

# Tekoälyn kehitystä CERNissä

Roope Niemi



**NexTGen**  
Next Generation Triggers

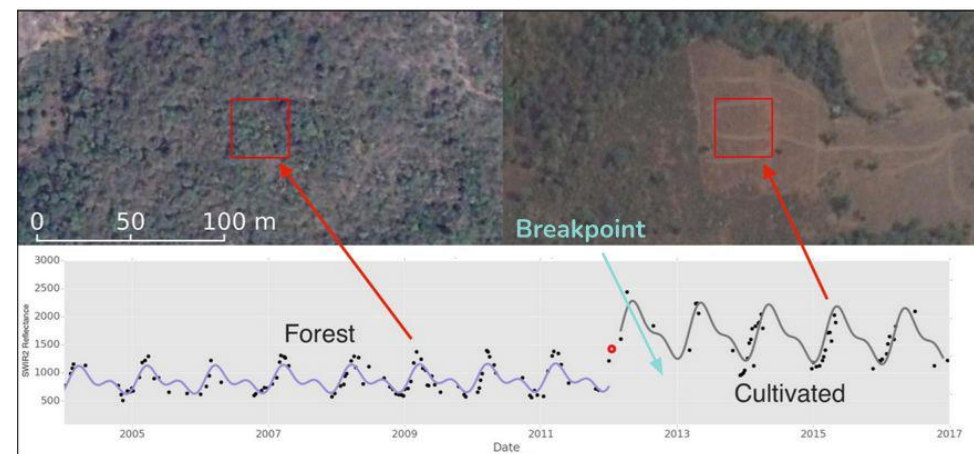
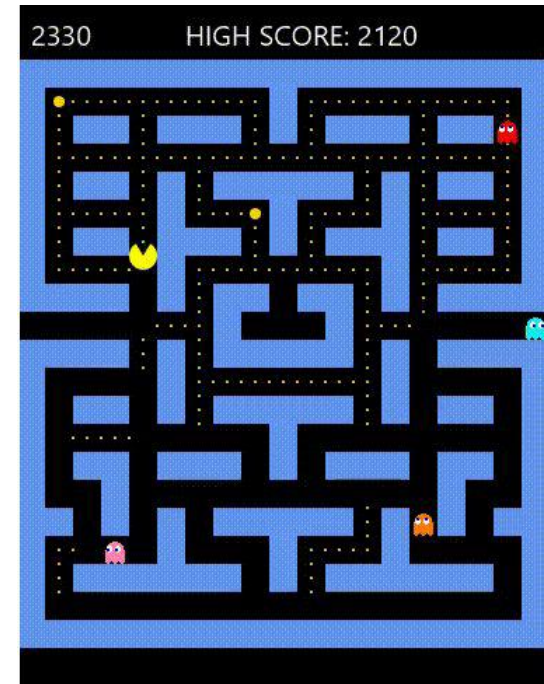
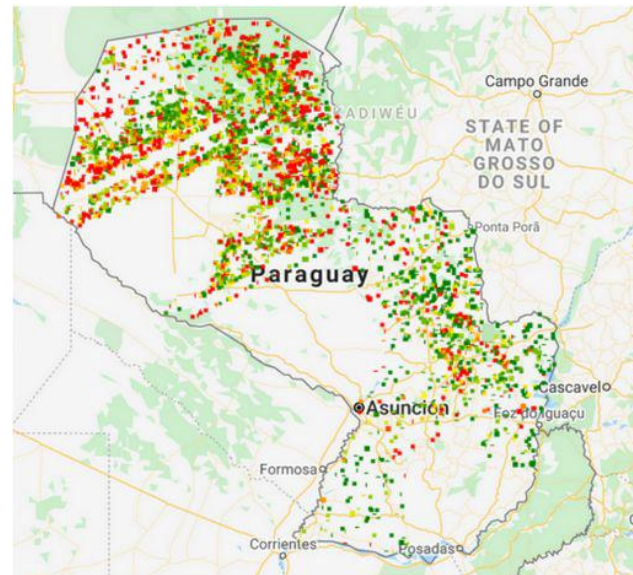
# Kuka olen?

