

# Machine Learning Methods in High-Energy Scattering

**Chris Leon**

6/24/2025




# Introduction

- Two useful reviews

## Machine Learning in Nuclear Physics

Boehnlein, Amber, et al.

"Colloquium: Machine learning in nuclear physics."  
*Reviews of modern physics* 94.3 (2022): 031003.

Alexander Scheinker 

*Accelerator Operations and Technology Division Applied Electrodynamics Group,  
Los Alamos National Laboratory, Los Alamos, New Mexico 87544, USA*

Computing and Software for Big Science (2024) 8:5  
<https://doi.org/10.1007/s41781-024-00113-4>

REVIEW



Artificial Intelligence for the Electron Ion Collider (AI4EIC)

# Possible Uses

- Difficult to model/simulate, but you have lots of data
- Time is crucial
- Care more about predictions than understanding

**NOT** always the right tool for the job

# Categories

- **Functional Approximation**
- **Classification**
- **Generation**
- **Chores**

# Functional Approximation

# Feed-forward, Dense Neural Network

- Function  $\mathbb{R}^m \rightarrow \mathbb{R}^n$  composed of more elementary functions:

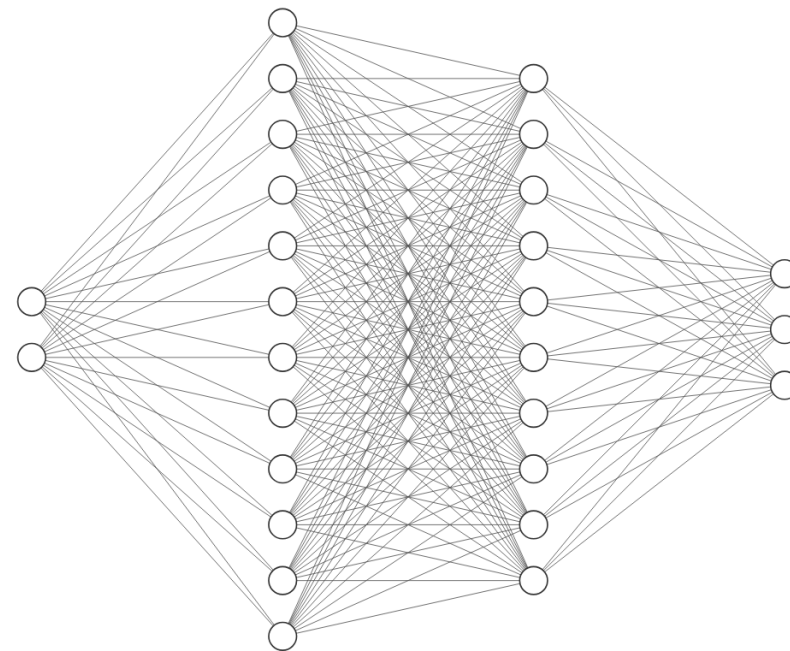
$$f(\mathbf{x}) = A_n \circ \sigma \circ \cdots \circ A_1 \circ \sigma(\mathbf{x}),$$

where:

$$A(\mathbf{x}) = W\mathbf{x} + \mathbf{b},$$

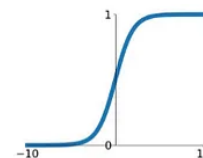
and  $\sigma$  is an ‘activation function’ (aka, non-linear function):

- $\sigma(\mathbf{x}) = (\sigma(x_1), \sigma(x_2) \dots \sigma(x_m))$



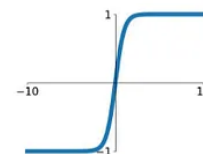
**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



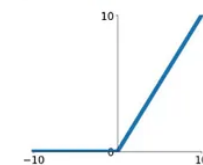
**tanh**

$$\tanh(x)$$



**ReLU**

$$\max(0, x)$$



# Universal Function Approximation Theorem

(Informal): *A deep enough NN can approximate any function,*  
 $\mathbb{R}^m \rightarrow \mathbb{R}^n$

## Neural Network:

$$f(\mathbf{x}) = A_m \circ \dots \circ \sigma \circ A_2 \circ \sigma \circ A_1(\mathbf{x})$$

- Parameters: Weights & biases
- Doing more functional compositions (“adding” more layers) to improve

## Taylor Series (analogy):

$$y = a_0 + a_1x + \dots a_nx^n$$

- Parameters: Coefficients
- Adding higher order terms to improve

# Training Procedure

- Learning as an optimization problem
- Gradient descent (or variation) to determine parameters
- Train/validation/test split

Want to minimize:

$L$

Example:

$$L = MSE = \frac{1}{N} \sum_i (y_i - \hat{y}_i)^2$$

Training

Validation

Test

**Machine learns from this set**  
(‘lecture notes, homework’)

**Used to monitor learning;**  
**Not just memorizing**  
(‘quizzes’)

**Final Evaluation**  
(‘final exam’)



# NNPDF

- PDF sets start with parameterizes  $f_i(x, Q_0^2)$

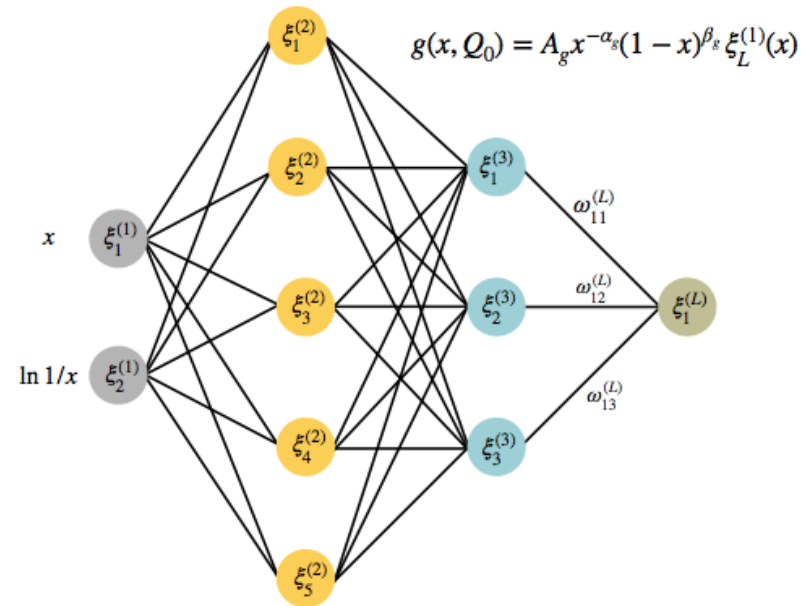
Example: CJ15 for valence u/d:

$$xf(x, Q_0^2) = a_0 x^{a_1} (1-x)^{a_2} (1 + a_3 \sqrt{x} + a_4 x).$$

- Parameterizes  $f_i(x, Q_0^2)$  with NN
- NNPDF 1.0 released 2008



<https://nnpdf.mi.infn.it/>



For users ▾ Docume

NNPDF code

Unpolarized PDF sets

Polarized PDF sets

Nuclear PDFs

Fragmentation  
functions

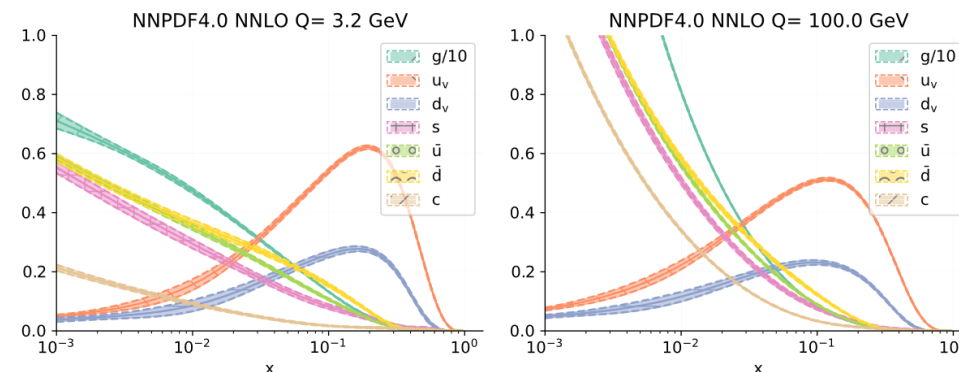
Neutrino Structure  
Functions

Tools

- Order: up to NNLO
- $N_{data} = 4,618$
- Written in TensorFlow (past NNPDF's used in-house C++ code)
- Use of GPU's and TensorFlow sped fitting time by factor of  $\times 140$
- Gradient descent optimization (vs. genetic algorithm in 3.1)

## The path to proton structure at 1% accuracy

NNPDF Collaboration



Evolution basis:

$$\Sigma = u + \bar{u} + d + \bar{d} + s + \bar{s} + 2c,$$

$$T_3 = (u + \bar{u}) - (d + \bar{d}),$$

$$T_8 = (u + \bar{u} + d + \bar{d}) - 2(s + \bar{s})$$

$$V = (u - \bar{u}) + (d - \bar{d}) + (s - \bar{s}),$$

$$V_3 = (u - \bar{u}) - (d - \bar{d}),$$

$$V_8 = (u - \bar{u} + d - \bar{d}) - 2(s - \bar{s}).$$

$$T_{15} = (u + \bar{u} + d + \bar{d} + s + \bar{s}) - 3(c + \bar{c})$$

In addition:  $g$

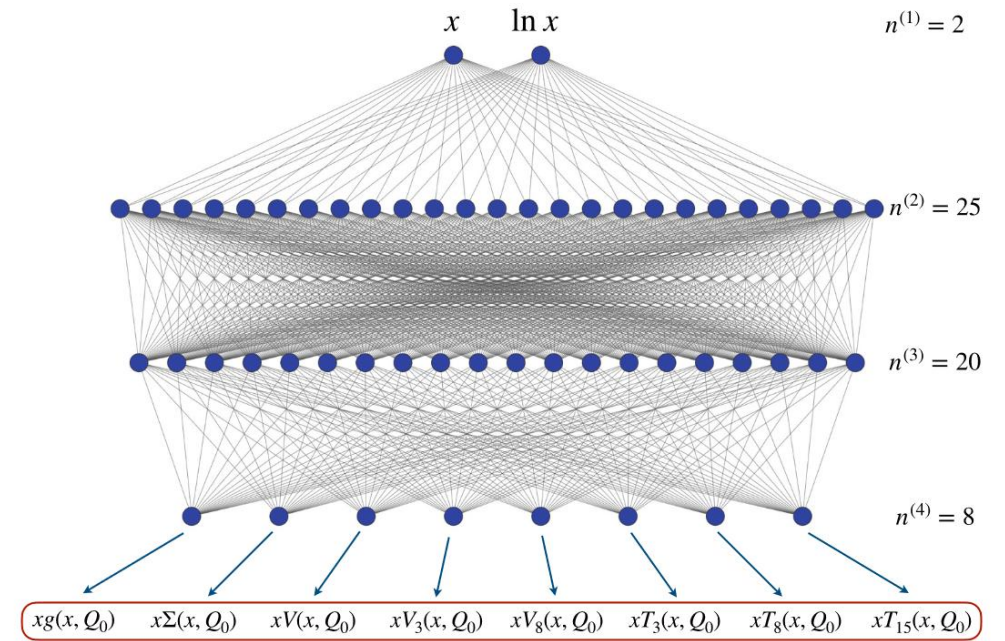
# NNPDF 4.0 – Training Procedure

- Single network for all PDF's

$$xf_k(x, Q_0; \theta) = A_k x^{1-\alpha_k} (1-x)^{\beta_k} \text{NN}_k(x; \theta),$$

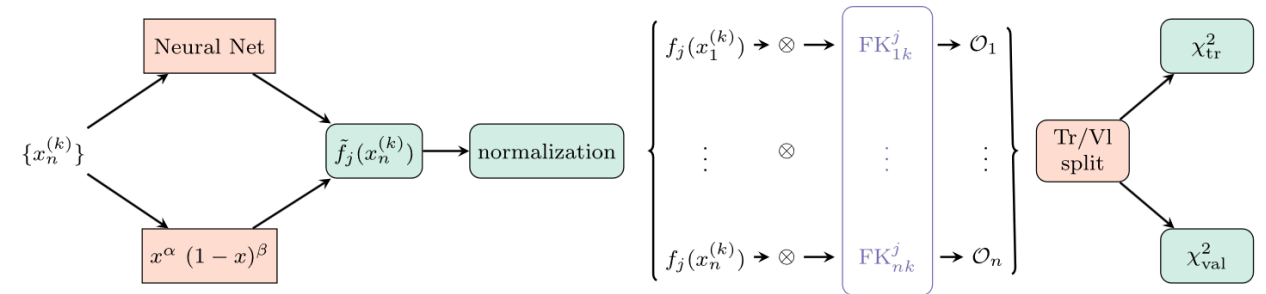
$$k = 1, \dots, 8,$$

- 763 free parameters (NNPDF 3.1: 296)



Activation function

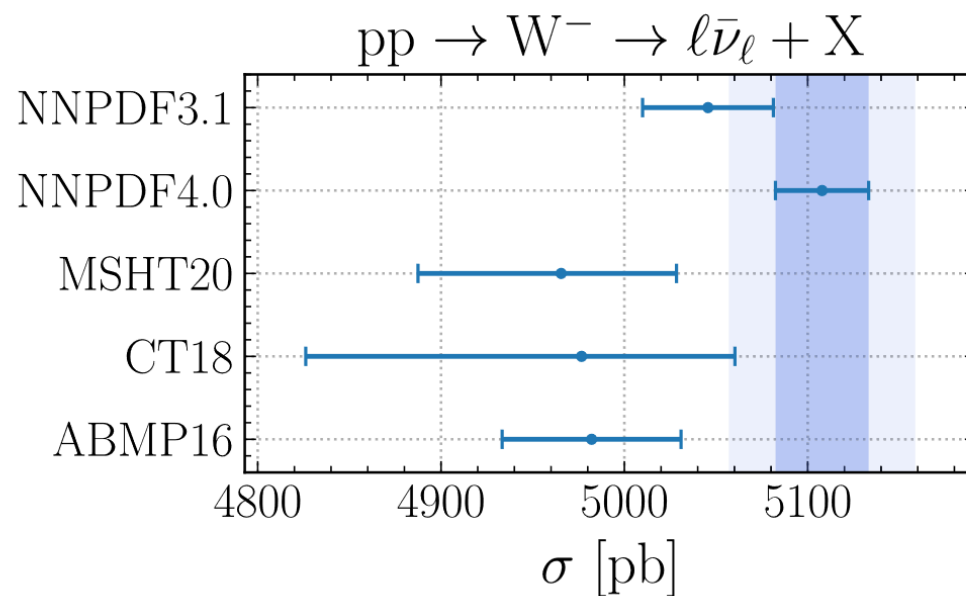
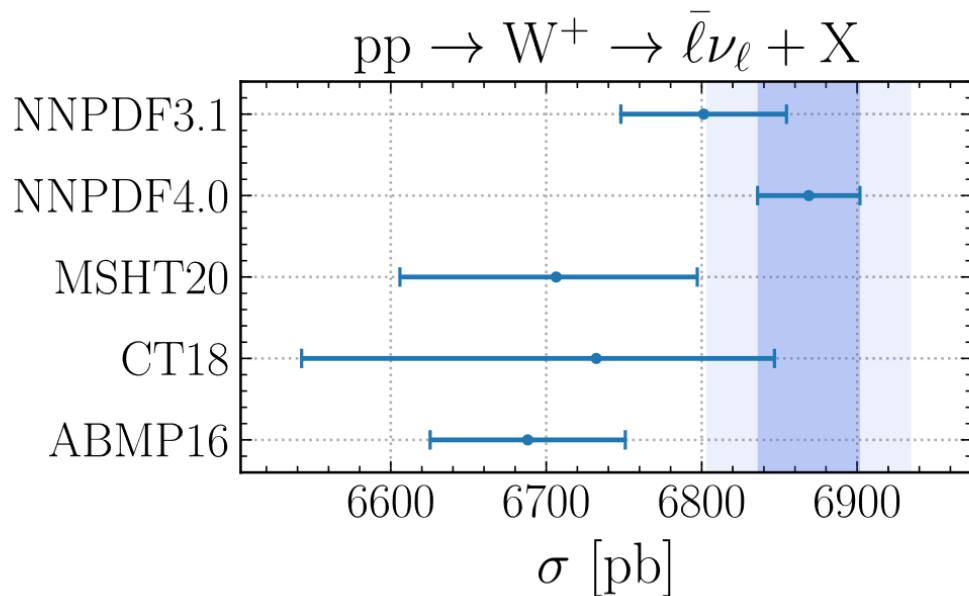
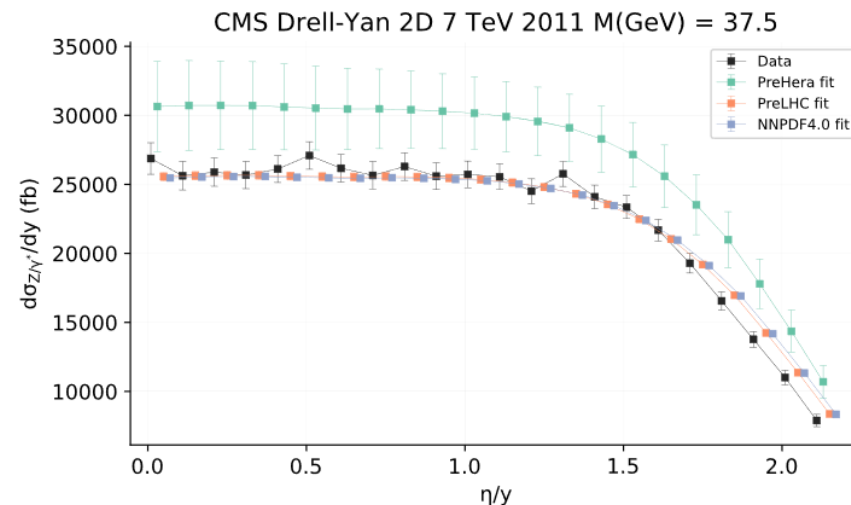
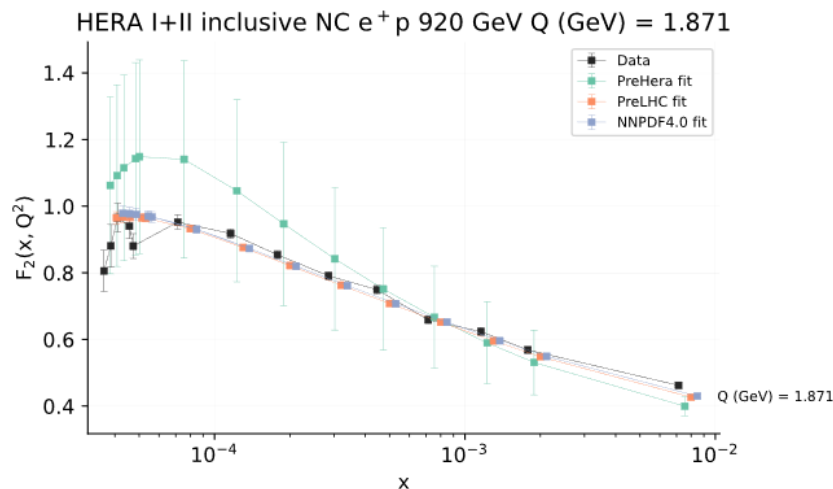
hyperbolic tangent



