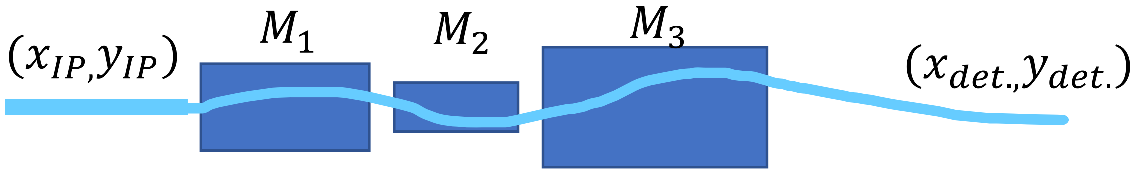
Machine-Learning for Roman Pots Reconstruction

David Ruth, Alex Jentsch, Sakib Rahman

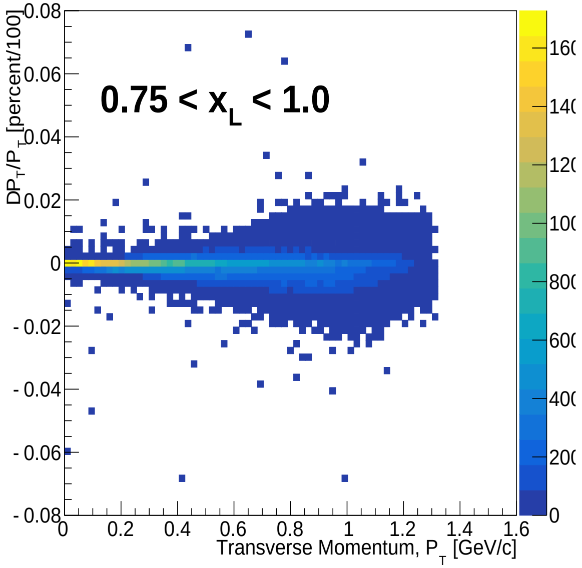
Roman Pots Reconstruction

$$\begin{pmatrix} x_{ip} \\ \theta_{x,ip} \\ y_{ip} \\ \theta_{y,ip} \\ z_{ip} \\ \Delta p/p \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \\ c_0 & c_1 & c_2 & c_3 & c_4 & c_5 \\ d_0 & d_1 & d_2 & d_3 & d_4 & d_5 \\ e_0 & e_1 & e_2 & e_3 & e_4 & e_5 \\ f_0 & f_1 & f_2 & f_3 & f_4 & f_5 \end{pmatrix} \begin{pmatrix} x_{det.} \\ \theta_{x,det.} \\ y_{det.} \\ \theta_{y,det.} \\ z_{det.} \\ \Delta p/p \end{pmatrix}$$