- CERNissä syyskuusta 2024 lähtien QUEST sopimuksella (2-3v) tekoälyn parissa
- Vuoristossa töiden ulkopuolella niin paljon kuin mahdollista



# Miten päädyin tänne?

- Sotungin lukio 2008-2011. En tiennyt mitä halusin tehdä lukiossa tai sen jälkeen
- 2012-2017: Vartijana työskentelyä pääkaupunkiseudulla. Antoi motivaatiota opiskella
- 2017-2022: Opiskelua Helsingin yliopistossa Kumpulassa (Tietojenkäsittelytiede, Datatiede). Ensimmäinen IT-alan työpaikka 2018-2019
- 2021-2024: Työskentelyä Nokialla. Gradu heille, jonka pohjalta jatkotyöpaikka heiltä kehittämään tekoälyä 6G-verkkoihin. Työskentelyn ohessa hakuja CERNiin

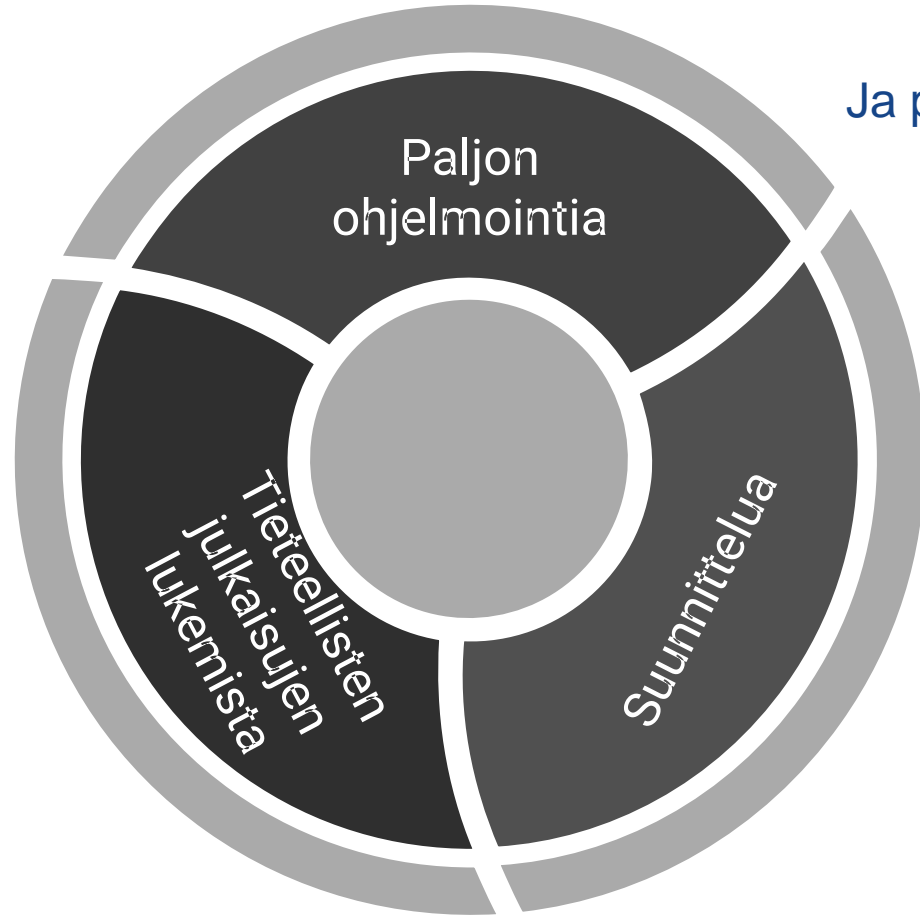


# Mitä teen CERNissä

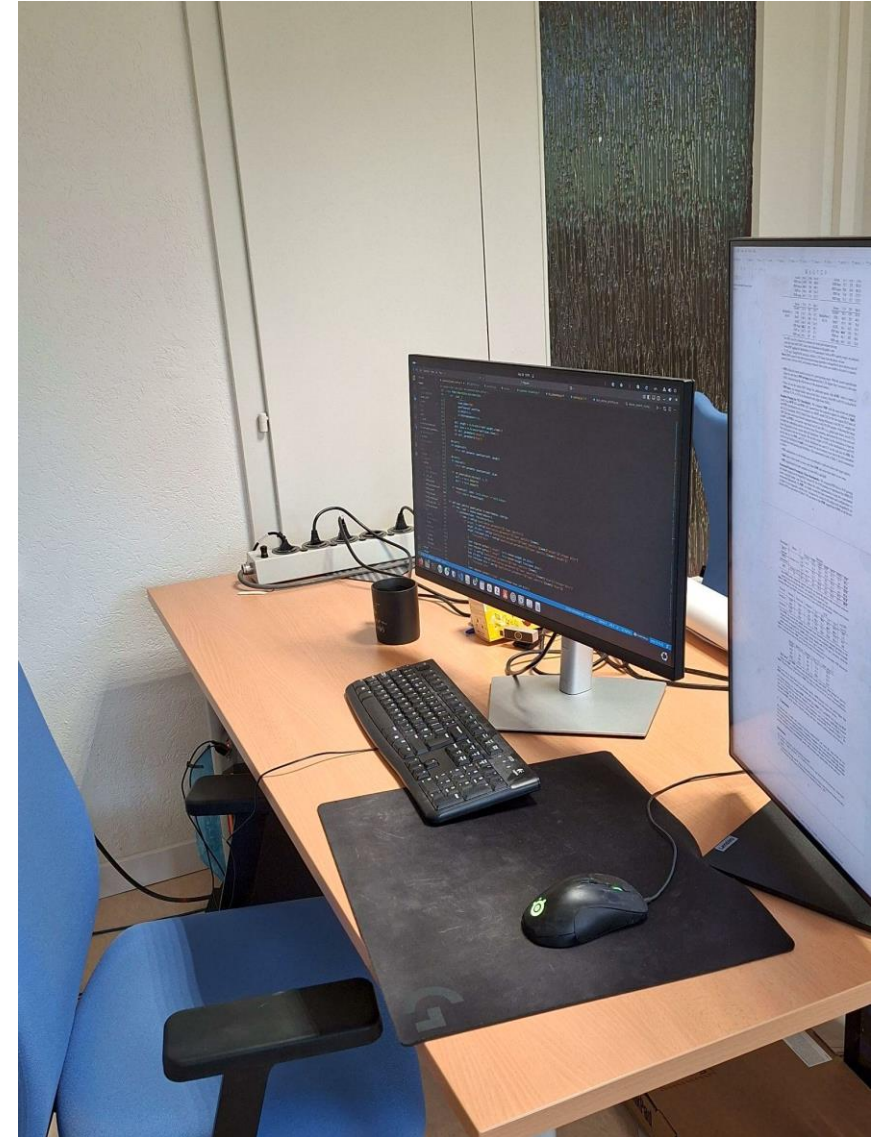
- Työskentelen tekoälyn suoraan laitteistolle (hardware) optimoimisen parissa. Tarkoitus tehdä tekoälymalleista pieniä, ketteriä mutta tarkkoja
- Vastaavanlaista työtä voi olla esimerkiksi tekoälyn kehittämisessä älykelloihin, puhelimiin, itseajaviin autoihin, 5-6G puhelinverkkoihin, droneihin, satelliitteihin. Mihin tahansa missä on rajattu määrä laskentatehoa ja tekoälyn pitää olla nopeaa
- CERNissä nopeudet ääripäässä (nanosekunteja)



# Mitä teen CERNissä



Ja paljon kahvia!



