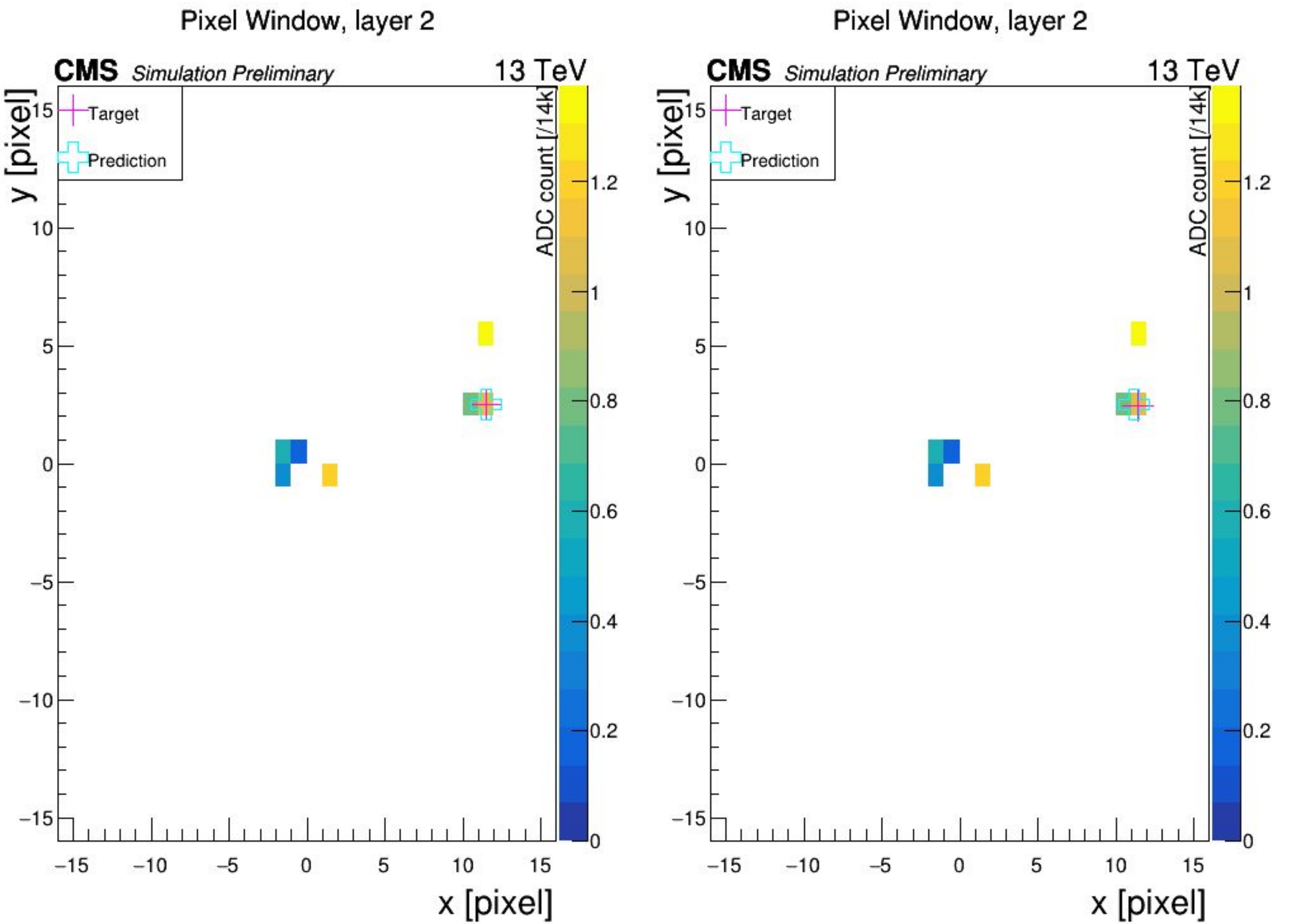
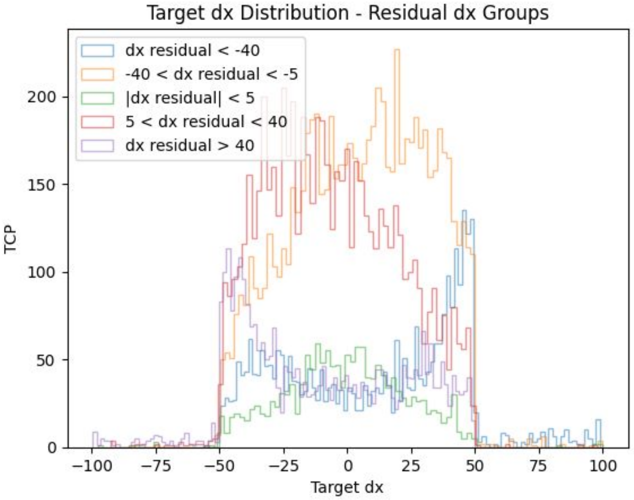
Résumé of Summer Research

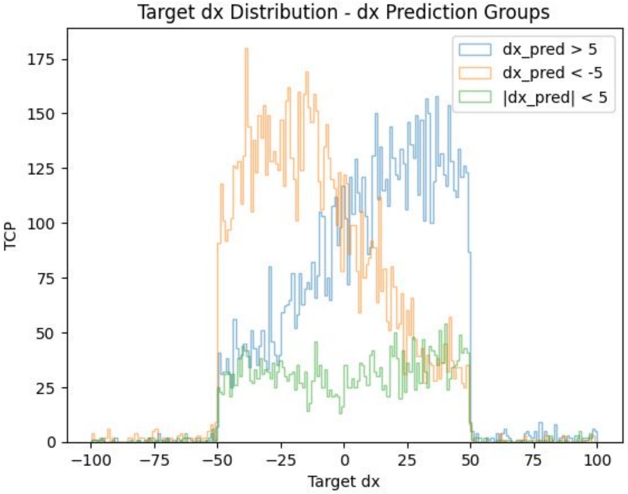
Liam O’Shaughnessy - 25/09/23

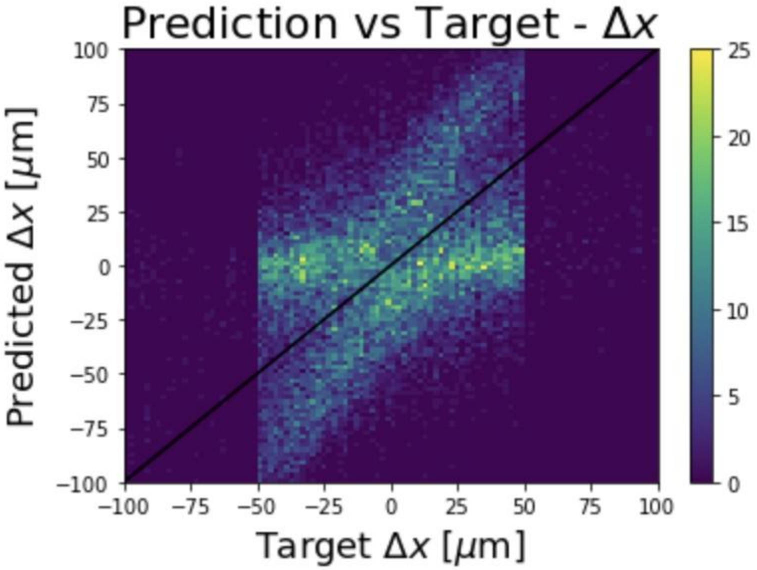

Pixel Window, layer 2
CMS Simulation Preliminary
13 TeV
Target
Prediction
y [pixel]
x [pixel]
ADC count [/14k]
Pixel Window, layer 2
CMS Simulation Preliminary
13 TeV
Target
Prediction
y [pixel]
x [pixel]
ADC count [/14k]

