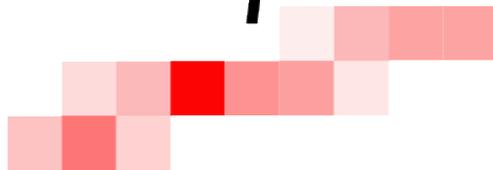


# *On-detector regression of particle kinematics with smartpixels*



Jennet Dickinson for the smartpixels team

October 9, 2025

# Interesting collisions are discarded at the LHC

- The silicon tracking detectors produce **too much data** to process at the collision frequency of 40 MHz
- Data from these detectors is only read out for triggered events
- **This limits our physics reach.**



Hadronic  
Higgs decays

?

Long-lived  
particles

# Interesting collisions are discarded at the LHC

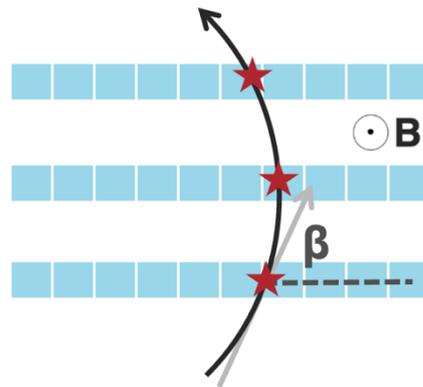
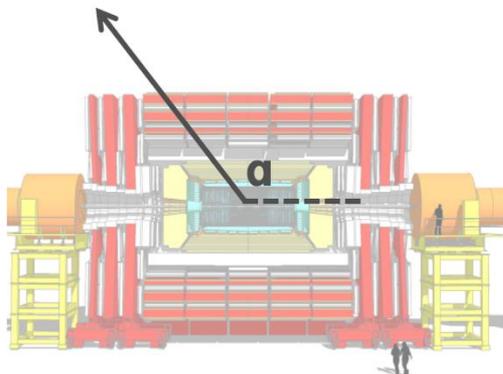
- The silicon tracking detectors produce **too much data** to process at the collision frequency of 40 MHz
- Data from these detectors is only read out for triggered events
- **This limits our physics reach.**
- *smartpixels* uses **machine learning** to perform physics-motivated **data reduction** within the pixelated area of the readout chip (ASIC)



# What to preserve?

- Current scheme: pixel address + charge for pixels above threshold
- What parameters will we later use for tracking?

- ✅ Hit position ( $x, y$ )
- 😬 Azimuthal incidence angle ( $\beta$ )
- 😬 Polar incidence angle ( $\alpha$ )
- 😬 Uncertainties on  $x, y, \alpha, \beta$
- 😬 Full covariance matrix



# Smartpixels regression

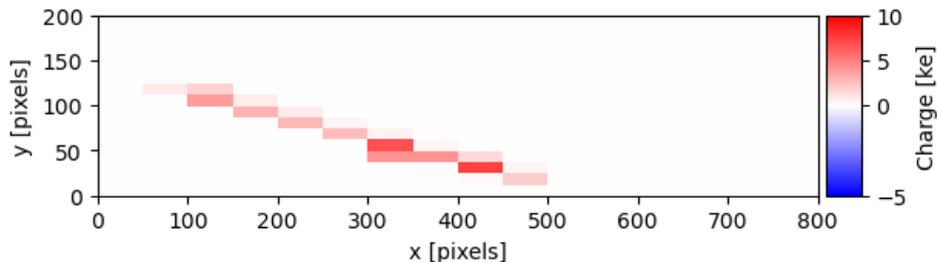
- Can we calculate these parameters from one silicon layer using ML?
  - Yes
- Can we do this within the tight resource constraints of an ASIC?
  - It looks like yes, at the cost of some performance
- What can we send off-detector given bandwidth constraints? Which parameters are the most important?
  - TBD. We estimated 3 and chose  $x$ ,  $y$ ,  $\beta$



# Simulated dataset

*Forthcoming on Zenodo!*

- All models presented use the same dataset of simulated interactions  $\pi^+/\pi^-$  in an array of 16 x 16 pixels



12.5x50 um pitch, 100 um thick

Located at radius of 30 mm

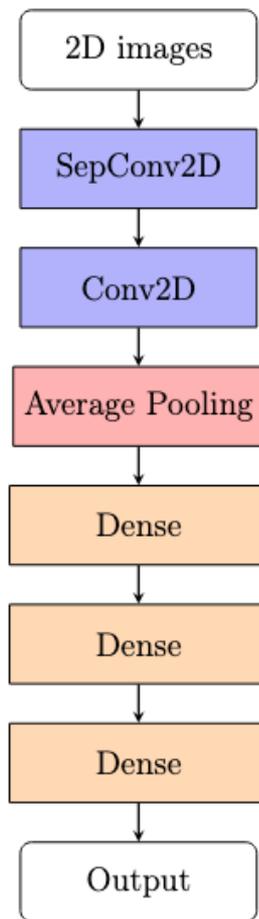
3.8 T magnetic field, -100V bias

Charge is sampled every 200 ps

- About 550,000 clusters. 80% / 20% used for training / testing
  - Clusters extending beyond the 16x16 pixel array are excluded