# Taustaa detectoreihin liittyen

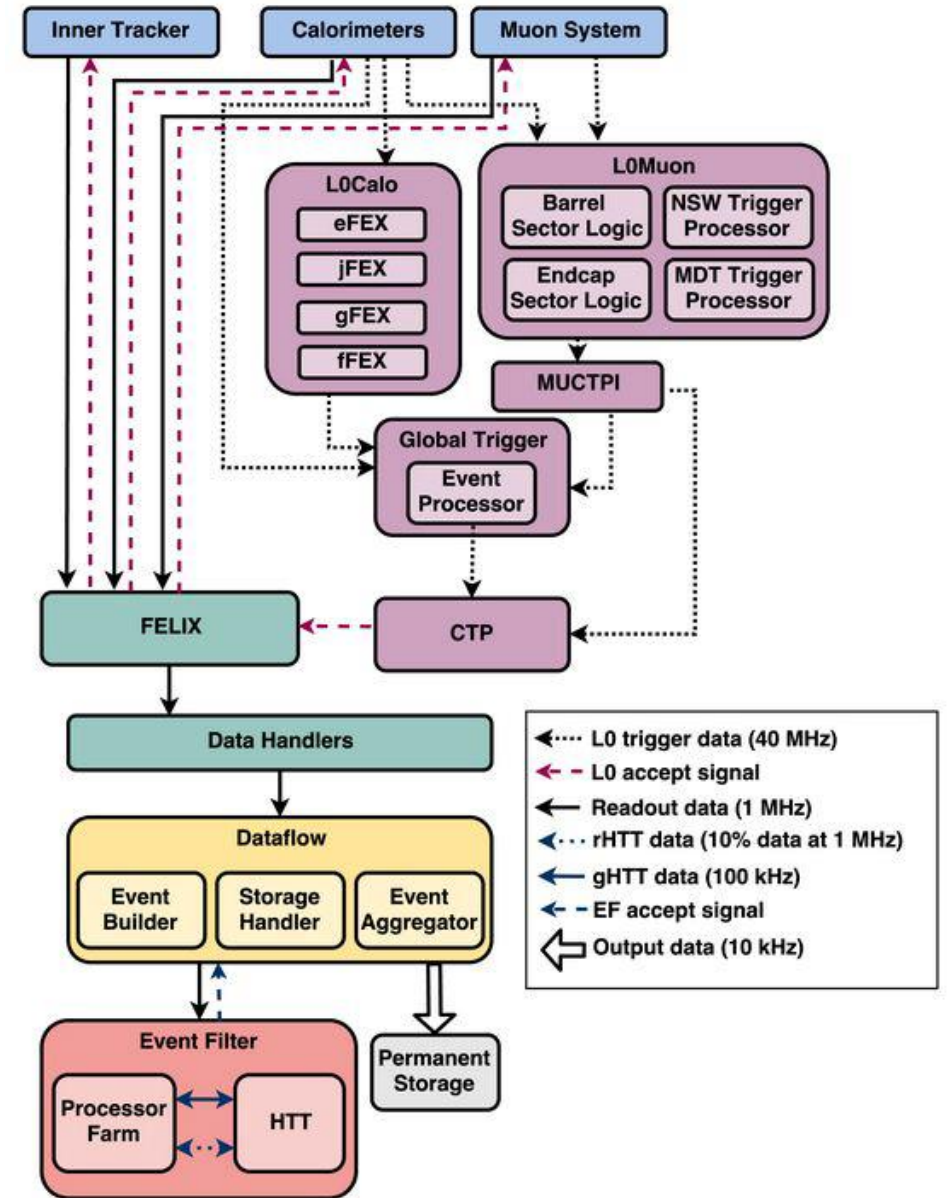
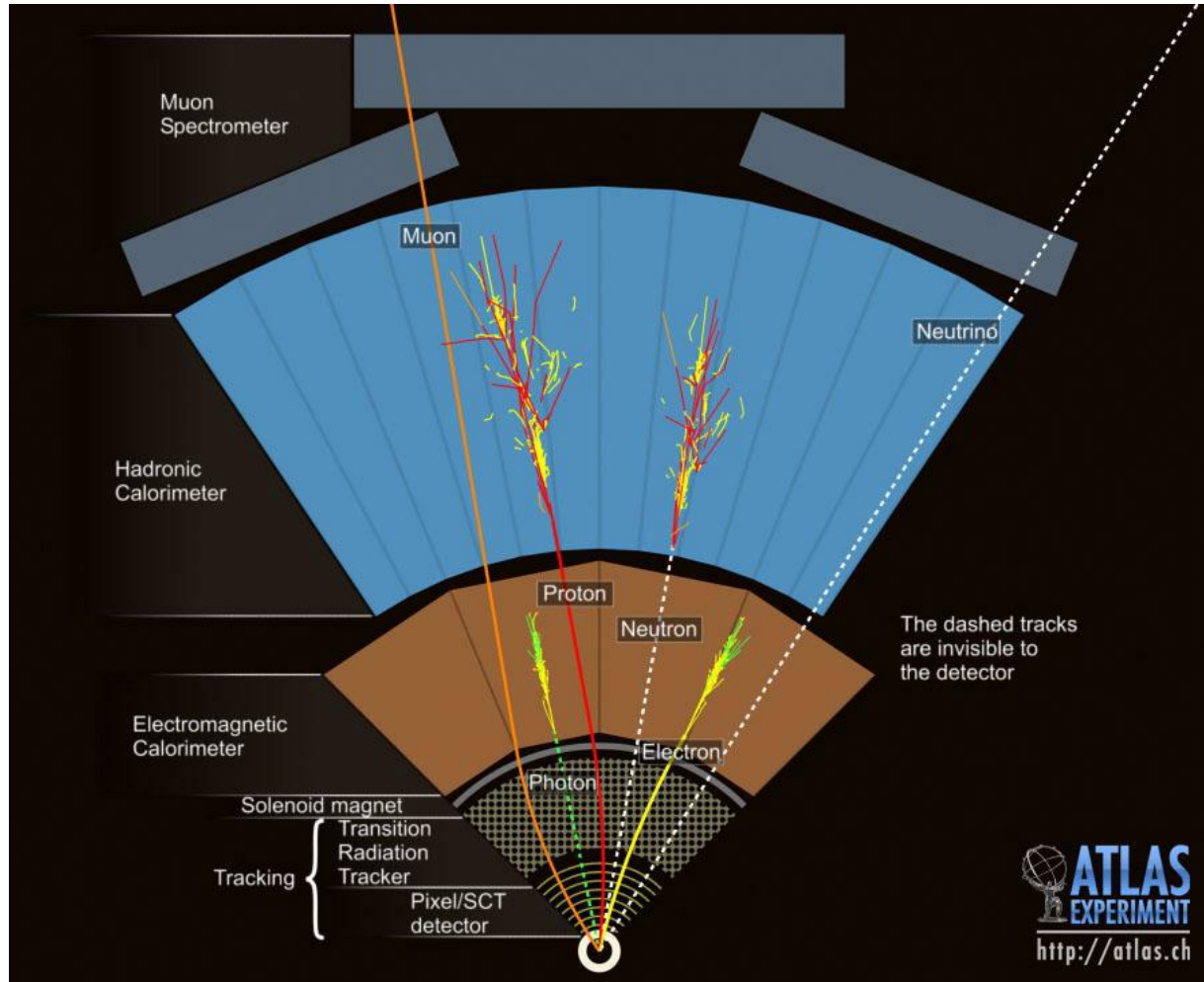
- Inner detector / tracker: Saa selville varautuneiden hiukkasten lentorata, liikemäärä, varauksen merkki (+-), lähtöpiste
- Electromagnetic calorimeter: pysäyttää fotonit ja elektronit. Mittaa niiden energiamäärän
- Hadronic calorimeter: Pysäyttää hadronit (pionit, kaonit, protonit, neutronit). Mittaa niiden energiamäärät

