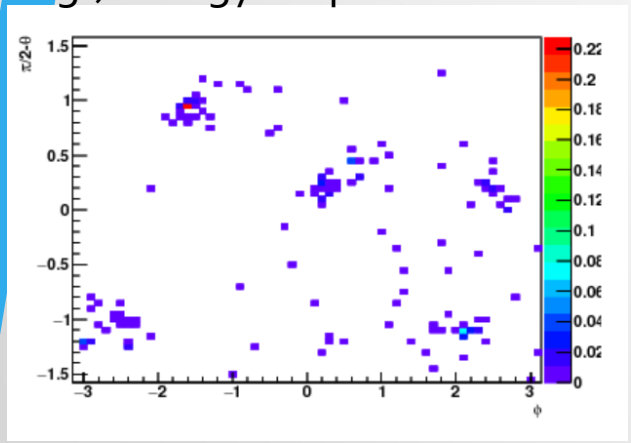
Trial

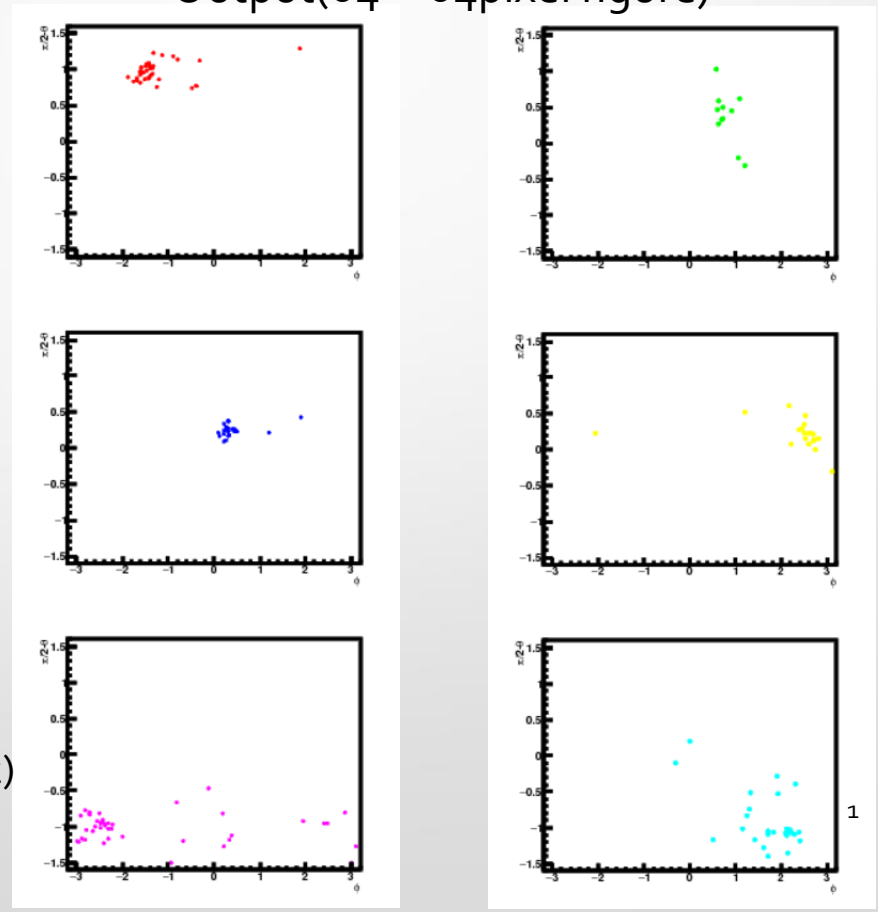
- Use Keras & tensorflow backend
- Using a certain map(s) of each event, estimate color of each track
  - Do not consider color-singlet state

Input(64 × 64pixel figure)
e.g.) energy map



9 input images
Energy map
Charge map
Dosig map + direction vector(x, y, z)
Zosig map
Ecal map
Hcal map