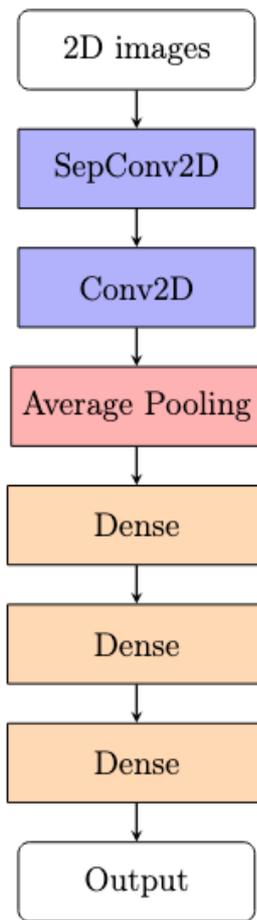
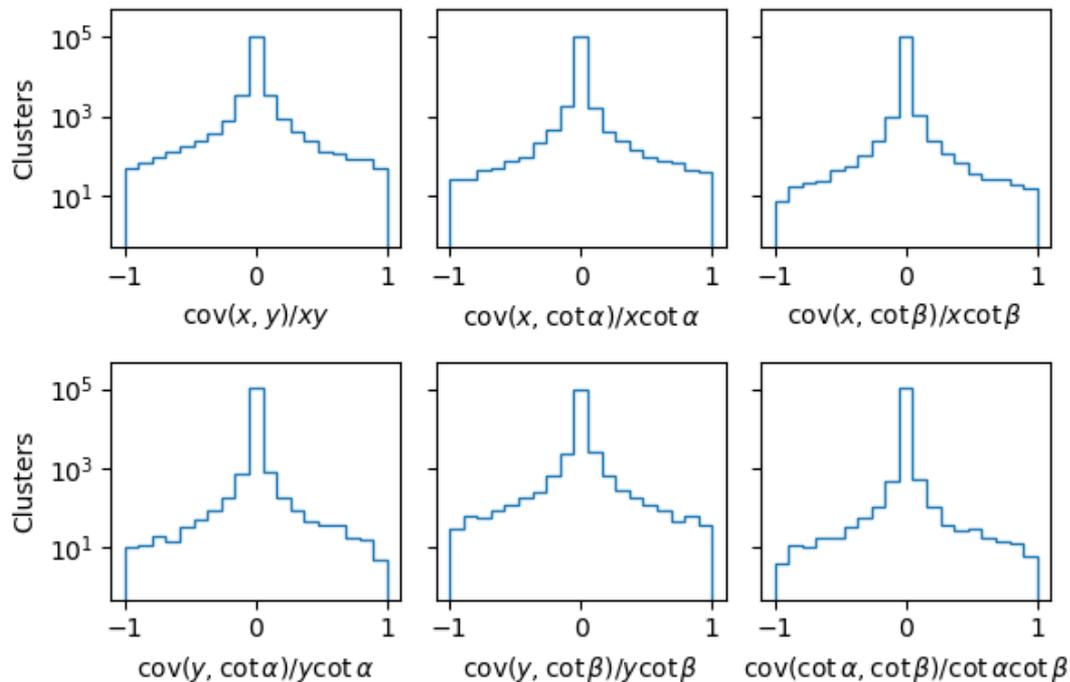
# Max model: predict everything

- Network architecture composed of **2D convolutional** and **dense** layers
  - 2D conv are commonly used for image processing
- Take the likelihood of each cluster to be a **4D Gaussian**
  - Predict 14 parameters: means of  $x$ ,  $y$ ,  $\alpha$ ,  $\beta$  + 10 elements of the covariance matrix
  - Use  $2\Delta\text{NLL}$  as the loss function in the network training