Summary

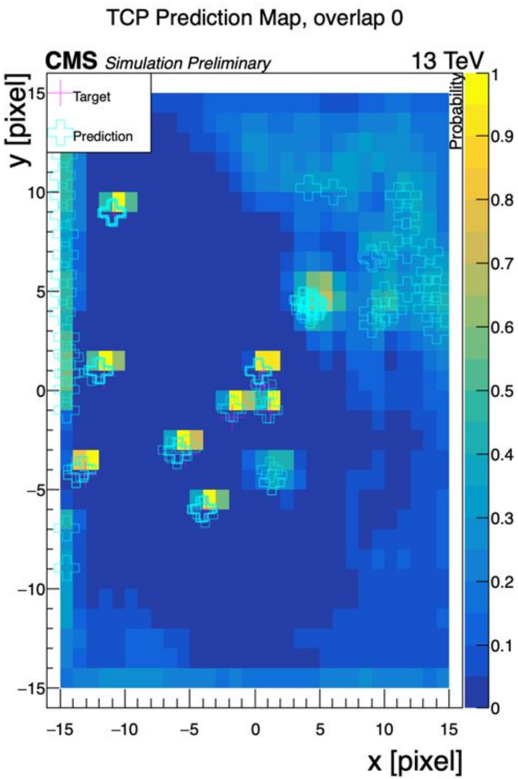
- Have been examining target values of track crossing point parameters (dx, dy, jet eta, etc.) compared to DeepCore predictions of these parameters
- Have been looking at interesting cases where DeepCore fails to predict correctly and why e.g. the dx boomerang, the TCP plots with odd boundaries
- Trying to break these down to see if odd splits can be explained by TCP's being in certain parameter regime, having charge issues, etc.

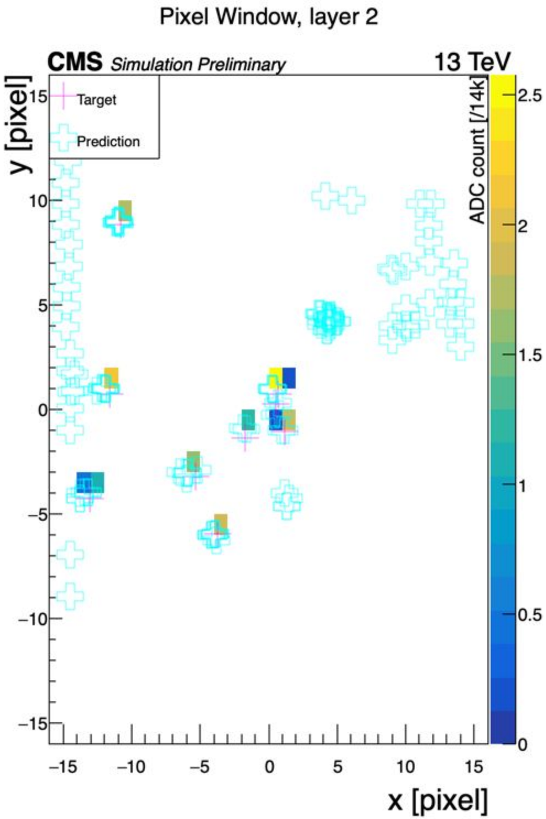




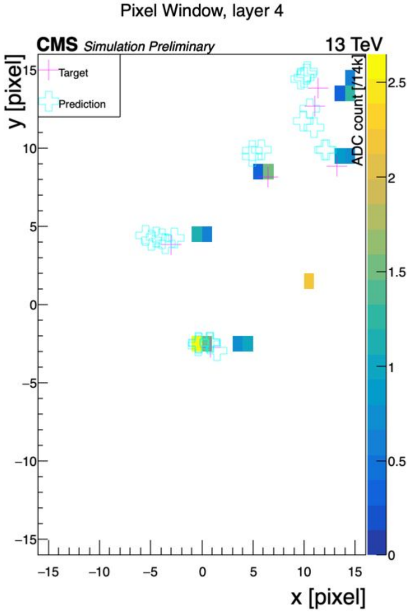


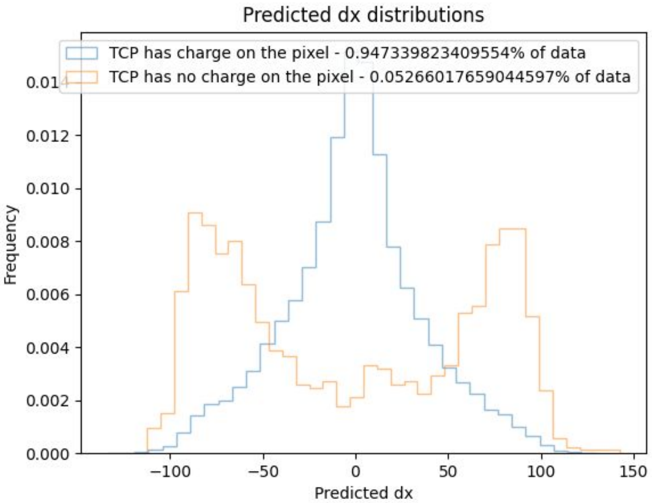
Tried to split the boomerangs by residual, by positive/negative/zero dx prediction, by eta and pt -> does not appear to cleanly split.

