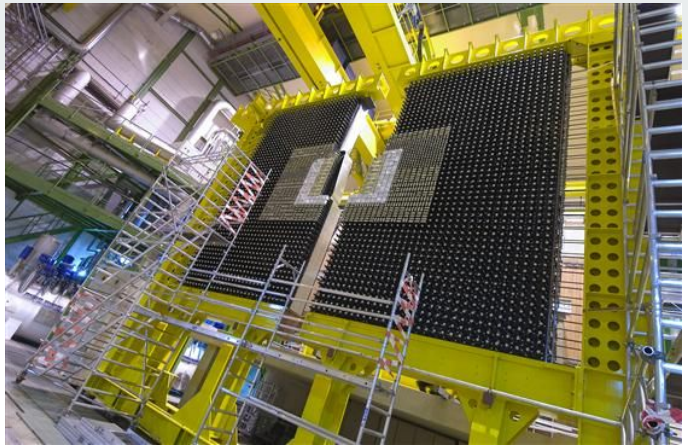
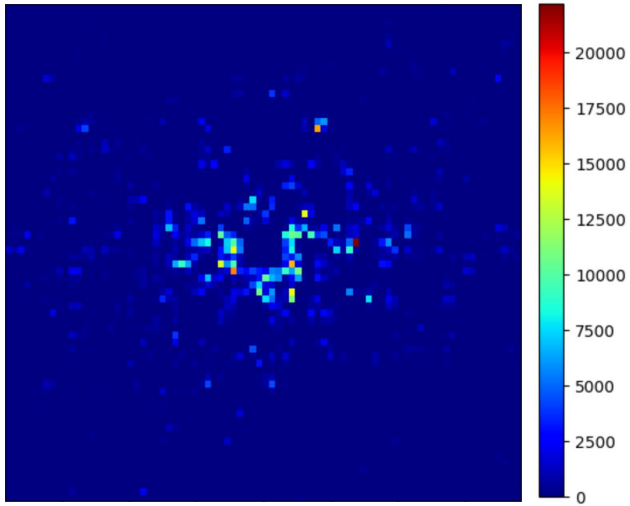
# Deep Learning approach to Cellular Automaton CALO reconstruction in LHCb

Núria Valls
DS4DS - La Salle URL Barcelona
I Workshop on Computing & Software de la Red Española de LHC
28th April 2021

# Introduction: ECAL

- **2D Shashlik grid** of modules → absorber plates and scintillator tiles (active material)
- Three regions:
  a. **Inner** → 4x4 cm² cell size
  b. **Middle** → 6x6 cm² cell size
  c. **Outer** → 12x12 cm² cell size
- Measures integrated energy → **overlapping energy** clusters.
- 2D images as **raw data**: 12 bit precision
- **Cluster**: group of cells that contain the energy of a particle (3x3)