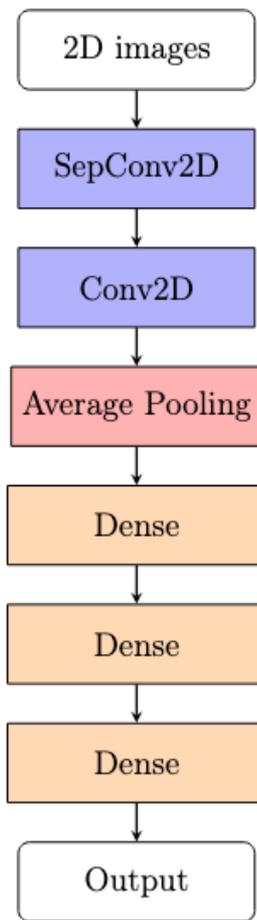
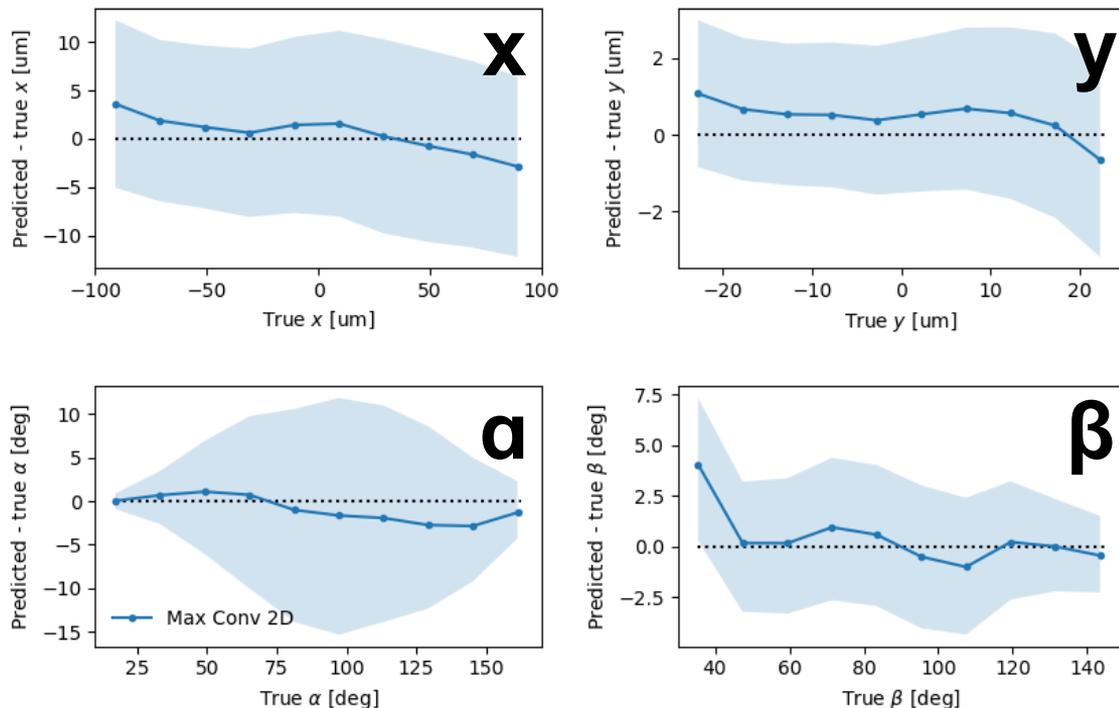
# Max model: predict everything

- Off-diagonal elements of the covariance matrix are all 0:



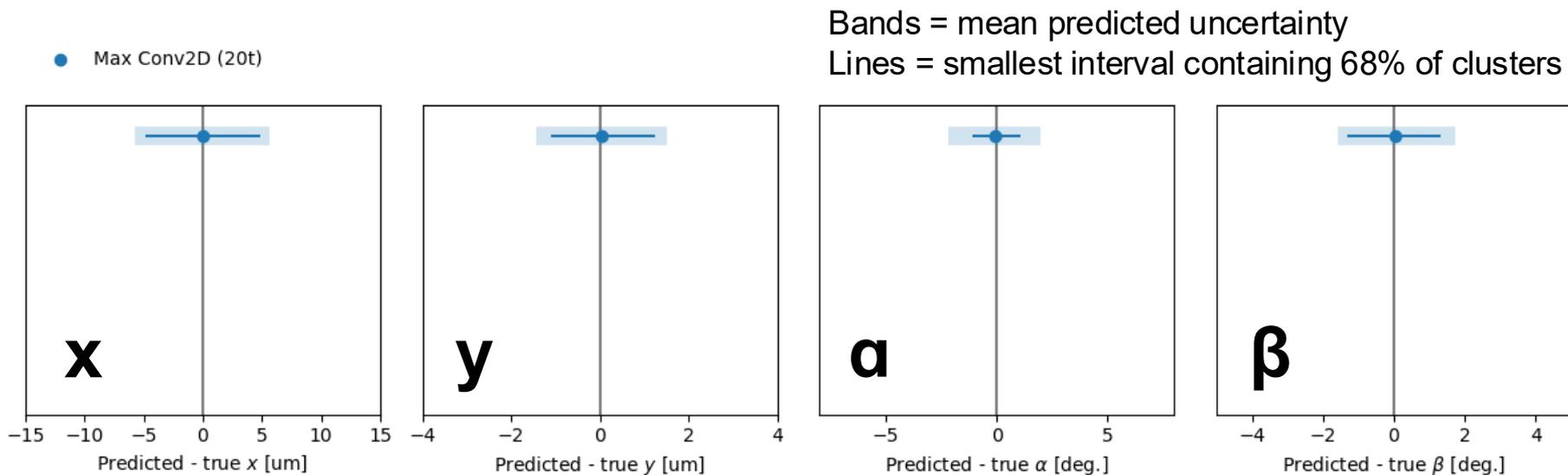
# Max model: predict everything

- Regression of the other parameters works well:



# Max model: predict everything

- Regression of the other parameters works well:

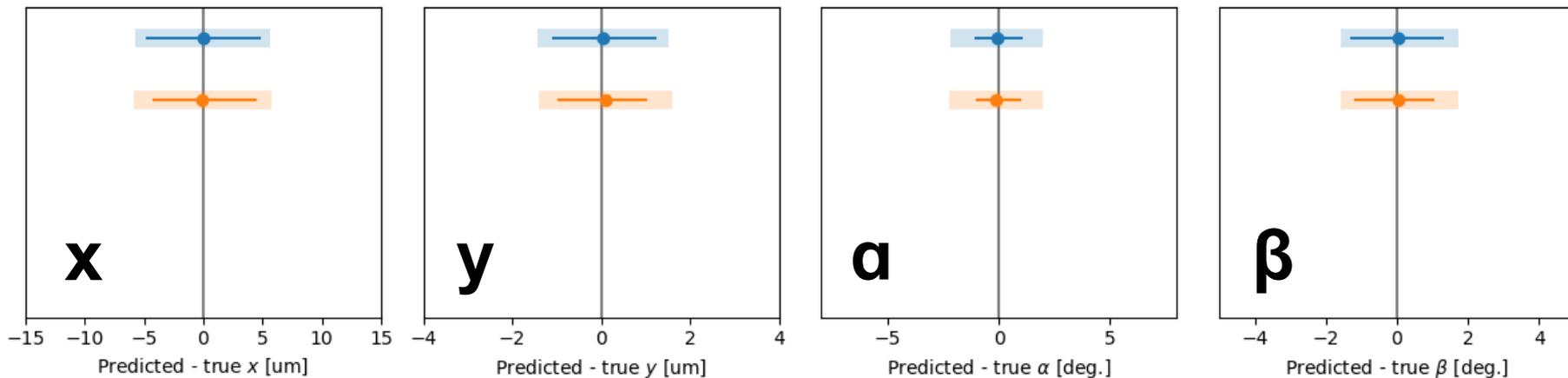


# Full model: forget the correlations

- Same architecture, but likelihood is a product of **4 x 1D Gaussians**
  - Predict 8 parameters: means of  $x$ ,  $y$ ,  $\alpha$ ,  $\beta$  + 4 uncertainties
  - Use  $2\Delta\text{NLL}$  as the loss function in the network training

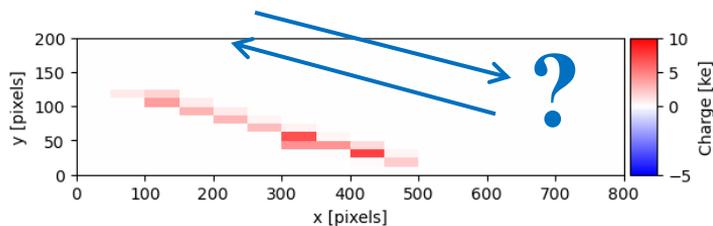
● Max Conv2D (20t)  
● Full Conv2D (20t)

Bands = mean predicted uncertainty  
Lines = smallest interval containing 68% of clusters

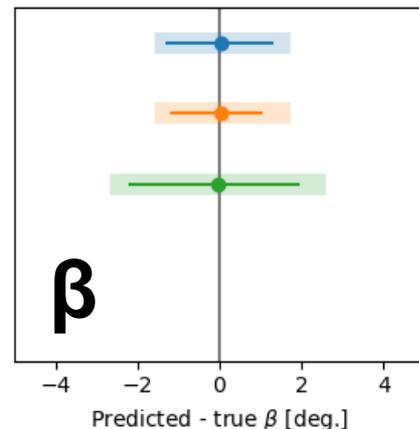
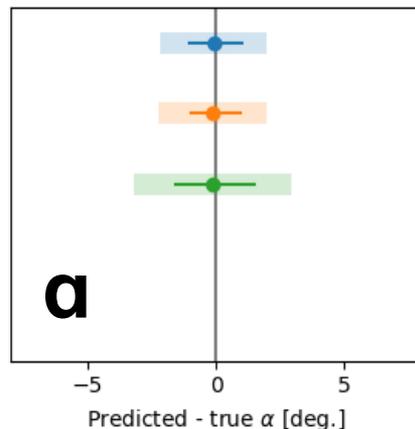
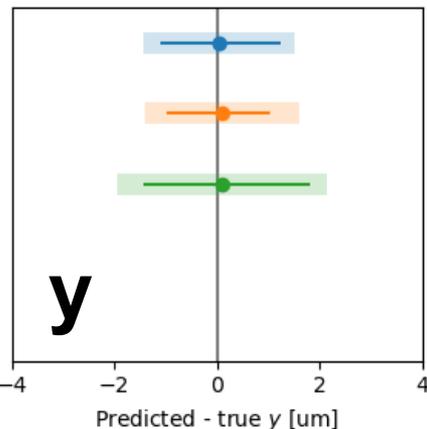
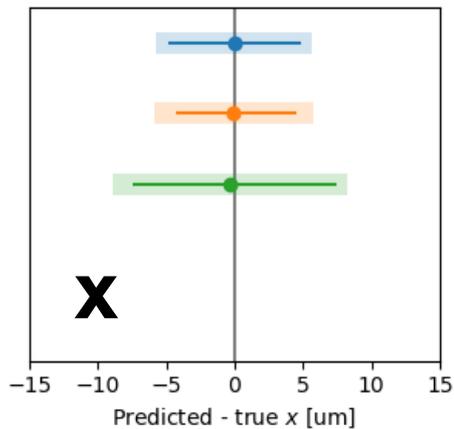


# Reality check: sampling rate

- Our ASIC will not sample 20 times in 4 ns
  - We have to live with sampling with **twice in 4ns.**
  - Once is not enough!

