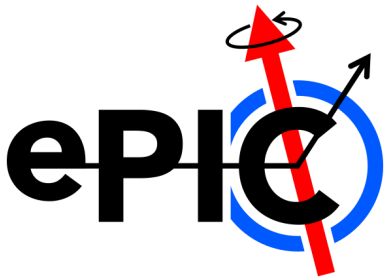
Deconvolution of Momentum Transfer Distributions Through Deep Learning

Exclusive, Diffraction, Tagging Physics Working Group

Maci Kesler



Joint ePIC/EICUG Meeting
University of Glasgow, Scotland, UK

July 11 - 17, 2026

KENT STATE UNIVERSITY

Center for Nuclear Research

Introduction: Exclusive Vector Meson (VM) Production

Major goals of the EIC

- 3D structure of nuclei
- **Gluon saturation**
- Origin of proton spin
- Quark and gluon confinement

Introduction: Exclusive Vector Meson (VM) Production

Major goals of the EIC

- 3D structure of nuclei
- **Gluon saturation**
- Origin of proton spin
- Quark and gluon confinement

Critical Measurement: Exclusive VM Production

- Deflection of VM **measures spatial distribution of gluons**
- Probe to gluon density $\rightarrow$ **saturation**

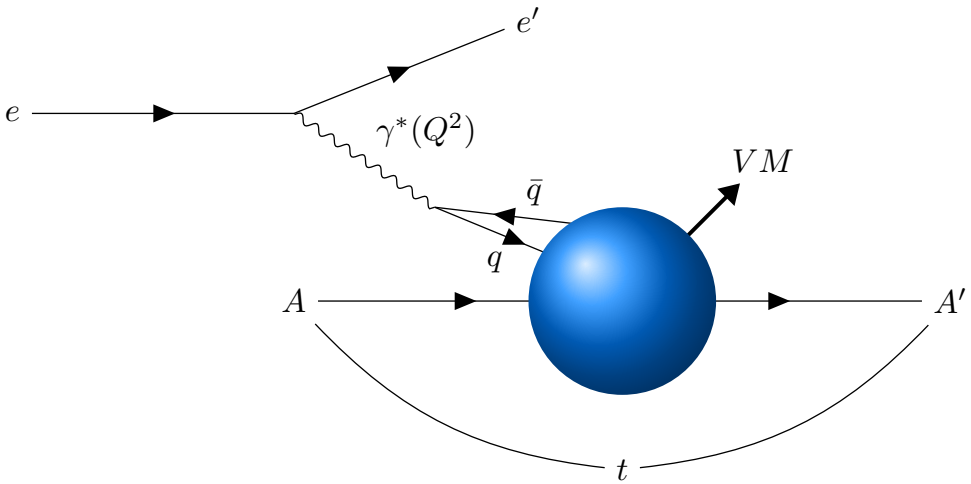
Introduction: Exclusive Vector Meson (VM) Production

Major goals of the EIC

- 3D structure of nuclei
- **Gluon saturation**
- Origin of proton spin
- Quark and gluon confinement

Critical Measurement: Exclusive VM Production

- Deflection of VM **measures spatial distribution of gluons**
- Probe to gluon density → **saturation**



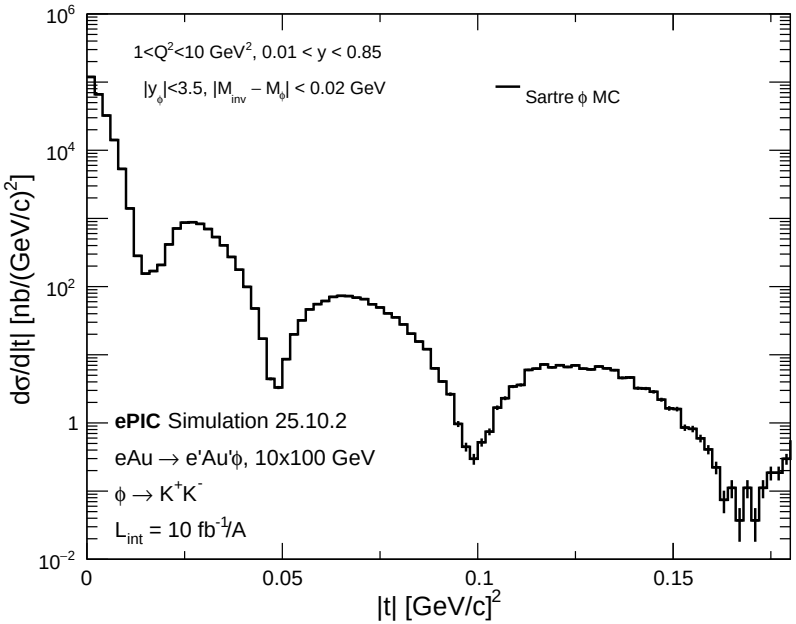
Distribution of momentum transfer ($|t|$) :