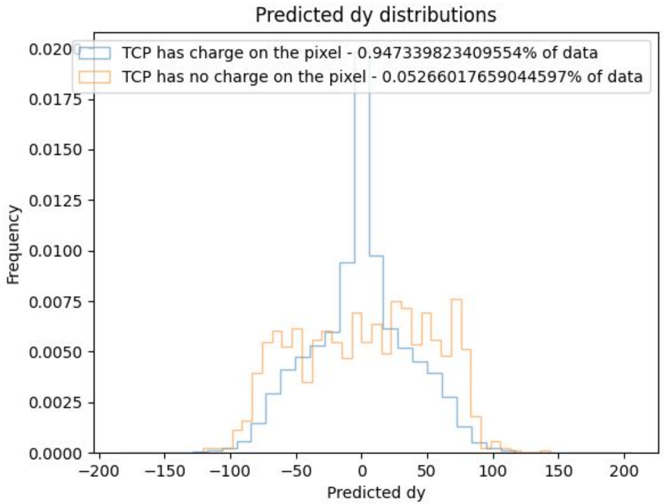


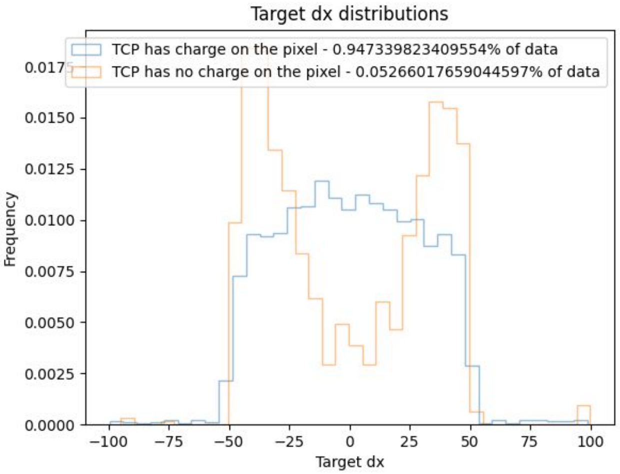


From the DeepCore visualizer, these had a lot of bad predictions at lower thresholds along the edges, tried to figure out what was going on here. When I raised the threshold on the TCP check, I believe many of the weird ones disappeared, but can recheck this. This also prompted question about if the linear propagation is causing issues/is way off.









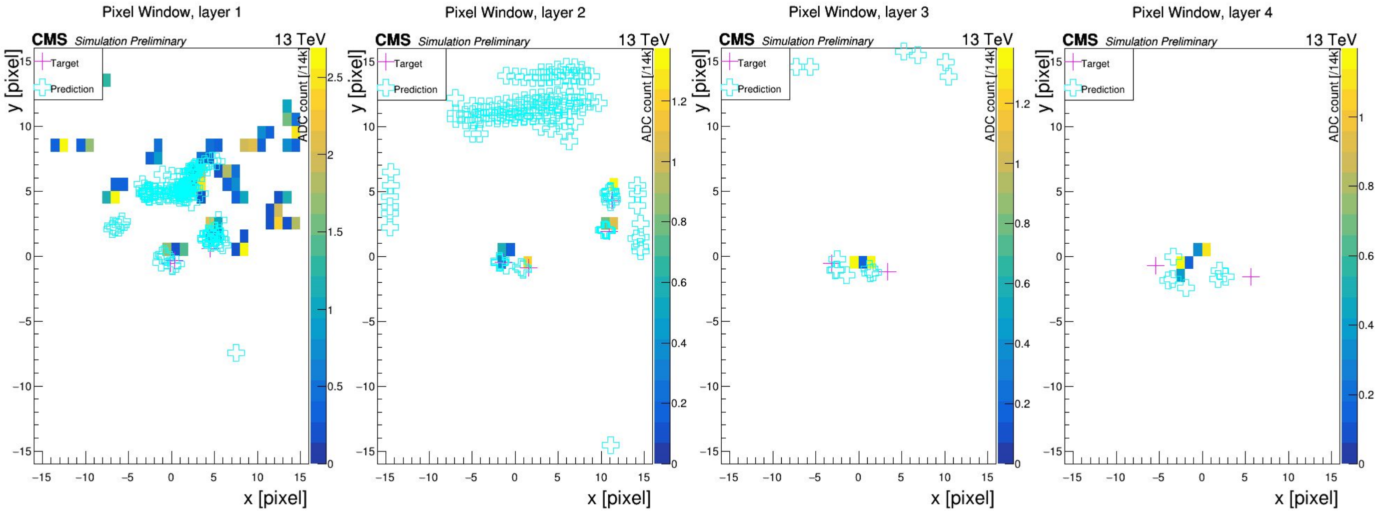
At the end, looked at if "chargeless" TCP's were impacting - certainly have different distributions but only 5% of the TCP's we get are chargeless

Current Tasks

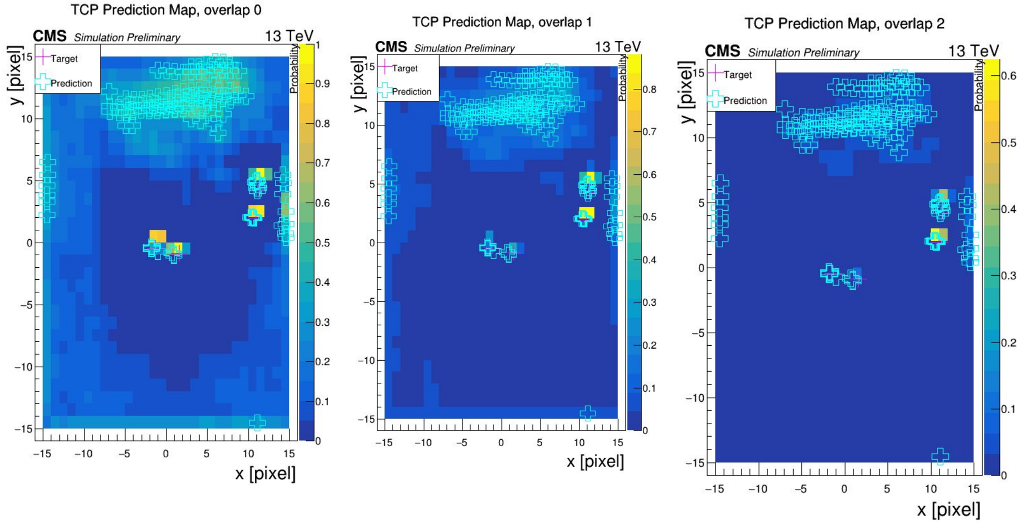
- New notebook with more data/methods
- Investigate propagation between layers - how to improve on linear prop
- Visualizer checking