✓ So in short:

- Inner detector = "where did the charged particle go, and how much momentum did it have?"
- Calorimeters = "how much energy did the particle carry, and was it EM or hadronic?"

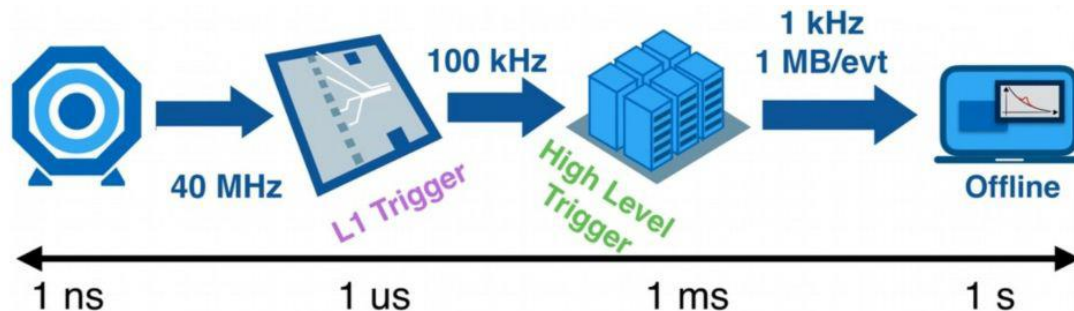
- Lisäksi muon spectrometer: tunnista myonit, niiden liikemäärä ja liikerata
- [Animation](#)