- Fourier conjugate to impact parameter
- Reflects the spatial profile
- **Gluon imaging**

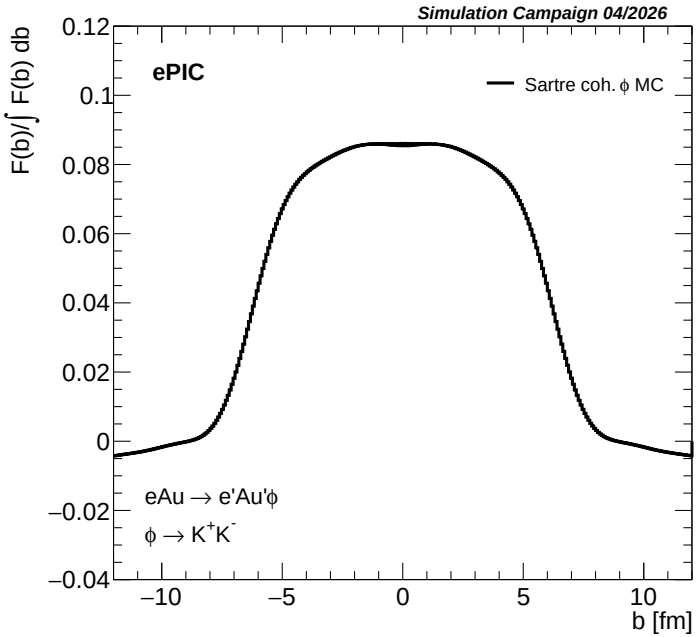
Saturation models predict at EIC **more than** 20% **of the cross-section will be diffractive**
EIC YR: https://doi.org/10.1016/j.nuclphysa.2022.122447

Goal: Precise Measurement of Diffractive Minima

MC $|t|$ Distribution



Transformation of MC $|t|$ Distribution

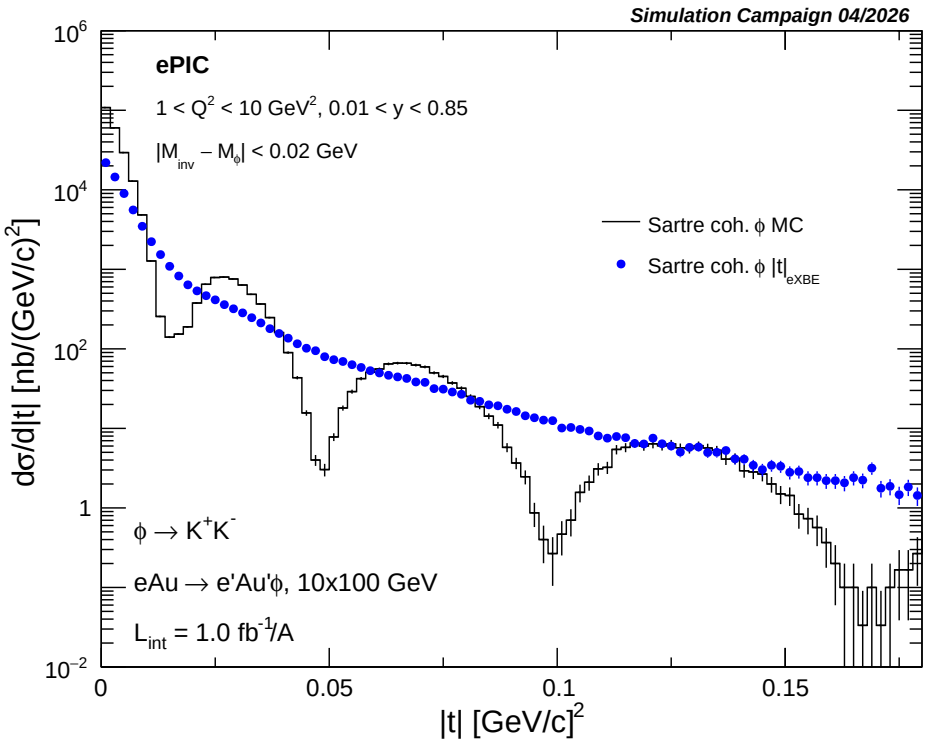


Diffractive Minima ⇔ Spatial Distribution

- Small **errors in minima** = large **errors in density**
- Smaller $|b|$ determined by larger $|t|$