**Fig. 4** Diagrammatic representation of the calculation of the  $\chi^2$  in the NNPDF fitting framework as a function of the values of  $\{x_n^{(k)}\}$  for the different datasets. Each block indicates an independent component

# NNPDF 4.0 - Results

- Results



# Neural Operators

- Often you don't want to solve just single function

E.g, given different initial field condition, want the evolution of system

$$function1 \rightarrow function2$$

Theorem (informal): neural networks can approximate an operator

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

# Neural Operators

Journal of Machine Learning Research 23 (2022) 1-97

Submitted 12/21; Revised 10/22; Published 12/22

## Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs

Nikola Kovachki<sup>†</sup>

Zongyi Li<sup>\*</sup>

Burigede Liu

Kamyar Azizzadenesheli

Kaushik Bhattacharya

Andrew Stuart

Anima Anandkumar

NKOVACHKI@NVIDIA.COM *Nvidia*

ZONGYILI@CALTECH.EDU *Caltech*

BL377@CAM.AC.UK *Cambridge University*

KAMYARA@NVIDIA.COM *Nvidia*

BHATTA@CALTECH.EDU *Caltech*

ASTUART@CALTECH.EDU *Caltech*

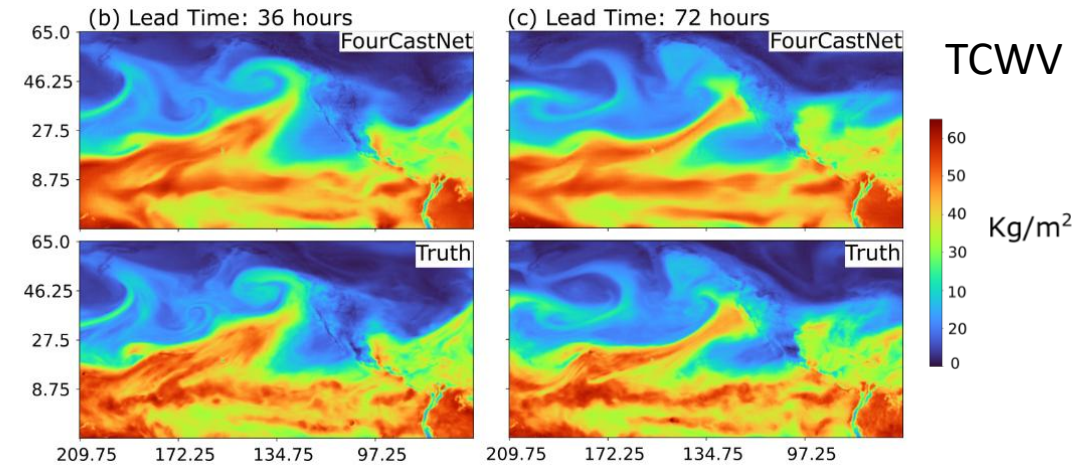
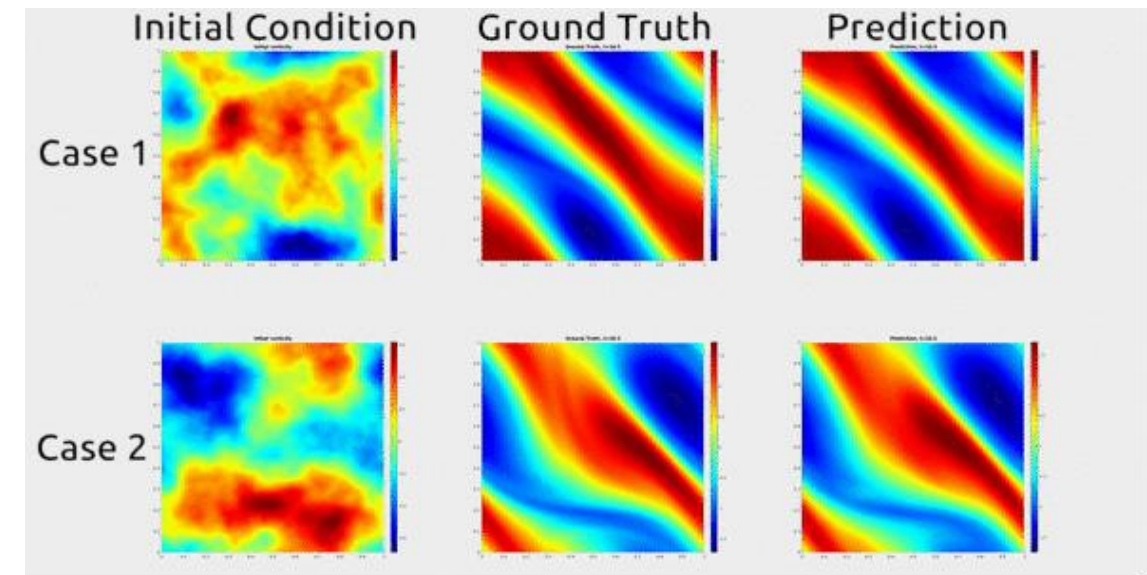
ANIMA@CALTECH.EDU *Caltech*

- Continuous generalization of regular NN's layers (basically)

$$u_{i+1}(x) = \sigma \left( \int dx' \kappa(x, x') u_i(x') + b(x) \right)$$

- Surrogate models: trained on data from simulations

- Often faster than numerical simulations



FourCastNet: A global data-driven high-resolution weather model arXiv: 2202.11214



# Fourier Neural Operators

- FNO fundamental layer, choose translation invariant  $\kappa(x, x')$ :

$$u_{i+1}(x) = \sigma \left( \int dx' \kappa(x - x') u_i(x') + b(x) \right)$$

- For integral:

$$\int dx' \kappa(x - x') u(x') \Rightarrow \sum_{i=1}^N \kappa(x_j - x'_k) u(x'_k) \text{ for each } x_j$$

Convolution theorem: convolution in position space becomes product in Fourier space

$$\mathcal{F}^{-1} \left( \mathcal{F} \left( \sum_{i=1}^N \kappa(x_j - x'_k) u(x'_k) \right) \right) = \mathcal{F}^{-1}(\mathcal{F}(\kappa) \mathcal{F}(u))$$

- Advantage:  $\mathcal{O}(N^2) \rightarrow \mathcal{O}(N \ln N)$  using FFT

# Scattering with NO

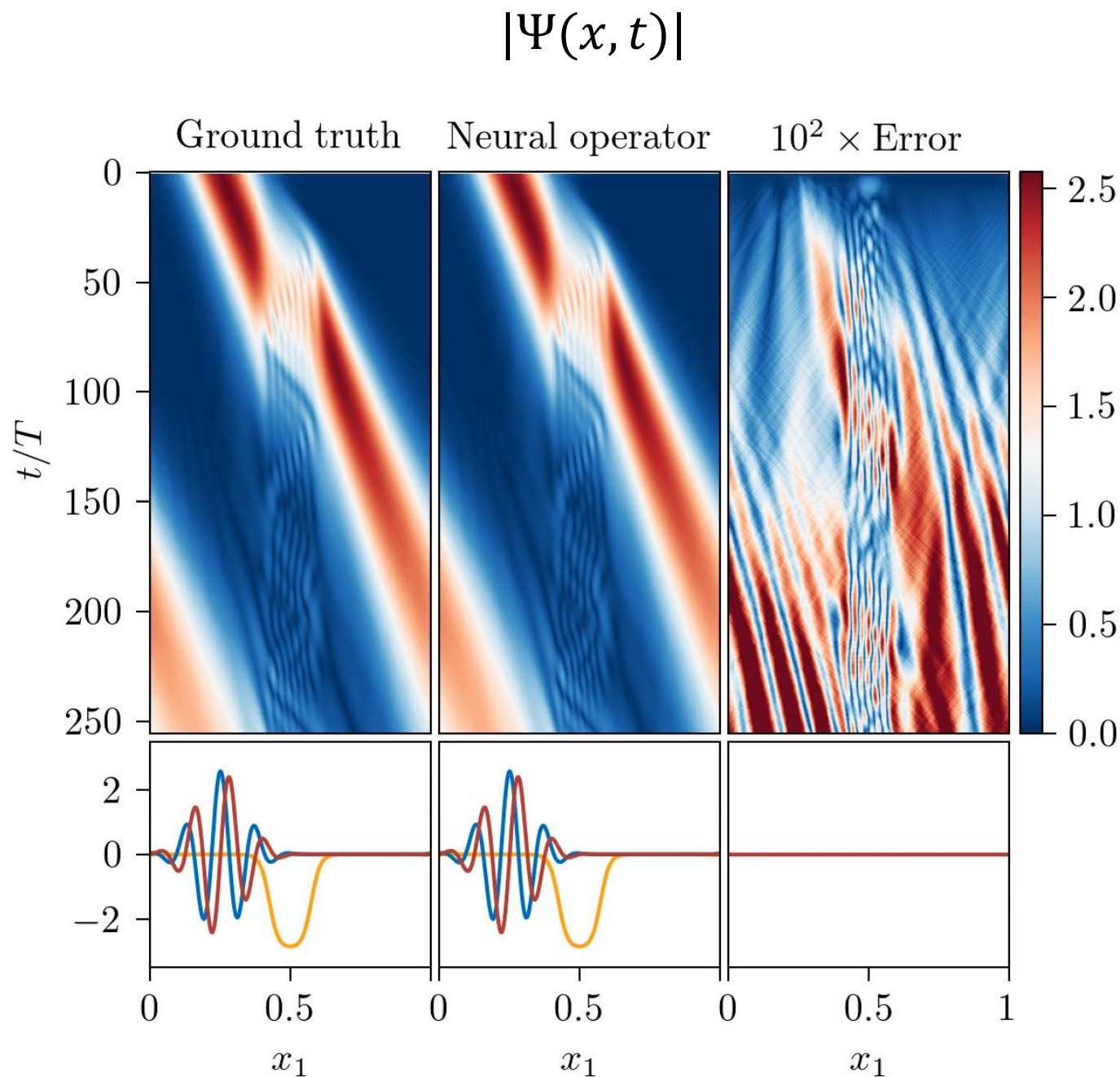
- For non-relativistic scattering:

$$\Psi(\vec{x}, T) = S[V(\vec{x})]\Psi(\vec{x}, 0).$$

$$S[V(\vec{x})] = \mathcal{T}e^{-i/\hbar \int_0^T \hat{H}[V(\vec{x})]dt}$$

$$\Psi(\vec{x}, T) \stackrel{?}{=} \mathcal{N}[V(\vec{x}), \Psi(\vec{x}, 0)].$$

$$\Psi_{\text{NO}}(\vec{x}, kT) = \mathcal{N}^k[V(\vec{x}), \Psi(\vec{x}, 0)],$$



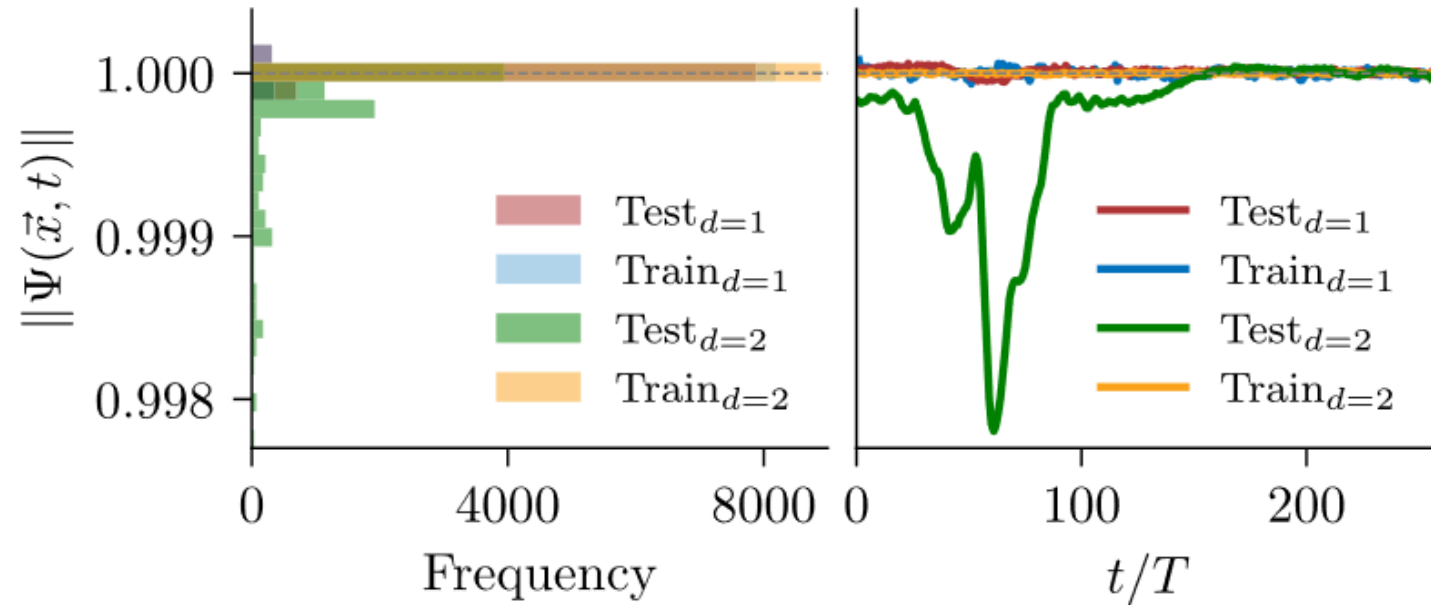


# Scattering with NO

- Faster, less memory intense than standard Crank-Nicholson (CN) method

$d$	$N$	CN time	CN memory	NO time	NO memory	Error
<b>1</b>	<b>256</b>	0.013	0.004	0.002	0.13	0.0005
1	2048	0.2	0.3	0.003	0.14	0.004
1	16384	56	20	0.003	0.2	0.004
1	32768	$\times$	>40	0.003	0.3	$\times$
<b>2</b>	<b>64</b>	0.5	0.6	0.005	0.16	0.0011
2	128	22	10	0.005	0.2	0.03
2	256	$\times$	>40	0.005	0.4	$\times$