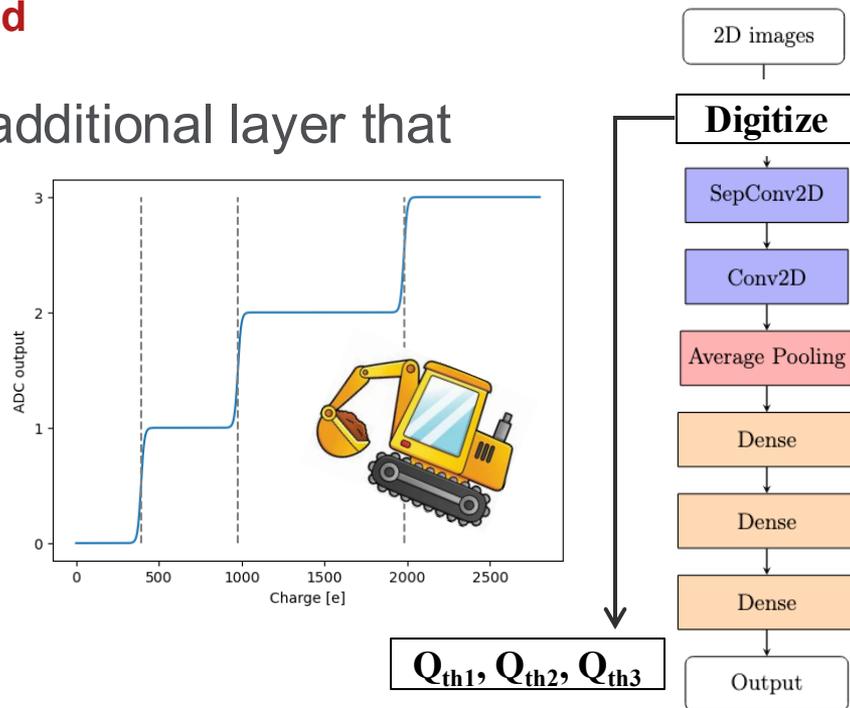


- Max Conv2D (20t)
- Full Conv2D (20t)
- Full Conv2D (2t)



# Reality check: digitization

- Our ASIC will not measure signal charge with electron-level precision
  - Need to **digitize** with a **realistic threshold**
- Strategy: train the network with one additional layer that performs input digitization
  - Just for information, NOT for implementation on-ASIC!
  - Treat thresholds  $Q_{th1}$ ,  $Q_{th2}$ ,  $Q_{th3}$  as trainable parameters
  - Add 80e Gaussian noise to the training dataset so  $Q_{th1}$  won't be too low



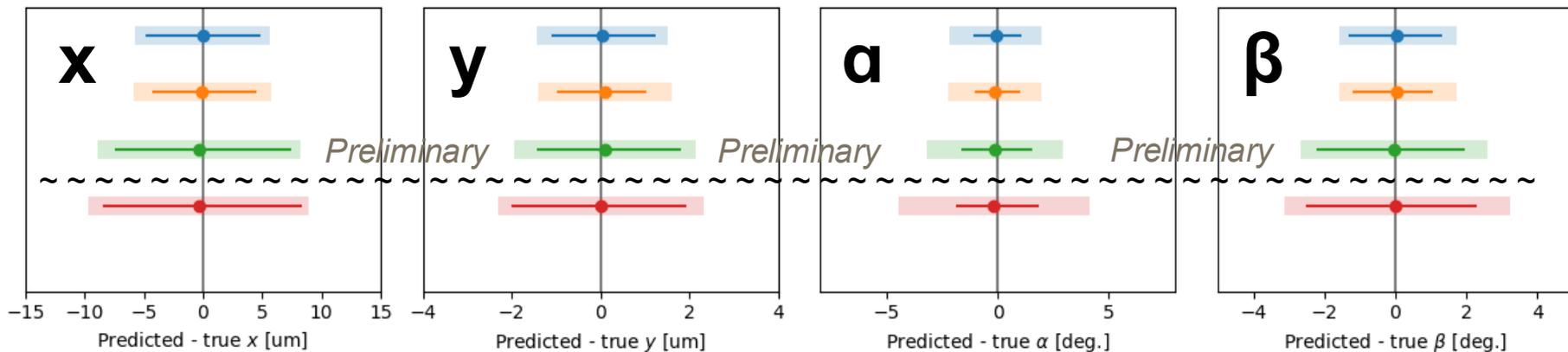
# Reality check: digitization

- With preliminary values for digitization:



00	$(-\infty, 392] e$
01	$(392, 977] e$
10	$(977, 1980] e$
11	$(1980, \infty) e$

- Max Conv2D (20t)
- Full Conv2D (20t)
- Full Conv2D (2t)
- Full Conv2D (digitized)