The Experimental Challenge

Measurements of the $|t|$ distribution encounter **2 primary challenges:**



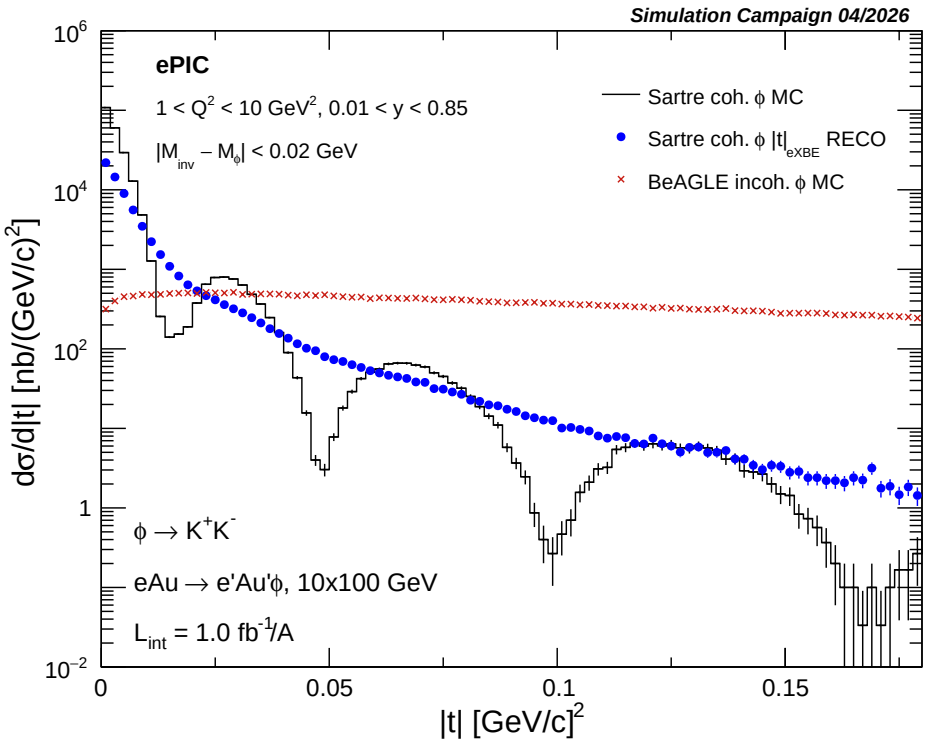
1. **Limited resolution in measuring $|t|$:**
   - **Mainly momentum resolution of e'**
     - Tiny angular error $\rightarrow$ large p_T error

Best method of $|t|$ reconstruction $\rightarrow$ $|t|_{\mathrm{eXBE}}$ **(blue circles)**

EIC YR: https://doi.org/10.1016/j.nuclphysa.2022.122447

The Experimental Challenge

Measurements of the $|t|$ distribution encounter **2 primary challenges:**



1. **Limited resolution in measuring $|t|$:**
   - **Mainly momentum resolution of e'**
     - Tiny angular error $\rightarrow$ large p_T error
2. **Overwhelming incoherent background:**
   - **Red x's**
   - Detector cannot suppress all incoherent production

Best method of $|t|$ reconstruction $\rightarrow$ $|t|_{\mathrm{eXBE}}$ **(blue circles)**

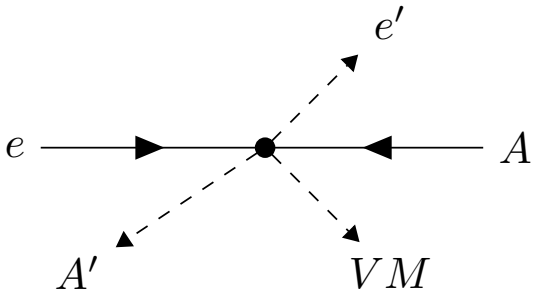
EIC YR: https://doi.org/10.1016/j.nuclphysa.2022.122447

Extracting $|t|$

Reconstruct from exclusive VM production

$$|t| = (P_A - P_{A'})^2$$

To access $|t|$: need complete final state

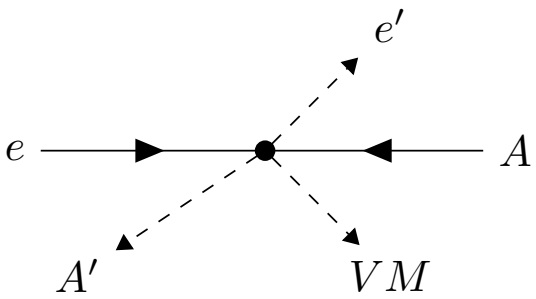


Extracting $|t|$

Reconstruct from exclusive VM production

$$|t| = (P_A - P_{A'})^2$$

To access $|t|$: need complete final state



Cannot measure $P_{A'}$

- For heavy nuclei $\rightarrow$ stays within beam envelope
- Tiny momentum change + tiny angular deflection