- Unitarity mostly obeyed

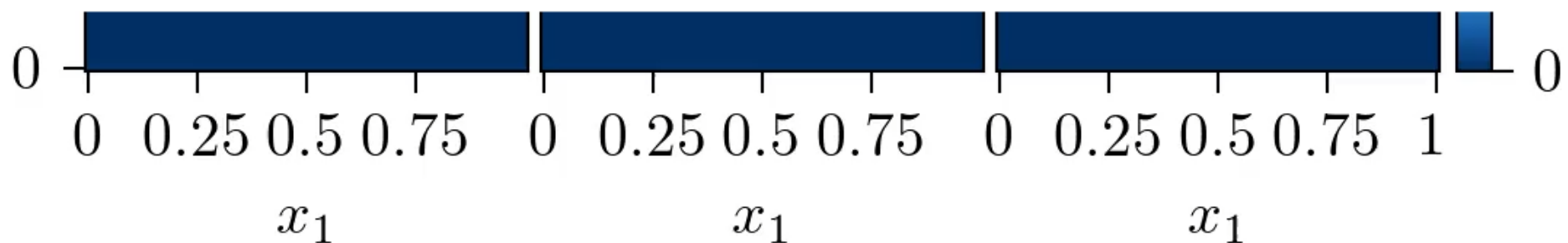


# Scattering with NO

- 2D Double Slit Experiment

$$|\Psi(\boldsymbol{x}, t)|$$

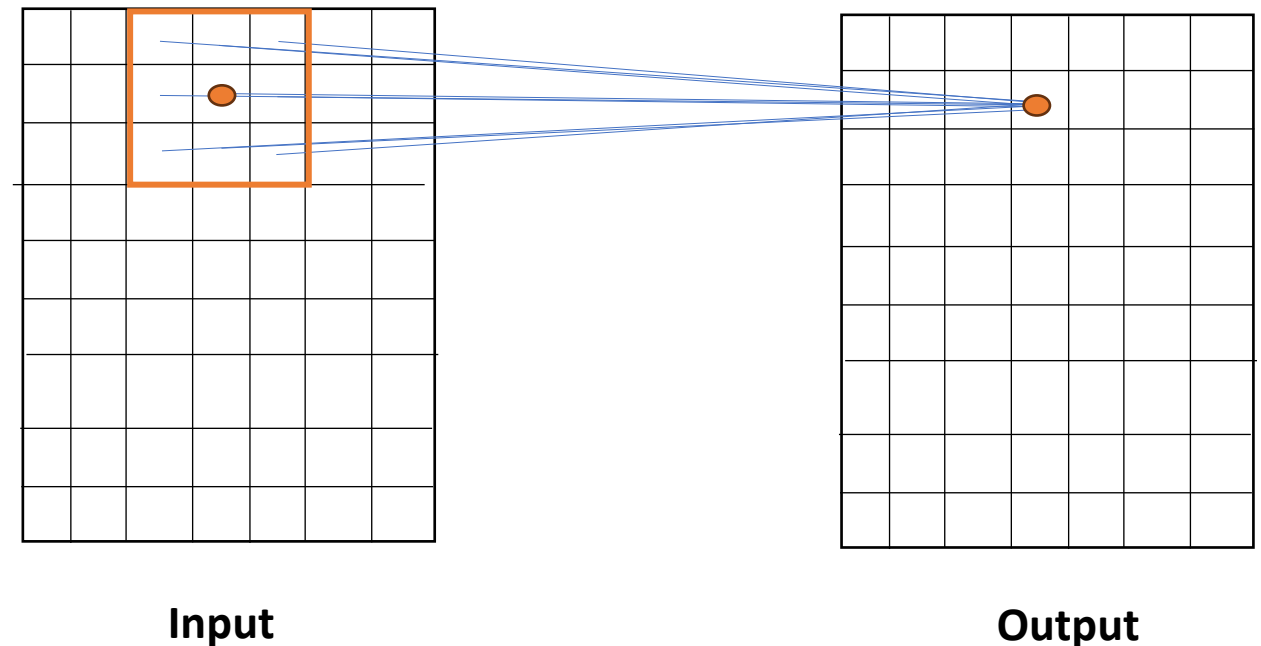
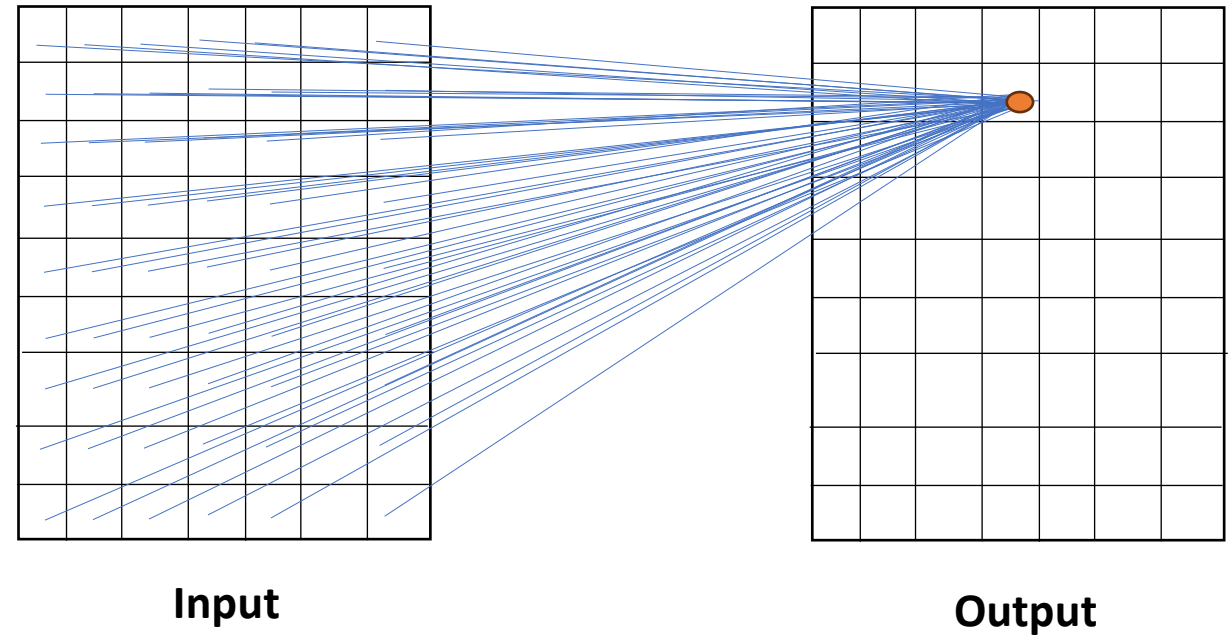
errors. For example, given that neural operators represent physics in unconventional ways, would cheaply obtainable training data (e.g., coming from exactly solvable systems) suffice to learn solutions to conventionally difficult problems? We leave a study of this provocative question until future work.



# Classification

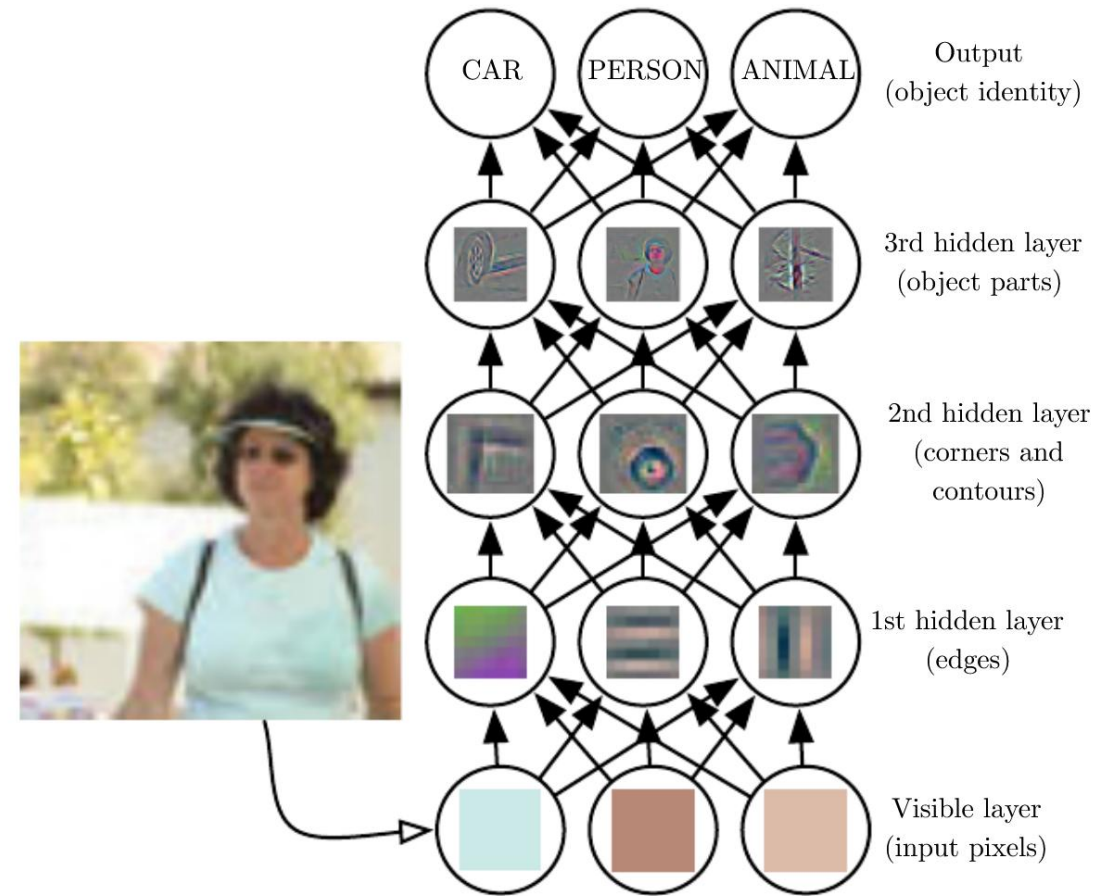
# Convolutional Neural Network

- For  $128 \times 128$  image  
 $N \sim 10^4$  pixels
- For one dense layer,  
there are  $N^2 \sim 10^8$   
weight parameters
- CNN: only non-zero  
weights around point
- Weights sharing:  
same for every point  
 $10^8 \rightarrow 9$



# Convolutional Neural Network

- Build up complexity



Zeiler and Fergus (2014).

# Deep learning in color

- Difficult to distinguish light quark and gluon jets
- Jet “images” : intensity at local calorimeter deposits

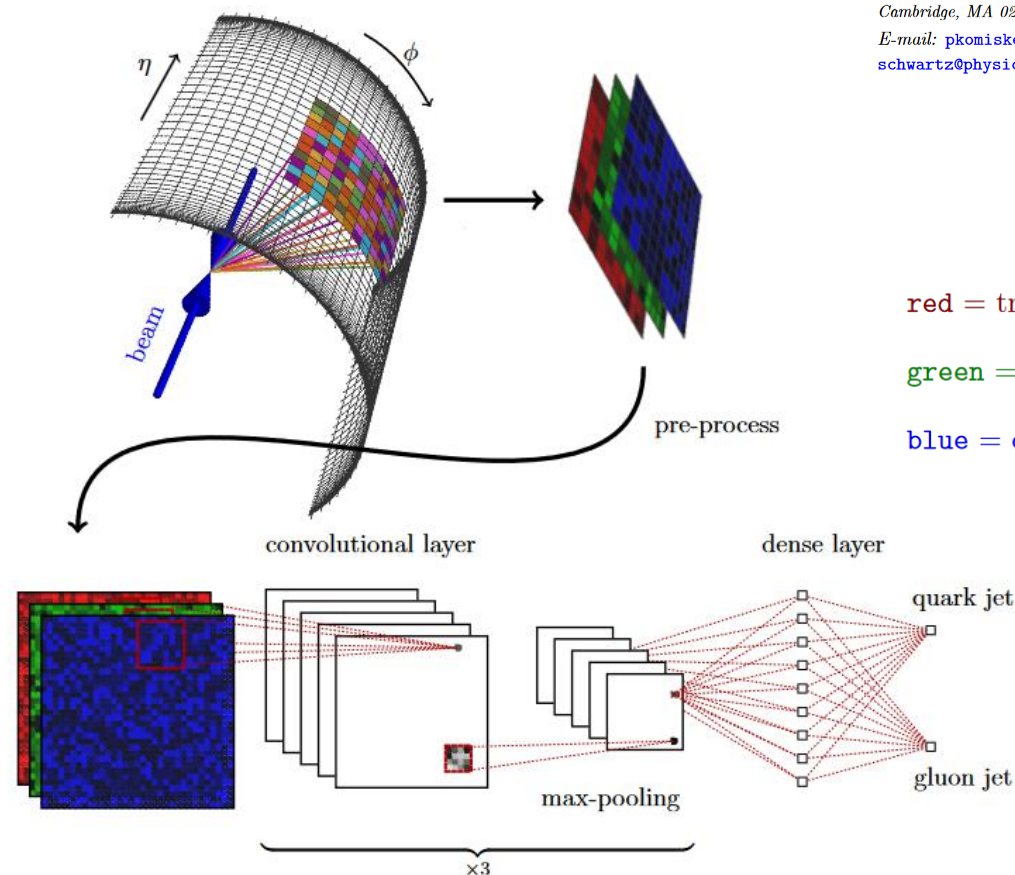
## Deep learning in color: towards automated quark/gluon jet discrimination

Patrick T. Komiske,<sup>a</sup> Eric M. Metodiev<sup>a</sup> and Matthew D. Schwartz<sup>b</sup>

<sup>a</sup>Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

<sup>b</sup>Department of Physics, Harvard University, Cambridge, MA 02138, U.S.A.

E-mail: [pkomiske@mit.edu](mailto:pkomiske@mit.edu), [metodiev@mit.edu](mailto:metodiev@mit.edu), [schwartz@physics.harvard.edu](mailto:schwartz@physics.harvard.edu)



red = transverse momenta of charged particles

green = the transverse momenta of neutral particles

blue = charged particle multiplicity

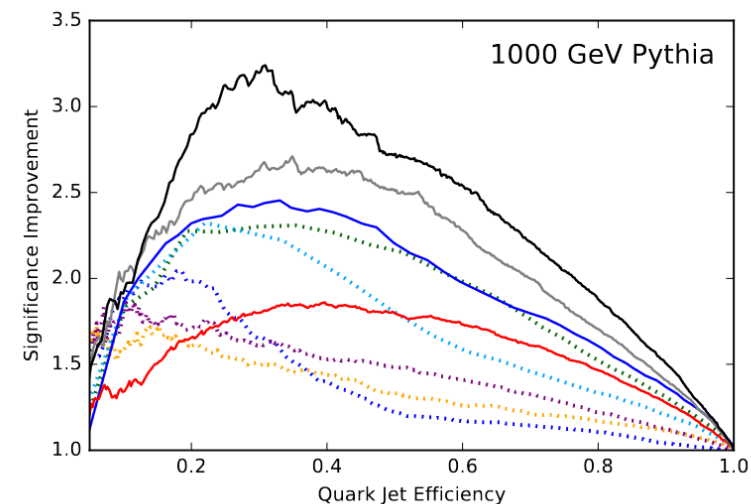
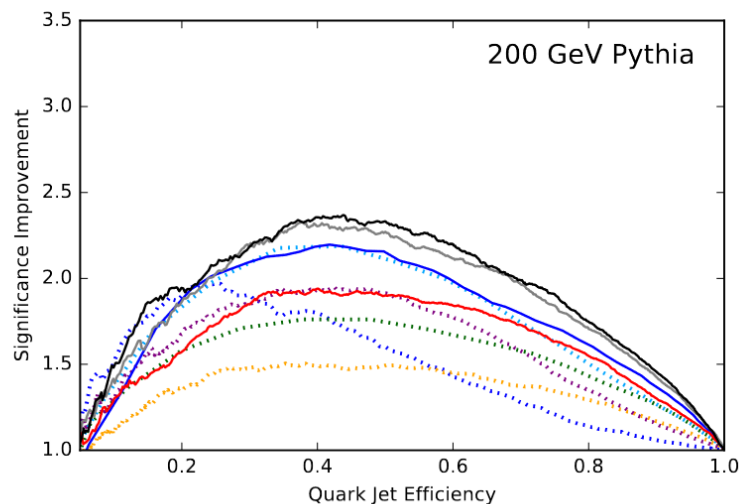
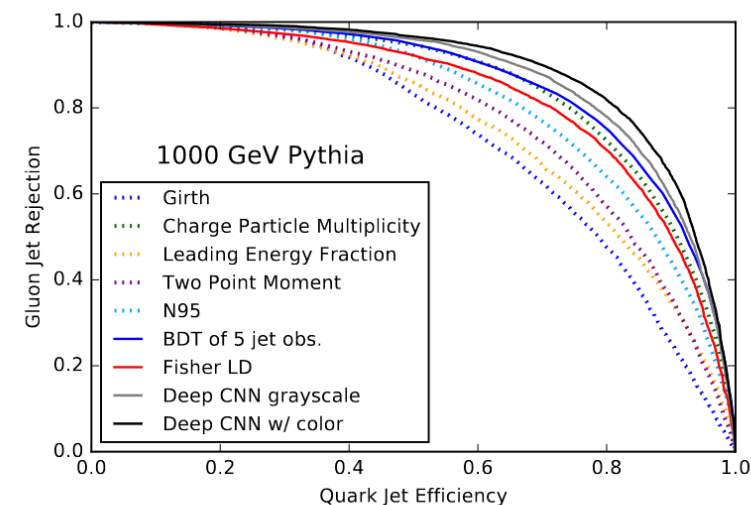
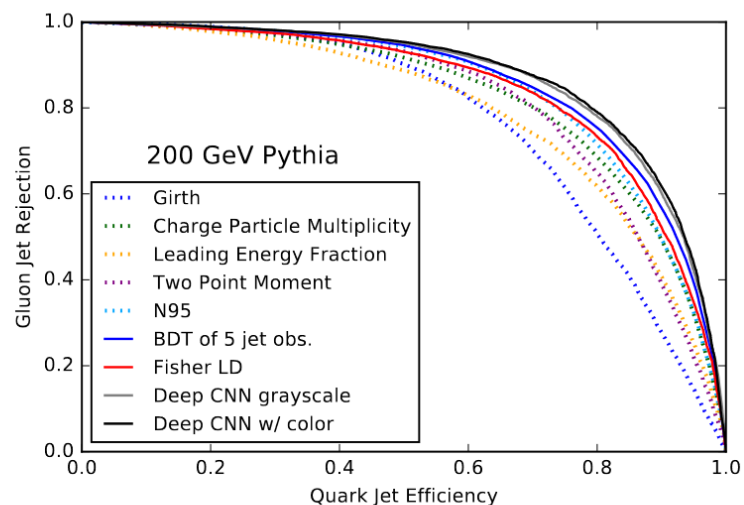
# Data