Linear Propagation

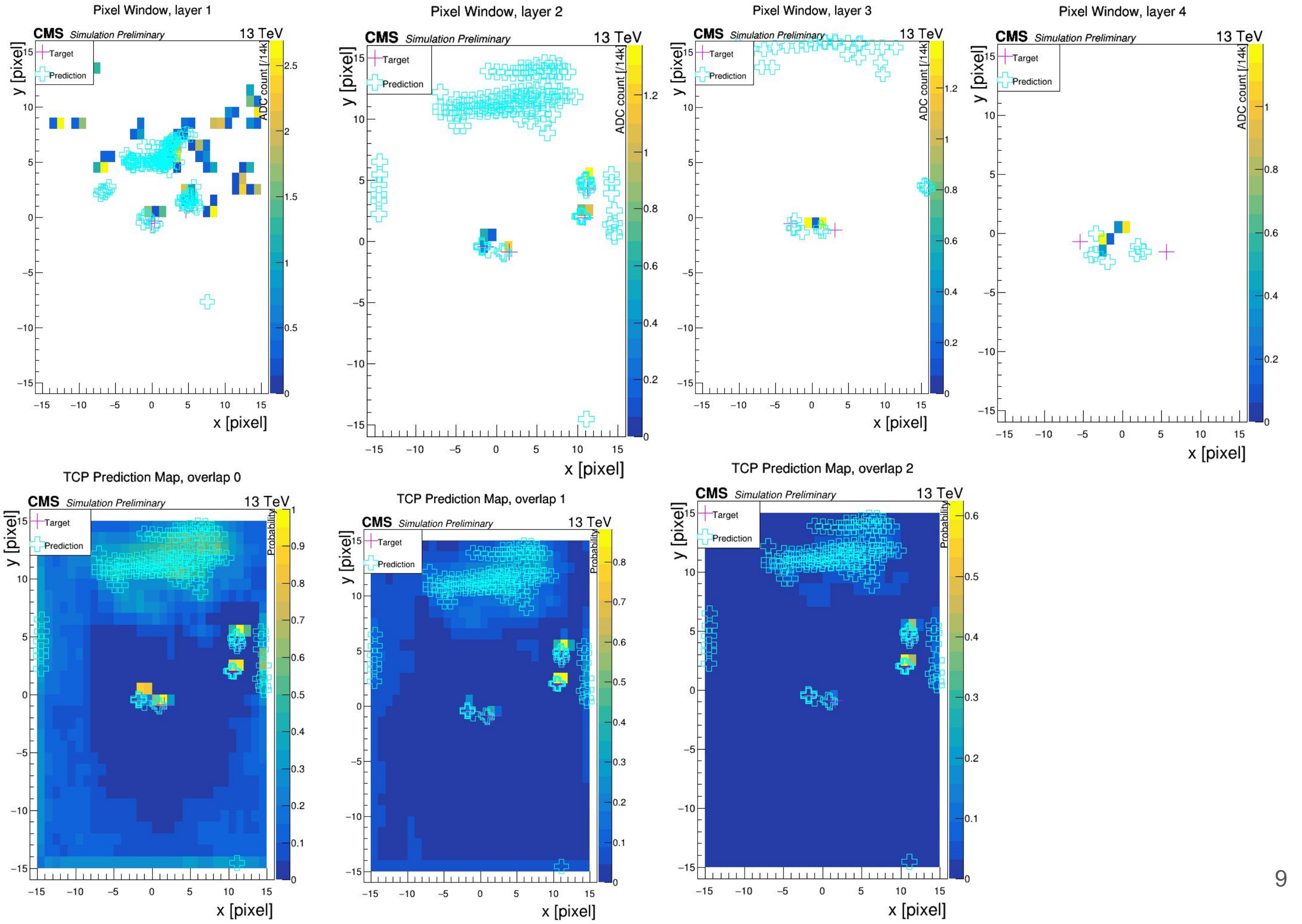
“The four barrel layers are radially located at r_1 = 29 mm, r_2 = 68 mm, r_3 = 109 mm, and r_4 = 160 mm” - the r1 was set at 30, the r3 was set at 102. “Pixel detector consists of 66M pixels (100 × 150 μm)” - these values are correct for x and y distances of pixels