Extracting $|t|$

Reconstruct from exclusive VM production

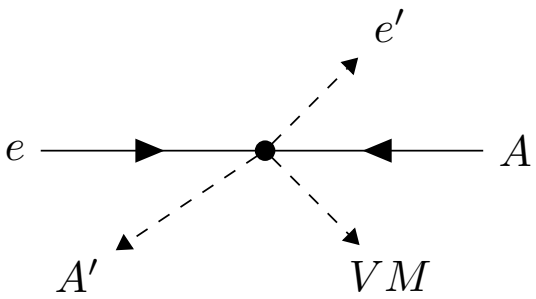
$$|t| = (P_A - P_{A'})^2$$

To access $|t|$: need complete final state

Cannot measure $P_{A'}$

- For heavy nuclei $\rightarrow$ stays within beam envelope
- Tiny momentum change + tiny angular deflection

- $|t|_{\mathrm{BABE}}$ (method E): $|t| = (P_{VM} + P_{e'} - P_e)^2$
  - Small error/inaccuracy has large effect on $|t|$



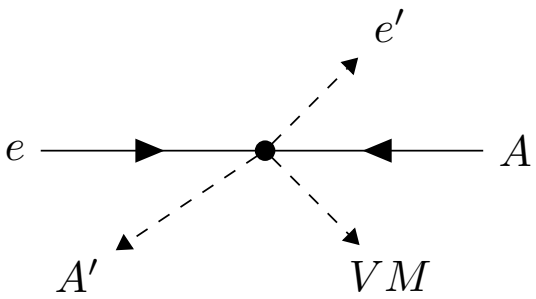
EIC YR: https://doi.org/10.1016/j.nuclphysa.2022.122447

Extracting $|t|$

Reconstruct from exclusive VM production

$$|t| = (P_A - P_{A'})^2$$

To access $|t|$: need complete final state



Cannot measure $P_{A'}$

- For heavy nuclei → stays within beam envelope
- Tiny momentum change + tiny angular deflection

- $|t|_{\mathbf{BABE}}$ (method E): $|t| = (P_{VM} + P_{e'} - P_e)^2$
  - Small error/inaccuracy has large effect on $|t|$
- $|t|_{\mathbf{eX}}$ (method A): $|t| = [\mathbf{p}_T(e') + \mathbf{p}_T(VM)]^2$
  - Underestimates true t, valid only for small $|t|$ and small Q^2

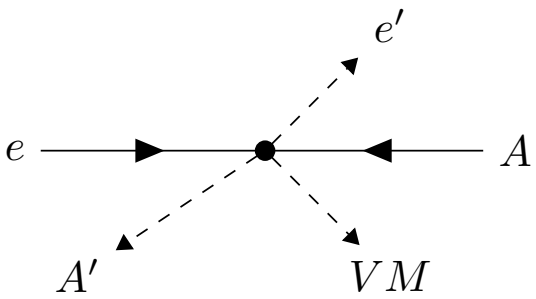
EIC YR: https://doi.org/10.1016/j.nuclphysa.2022.122447

Extracting $|t|$

Reconstruct from exclusive VM production

$$|t| = (P_A - P_{A'})^2$$

To access $|t|$: need complete final state



Cannot measure $P_{A'}$

- For heavy nuclei → stays within beam envelope
- Tiny momentum change + tiny angular deflection

- $|t|_{\mathbf{BABE}}$ (method E): $|t| = (P_{VM} + P_{e'} - P_e)^2$
  - Small error/inaccuracy has large effect on $|t|$
- $|t|_{\mathbf{eX}}$ (method A): $|t| = [\mathbf{p}_T(e') + \mathbf{p}_T(VM)]^2$
  - Underestimates true t, valid only for small $|t|$ and small Q^2
- $|t|_{\mathbf{eXBE}}$ (method L): $|t| = (P_A - P_{A'}^{\mathrm{corr}})^2$
  - Only applies to coherent events

EIC YR: https://doi.org/10.1016/j.nuclphysa.2022.122447

Extracting $|t|$

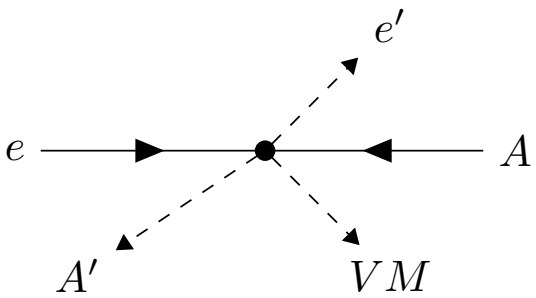
Reconstruct from exclusive VM production

$$|t| = (P_A - P_{A'})^2$$

To access $|t|$: need complete final state

Cannot measure $P_{A'}$

- For heavy nuclei → stays within beam envelope
- Tiny momentum change + tiny angular deflection