Raw data of an example event

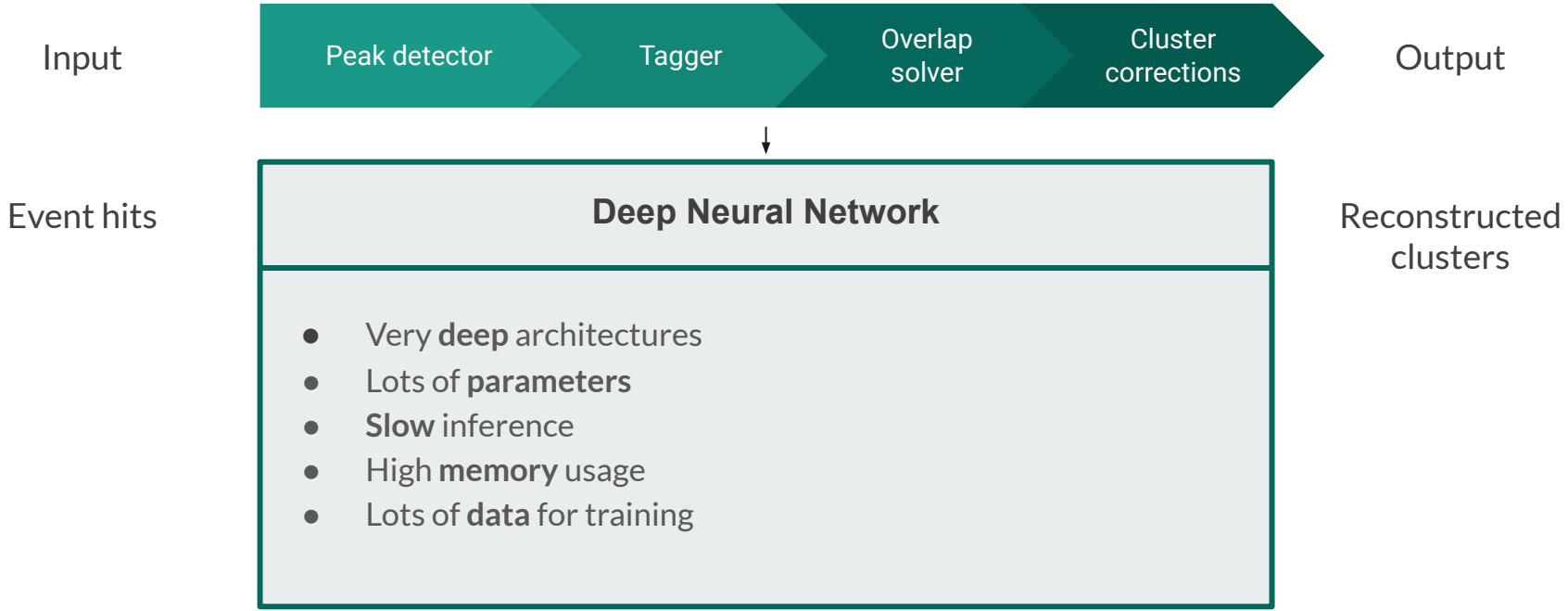
# The current approach

| Peak detector | Cluster reconstructor | Overlap solver | Cluster corrections |
| --- | --- | --- | --- |
| Find local maxima. | Tag energy cells around maxima. | Fraction energy values for overlapping cells. | Can be done with reconstructed data. |
| One iteration for all cells at least. | One iteration through all the cells at least. | Needs more than local information. | |
| Is “CA-like”. | Is a CA. | | |

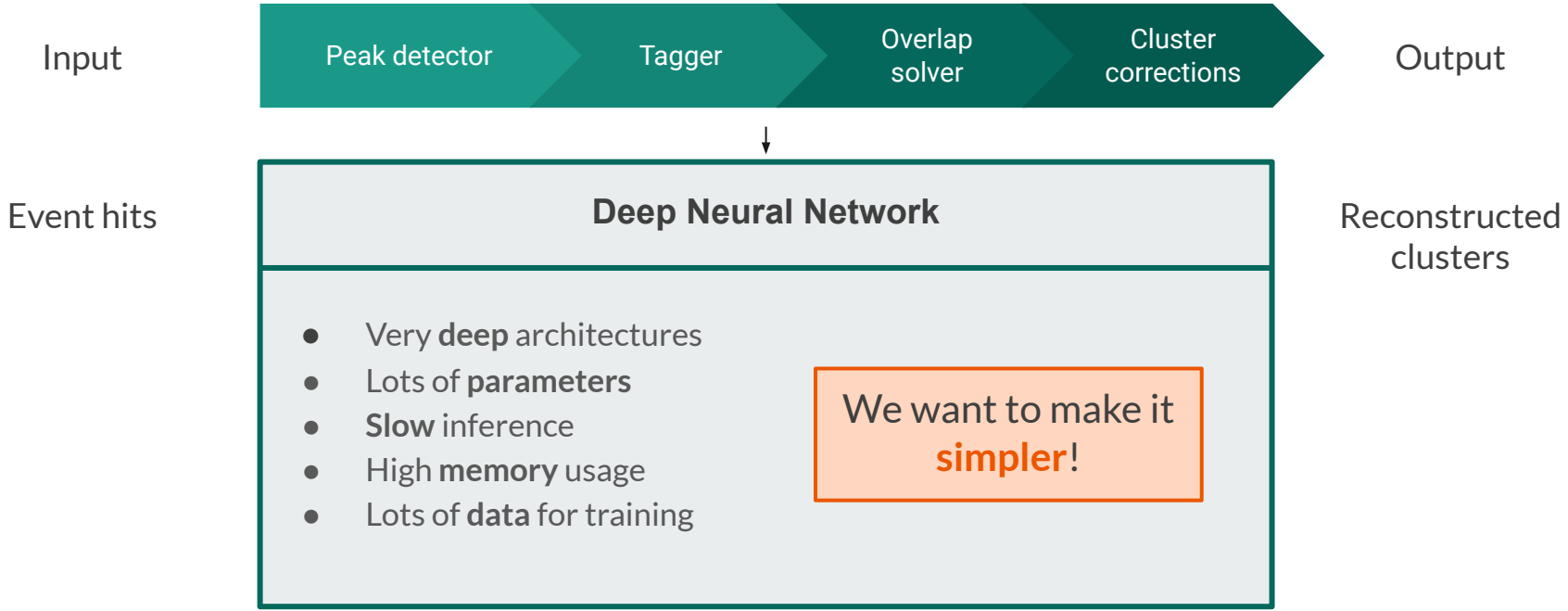
Calorimeter reconstruction takes 25% of High Level Trigger 2 sequence time