# Triggerit



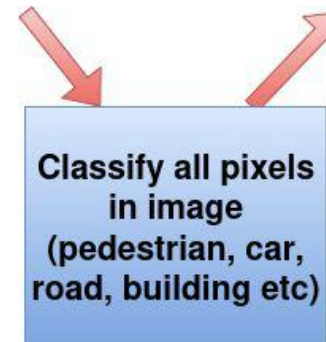
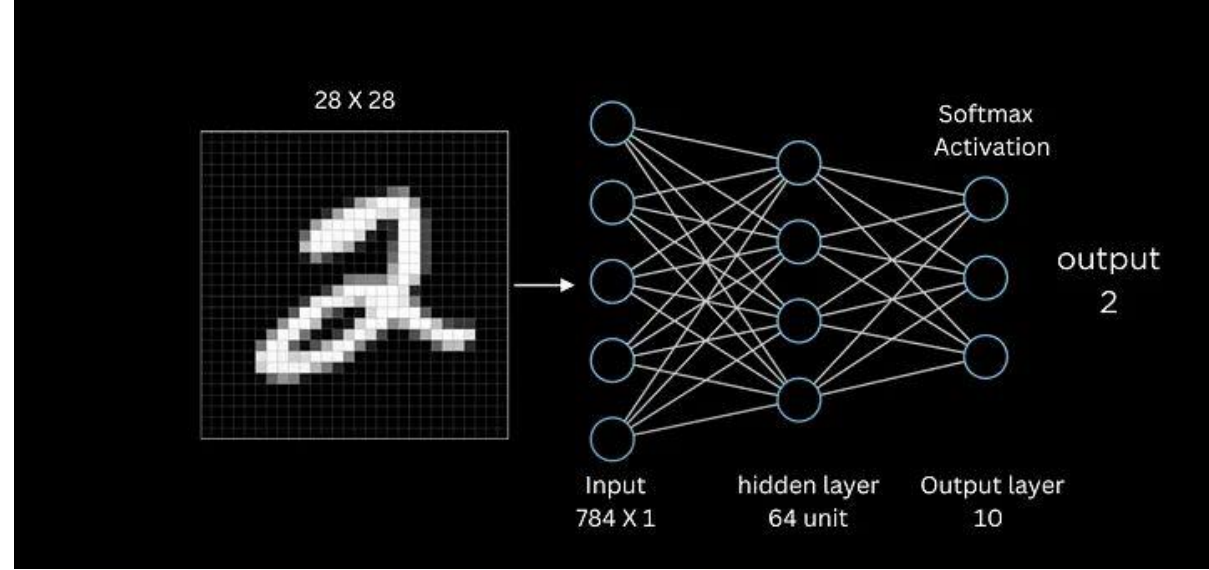
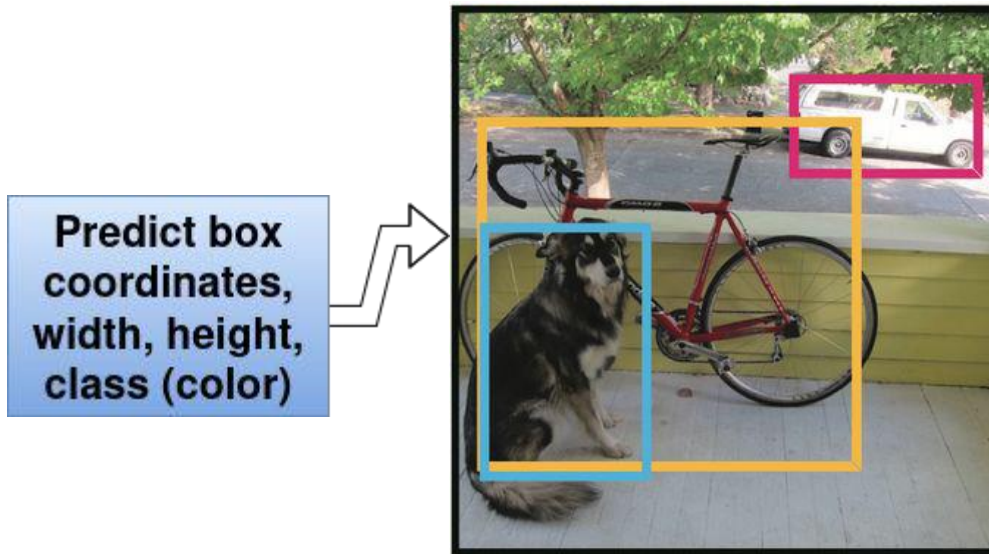
# Triggerit

- Hiukkasten törmäyksistä kertyy paljon dataa, kymmeniä teratavuja sekunnissa.
- Triggerit filtteröivät dataa, tarkoitus valita vain relevantti ja hyödyllinen data, hylkää loput. Suht uutena asiana tekoälyn hyödyntäminen triggerissä.