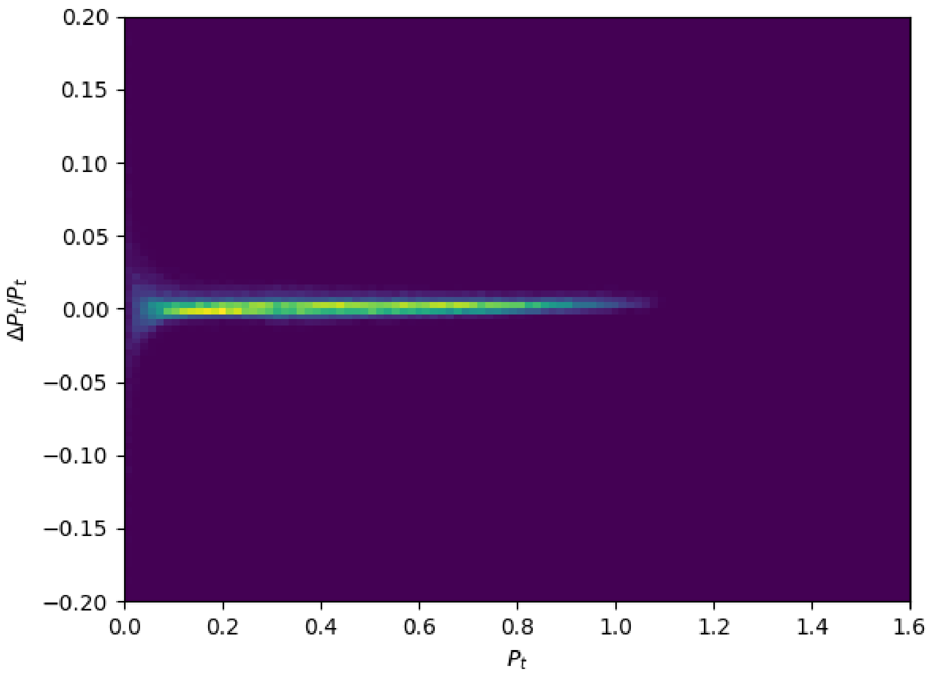


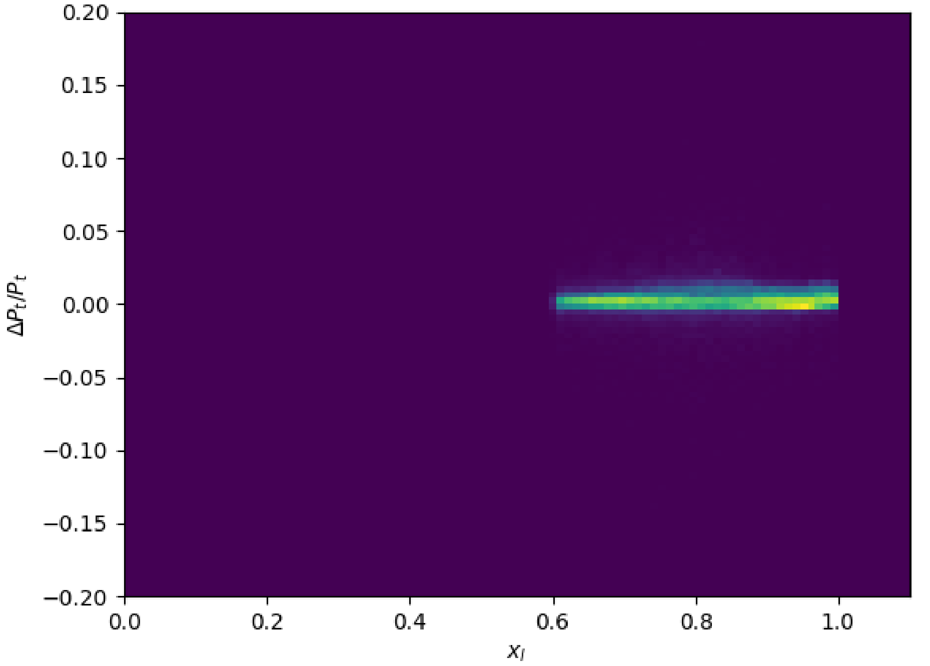
$M_{transfer}$
$= M_1 M_2 M_3 \ldots$

- Transfer matrix gives info on transport through the various magnets between IP and RP Detectors



- Current dynamic method performs well but only at high x_L
- Performance also suffers for high P_t
- Assumes linearity to work
- Assumes particles are coming from the center of the main detector
- Complex study needs to be re-done for every change in beamline configuration
- **Try machine learning application instead**





- **Framework:** PyTorch
- **Architecture:** Multi-Layer Perceptron
- **5 Independent Models:**

$$\begin{pmatrix} x \\ \theta_x \\ y \\ \theta_y \end{pmatrix} \rightarrow (P_z)$$

$$\begin{pmatrix} x \\ \theta_x \\ P_z \end{pmatrix} \rightarrow (P_x)$$

$$\begin{pmatrix} y \\ \theta_y \\ P_z \end{pmatrix} \rightarrow (P_y)$$

$P_t < 0.3$

$$\begin{pmatrix} x \\ \theta_x \\ P_z \end{pmatrix} \rightarrow (P_x)$$

$P_t < 0.3$

$$\begin{pmatrix} y \\ \theta_y \\ P_z \end{pmatrix} \rightarrow (P_y)$$

- **5 Hidden Layers, 128 Neurons**
- **Loss Function:** Huber Loss
- **Optimizer:** Adam

- Performance is excellent for P_z and shows little dependence on x_L
- P_t performance is good, but needs further optimization, and performance still slightly worse at low P_t
- Currently trying to run network on ifarm at Jlab, but not straightforward to use these computing resources – may have to use JupyterHub