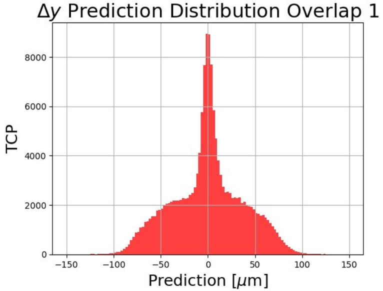


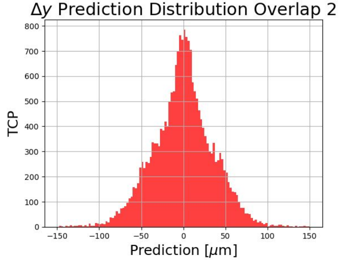
Plots with new lin prop

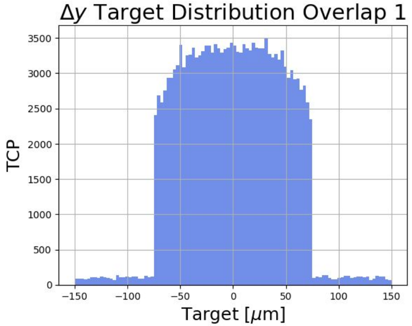


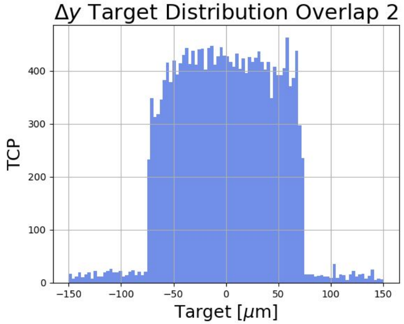
Plots with old lin prop

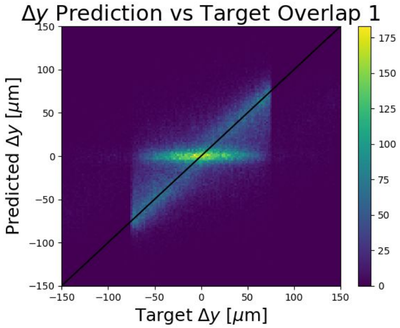


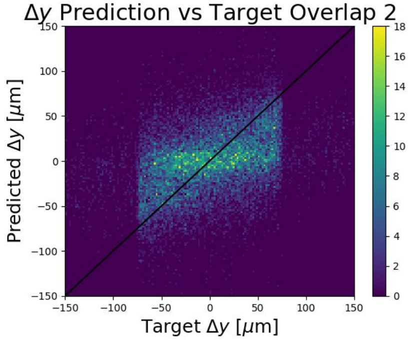




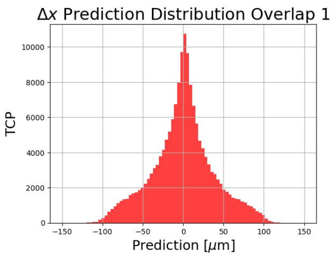


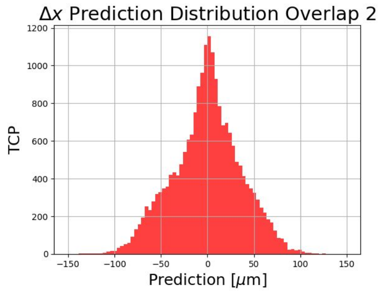


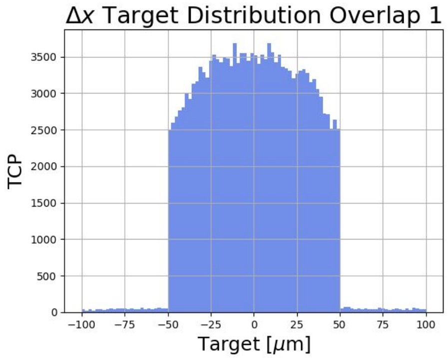


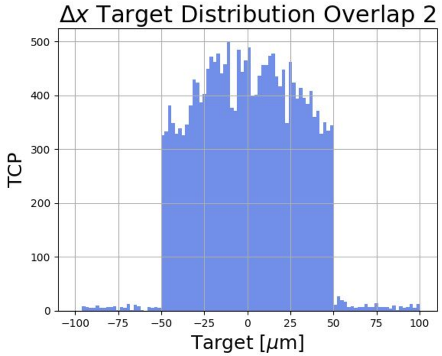


Targ + pred both sent to 0 for overlap 3

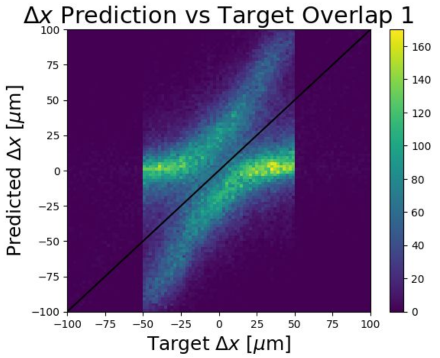


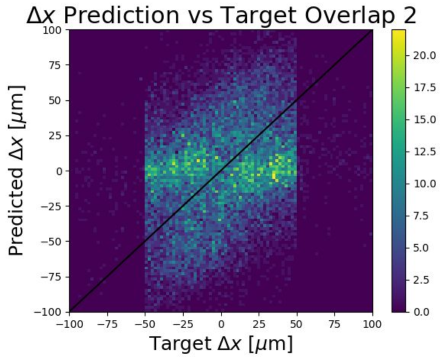


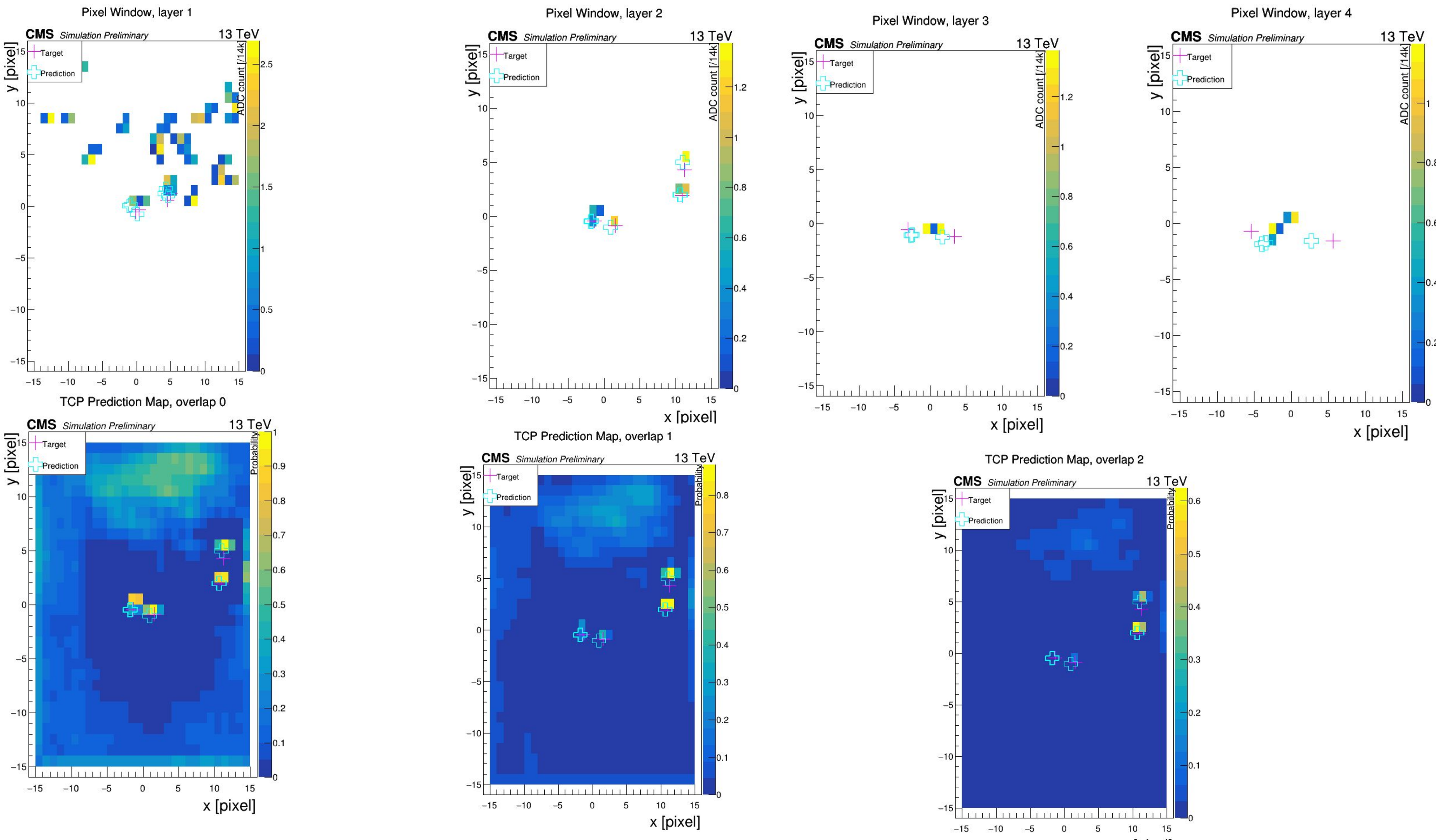


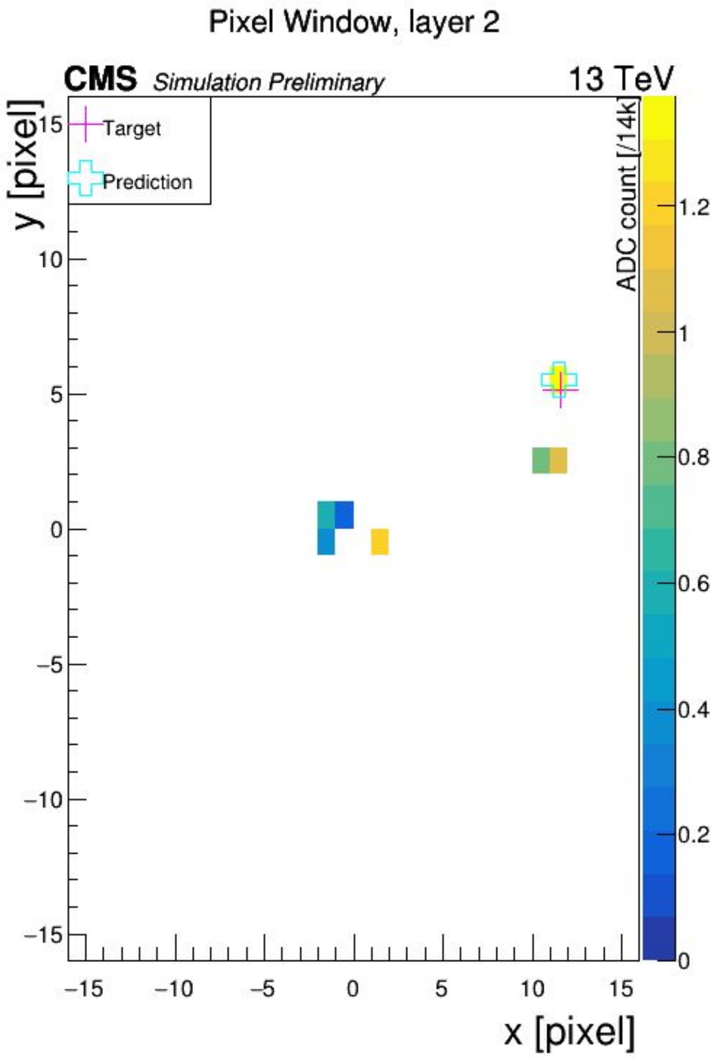


Targ + pred both sent to 0 for overlap 3





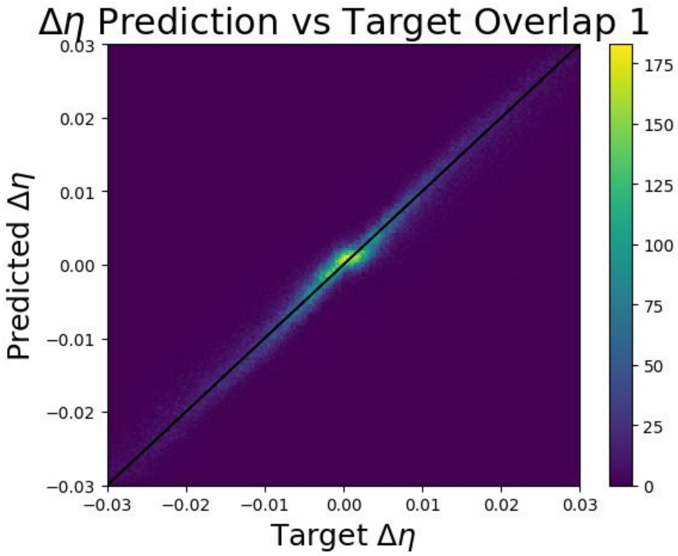

Pixel Window, layer 1