- $|t|_{\mathbf{BABE}}$ (method E): $|t| = (P_{VM} + P_{e'} - P_e)^2$
  - Small error/inaccuracy has large effect on $|t|$
- $|t|_{\mathbf{eX}}$ (method A): $|t| = [\mathbf{p}_T(e') + \mathbf{p}_T(VM)]^2$
  - Underestimates true t, valid only for small $|t|$ and small Q^2
- $|t|_{\mathbf{eXBE}}$ (method L): $|t| = (P_A - P_{A'}^{\mathrm{corr}})^2$
  - Only applies to coherent events

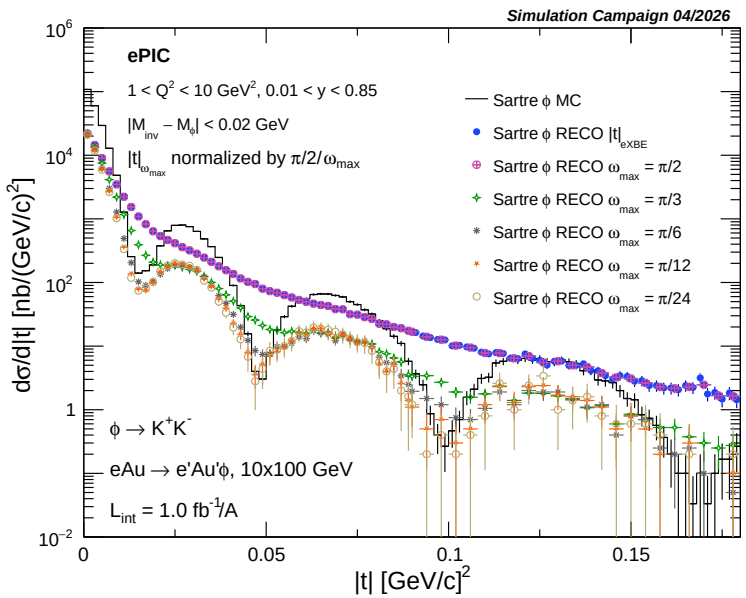
EIC YR: https://doi.org/10.1016/j.nuclphysa.2022.122447

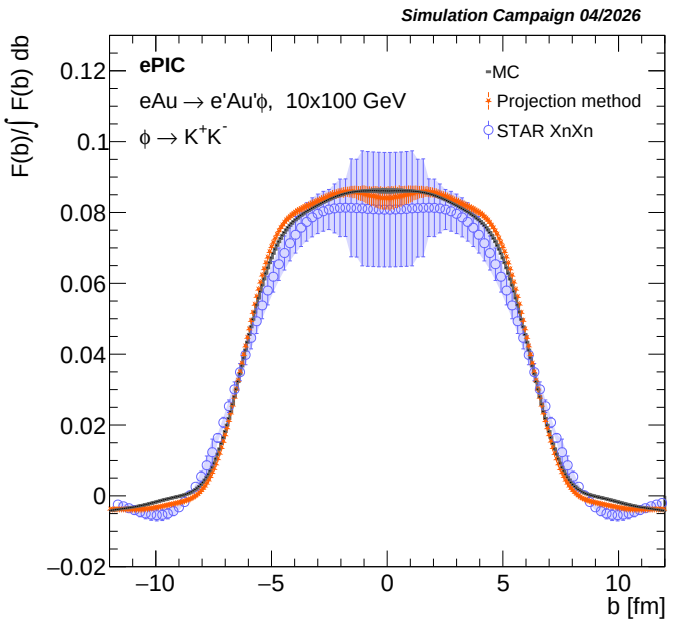
- **Projected-$|t|$ Method**: Add projection + wedge cut to $|t|_{\mathrm{eXBE}}$
  - Loss of statistics

M. Kesler, A. I. Sheikh, et. al., 10.1016/j.physletb.2026.140585

Resulting $|t|$ Distribution and Transformation After Using Projected-$|t|$ Method

Significant improvement in both distributions!





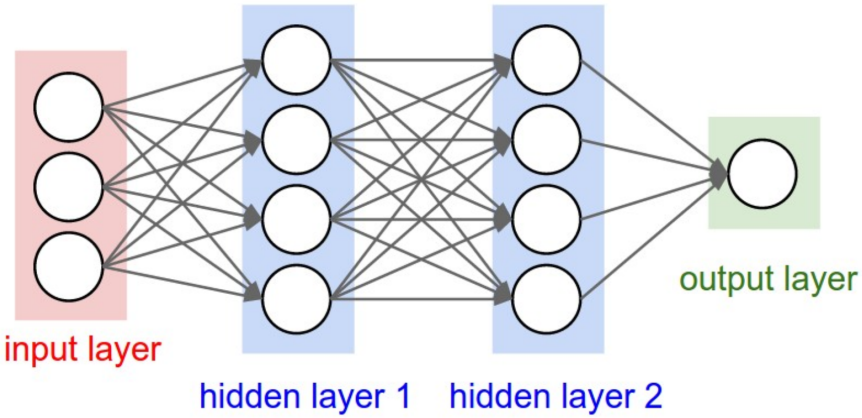
STAR Collaboration, 10.1103/PhysRevC.96.054904

- Increase $\omega_{\max}$:
  - **Ability to resolve dips**

- Using $\omega_{\max} = \pi/12$
- Need **precise location of minima**

Can AI Do Better?

