physics (2023)