LHCb Public Results

# The current approach in Deep Learning



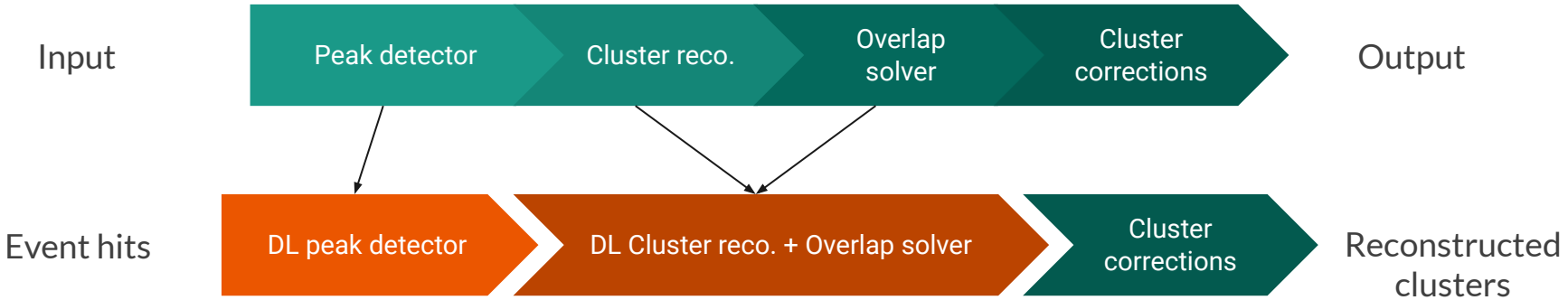
# The current approach in Deep Learning



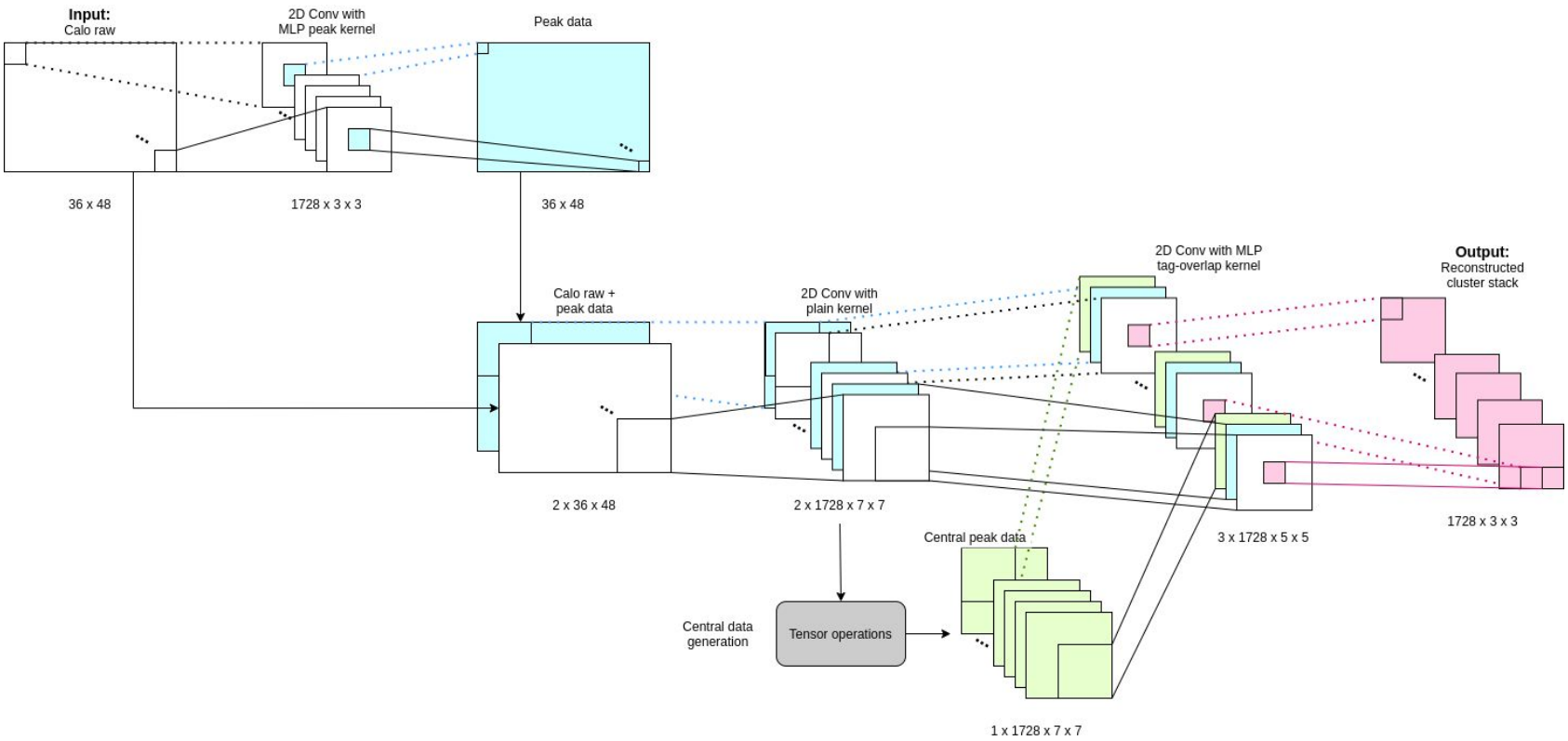
# Our Deep Learning approach

**Base idea:** [1] → The rules of a Cellular Automata can be **learned** by a deep convolutional architecture.

[1] Gilpin, W. (2019). Cellular automata as convolutional neural networks. *Physical Review E*, *100*(3), 032402.



# Overall structure



# Peak finder

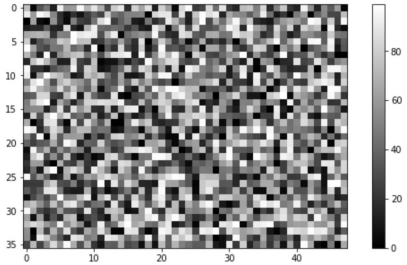