PyTorch Model: Multilayer Perceptron (MLP)



- Deep learning artificial NN
- Stacked layers of neurons
- Good for tabulated data
- Can **learn non-linear patterns in data**

Goal: train model to recover $|t|$ distribution

- Inverse supervised learning: learns **relationship between input & output**
- Give model:
  - **Output:** $|t|_{\mathrm{MC}}$
  - **Input:** kinematics from eA collisions
- Split data into 70% training, 15% validation, and 15% testing

AI Deconvolution of $|t|$

- Regression unfolding: learns inverse mapping **from smeared to truth kinematics**

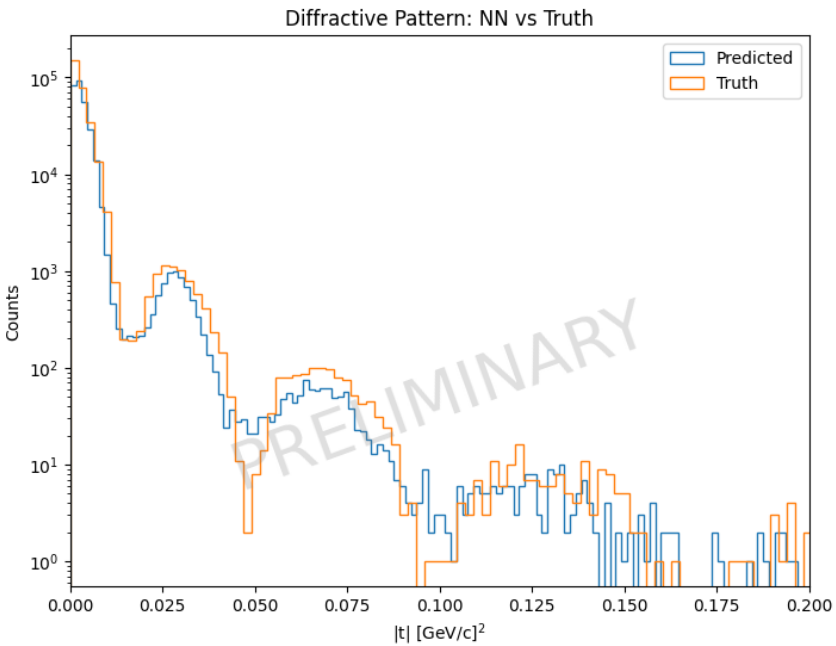
$$|t|_{\mathrm{MC}} \Rightarrow f(P_{e'}, P_{\mathrm{VM}}, m_{\mathrm{VM}}, Q^2_{\mathrm{RECO}}, x_{\mathrm{RECO}}, y_{\mathrm{RECO}}, \sqrt{|t|_x}, \sqrt{|t|_y})$$

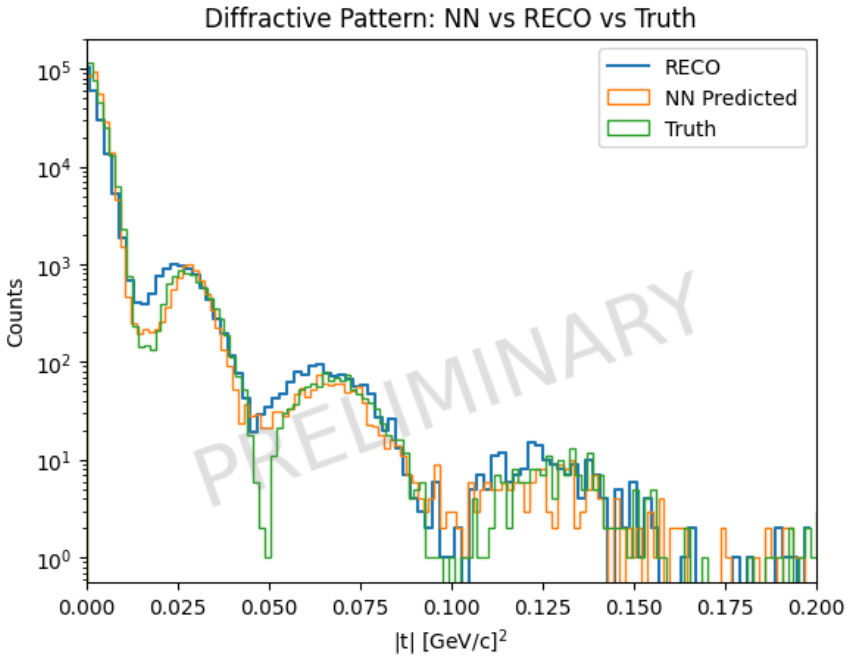
- Analysis done with **Carson Throckmorton, KSU**