Pixel Window, layer 2
Pixel Window, layer 3
Pixel Window, layer 4
TCP Prediction Map, overlap 0
TCP Prediction Map, overlap 1
TCP Prediction Map, overlap 2
CMS Simulation Preliminary
13 TeV
Target
Prediction
ADC count [/14k]
Probability
x [pixel]
y [pixel]

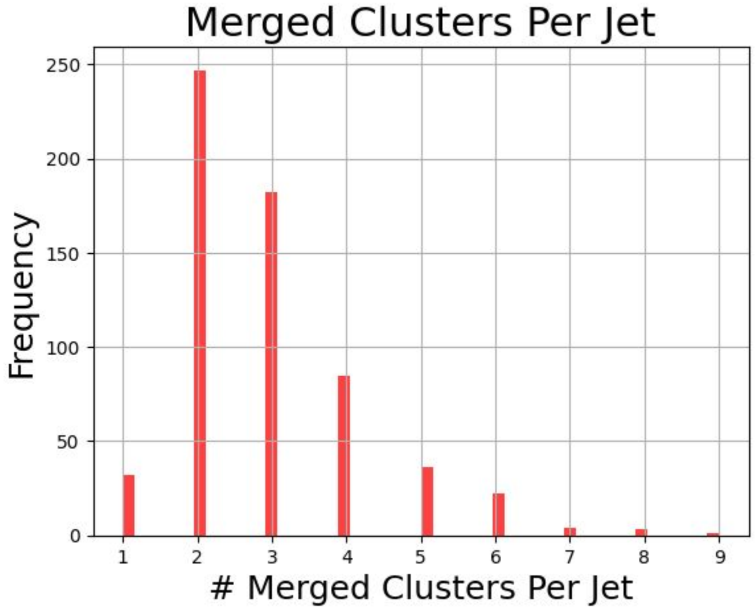


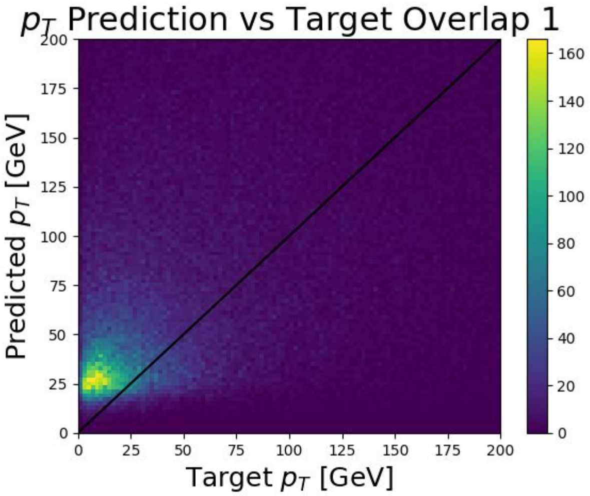
target(x,y,eta,phi)=
0.24070096015930176
-1.0722055435180664
-1.1713344563576173
-1.5955027470730188

prediction(x,y,eta,phi)=
0.028765248134732246
-0.003075879067182541
-1.2516465187072754
-1.7314609289169312

(big pixel)


Δη Prediction vs Target Overlap 1
Predicted Δη
Target Δη

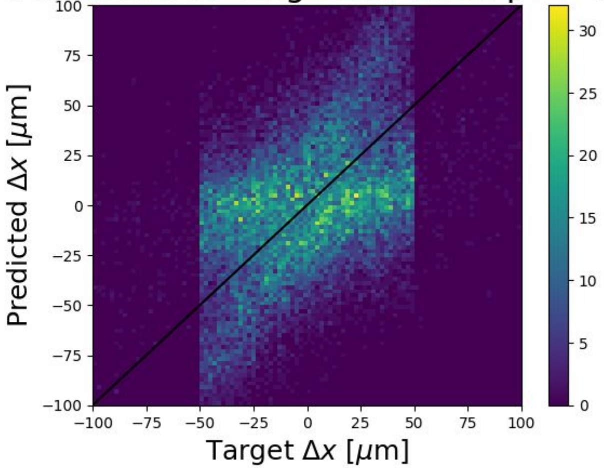

Merged Clusters Per Jet
Frequency
Merged Clusters Per Jet


p_T Prediction vs Target Overlap 1
Predicted p_T [GeV]
Target p_T [GeV]