The ruleset defines **comparisons**: Identify local maxima **independently of the numerical value scale**.

**Training:**
1k images of random values (0-99).

**Testing:**
with ECAL data
**0.9973 acc**

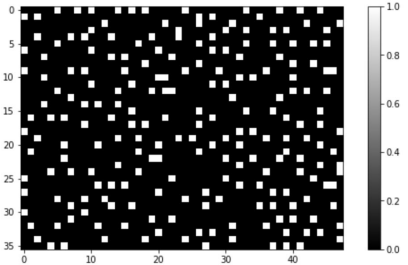
Input

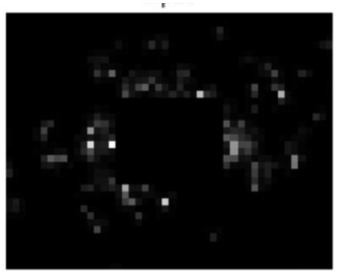


CNN

2D Conv. + Dense layers
**1272 parameters**

Output



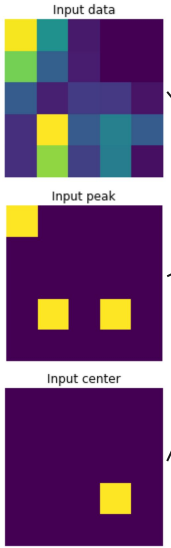


CNN

# Cluster reco + overlap



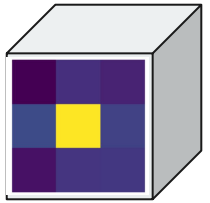
The ruleset defines **fractions**: Need to model **non-linearities** → 2D Conv. Layers don't work. We need a **MLP** to be the **kernel of the convolution**.