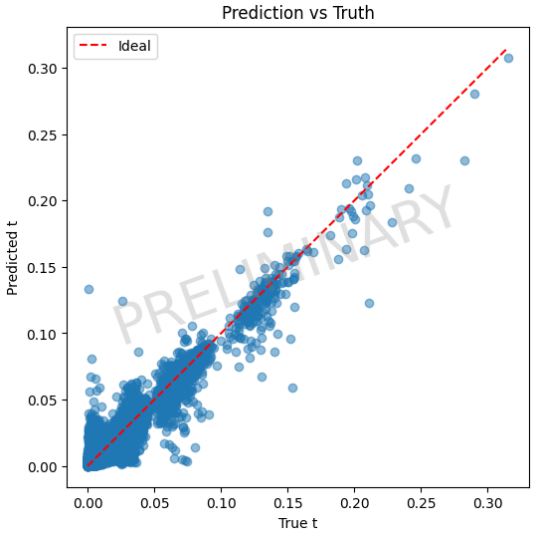
Optuna to determine best hyperparameters: https://optuna.org/

- Activation functions: ReLU and Leaky ReLU
- Optimizer: AdamW
- Loss function: Huber (Smooth L1)
- Scheduler: Plateau
- Epochs: 25
- Learning rate: 0.001
- Hidden layers: 4
- Number of neurons: [93,88,129,110]
- Dropout: [0.03,0.04,0.02,0.01]
- Weight decay: 0.02
- Batch size: 1024
- Gradient clip: 1.6

Results







Promising **results without the wedge cut**

- Captures more of first dip
- Peaks are narrowed

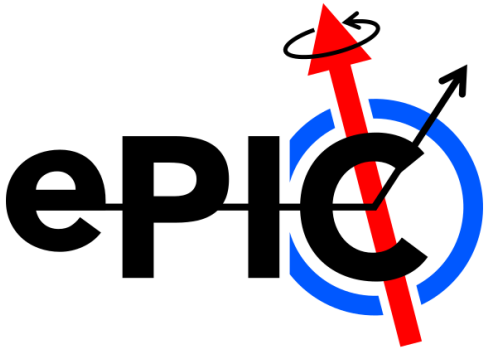
Summary

- Developed **deep learning model to resolve diffractive minima** in $|t|$ distribution
- Can resolve first minima very well
- Still very preliminary $\rightarrow$ **work in progress**

Next Steps:

- Fix narrow peaks
  - Try adding ω_{max} cut (from projected-$|t|$ method)
    - Use model to optimize this cut
- **Compare e' resolution from model with ePIC**
- Write model into scripts

Thank you :)



Backup Slides

Hyperparameters (1/2)

Activation functions: transform neuron's output before passed as input.

- Introduces **nolinearity to model**
- **ReLU:** avoids vanishing gradient problem
- **Leaky ReLU:** avoids dead neurons

Optimizer: update parameters & **minimize loss during training.**

- **AdamW:**
  - Combines momentum & RMPProp
  - Adaptive learning rate
  - Weight decay built-in

Loss function: quantify difference between predicted output & target. Guide optimization process. **Goal of training to minimize this loss.**

- **Huber (Smooth L1):**
  - Combines MSE & MAE
  - Good balance for physics regression

Hidden layers & **neurons:** perform summation (compute weighted sum of inputs) & activation (pass weighted sum through activation function).

- More neurons = more capacity
- Too many = overfitting

Hyperparameters (2/2)

Regularizaition Methods: **prevent overfitting** & generalize models.

- **Dropout:** randomly turns off neurons during training
- **Weight decay:** pushes weights toward zero each step

Learning rate: step size in gradient descent.

- Too high = unstable
- Too low = slow

Batch size: number of samples per gradient update.

- Large batch = smoother gradients & less noise

Gradient clipping: prevents gradients from growing too large.

- Stabilizes training

Epochs: number of passes through dataset.