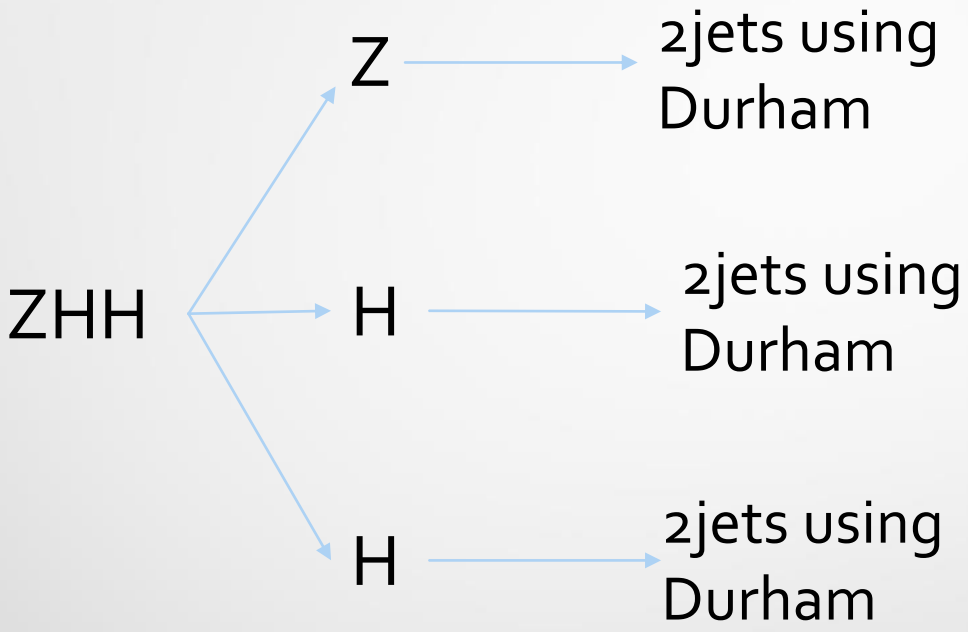
Output(64 × 64pixel figure)



+ no particle

Create answer

- Supervised learning - Create "answer" jets: perfect Durham jet clustering



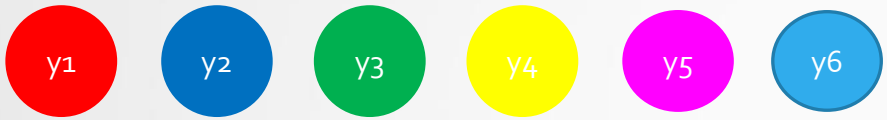
So far, do not consider color singlet state: number of jets is 6

ZHH→(qq)(bb)(bb)→6jets

Pseudo-labelling

- Output: inference of the probability of the color to be assigned
  - $\sum y_i = 1.0$

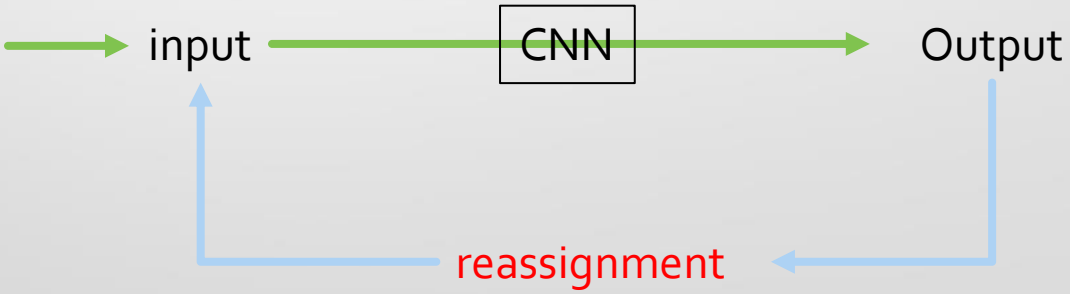
output



+ no particle

- The combination of color assignments is arbitrary, so assign them so that the loss function is minimized.
  - Using preliminary results after a training, re-assign the color combination
  - Minimize cross entropy $L = \frac{1}{n}\sum y_i \log p_j$