# Neuroverkot

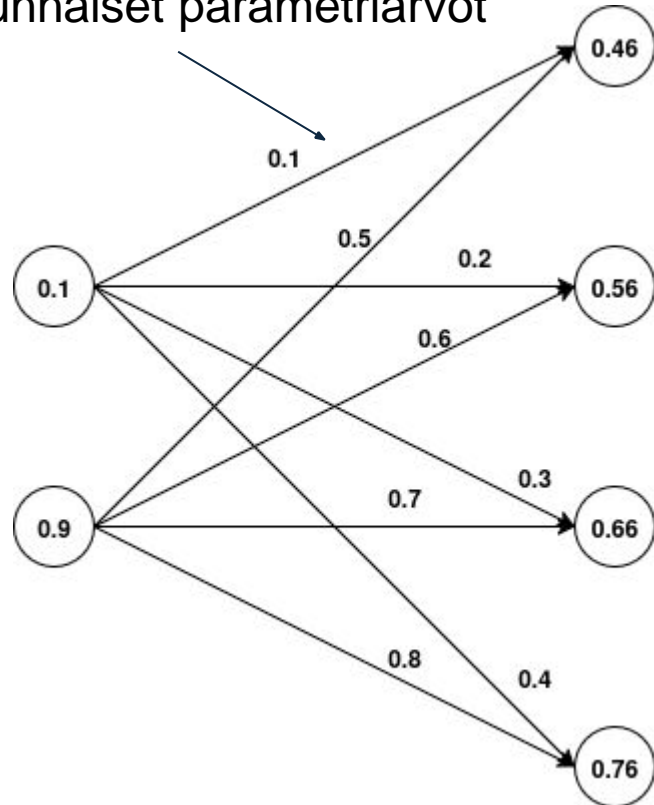
- Oppivat datasta. Koostuvat suurista määristä opittavia parametreja
- Luokittelua, regressiota, kuvan/tekstin/videon generaatiota yms.
- Ongelma pitää pystyä määrittämään matemaattisesti