- $pp$  at  $\sqrt{s} = 13 \text{ TeV}$
- 90 k training (10% val.)
- 10k testing

Range [GeV]	Events	Simulation
100 – 110	100 000	Pythia
200 – 220	100 000	Pythia + Herwig
500 – 550	100 000	Pythia
1000 – 1100	100 000	Pythia + Herwig

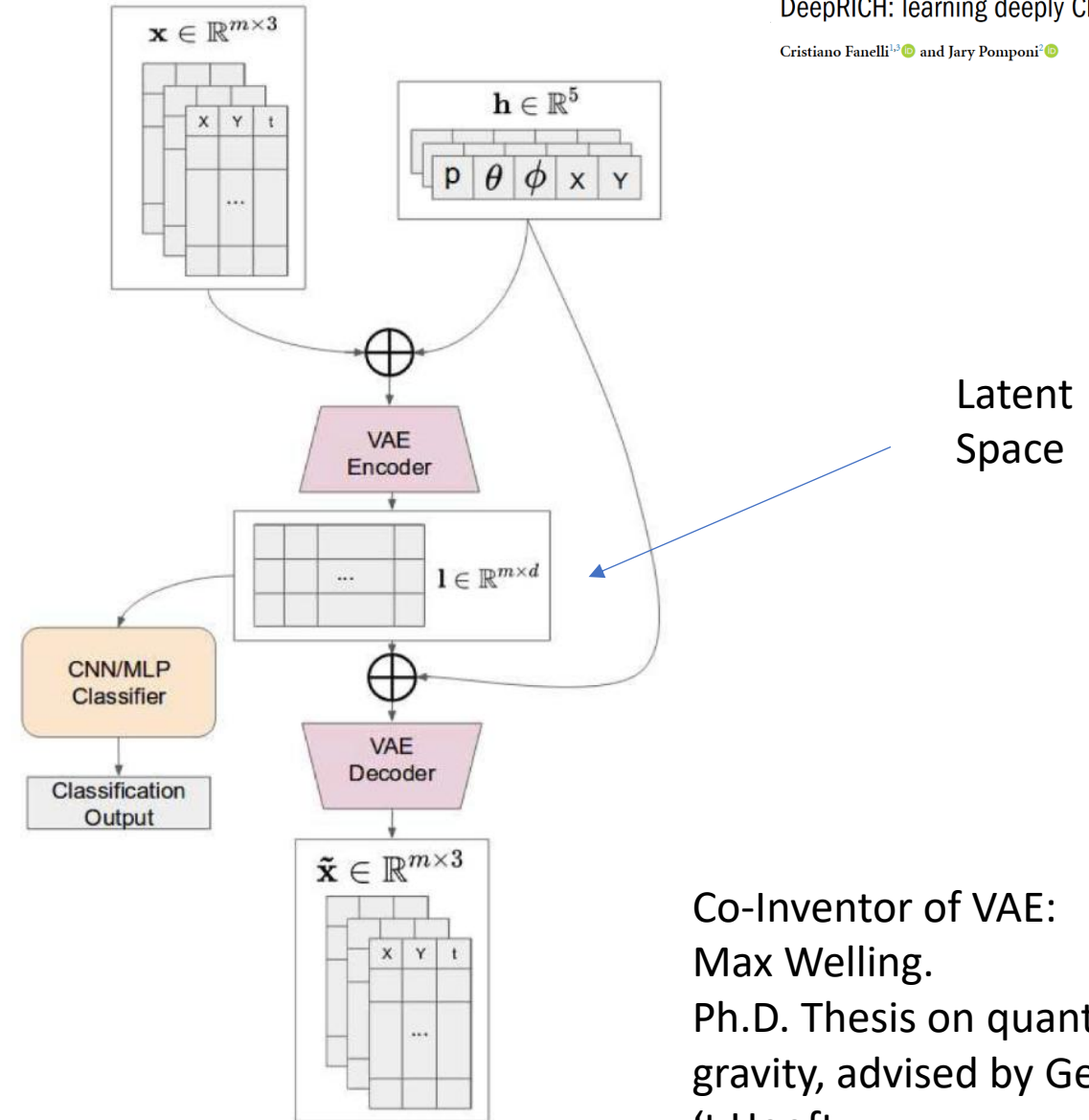
# Deep learning in color

- Matches or outperforms traditional variables
- Insensitive to whether Pythia/Herwig





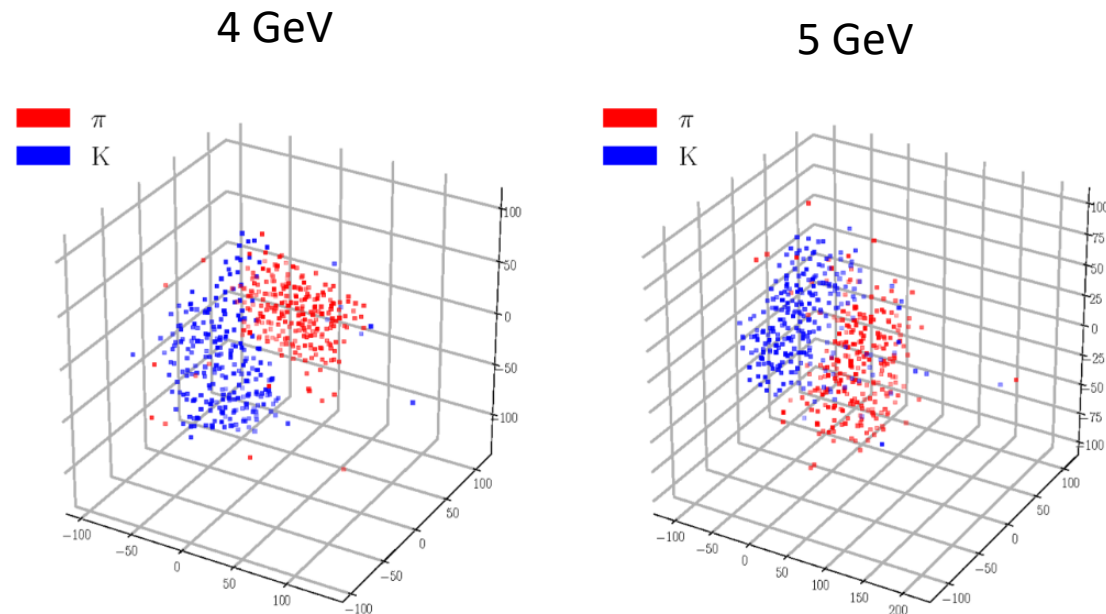
- For deflection of internally reflecting Cherenkov (DIRC) detector at GlueX
- Full simulation slow
- Variational autoencoder (VAE) forces data to lower dimensional latent space
- Classifier uses latent space to separate  $K$  &  $\pi$ 's



Co-Inventor of VAE:  
Max Welling.  
Ph.D. Thesis on quantum  
gravity, advised by Gerard  
't Hooft

# DeepRich

- Latent space



3D-visualization obtained from t-SNE

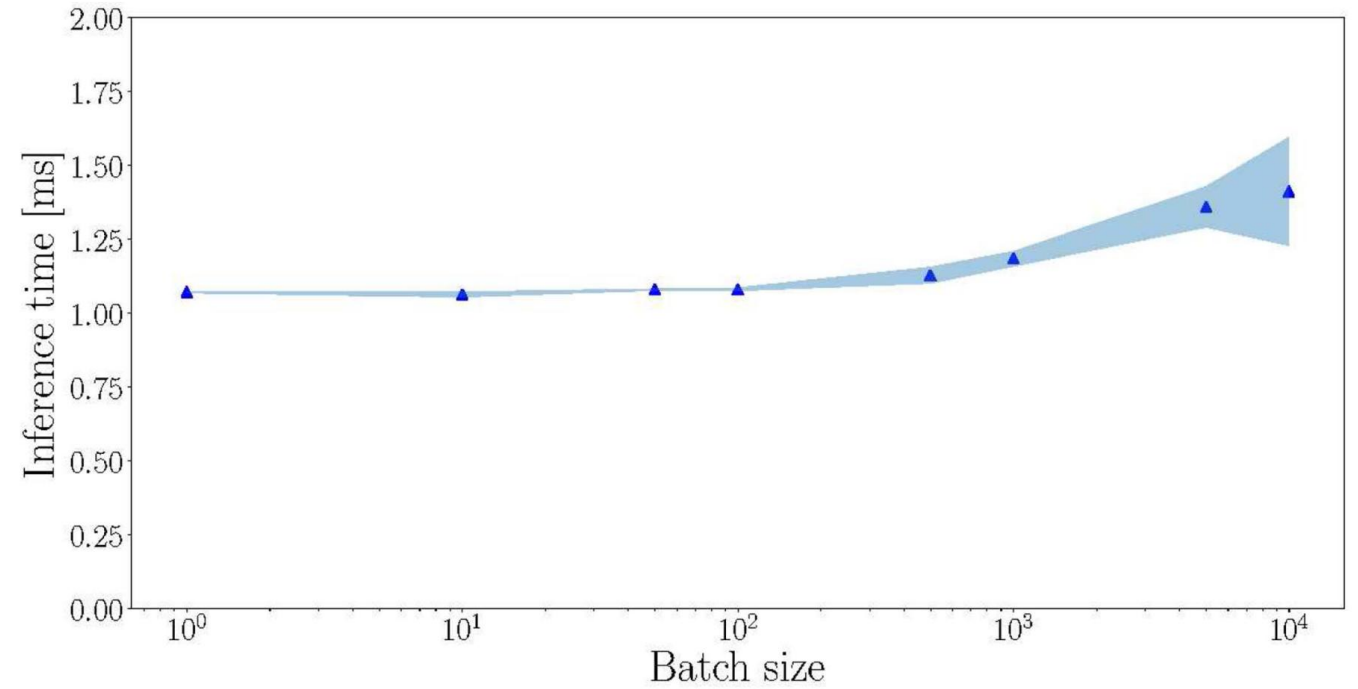
- Classification ability comparable to FastD

**Table 3.** The area under curve (%), the signal efficiency to detect pions  $\varepsilon_S$  and the background rejection of kaons  $\varepsilon_B$  corresponding to the point of the ROC that maximizes the product  $\varepsilon_S \cdot \varepsilon_B$ . The corresponding momenta at which these values have been calculated are also reported. This table is obtained by integrating over all the other kinematic parameters (i.e. a total of  $\sim 6k$  points with different  $\theta, \phi, X, Y$  for each momentum).

Kinematics	DeepRICH			FastDIRC		
	AUC	$\varepsilon_S$	$\varepsilon_B$	AUC	$\varepsilon_S$	$\varepsilon_B$
4 GeV/c	99.74	98.18	98.16	99.88	98.98	98.85
4.5 GeV/c	98.78	95.21	95.21	99.22	96.33	96.32
5 GeV/c	96.64	91.13	91.23	97.41	92.40	92.47

# DeepRich

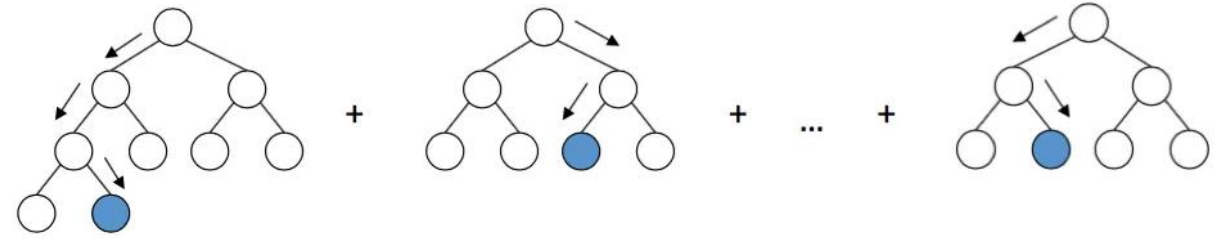
- Advantage: speed.
- Uses parallelism of GPU
- Can do  $10^4$  particles in  $\sim 1.4$  ms



# Boosted Decision Trees

**NOT** a neural network or deep learning approach

- Group of weak learning better than individual
- Ensemble of decision trees used in astrophysics as well (paper: random forest instead of BDT)



**The quenching of galaxies, bulges, and disks since cosmic noon**

**A machine learning approach for identifying causality in astronomical data**

Asa F. L. Bluck<sup>1,2,3</sup>, Roberto Maiolino<sup>1,2</sup>, Simcha Brownson<sup>1,2</sup>, Christopher J. Conselice<sup>4</sup>, Sara L. Ellison<sup>5</sup>, Joanna M. Piotrowska<sup>1,2</sup>, and Mallory D. Thorp<sup>5</sup>

- Developed for [Kaggle's Higgs Boson Challenge](#)
- Separate signal  $H \rightarrow \bar{\tau}\tau$  from background
- Features: # jets, derived mass, the  $\eta, \phi, E_T^{miss}$  of various particles, etc.
- Won “ML meet HEP” Award:  
*“an excellent compromise between performance and simplicity”*

## XGBoost: A Scalable Tree Boosting System

Tianqi Chen  
University of Washington  
tqchen@cs.washington.edu

Carlos Guestrin  
University of Washington  
guestrin@cs.washington.edu

[Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.](#)

Citations: ~53,000

The Higgs boson machine learning challenge

[JMLR: Workshop and Conference Proceedings 42:19-55, 2015](#)

“Many successful solutions are based on the XGBoost implementation of boosted decision trees”

# Boosted Decision Trees

Compared to deep learning methods:

- About as good for tabular data
- Faster to train
- Better interpretation

When data has locality or order (e.g. image or time series) not as competitive

## XGBoost is All You Need

What do I really mean by my (in)famous tagline, Part 1



BOJAN TUNGUZ  
JAN 20, 2025



49



7



1

Share

# XGBoost

# Classification

At  $> 4.9$  GeV, pions as well as electrons and positrons produce signal in High Threshold Cherenkov Counter at CLAS12, JLAB

- Used Boosted Decision Trees in electron/positron identification
- Performed better than neural network

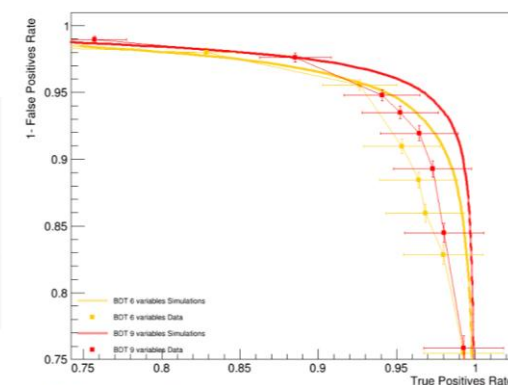
## Enhancing Lepton Identification in CLAS12 using Machine Learning Techniques

Mariana Tenorio-Pita, ODU

### Results

BDT 9/ Spring 2019	Data	Simulation
Signal	97.30%	99.36%
Background	10.73%	10.70%

*By having a cut in the response at -0.06, we observe that we are keeping most of the data while suppressing most of the background.*



*ROC curve of validation on simulation (solid) and data (dots) show consistency with each other.*

# Generation



# Generation

- We often need to generate samples

Examples:

- Monte Carlo for experimental analysis
- Lattice QCD: samples for calculations

Problem: these are often computationally expensive, need to generate many

# Diffusion

- Took ideas from non-equilibrium thermodynamics
- Add Noise to image

$$\mathbf{z}_t = \sqrt{1 - \beta_t} \mathbf{z}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t \quad (1)$$

$$\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, I)$$

Asymptotically:

$$\lim_{t \rightarrow \infty} \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, I)$$

---

## Deep Unsupervised Learning using Nonequilibrium Thermodynamics

---

Jascha Sohl-Dickstein  
Stanford University

JASCHA@STANFORD.EDU

Eric A. Weiss  
University of California, Berkeley

EWEISS@BERKELEY.EDU

Niru Maheswaranathan  
Stanford University

NIRUM@STANFORD.EDU

Surya Ganguli  
Stanford University

SGANGULI@STANFORD.EDU

[International conference on machine learning, 2015](#)