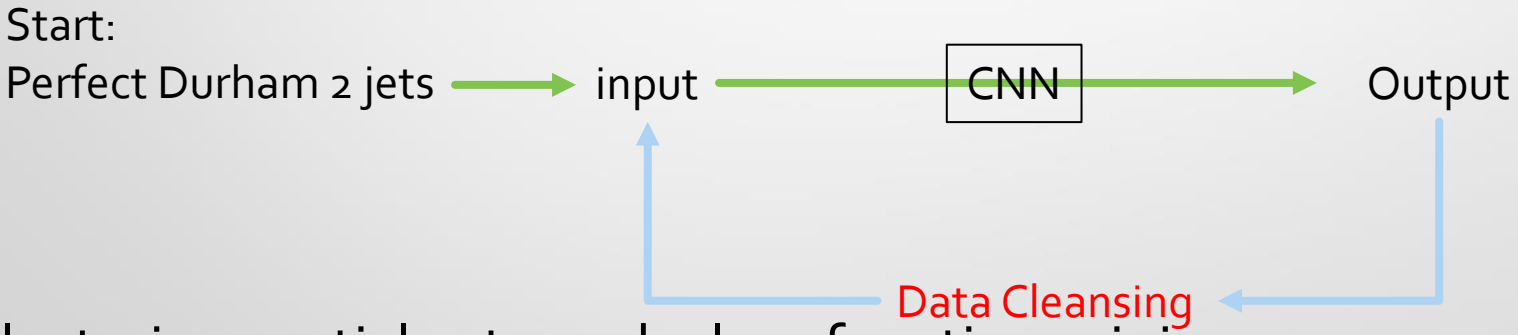
Start:
Energy ordering
Of jets



Data Cleansing

- Perfect Durham clustering is not always the best clustering into 2 jets for CNN

- By using the preliminary training weights, clustering into 2 jets is performed



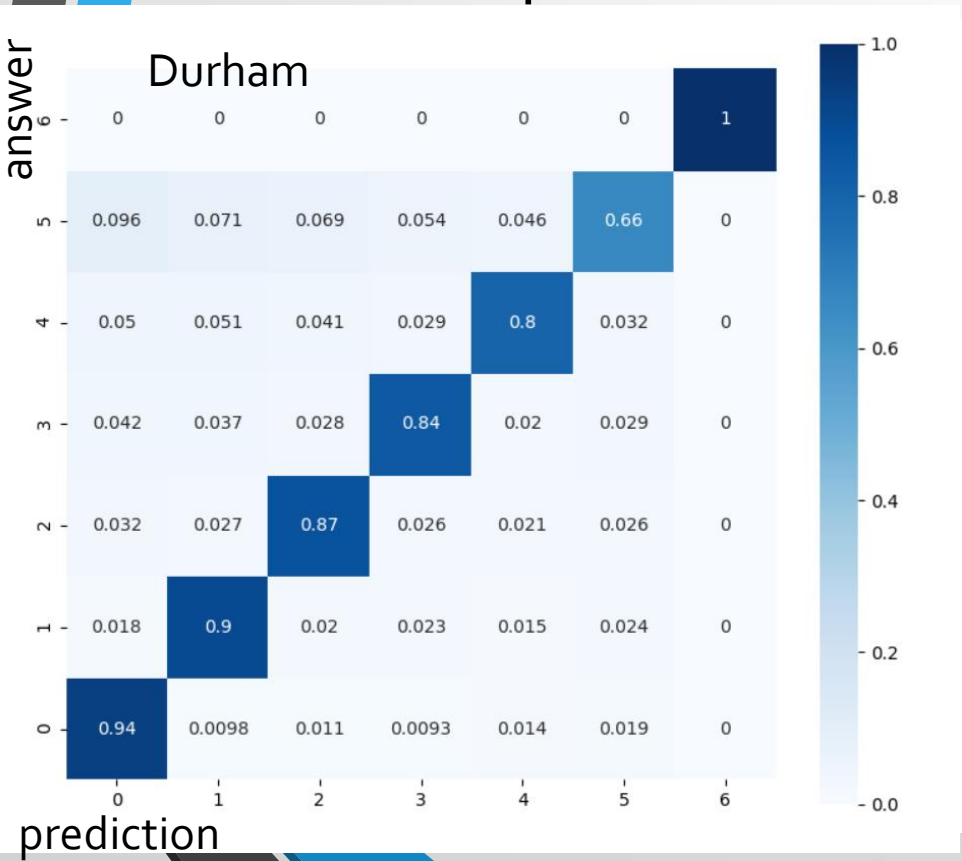
- Clustering particles to make loss function minimum
- First pseudo-labeling. After that, data cleansing
  - Is there better way?