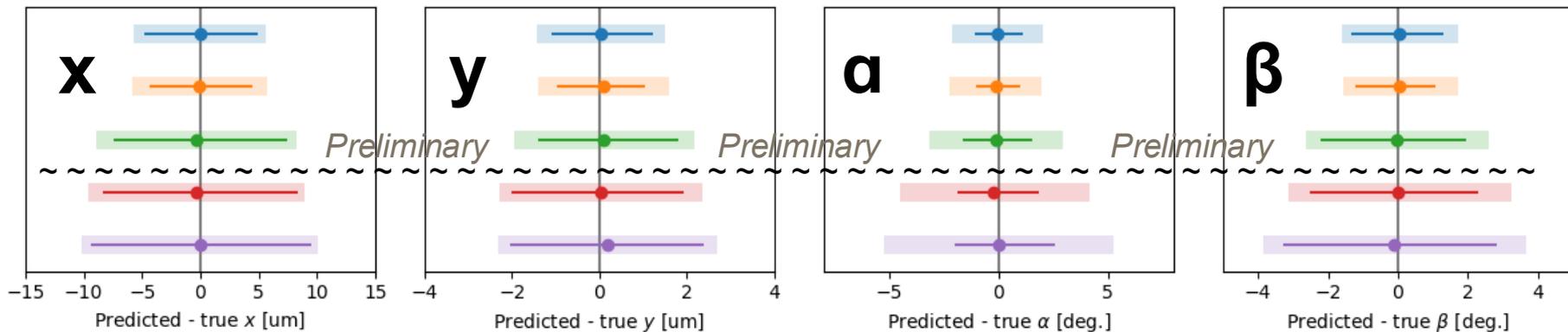


# Reality check: quantization

- Our ASIC will not store parameters with 32-bit floating point precision
  - Need to **quantize** all neural network weights and biases
- A first guess for this model:
  - Conv2d layers: 4-bit, Dense layers: 8-bit

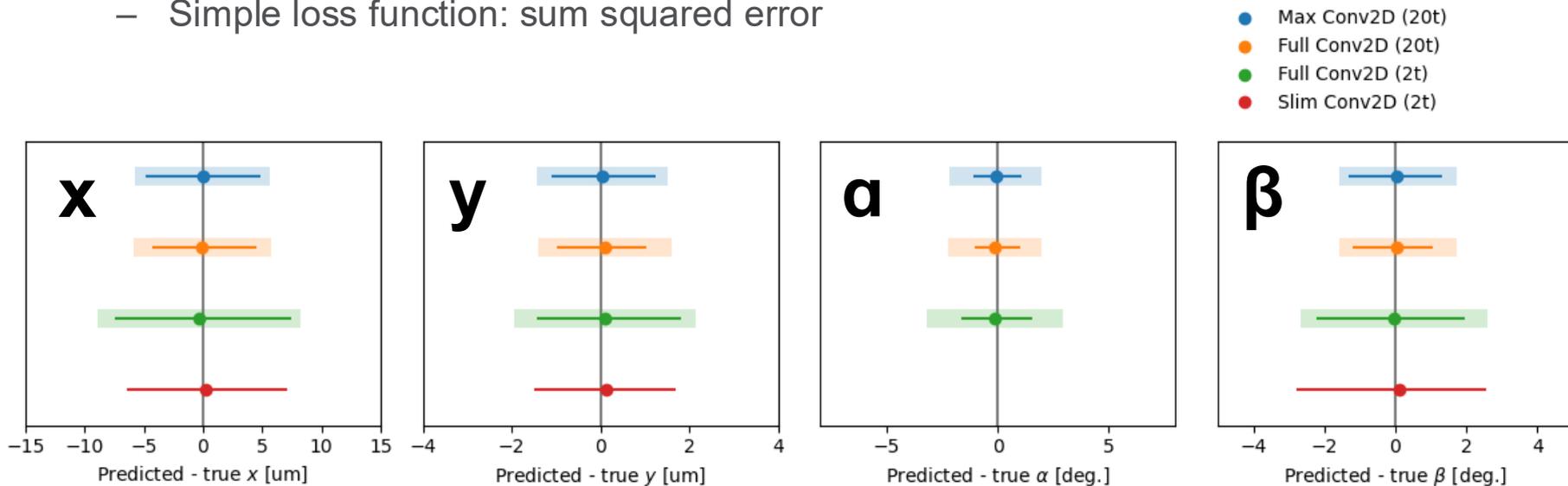


- Max Conv2D (20t)
- Full Conv2D (20t)
- Full Conv2D (2t)
- Full Conv2D (digitized)
- Full Conv2D (quantized)



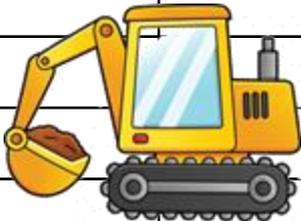
# Reality check: bandwidth

- Cannot transfer 10 8-bit numbers / cluster to the trigger every 25 ns
  - But 16 bits every 25 ns could be achievable
- Train a **slim model** that predicts only  $x$ ,  $y$ ,  $\beta$ 
  - Simple loss function: sum squared error



# On-ASIC resources

- What happens if we synthesize these models for 28nm CMOS?