**Exercise:** From Eq. (1) and assuming  $\beta_i \in (0,1)$ , show

$$\mathbf{z}_t = \sqrt{\alpha_t} \mathbf{z}_0 + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_t$$

where:

$$\alpha_t = \prod_{i=1}^t (1 - \beta_i)$$

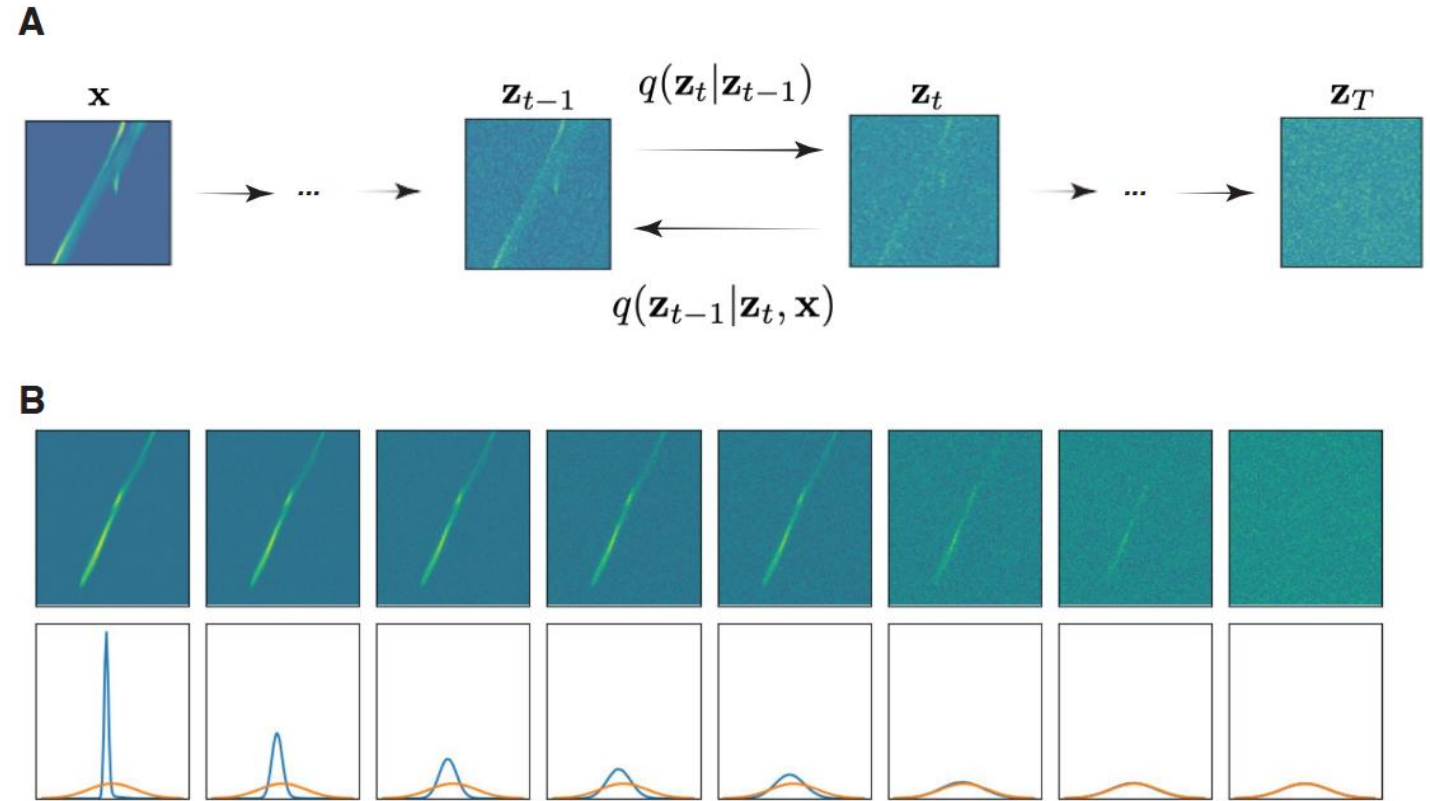
and, thus, the asymptotic limit  $\mathbf{z}_t \rightarrow \mathcal{N}(\mathbf{0}, I)$  holds.

Also,

$$\begin{aligned} & \mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{z}_0 \\ &= \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \sqrt{1 - \beta_t} \mathbf{z}_t + \sqrt{\frac{\beta_t (1 - \alpha_{t-1})}{1 - \alpha_t}} \boldsymbol{\epsilon}_t \end{aligned}$$

# Diffusion

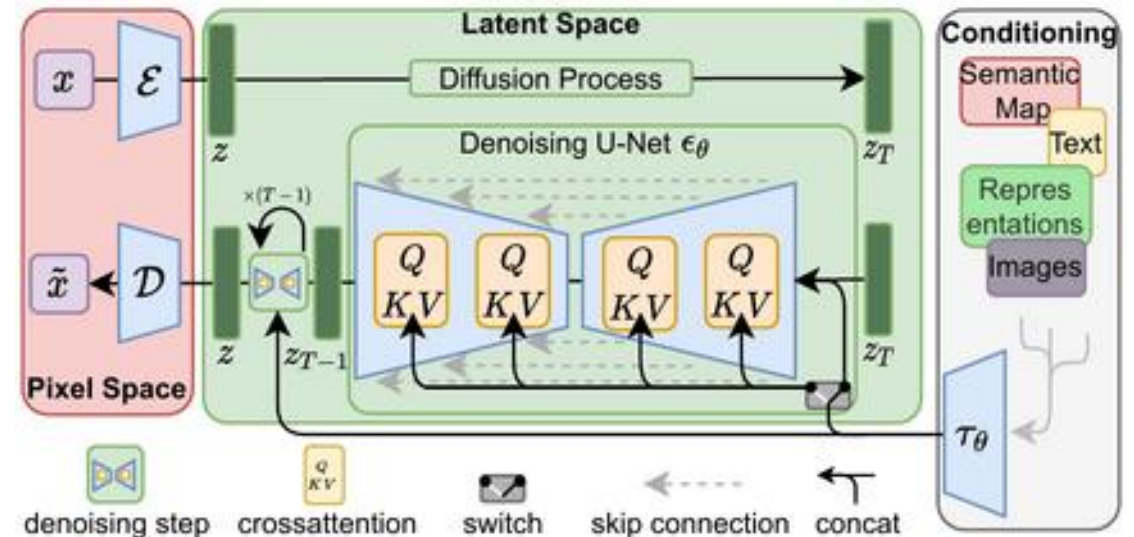
- Forward process: ‘adds’ noise sequentially
- Train neural network to ‘denoise’ image’, one step at a time.
- Distributions analytic since made from Gaussians



[Scheinker \(2024\)](#)

# Diffusion

- To generate image, start with Gaussian noise, then denoise
- Images high dimensional, often performed in “latent space” (lower dimensional representation of data)
- Can generate high quality ‘images’,



# Diffusion

- For EIC or JLab

- “Image”:  $\phi$  and  $\eta$  index

- “Intensity”: related to

$$z_i = \frac{2 p_{T,i}}{\sqrt{s}} \cos \eta_i$$

with massless approx. used.

$$\sum_i z_i^{meas.} \leq 2$$

- Channels:  $PID_i$ ,  
 $e^-, \pi^+, K^+$

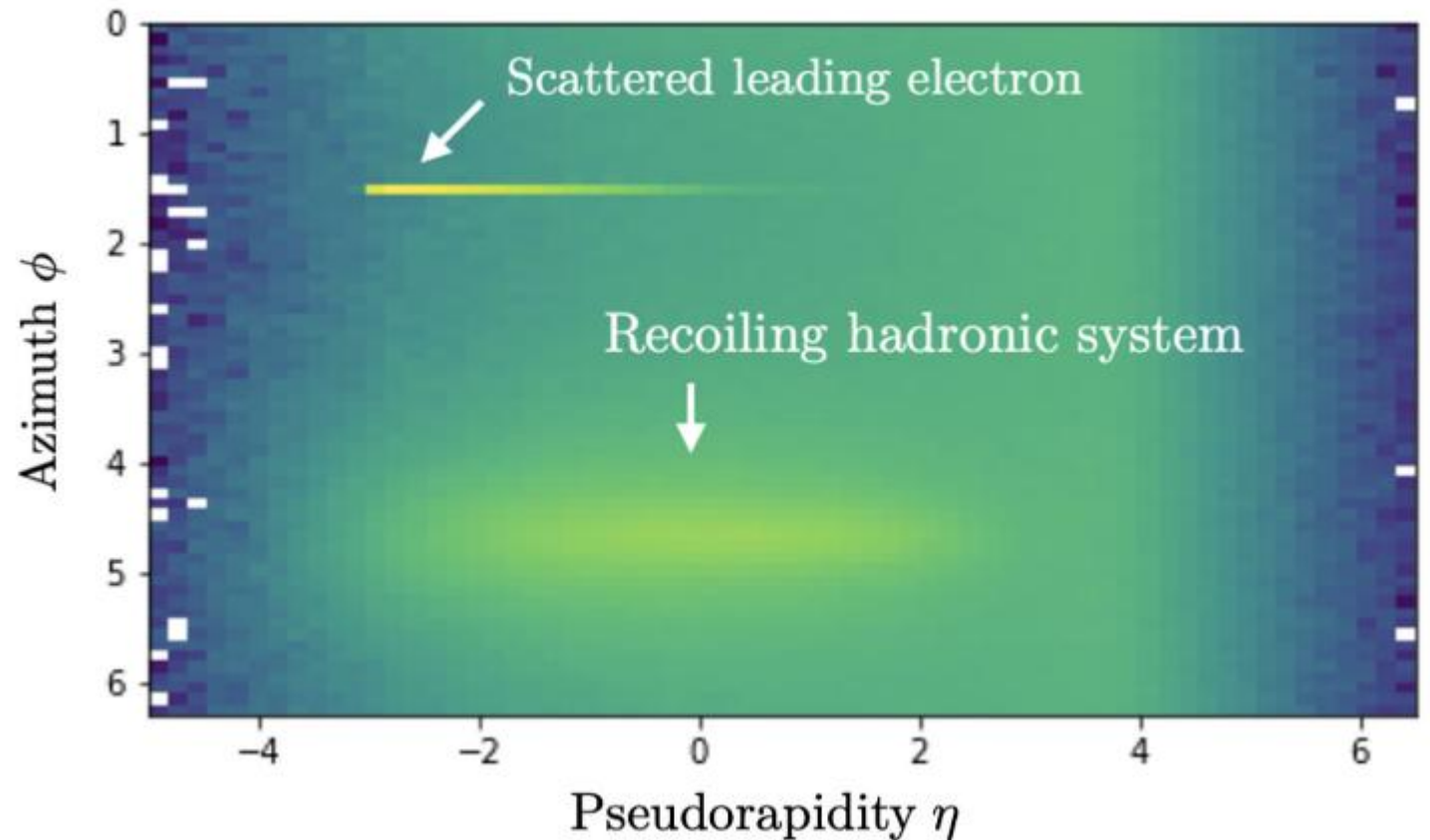
## Diffusion model approach to simulating electron-proton scattering events

Peter Devlin,<sup>1,\*</sup> Jian-Wei Qiu<sup>1,†</sup>, Felix Ringer<sup>1,2,‡</sup> and Nobuo Sato<sup>1,§</sup>

<sup>1</sup>Thomas Jefferson National Accelerator Facility, Newport News, Virginia 23606, USA

<sup>2</sup>Department of Physics, Old Dominion University, Norfolk, Virginia 23529, USA

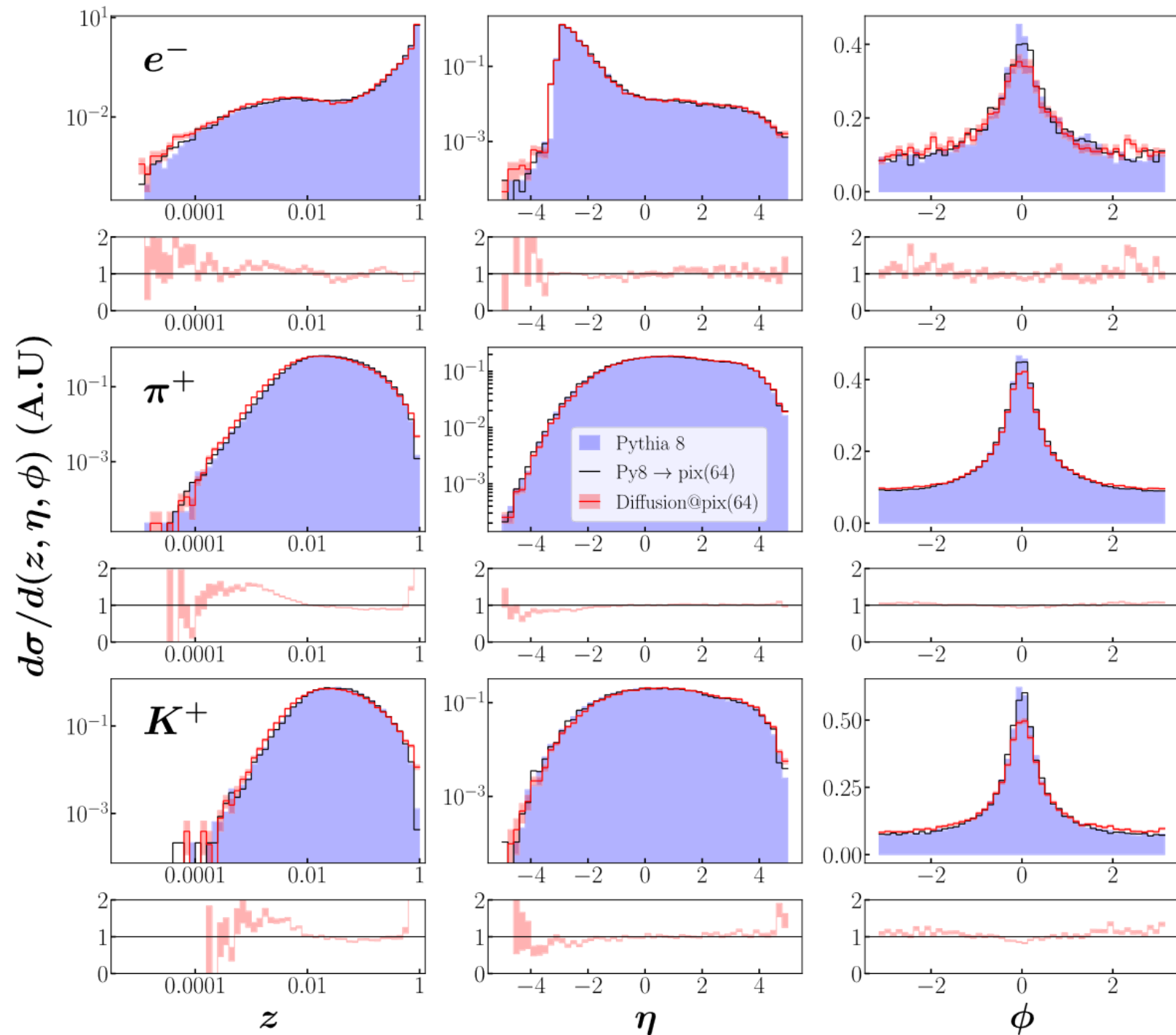
(Received 8 November 2023; accepted 27 June 2024; published 31 July 2024)



# Diffusion

- Results

- $Py8 \rightarrow pix(64)$ :  
pixelation





# Diffusion

- All generated events had:

$$\sum_i z_i^{meas} \leq 2$$

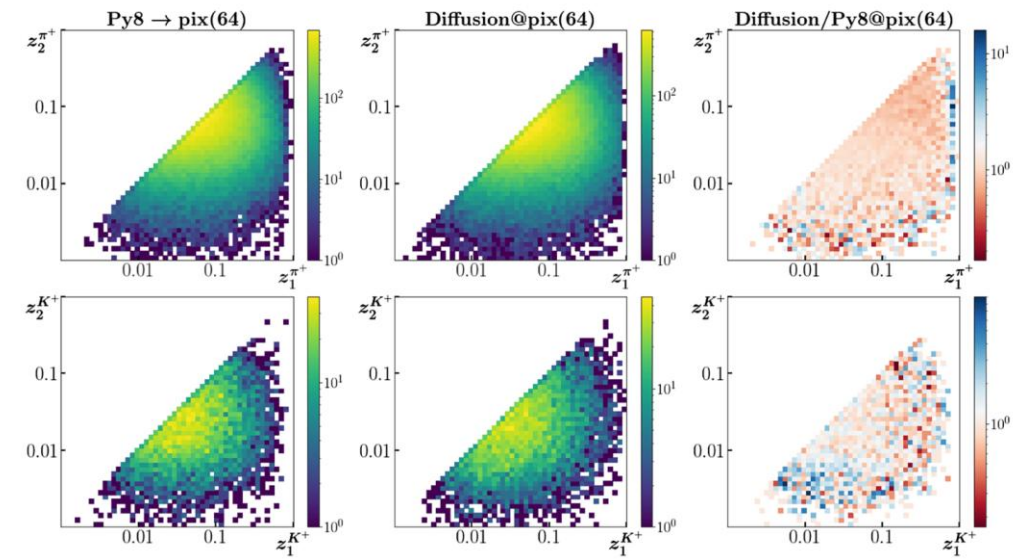
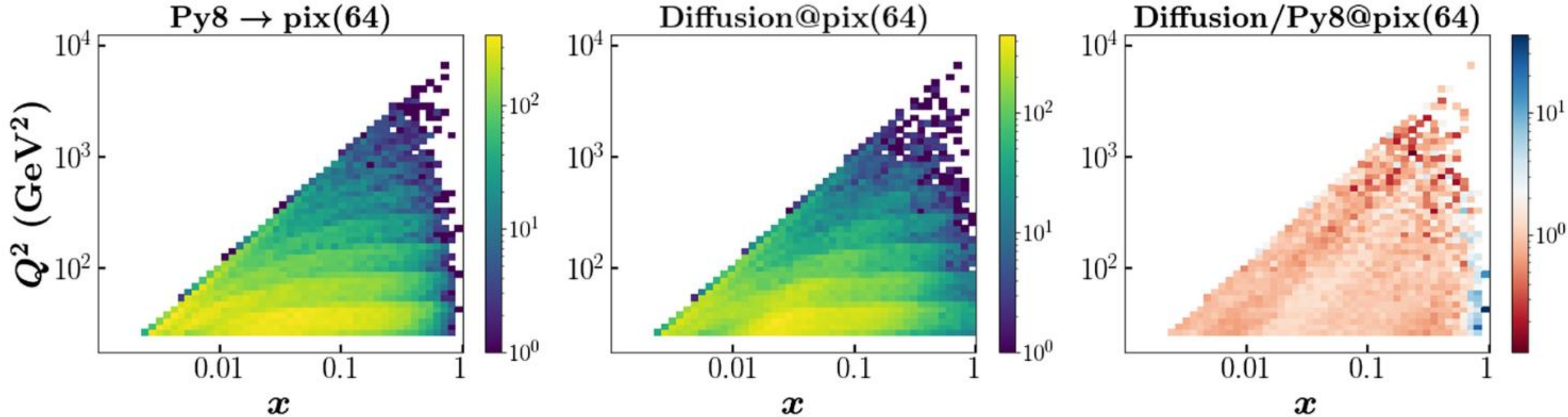


FIG. 8. Dihadron correlations comparing Pythia8 (left), the diffusion model (middle), and ratio to Pythia8 (right): leading and subleading pions  $\pi^+$  (upper row) and kaons  $K^+$  (lower row).