status

- Use ZHH→(qq)(bb)(bb): 6jets clustering
  - q: uds
- Use 80000 events for training(72000 train, 8000 validation)
  - Very weak or no over fitting can be seen
- Don't consider color singlet state for network training
- Input: 6 + 3 images output: 6 + 1 images

Comparison with Durham

- Seems to start to exceed Durham performance!
  - Mis-clustering to adjacent jets
  - Need to remove it as much as possible
- Need to improve an efficiency in circle to get better performance



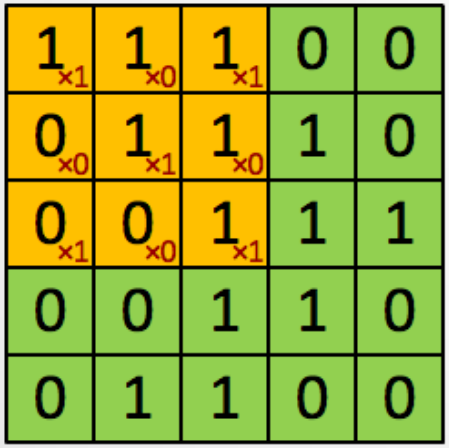
backups

Basics: convolution

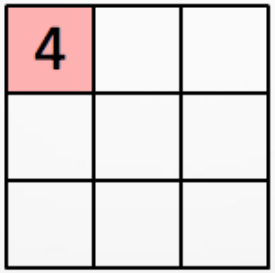
- Convolution: Apply the filters to extract the feature
  - Sum of the product of each pixel and filter weights:

$$y_{kl} = \sum_{i,j} w_{ij} \cdot x_{(k+i)(l+j)} \ (+b)$$

  - Slide filters over all the pixels



Image



Convolved Feature

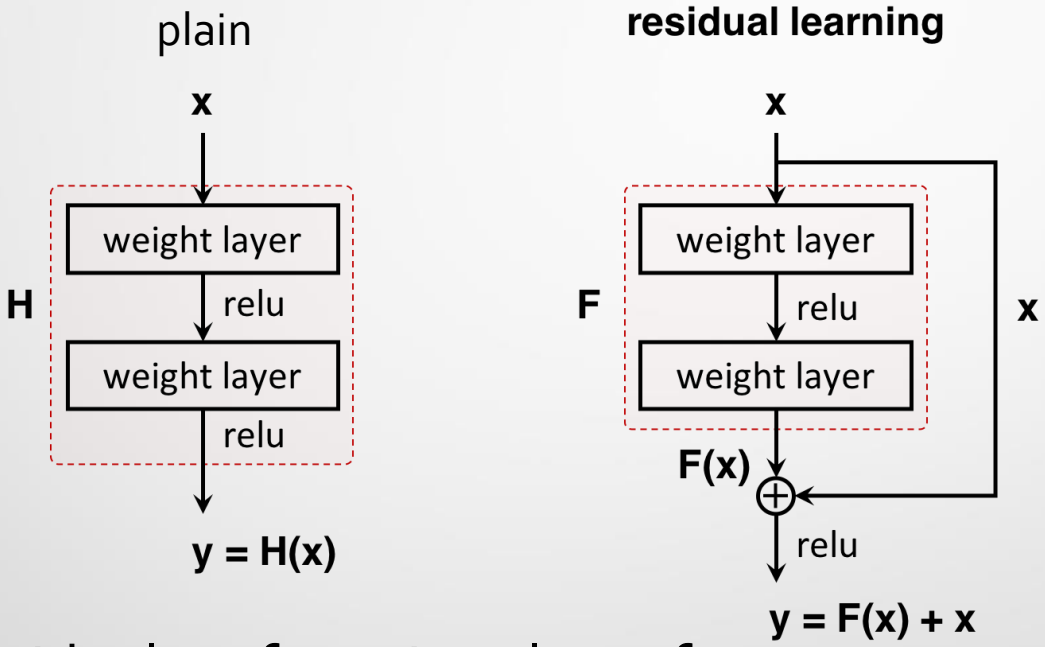
  - Filters are parameters: CNN can obtain them automatically
- After the convolutional operation, apply non-linear transform