Model	Architecture	# Inputs	# Outputs	# Parameters	Area [mm <sup>2</sup> ]	Power
Max (20t)	Conv 2D	5120	14	2121		
Full (20t)	Conv 2D	5120	8	2019		
Full (2t)	Conv 2D	512	8	1767		
Slim (2t)	Conv 2D	512	3	4322		
Slim (2t)	MLP, 16-bit inputs	512	3	2179		0.94 <i>Preliminary</i> ~ ~ ~ ~ ~ ~ ~ ~

28 nm CMOS



Slim models can still be further reduced!

# Summary

- On-chip intelligence has great potential to **reduce pixel detector data rates** at collider experiments
- **Regression** of the parameters we want for tracking can be performed on-ASIC and read off directly
  - Depending on bandwidth and physics needs, can provide some combination of hit positions and track angles
- More work to do, especially **synthesis and ASIC resource studies**
  - One year of seed funding awarded to Fermilab (DE-SCL0000038)
- Targeting a tape-out in 2027 (VIAS)

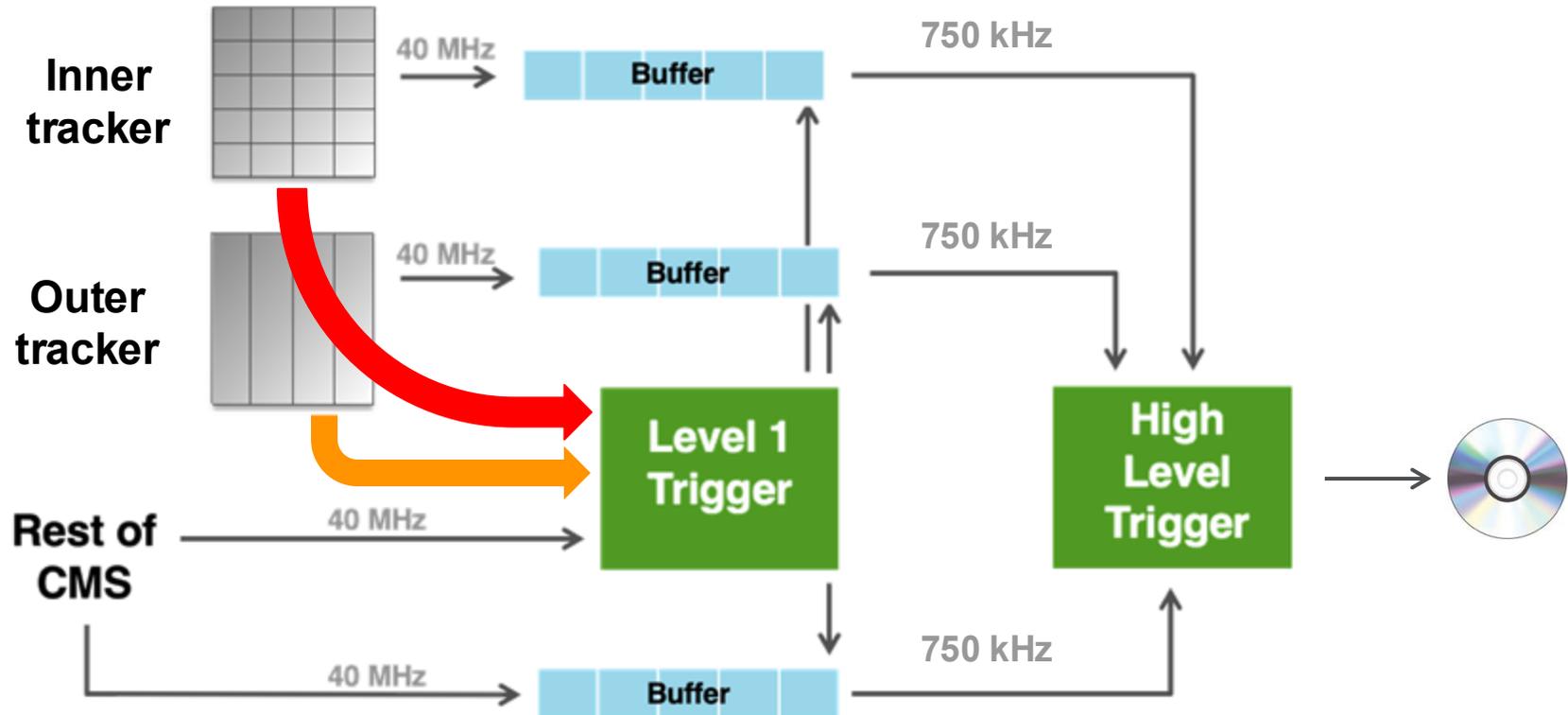
# The *smartpixels* team

- **Cornell University** Jennet Dickinson, Ben Weiss, Albert Zhou
- **Fermilab:** Douglas Berry, Giuseppe Di Guglielmo, Farah Fahim, Abhijith Gandrakota, Lindsey Gray, Jim Hirschauer, Ron Lipton, Benjamin Parpillon, Chinar Syal, Nhan Tran
- **Johns Hopkins University:** Petar Maksimovic, Morris Swartz
- **Northeastern University:** Nick Manganelli