# Chores



# Currents Trends

- A lot of open-source code to train on
- LLM's have improved in coding

## The New York Times

### Powerful A.I. Is Coming. We're Not Ready.

"If you really want to grasp how much better A.I. has gotten recently, talk to a programmer. A year or two ago, A.I. coding tools existed, but were aimed more at speeding up human coders than at replacing them.

**Today, software engineers tell me that A.I. does most of the actual coding for them, and that they increasingly feel that their job is to supervise the A.I. systems.**

Jared Friedman, a partner at Y Combinator, a start-up accelerator, recently said **a quarter of the accelerator's current batch of start-ups were using A.I. to write nearly all their code.** "A year ago, they would've built their product from scratch — but now 95 percent of it is built by an A.I.," he said."

<https://www.nytimes.com/2025/03/14/technology/why-im-feeling-the-agi.html>

## Comparing Large Language Models and Human Programmers for Generating Programming Code

Wenpin Hou\* and Zhichenf

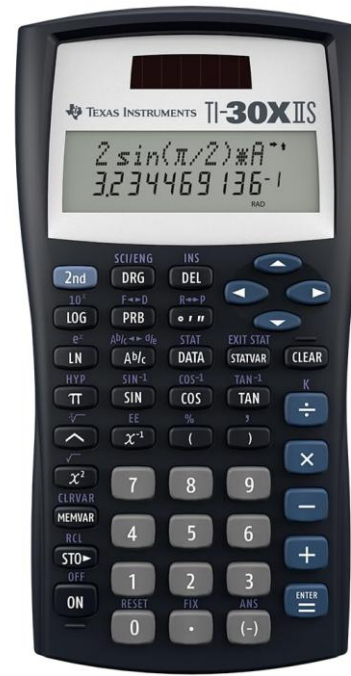


The performance of seven LLMs in generating programming code using various prompts and task difficulties is systematically evaluated. GPT-4 consistently outperforms other LLMs, with its performance varying across different task difficulties. GPT-4, employing the optimal prompt strategy, performs as well as most human participants in a contest. The study demonstrates strong capabilities of LLMs in generating programming languages and code, and the efficiency of the code generated by LLMs compared to human programmers. GPT-4 is also evaluated in generating code for front-end design, demonstrating that GPT-4 has the potential to be used for code generation and software development designed based on an optimal prompt strategy for LLMs for programming.

“These results suggest that **GPT-4 possesses superior programming abilities compared to most human programmers** participating in these contests, yet it still **falls short of surpassing the most elite human programmers** in most situations.”

# Historical Precedent

- Calculators, computers refocused what humans do
- Human in loop is still required



# Code Commenting

```
class AttentionBlock(layers.Layer):
    """Applies self-attention.

    Args:
        units: Number of units in the dense layers
        groups: Number of groups to be used for GroupNormalization layer
    """

    def __init__(self, units, groups=8, **kwargs):
        self.units = units
        self.groups = groups
        super().__init__(**kwargs)

        self.norm = layers.GroupNormalization(groups=groups)
        self.query = layers.Dense(units, kernel_initializer=kernel_init(1.0))
        self.key = layers.Dense(units, kernel_initializer=kernel_init(1.0))
        self.value = layers.Dense(units, kernel_initializer=kernel_init(1.0))
        self.proj = layers.Dense(units, kernel_initializer=kernel_init(0.0))

    def call(self, inputs):
        batch_size = tf.shape(inputs)[0]
        height = tf.shape(inputs)[1]
        width = tf.shape(inputs)[2]
        scale = tf.cast(self.units, tf.float32) ** (-0.5)

        inputs = self.norm(inputs)
        q = self.query(inputs)
        k = self.key(inputs)
        v = self.value(inputs)

        attn_score = tf.einsum("bhw, bhw->bhw", q, k) * scale
        attn_score = tf.reshape(attn_score, [batch_size, height, width, height * width])

        attn_score = tf.nn.softmax(attn_score, -1)
        attn_score = tf.reshape(attn_score, [batch_size, height, width, height, width])

        proj = tf.einsum("bhw,bhw->bhw", attn_score, v)
        proj = self.proj(proj)
        return inputs + proj
```

<https://github.com/keras-team/keras-io/blob/master/examples/generative/ddpm.py>

Asked LLM  
to document



```
class AttentionBlock(layers.Layer):
    """Applies self-attention.

    Args:
        units: Number of units in the dense layers.
        groups: Number of groups to be used for GroupNormalization layer.
    """

    def __init__(self, units, groups=8, **kwargs):
        # Initialize the class with specified parameters for units and group
        self.units = units
        self.groups = groups
        super().__init__(**kwargs)

        # GroupNormalization layer to normalize input tensor by groups.
        self.norm = layers.GroupNormalization(groups=groups)
        # Dense layers for creating query, key, and value matrices with cust
        self.query = layers.Dense(units, kernel_initializer=kernel_init(1.0))
        self.key = layers.Dense(units, kernel_initializer=kernel_init(1.0))
        self.value = layers.Dense(units, kernel_initializer=kernel_init(1.0))
        # Dense projection layer with kernel initializer set to zero.
        self.proj = layers.Dense(units, kernel_initializer=kernel_init(0.0))

    def call(self, inputs):
        # Extract batch size, height, and width from the input tensor.
        batch_size = tf.shape(inputs)[0]
        height = tf.shape(inputs)[1]
        width = tf.shape(inputs)[2]
        # Compute scale factor for the attention scores, inversely proportio
        scale = tf.cast(self.units, tf.float32) ** (-0.5)

        # Apply normalization to the input tensor.
        inputs = self.norm(inputs)
        # Generate query, key, and value matrices using dense layers.
        q = self.query(inputs)
        k = self.key(inputs)
        v = self.value(inputs)

        # Compute attention scores using Einstein summation for matrix multi
        attn_score = tf.einsum("bhw, bhw->bhw", q, k) * scale
        # Reshape attention scores to collapse the spatial dimensions into a
        attn_score = tf.reshape(attn_score, [batch_size, height, width, height * width])

        # Apply softmax to normalize attention scores across spatial dimensi
        attn_score = tf.nn.softmax(attn_score, -1)
        # Reshape back to split spatial dimensions after applying softmax.
        attn_score = tf.reshape(attn_score, [batch_size, height, width, height, width])

        # Compute the weighted sum of value matrices using the attention sco
        proj = tf.einsum("bhw,bhw->bhw", attn_score, v)
        # Pass the resulting projection through the projection dense layer.
        proj = self.proj(proj)
        # Return the residual connection by adding the input to the output c
        return inputs + proj
```

# Other Applications

- Translating Code
- Error handling
- Code optimization
- Improve prose writing

[1663 / Translate Fortran to C++ with AI and RAG](#)

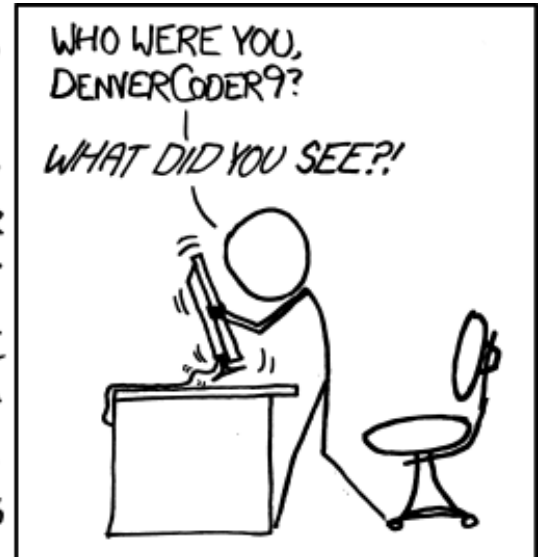
## **Translate Fortran to C++ with AI and RAG**

KYLE DICKMAN



Scientists are using artificial intelligence and large language models to rewrite old code in modern languages.

NEVER HAVE I FELT SO  
CLOSE TO ANOTHER SOUL  
AND YET SO HELPLESSLY ALONE  
AS WHEN I GOOGLE AN ERROR  
AND THERE'S ONE RESULT  
A THREAD BY SOMEONE  
WITH THE SAME PROBLEM  
AND NO ANSWER  
LAST POSTED TO IN 2003



# Note Translation

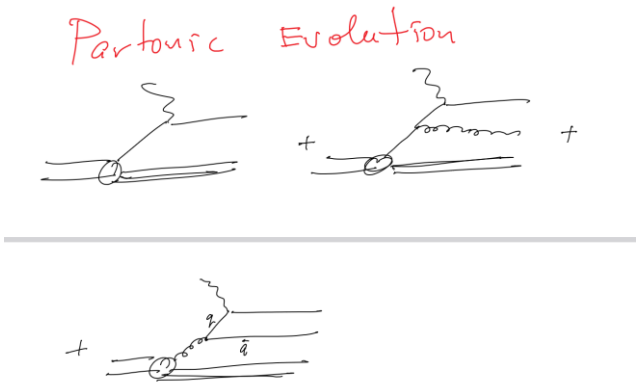
- Test: hand-written notes to Latex code with GPT-4o
- Sargsian's 2017 pQCD course
- Frank had better luck with Gemini

Lecture12  
Monday, November 13, 2017 3:05 PM  
Quarks Within Proton and Neutron

$$F_2(x) = \sum e_i^2 x f_i(x)$$
$$F_1(x) = \frac{1}{2} \sum e_i^2 f_i(x)$$
$$2 \times F_1(x) = F_2(x)$$
$$\frac{d\sigma}{dE d\Omega} = \sigma_{\text{Mott}} \left( \frac{F_2(x)}{2} + 2 \tan^2 \frac{\theta}{2} \frac{F_1(x)}{M} \right)$$

Quarks

u	2.4 MeV	$\frac{2}{3}$	$\frac{1}{2}$
d	4.8 MeV	$-\frac{1}{3}$	$\frac{1}{2}$
s	95 MeV	$-\frac{1}{3}$	$\frac{1}{2}$
c	1.275 GeV	$\frac{2}{3}$	$\frac{1}{2}$
b	4.18 GeV	$-\frac{1}{3}$	$\frac{1}{2}$
t	172.44	$\frac{2}{3}$	$\frac{1}{2}$



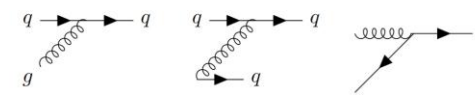
## Lecture 12: Quarks within the Proton and Neutron

### 1. Structure Functions and Cross Sections

$$F_2(x) = \sum e_i^2 \cdot x f_i(x)$$
$$F_1(x) = \frac{1}{2} \sum e_i^2 f_i(x)$$
$$2F_1(x) = F_2(x)$$
$$\frac{d\sigma}{dE d\Omega} = \sigma_{\text{Mott}} \left( \frac{F_2(x)}{\nu} + 2 \tan^2 \frac{\theta}{2} \cdot \frac{F_1(x)}{M} \right)$$

Quark	Mass	Charge q	Spin
u	2.4 MeV	$\frac{2}{3}$	$\frac{1}{2}$
d	4.8 MeV	$-\frac{1}{3}$	$\frac{1}{2}$
s	95 MeV	$-\frac{1}{3}$	$\frac{1}{2}$
c	1.275 GeV	$\frac{2}{3}$	$\frac{1}{2}$
b	4.18 GeV	$-\frac{1}{3}$	$\frac{1}{2}$
t	172.44 GeV	$\frac{2}{3}$	$\frac{1}{2}$

### 8. Partonic Evolution (Feynman Diagrams)



# Conclusion

- ML methods becoming more common
- Many possible uses in research: functional approximation, classification and generation
- Can automate much drudgery

# Tutorial

- Link to Google Colab:

[https://colab.research.google.com/drive/11hfIV9LgMQQC3n2tAT4\\_Ldj0qnJFn684?usp=sharing](https://colab.research.google.com/drive/11hfIV9LgMQQC3n2tAT4_Ldj0qnJFn684?usp=sharing)

- **Need to sign into Google Account** to run
- Can download notebook from Indico page, but need TensorFlow



# Resources

## Good coding practices

- The Good Research Code Handbook ([online](#))

## Neural Networks in Nuclear Physics

- Boehnlein, Amber, et al. "[Colloquium: Machine learning in nuclear physics.](#)" *Reviews of Modern Physics* 94.3 (2022): 031003.

## Tensorflow Tutorials

- [TensorFlow 2 quickstart for beginners](#)
- [Convolutional Neural Network \(CNN\)](#)

# Resources - Basics

## General

- Textbook: Goodfellow, Bengio, and Courville. *Deep learning*. 2016.
  - Available [online](#), with [lectures](#)
  - Topics listed below can be found here

## Neural Networks

- [But what is a neural network?](#) | 3Blue1Brown (YouTube, Chapter 1 of 4 in ML series)

## CNNs

- [But what is a convolution?](#) | 3Blue1Brown (YouTube)
- Stanford University: [Introduction to Convolutional Neural Networks for Visual Recognition](#) (YouTube, whole course)

# Resources – Advanced Topics

## PINNs

- Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." [Journal of Computational physics 378](#) (2019): 686-707.

## NN Optimization

- Smith, Samuel L., et al. "Don't decay the learning rate, increase the batch size." *arXiv preprint [arXiv:1711.00489](#)* (2017).

## Uncertainty Quantification in NNs

- Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." [international conference on machine learning](#). PMLR, 2016.

## Neural Operators

- Kovachki, Nikola, et al. "Neural operator: Learning maps between function spaces." *arXiv preprint [arXiv:2108.08481](#)* (2021).