Build the NN Model

```
best = study.best_params
for x in best:
    print(f"{x}: {best[x]}")
```

```
hidden_dim_0: 93
activation_0: leaky_relu
dropout_0: 0.03007150402385417
hidden_dim_1: 88
activation_1: relu
dropout_1: 0.043018768754540145
hidden_dim_2: 129
activation_2: relu
dropout_2: 0.021246484519209095
hidden_dim_3: 110
activation_3: leaky_relu
dropout_3: 0.01186743778614358
optimizer: adamw
lr: 0.00143994192136382
weight_decay: 0.021329785380376887
batch_size: 1024
grad_clip: 1.6347382449190462
scheduler: plateau
loss_fn: smooth_l1
```

```
class Dynamic_Regressor(nn.Module):
    def __init__(self, input_dim, layers, activations, dropouts):
        super().__init__()

        modules = []
        prev_dim = input_dim

        for i, hidden_dim in enumerate(layers):
            modules.append(nn.Linear(prev_dim, hidden_dim))
            modules.append(activations[i])
            modules.append(nn.Dropout(dropouts[i]))
            prev_dim = hidden_dim

        modules.append(nn.Linear(prev_dim, 1))
        self.net = nn.Sequential(*modules)

    def forward(self, x):
        return self.net(x).squeeze(-1)
```

- Build **dynamic model** to take in hyperparameters as variables
- Optuna loops over trials to **find best params**

Train the NN Model

```python
train_loader = DataLoader(train_data_custom, batch_size=best["batch_size"], shuffle=True)
val_loader   = DataLoader(val_data_custom,   batch_size=best["batch_size"], shuffle=False)
test_loader  = DataLoader(test_data_custom,  batch_size=best["batch_size"], shuffle=False)

# train with best hyperparams
for epoch in range(epochs):
    final_model.train()
    train_loss = 0.0

    for Xb, yb in train_loader:
        pred = final_model(Xb)
        loss = loss_fn(pred, yb.squeeze(-1))

        optimizer.zero_grad()
        loss.backward()

        if best["grad_clip"] > 0:
            nn.utils.clip_grad_norm_(final_model.parameters(), best["grad_clip"])

        optimizer.step()

        train_loss += loss.item() * Xb.size(0)

    train_loss /= len(train_loader.dataset)

    final_model.eval()
    val_loss = 0.0
    with torch.no_grad():
        for Xb, yb in val_loader:
            pred = final_model(Xb)
            loss = loss_fn(pred, yb.squeeze(-1))
            val_loss += loss.item() * Xb.size(0)

    val_loss /= len(val_loader.dataset)

    if sched_name == "plateau":
        scheduler.step(val_loss)
    elif scheduler:
        scheduler.step()
    print(f"Epoch {epoch+1:03d}|"f"Train Loss: {train_loss:.4e}|"f"Val Loss: {val_loss:.4e}")
```

```
Epoch 001|Train Loss: 1.1757e-05|Val Loss: 5.2477e-06
Epoch 002|Train Loss: 5.1826e-06|Val Loss: 4.0062e-06
Epoch 003|Train Loss: 4.1286e-06|Val Loss: 3.0839e-06
Epoch 004|Train Loss: 3.7595e-06|Val Loss: 4.8740e-06
Epoch 005|Train Loss: 3.3640e-06|Val Loss: 2.7441e-06
Epoch 006|Train Loss: 3.2639e-06|Val Loss: 2.5694e-06
Epoch 007|Train Loss: 2.9455e-06|Val Loss: 2.8776e-06
Epoch 008|Train Loss: 2.9294e-06|Val Loss: 2.5884e-06
Epoch 009|Train Loss: 2.8543e-06|Val Loss: 2.0923e-06
Epoch 010|Train Loss: 2.6574e-06|Val Loss: 2.4854e-06
Epoch 011|Train Loss: 2.7296e-06|Val Loss: 2.1832e-06
Epoch 012|Train Loss: 2.7928e-06|Val Loss: 4.2469e-06
Epoch 013|Train Loss: 3.1314e-06|Val Loss: 2.6530e-06
Epoch 014|Train Loss: 2.3284e-06|Val Loss: 1.9020e-06
Epoch 015|Train Loss: 2.0898e-06|Val Loss: 1.7873e-06
Epoch 016|Train Loss: 2.0311e-06|Val Loss: 1.8015e-06
Epoch 017|Train Loss: 1.9738e-06|Val Loss: 1.8676e-06
Epoch 018|Train Loss: 1.9468e-06|Val Loss: 1.8252e-06
Epoch 019|Train Loss: 1.9099e-06|Val Loss: 1.7999e-06
Epoch 020|Train Loss: 1.8490e-06|Val Loss: 1.7691e-06
Epoch 021|Train Loss: 1.8374e-06|Val Loss: 1.7474e-06
Epoch 022|Train Loss: 1.8344e-06|Val Loss: 1.7612e-06
Epoch 023|Train Loss: 1.8269e-06|Val Loss: 1.7493e-06
Epoch 024|Train Loss: 1.8265e-06|Val Loss: 1.7506e-06
Epoch 025|Train Loss: 1.8315e-06|Val Loss: 1.7432e-06
```

- **Left:** training loop with validation
- **Right:** accumulated loss from each epoch