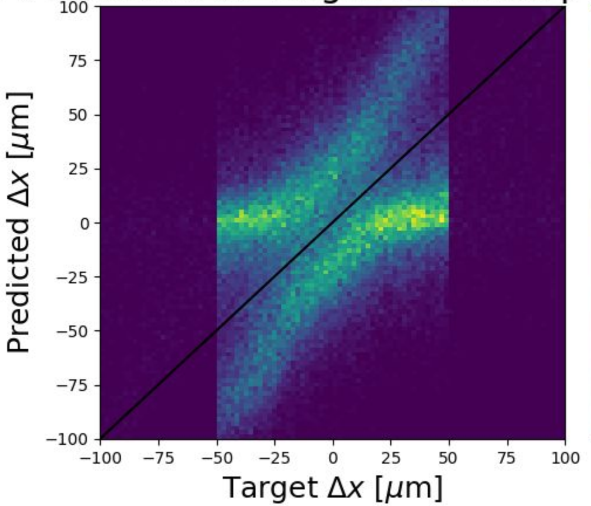
0.8125

1.0

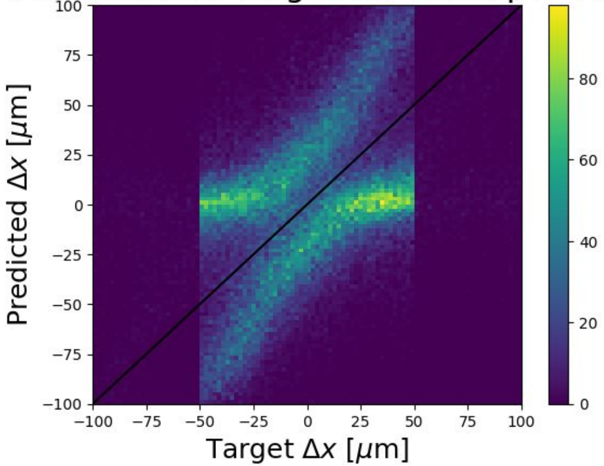
Δx Prediction vs Target for Track pT > 100



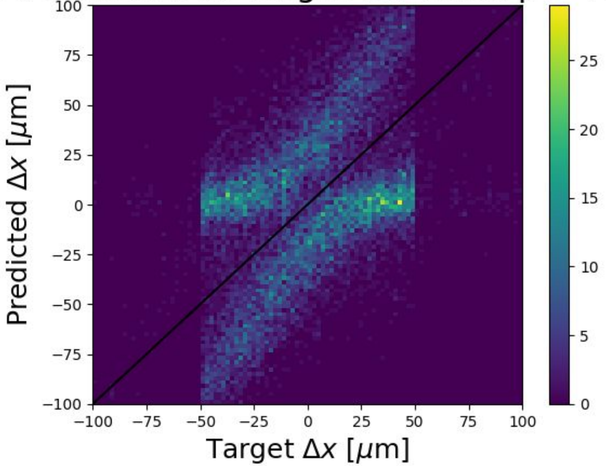
Δx Prediction vs Target for Track pT > 10

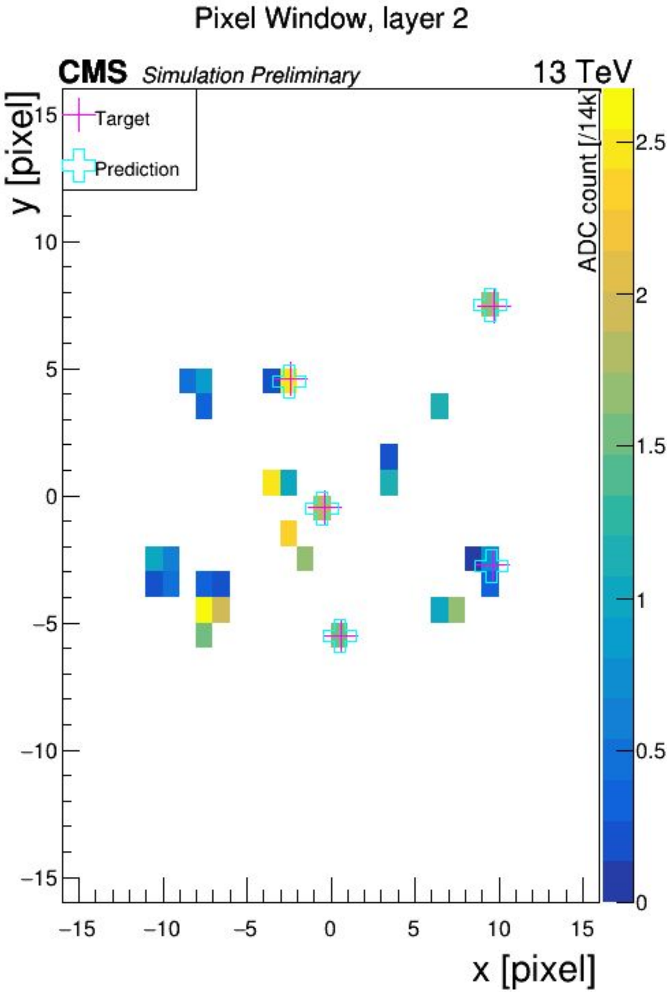


Δx Prediction vs Target for Track pT < 100



Δx Prediction vs Target for Track pT < 10

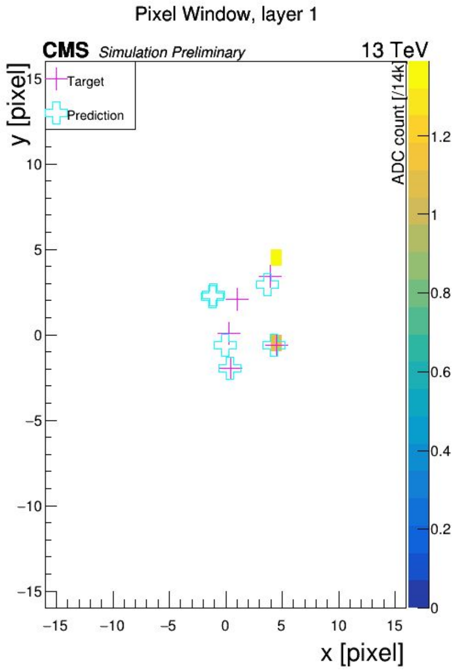


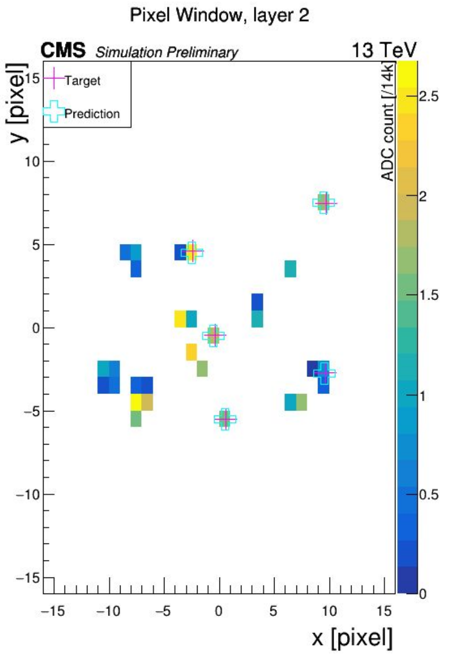


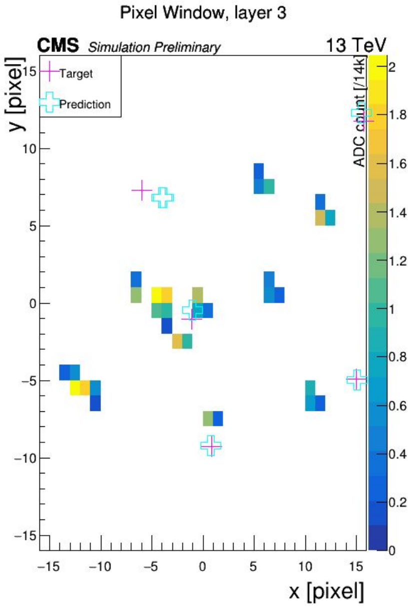
New Pred, bin (x,y): -3.0 4.0
target(x,y,eta,phi,pt)=
0.20136487483978271
0.355228066444397
-0.9074209666935618
0.8715723045369828
2.3911147441465905
prediction(x,y,eta,phi,pt)=
0.013919083401560783
-0.07324044406414032
-0.7944803237915039
0.3417768180370331
44.34525680541992