# Backup Slides

# ML Action Parameters in LQCD

- Parameter estimation
- Uses SU(2) Gauge symmetry
- Gauge links: 4 dimension
- Each as 4, therefore 16
- Grid:  $12^3 \times 36$

PHYSICAL REVIEW D **97**, 094506 (2018)

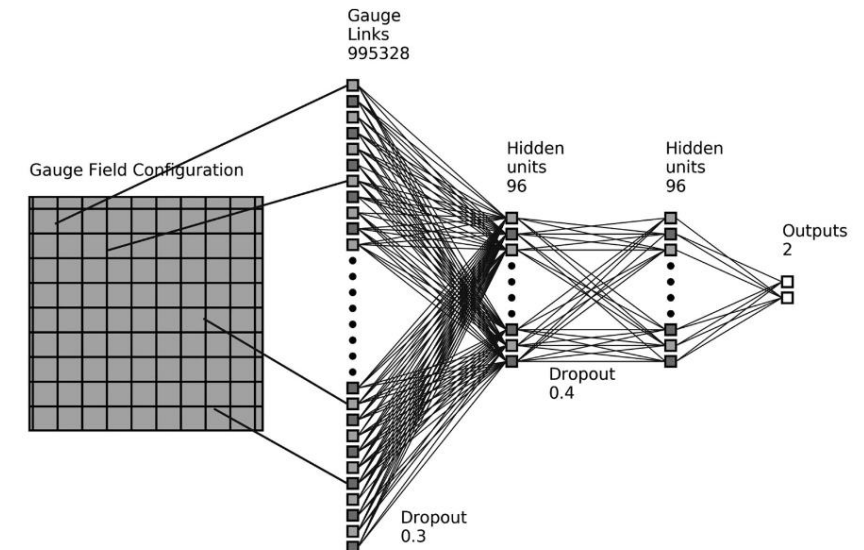
## Machine learning action parameters in lattice quantum chromodynamics

Phiala E. Shanahan,<sup>1,2</sup> Amalie Trewartha,<sup>2</sup> and William Detmold<sup>3</sup>

<sup>1</sup>*Department of Physics, College of William and Mary, Williamsburg, Virginia 23187-8795, USA*

<sup>2</sup>*Jefferson Laboratory, 12000 Jefferson Avenue, Newport News, Virginia 23606, USA*

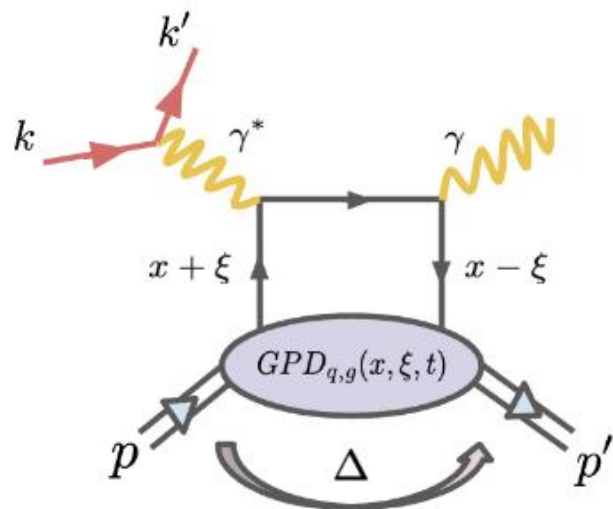
<sup>3</sup>*Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*



- For  $ep \rightarrow e'p'\gamma'$ ,

$$\frac{d^5\sigma}{dx_{Bj}dQ^2d|t|d\varphi d\varphi_S} = \Gamma|T|^2.$$

DVCS:



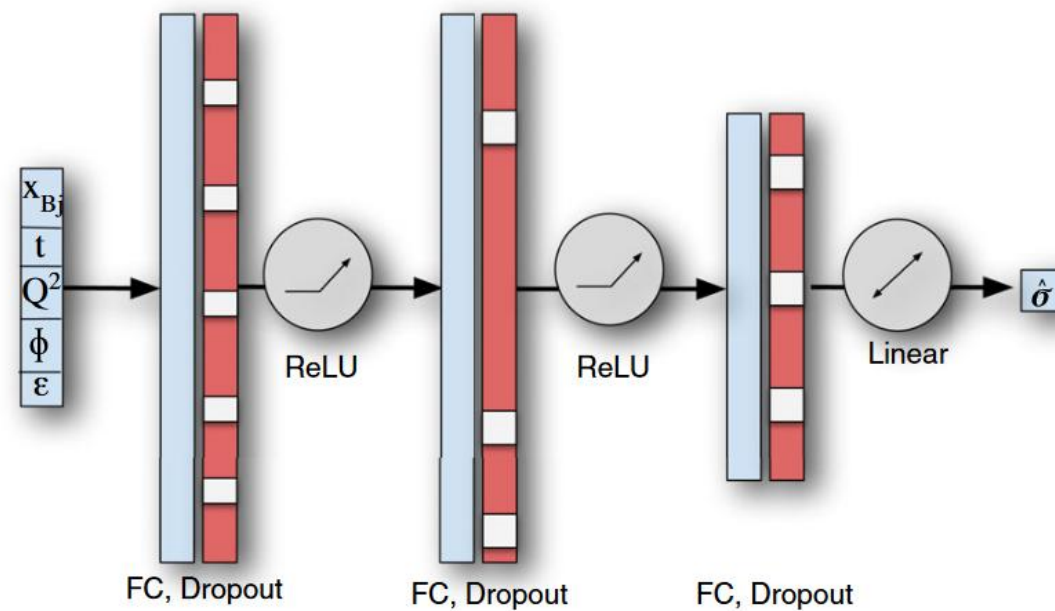
- But  $T = T_{DVCS} + T_{BH}$

## Deep learning analysis of deeply virtual exclusive photoproduction

Jake Grigsby,<sup>\*</sup> Brandon Kriesten,<sup>†</sup> Joshua Hoskins<sup>⊙,‡</sup> and Simonetta Liuti<sup>⊙,§</sup>  
 Department of Physics, University of Virginia, Charlottesville, Virginia 22904, USA

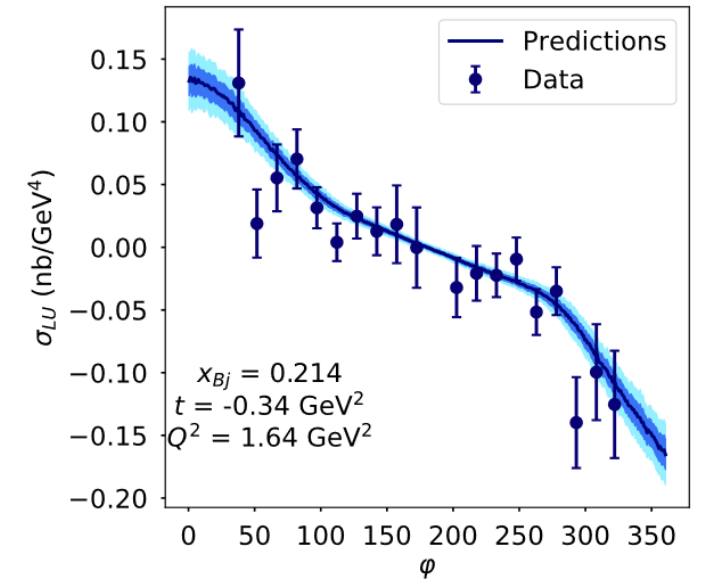
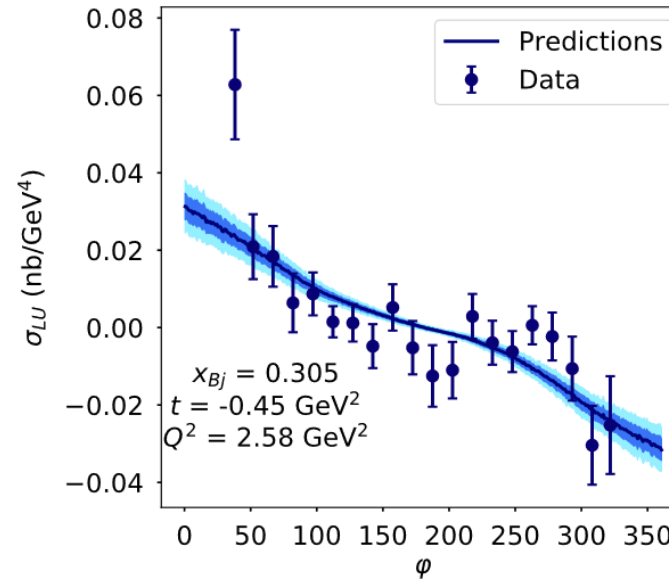
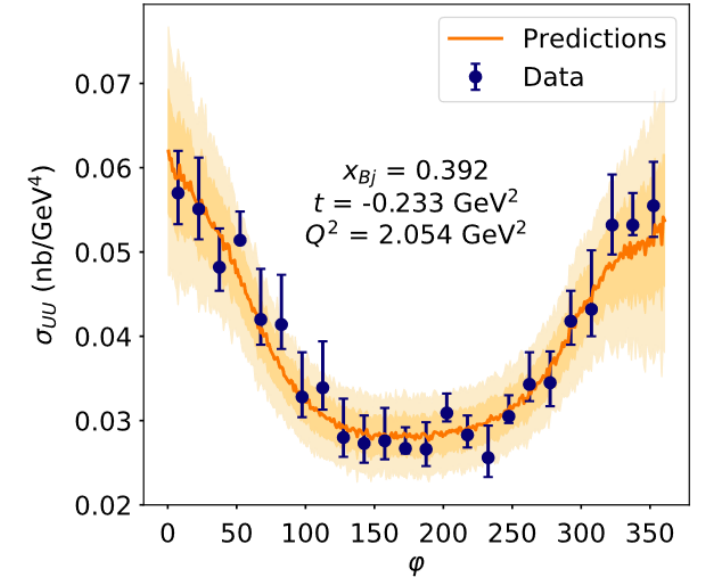
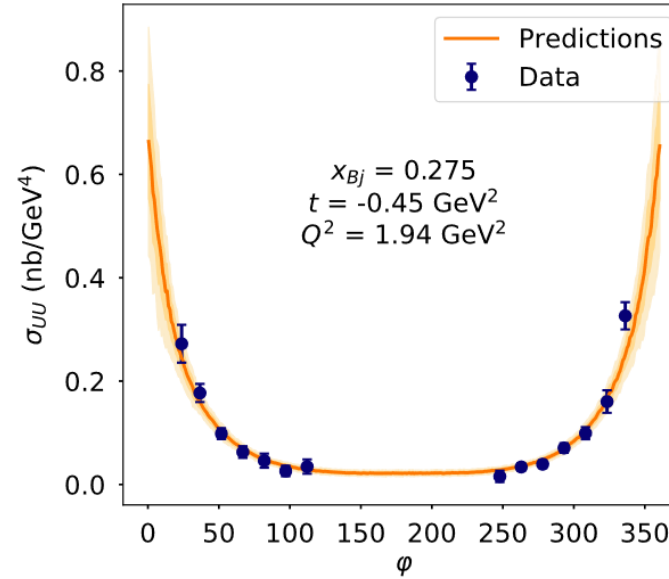
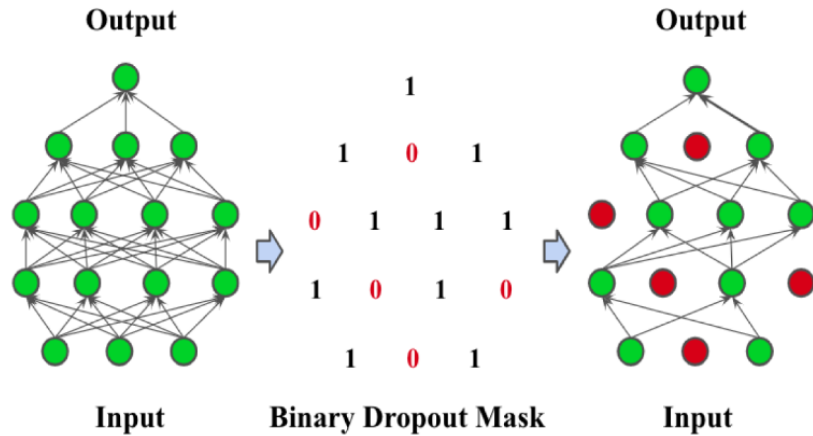
Peter Alonzi<sup>⊙||</sup>  
 School of Data Science, University of Virginia, Charlottesville, Virginia 22904, USA

Matthias Burkardt<sup>¶</sup>  
 New Mexico State University, Department of Physics,  
 Box 30001 MSC 3D, Las Cruces, New Mexico 88003, USA



# FemtoNet

- *Dropout* - prevents overfitting, leads to neurons to look at independent features, avoid correlations



# LQCD Ensemble Generation

- Potential for lattice QCD
- Computational methods reach critical slowing down
- Unlike with images, probability distribution known

## Image generation

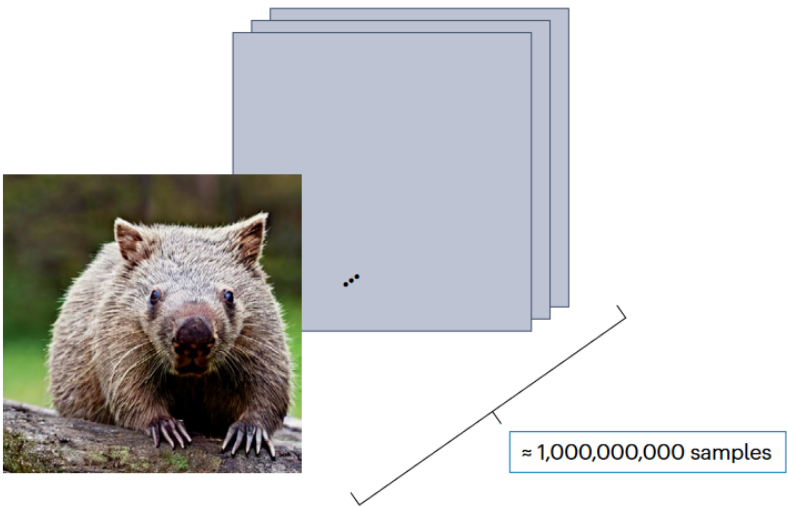
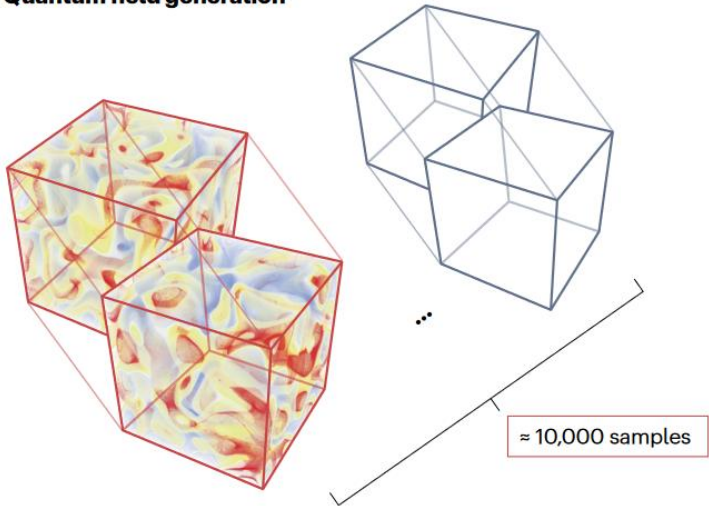


Image geometry	512 × 512
RGB pixel variables	× 3
	≈ 1,000,000 dof

**Target**  
Subjective high quality per sample

**Symmetries**  
Few approximate symmetries  
(for example, reflection, small translations)

## Quantum field generation



Lattice geometry	256 × 256 × 256 × 512
SU(3) link variables	× 4 × 8
	≈ 100,000,000,000 dof

**Target**  
Objective distribution  $p(U) = e^{-S(U)}/Z$

**Symmetries**  
High-dimensional exact symmetries  
(for example, translations, gauge symmetry)

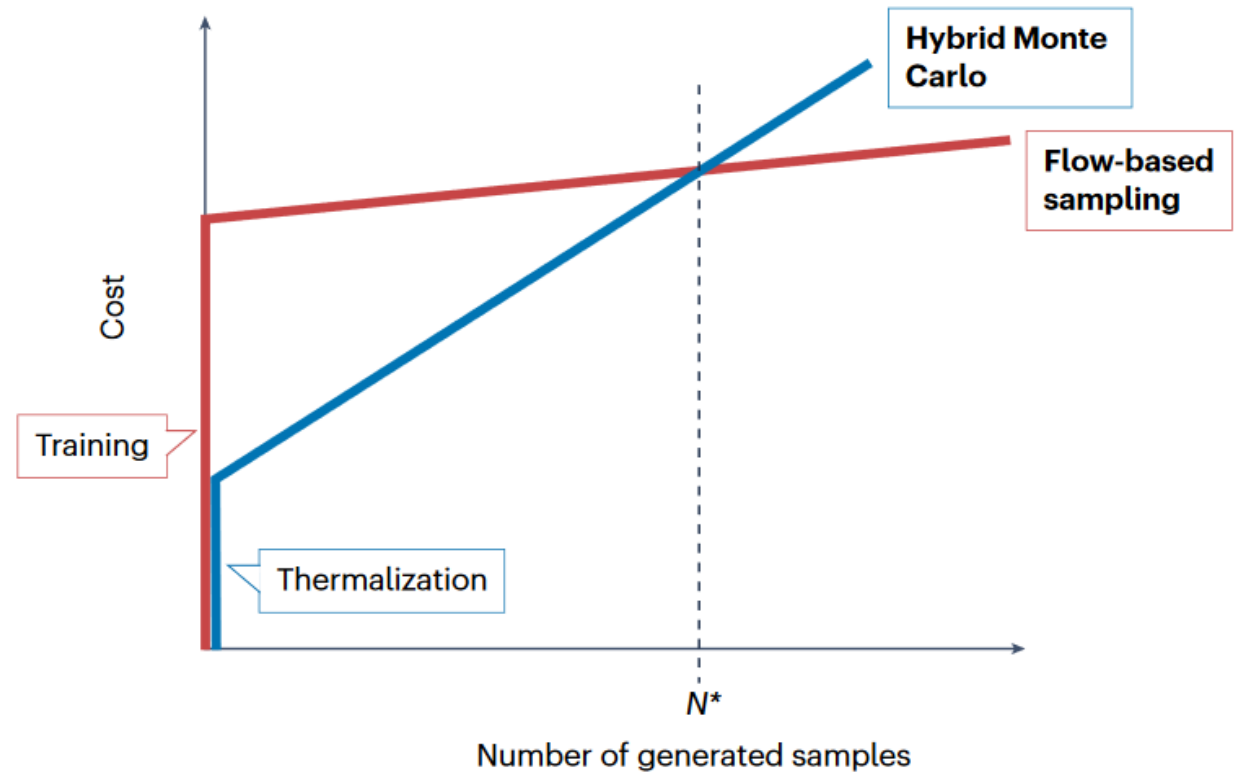
## Advances in machine-learning-based sampling motivated by lattice quantum chromodynamics