$$z_{kl} = \sigma(y_{kl})$$

  - "Non-linear" is important to get good expression
- Stack these operations

Basics: Residual convolution

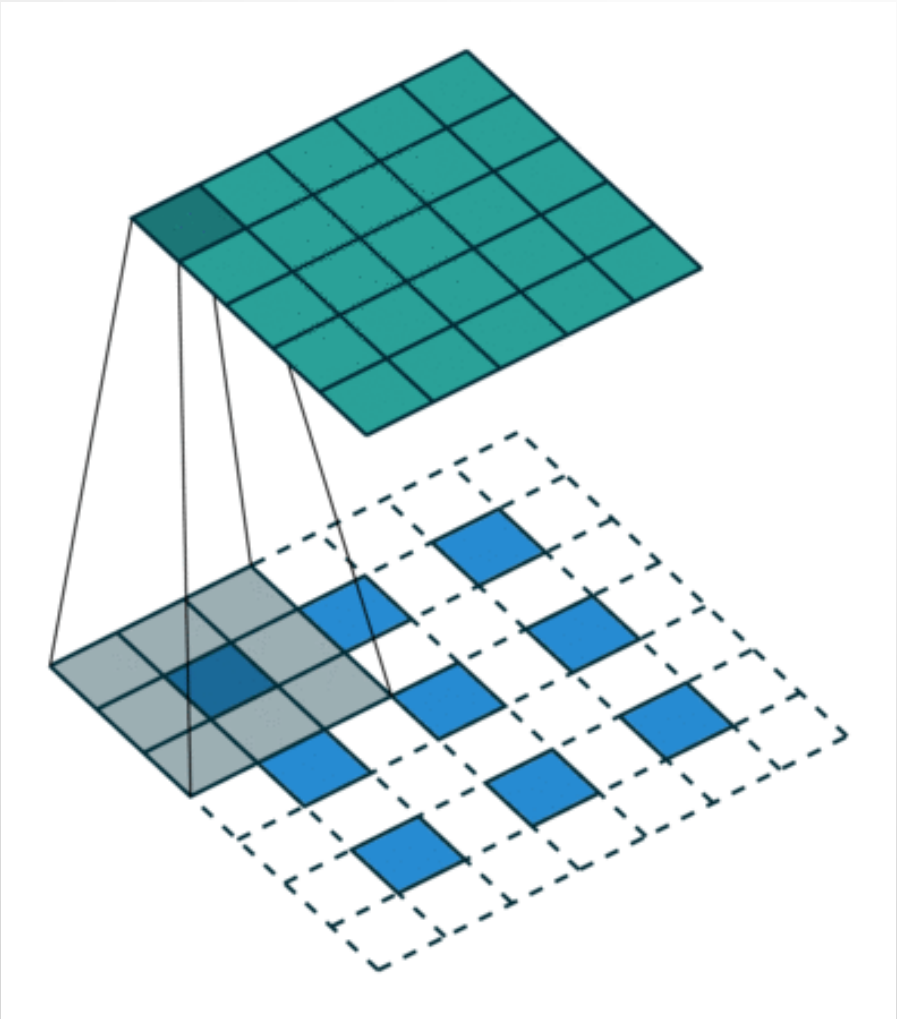
- Stream is divided into 2 paths:
  - Path with convolution
  - Path without any operation
- Sum up these 2 path in downstream