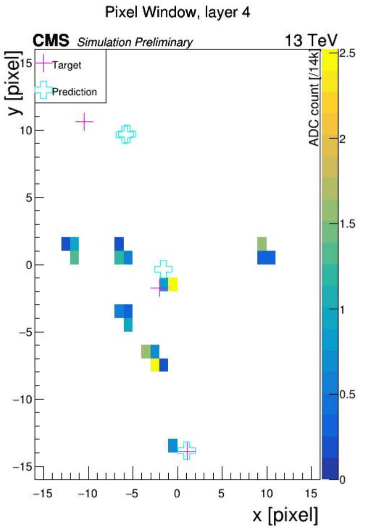
New Pred, bin (x,y): 0.0 -6.0
target(x,y,eta,phi,pt)=
0.22538554668426514
-0.05873811244964597
1.266343376705109
-0.05119968558644494
95.07100903743853
prediction(x,y,eta,phi,pt)=
0.08322035521268845
0.04299357533454895
1.2575995922088623
-0.05193641781806946
42.94028091430664

Add jet pt and jet eta

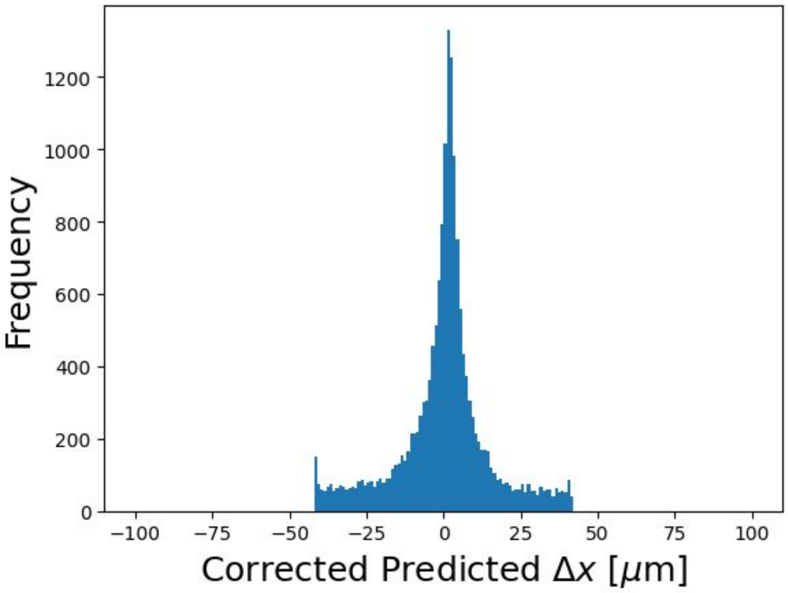








Add pred and targ dx, pt

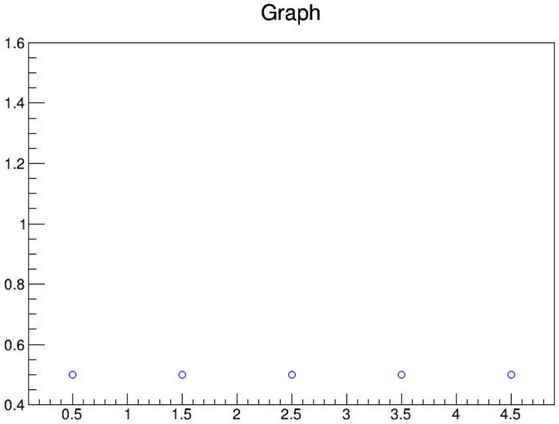


5th degree polynomial fit - extended range

Probability-based color gradient for prediction crosses

This is not working to change the color of the points - test case:

```
for i in range(num_points):
        x = i + 0.5
        y = 0.5
        color = ROOT.kBlue + i
        points.SetPoint(i, x, y)
        points.SetMarkerStyle(20)
        points.SetMarkerColor(color)
points.Draw("AP")
```



Why are these all the same color?

```cpp
class TAttMarker {

protected:
   Color_t    fMarkerColor;       ///< Marker color
   Style_t    fMarkerStyle;       ///< Marker style
   Size_t     fMarkerSize;        ///< Marker size

public:
   TAttMarker();
   TAttMarker(Color_t color, Style_t style, Size_t msize);
   virtual ~TAttMarker();
           void     Copy(TAttMarker &attmarker) const;
   virtual Color_t  GetMarkerColor() const {return fMarkerColor;} ///< Return the marker color
   virtual Style_t  GetMarkerStyle() const {return fMarkerStyle;} ///< Return the marker style
   virtual Size_t   GetMarkerSize()  const {return fMarkerSize;}  ///< Return the marker size
   virtual void     Modify();
   virtual void     ResetAttMarker(Option_t *toption="");
   virtual void     SaveMarkerAttributes(std::ostream &out, const char *name, Int_t coldef=1, Int_t stydef=1, Int_t sizdef=1)
   virtual void     SetMarkerAttributes();  // *MENU*
   virtual void     SetMarkerColor(Color_t mcolor=1) { fMarkerColor = mcolor;} ///< Set the marker color
   virtual void     SetMarkerColorAlpha(Color_t mcolor, Float_t malpha);
   virtual void     SetMarkerStyle(Style_t mstyle=1) { fMarkerStyle = mstyle;} ///< Set the marker style
   /// Set the marker size.
   /// Note that the marker styles number 1 6 and 7 (the dots), cannot be scaled.
   /// They are meant to be very fast to draw and are always drawn with the same number of pixels;
   /// therefore this method does not apply on them.
```

```cpp
////////////////////////////////////////////////////////////////////////////////
/// Set x and y values for point number i.

void TGraph::SetPoint(Int_t i, Double_t x, Double_t y)
{
   if (i < 0) return;
   if (fHistogram) SetBit(kResetHisto);

   if (i >= fMaxSize) {
      Double_t **ps = ExpandAndCopy(i + 1, fNpoints);
      CopyAndRelease(ps, 0, 0, 0);
   }
   if (i >= fNpoints) {
      // points above i can be not initialized
      // set zero up to i-th point to avoid redefinition
      // of this method in descendant classes
      FillZero(fNpoints, i + 1);
      fNpoints = i + 1;
   }
   fX[i] = x;
   fY[i] = y;
   if (gPad) gPad->Modified();
}
```

X and Y arrays for graph, access single pointer for color

How CERN does it:

```
if (not ON_DATA) :
        graphTargetTot[jet][1].Draw("SAME P")
graphPredTot[jet][1].Draw("SAME P")
```

ROOT forums say it is impossible to do multiple colors in the same graph, would need a TMultiGraph/overlay graphs