Kyle Cranmer<sup>1</sup>, Gurtej Kanwar<sup>2</sup>, Sébastien Racanière<sup>3</sup>, Danilo J. Rezende<sup>3</sup> & Phiala E. Shanahan<sup>4,5</sup>✉



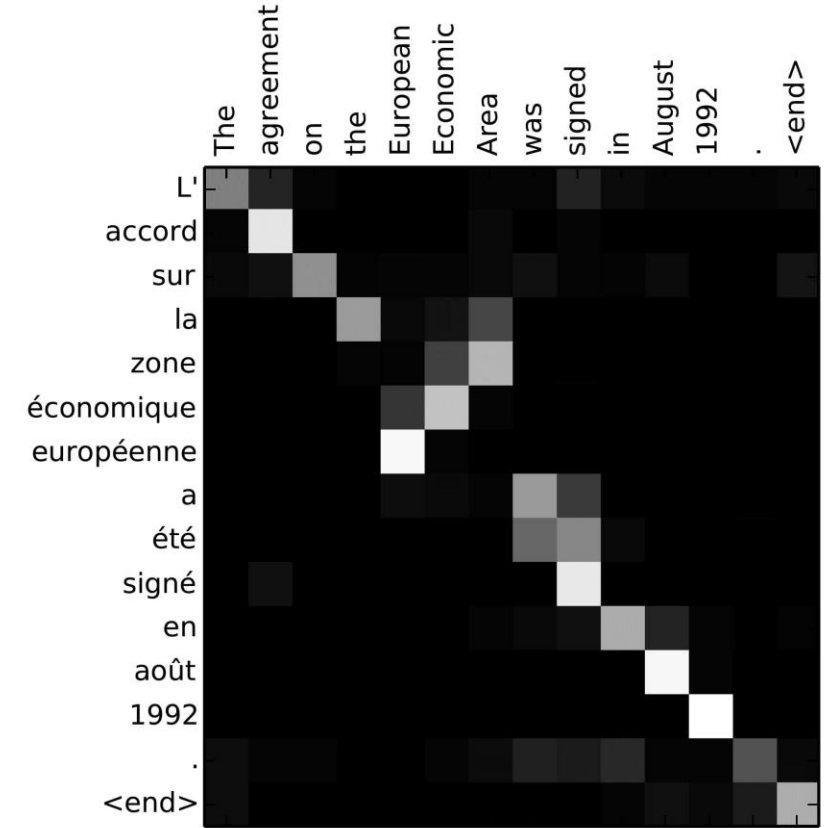
# LQCD Ensemble Generation

- Eventually should outpace traditional hybrid Monte Carlo
- We want symmetries to hold, asymptotic guarantees
- Much work done on simpler problems – 2D, scalar theories, etc.



# Transformer

- Uses attention mechanism
- Technology behind ChatGPT (generative pre-trained transformer)



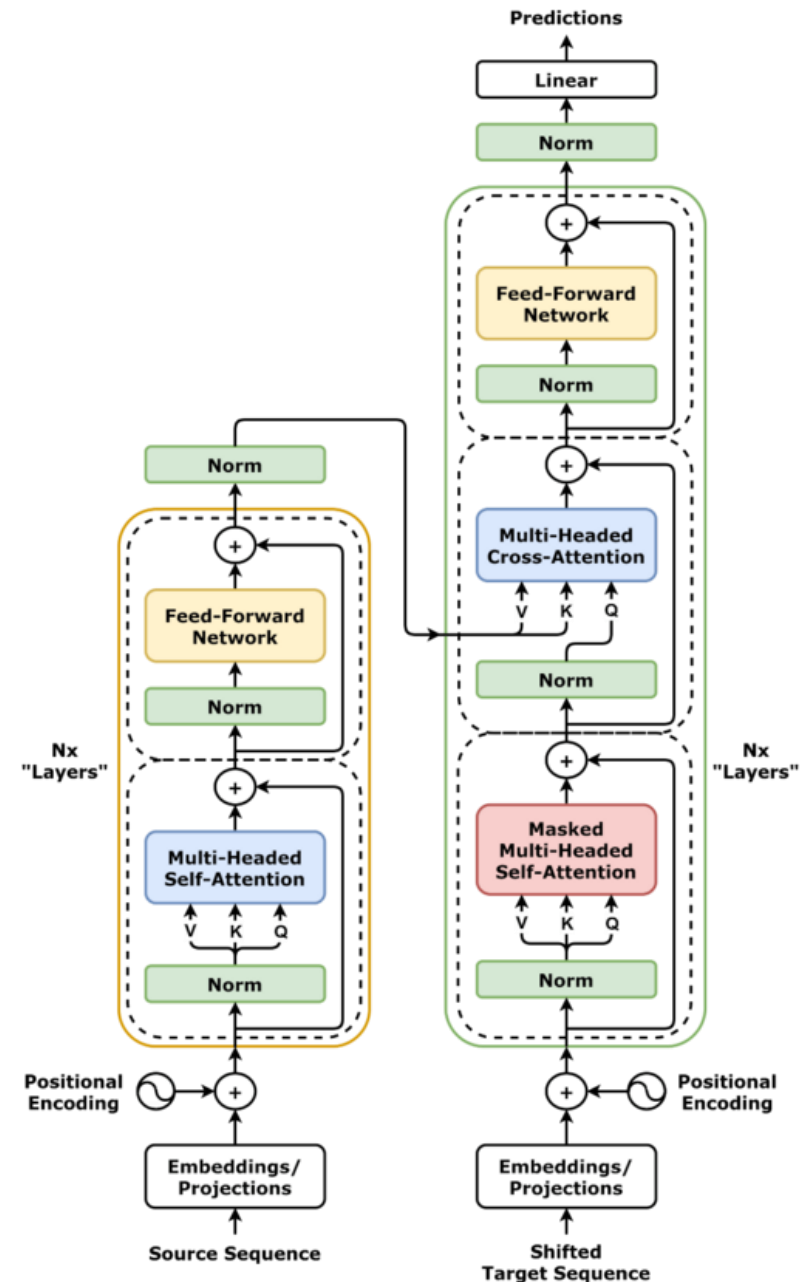
# Transformers

## Advantages:

- Can pick out complex and long-term correlations
- Computationally efficient to train

## Disadvantage:

- Usually requires a lot of data to learn properly



# QCD Jets with Transformers



PUBLISHED FOR SISSA BY SPRINGER

RECEIVED: March 21, 2023

REVISED: May 30, 2023

ACCEPTED: June 19, 2023

PUBLISHED: June 27, 2023

- Authors use analogy:

Learning the language of QCD jets with transformers

Thorben Finke,<sup>a</sup> Michael Krämer,<sup>a</sup> Alexander Mück<sup>a</sup> and Jan Tönshoff<sup>b</sup>

*constituents in jets  $\Leftrightarrow$  words in sentence*

- Want to do Density Estimation.
- E.g, for # constituents of jets or momentum.

# Approach

Preprocess data to be more like natural language

- **Order:** constituents ordered by  $p_T$  (greatest first) to get “grammar”

$$x_1 x_2 \dots x_n$$

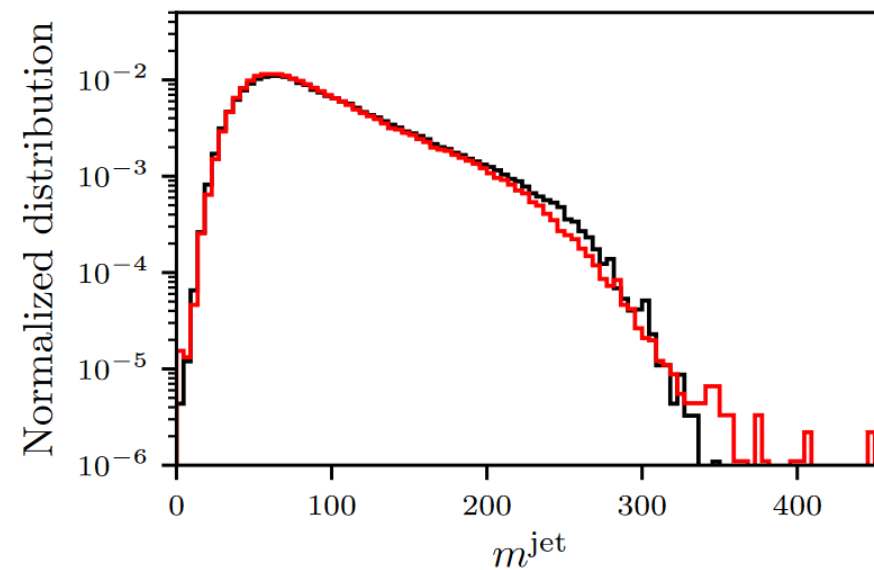
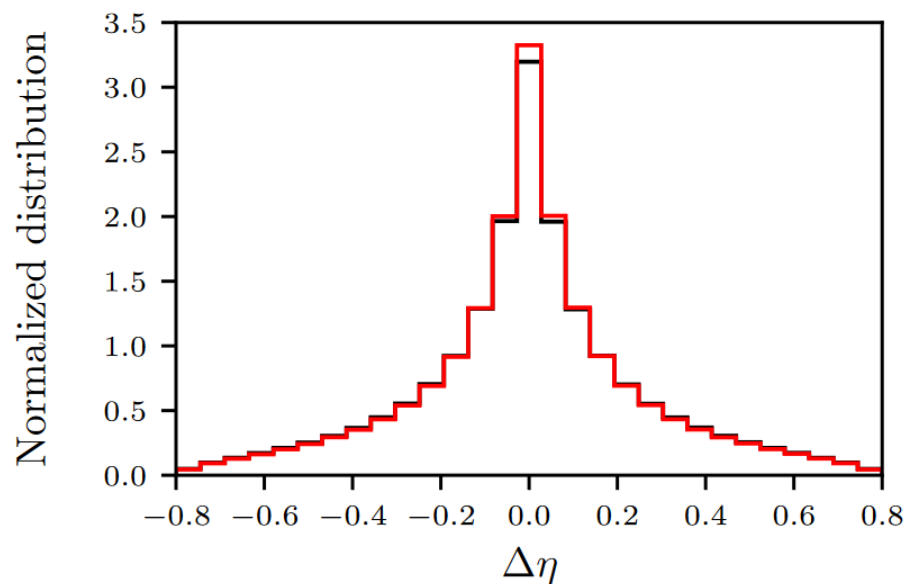
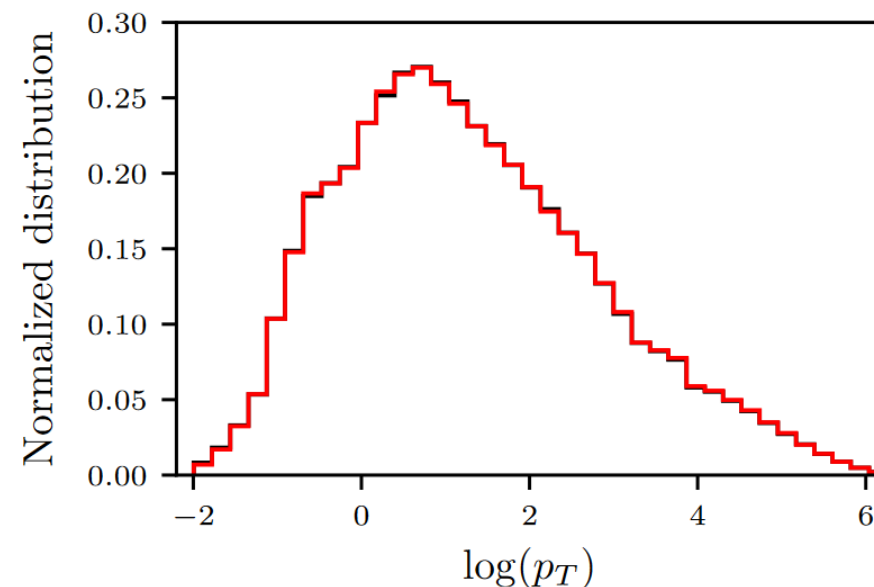
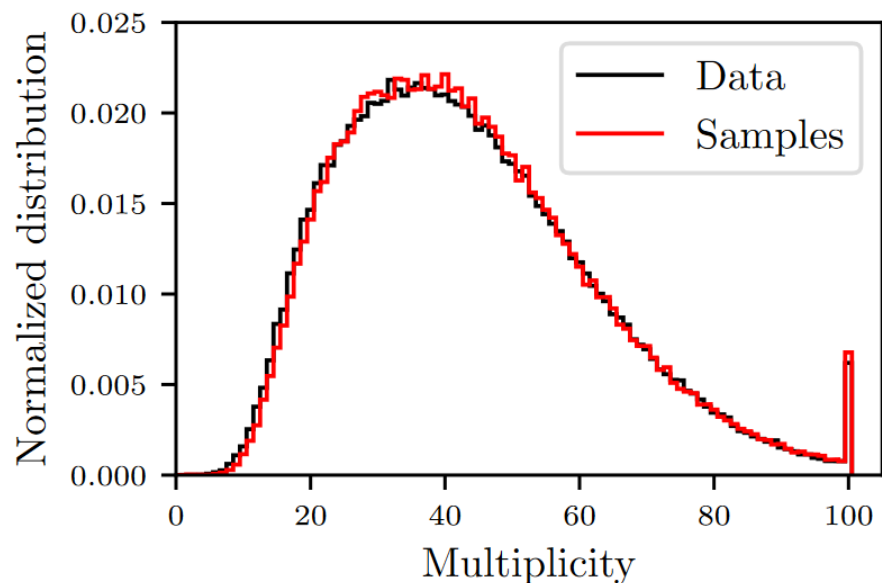
- **Discretize:** Bin data

Other:

- Take only 50 highest  $p_T$  constituents (dataset has highest 200).
- “Stop token” added if <50 constituents and padded
- Select 600k jets for training, 200k for val/test each

# Results

- Histograms
- Trained only on 50, but were able to accurately predict up to 100



# Gradient Descent

$$\theta_j \rightarrow \theta_j - \epsilon \frac{\partial L}{\partial \theta_j}$$

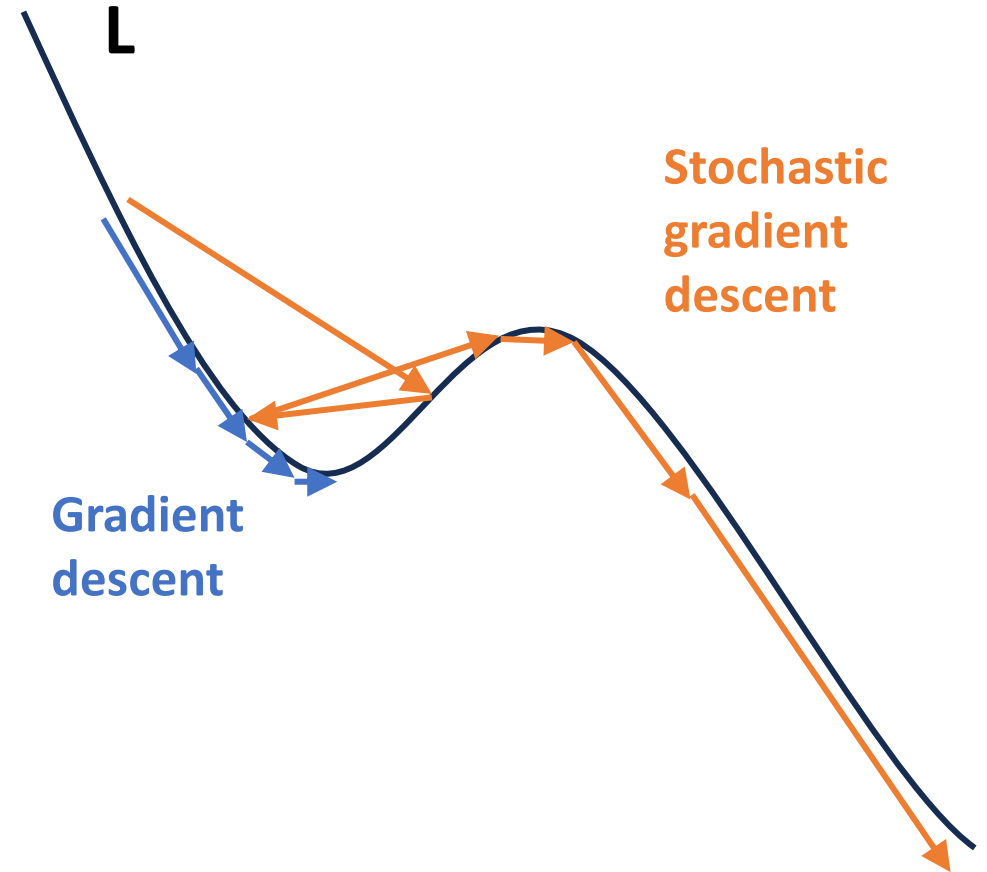
where  $\epsilon$  is 'learning rate'; hyperparameter

**Stochastic gradient descent:** use only part of data ("batch") for  $L$  every parameters update.

- More computationally efficient
- Less likely to end up stuck in local stationary point, explore more

Data

batch	batch	batch	batch	batch	batch
-------	-------	-------	-------	-------	-------



$$\begin{aligned} L_B &= L + (L_B - L) \\ &= L + \eta \end{aligned}$$