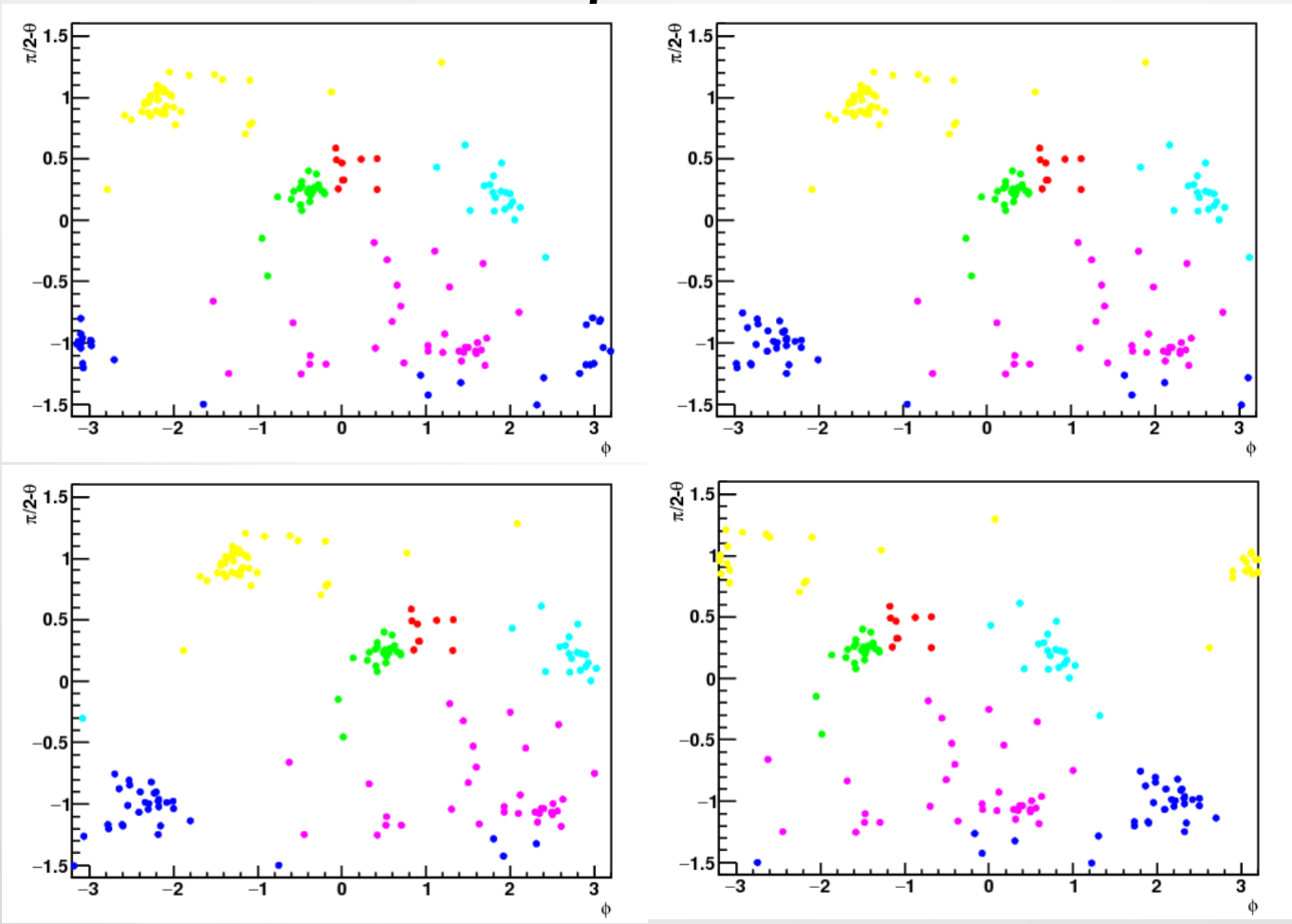
- Can learn "Residuals" of previous layer features
- Can construct very deep network
  - >100 layers can be constructed
  - Deeper will be better performance

Basics: Transposed convolution

- Reverse operation of convolution
  - After adding padding, do convolution
  - Use for upsampling



Data Augmentation



- Random shift for x axis
  - Considering periodic condition of ϕ angle (f(Φ+2π) = f(Φ))
  - To suppress over fitting
- Add random y-flip (I think not good from physics point of view, but suppress over-fitting is important)