**Regression MLP**:
5 Hidden Layers
(~100k params)

X
Reconstructed single cell value


Aggregate 9 cell values to get the reconstructed cluster



Stack the clusters of all the windows to get the **reconstructed cluster stack** of an event.

**Dataset:** 200k balanced 5x5 images from 2k events

**1.05% relative error** in **testing**

# Results

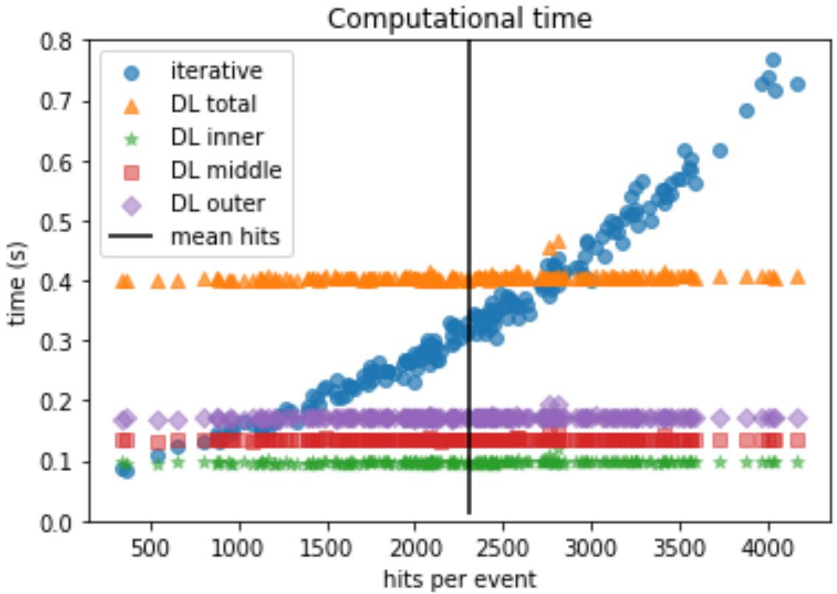
Local comparison between an **iterative method** (same complexity as the current algorithm) and the proposed **DL method**.

Relative **difference** of **reconstructed energy per cluster** with simulated events:
Mean **0.055 ± 0.104** and **0.083 ± 0.153**

→ Nearly **constant behavior** with independence of the events complexity.

→ Only faster in **25%** of the events.

→ If **region-parallelized** it is faster in **89%** of the events.



# Conclusions

- We achieved to **reproduce** the steps of the current reconstruction in a **linear complexity algorithm**.
- The **volume of data** needed for the network to **generalize knowledge** is much less than training with the whole calo images.
- We can use **artificially generated data** to train the networks and not depend on simulations.
- Each step can be tuned and retrained **adding new features** (e.g. train directly with montecarlo data).
- Once trained, the **reconstruction cost is independent from the event complexity**.

# Thank you

Ask questions!

Núria Valls - nuria.valls.canudas@cern.ch

# Backup

# Formulation of the peak finder

CA-like formulation:

- Output states: 2
- Neighbourhood: 8 cells
- Ruleset:

$$f(c_{i,j}^{t+1}) = \begin{cases} 1, & \text{if } c_{i,j}^{t} > c_{i+M,j+N}^{t} \text{ for } M \in \{-1,0,1\},\ N \in \{-1,0,1\} \\ 0, & \text{otherwise} \end{cases}$$

Where the energy value of each cell is defined as $c_{i,j}^{t}$ nd the case M=0, N=0 is excluded.