- **Oak Ridge National Laboratory:** Shruti R. Kulkarni, Aaron Young
- **Purdue University:** Mia Liu, **Arghya Ranjan Das, Shiqi Kuang, Ana Sofía Calle Muñoz**
- **University of Chicago:** Karri DiPetrillo, Anthony Badea, Eliza Howard, Daniel Abadjiev
- **University of Colorado Boulder:** Keith Ulmer, Jannicke Pearkes
- **University of Illinois Chicago:** Corrinne Mills, Danush Shekar, Mohammad Abrar Wadud, Jieun Yoo
- **University of Illinois Urbana-Champaign:** Mark S. Neubauer, **David Jiang**

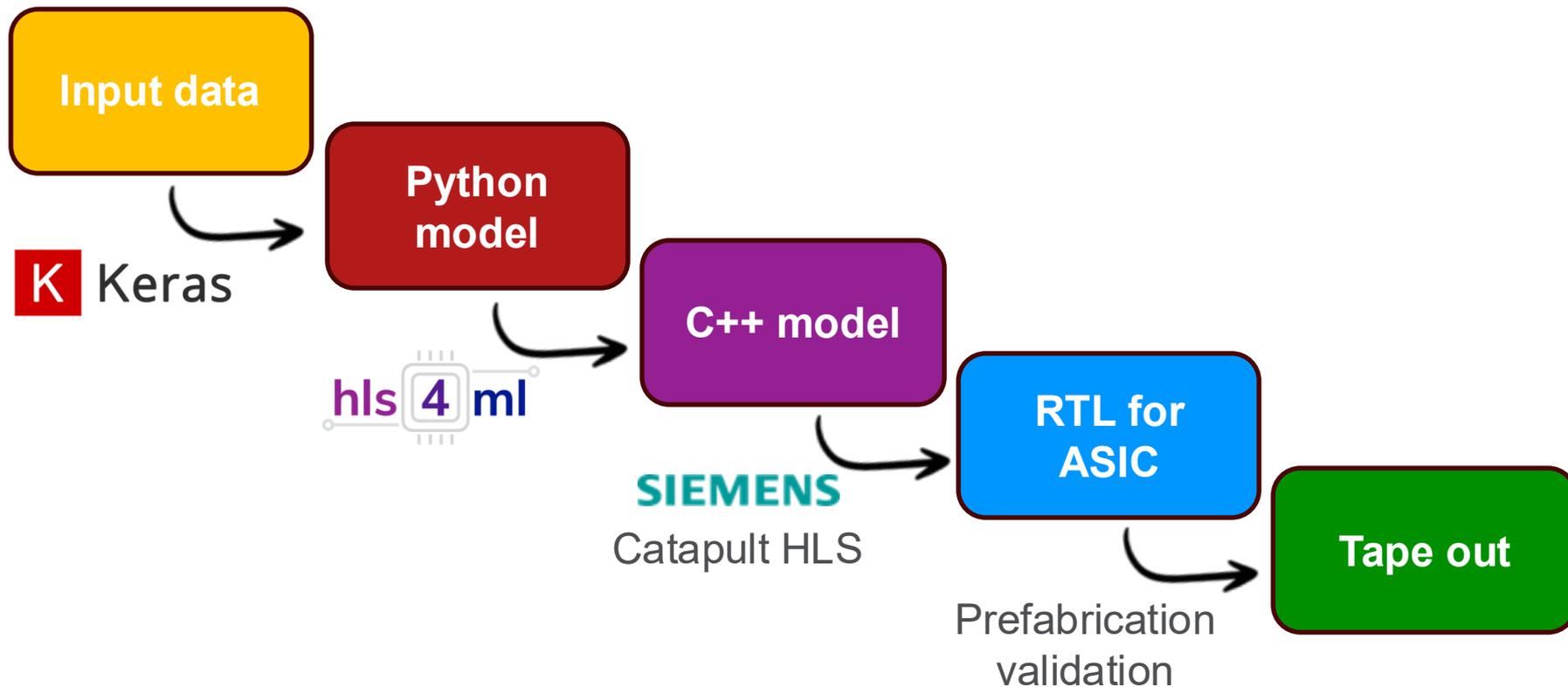


# Backup

# CMS with smartpixels



# Hardware design flow



# Regressing an uncertainty

- When training an ML algorithm, you minimize a **loss function**
- One common choice of loss function is mean-squared error (MSE)

$$\frac{1}{N} \sum_{i=1}^N (y_i - \mu_i)^2$$

This is equivalent to assuming each training input has a unit-Gaussian likelihood, and minimizing NLL to predict mean  $\mu_i$

- Can generalize to a Gaussian of any width

$$\frac{1}{N} \sum_{i=1}^N \left( \frac{y_i - \mu_i}{\sigma_i} \right)^2$$

Minimize this to predict mean  $\mu_i$  and uncertainty  $\sigma_i$  simultaneously

- Can generalize further: choose any likelihood and minimize NLL

# Hardware design flow