→ Identify local maxima **independently of the numerical value scale:**

- Generate data for the training where the **ruleset is uniformly represented**.
- Dataset: **1k** images with random values from **0 to 99**.
- Network architecture: 2D Conv. + Dense layers → **1272 parameters** → **0.9973 acc**

# Formulation of the cluster reco + overlap

CA-like formulation:

- Output states: **4096** (2^12)
- Neighbourhood: **24** cells
- Ruleset:

$$f(c_{i,j}^{t+1}) = \begin{cases} c_{i,j}^{t}, & \text{if is central seed} \\ \frac{c_{i,j}^{t} C_0}{\sum_{k=0}^{K} C_k}, & \text{for } K = num\_seeds \text{ if } c_{i,j}^{t} > 0 \text{ and is not central seed} \\ 0, & \text{otherwise} \end{cases}$$

The rules to learn become **non-linear**:

- 2D convolutional layer: kernel multiplied through the image.
  → The kernel **cannot model non-linearities**

- Dense layer: can model non-linearities but the **neighbouring information is already lost**.

→ Train a **MLP** to be the **kernel** of the convolution.

# Second step: cluster reco + overlap

CA-like formulation:

- Output states: **4096** (2^12) → ECAL precision. This is “forcing” the network to be regressive.

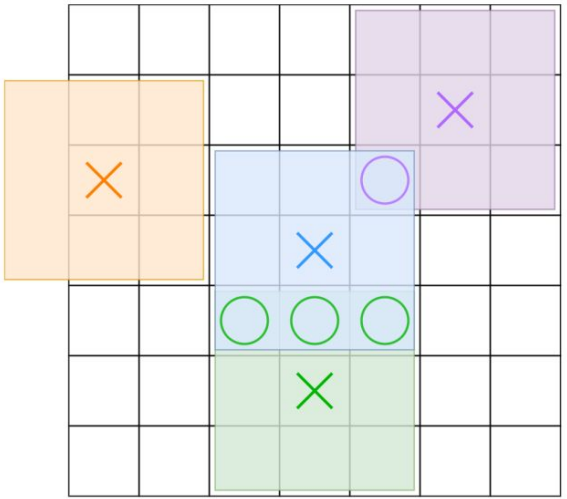
# Second step: cluster reco + overlap

CA-like formulation:

- Output states: **4096** (2^12)
- Neighbourhood: 24 cells

1. Tag cells around the peaks (X)
2. Fraction the energy of overlapping cells (O)

→ We need a **7x7 window** around each peak that outputs a 3x3 image (reconstructed cluster)



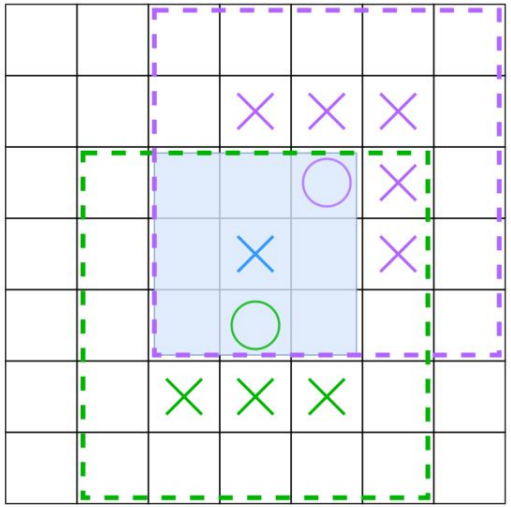
# Second step: cluster reco + overlap

CA-like formulation:

- Output states: **4096** (2^12)
- Neighbourhood: **24** cells (5*5)

1. Tag cells around the peaks (X)
2. Fraction the energy of overlapping cells (O)

→ We need a **5x5 window** around the central peak that **generates the energy value of the central cell in the reconstructed cluster.**



# Requirements

- **Test environment:** Intel Core i7-6500U CPU @ 2.50 GHz x4, memory of 15.5 GB, disk capacity of 512.1 GB with Ubuntu 20.04.1 LTS 64-bit OS.
- All the algorithms are coded in **Python 3.8**.
- All the models have been trained with **Tensorflow 2.3.0**.

# Next steps

- Test efficiency and time performance inside **Moore** (Rec).
  - How can we make an **efficient inference** of the models in **c++**?
- Add **MC information** to the cluster reco+ overlap model.
- Adapt the kernel for **irregular geometry** using Graph Neural Networks.
- Check if a **parallelized** version of this algorithm could fit in **HLT1** time requirements.