# Esimerkki: Dense layer (tiheä kerros)

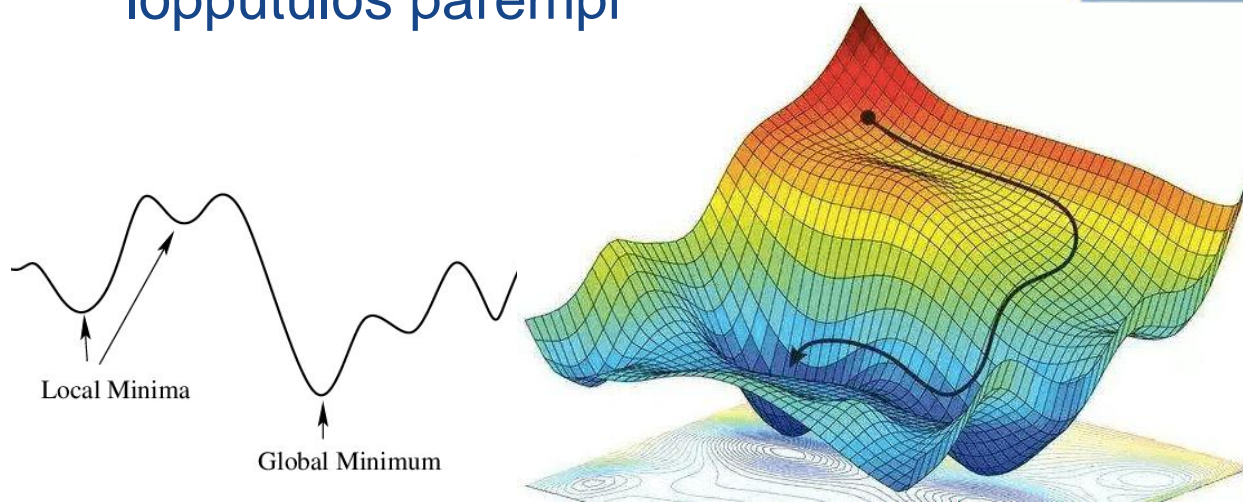
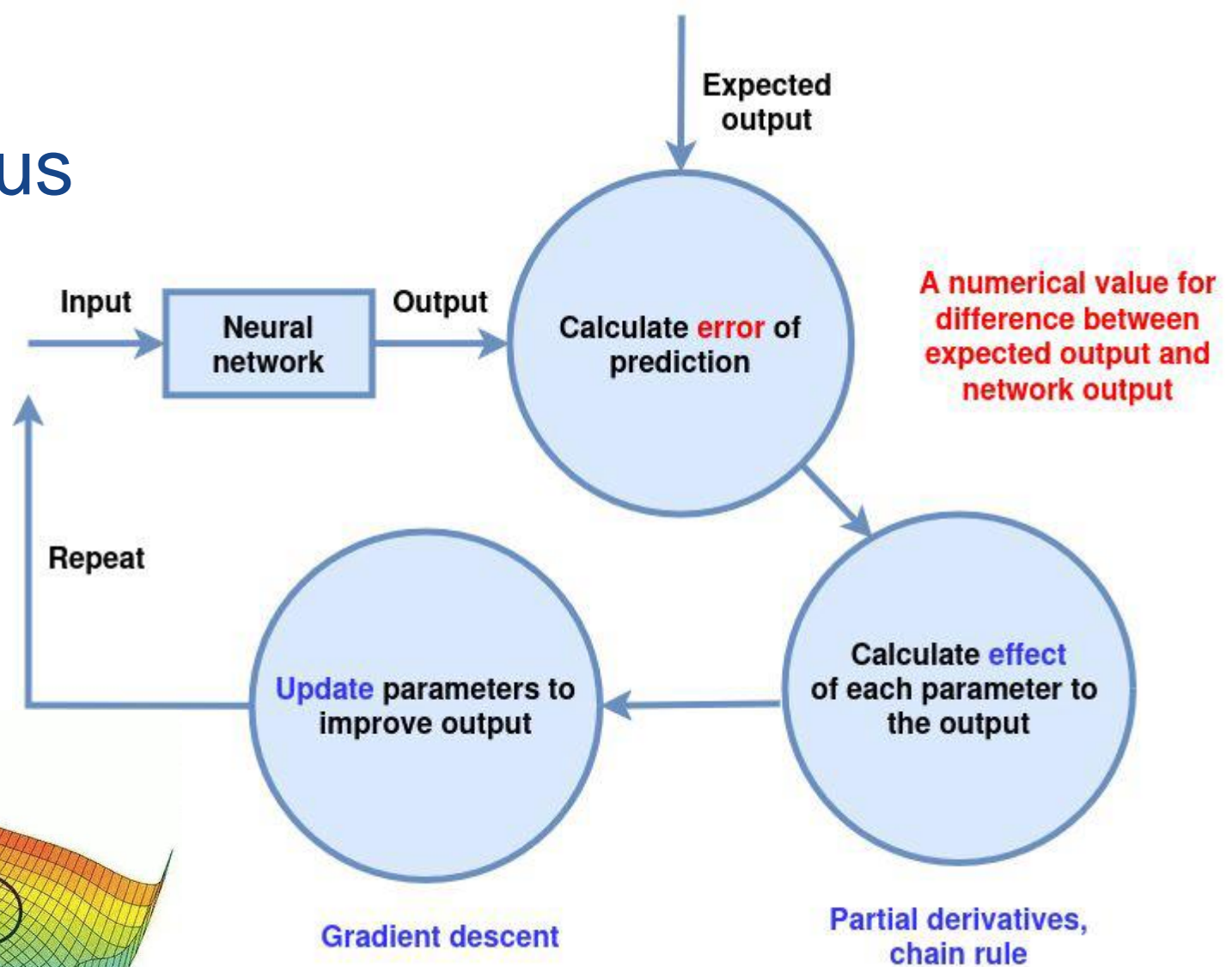
Aluksi satunnaiset parametriarvot



$$\begin{bmatrix} 0.1 & 0.9 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.46 & 0.56 & 0.66 & 0.76 \end{bmatrix}$$

# Neuroverkkojen koulutus

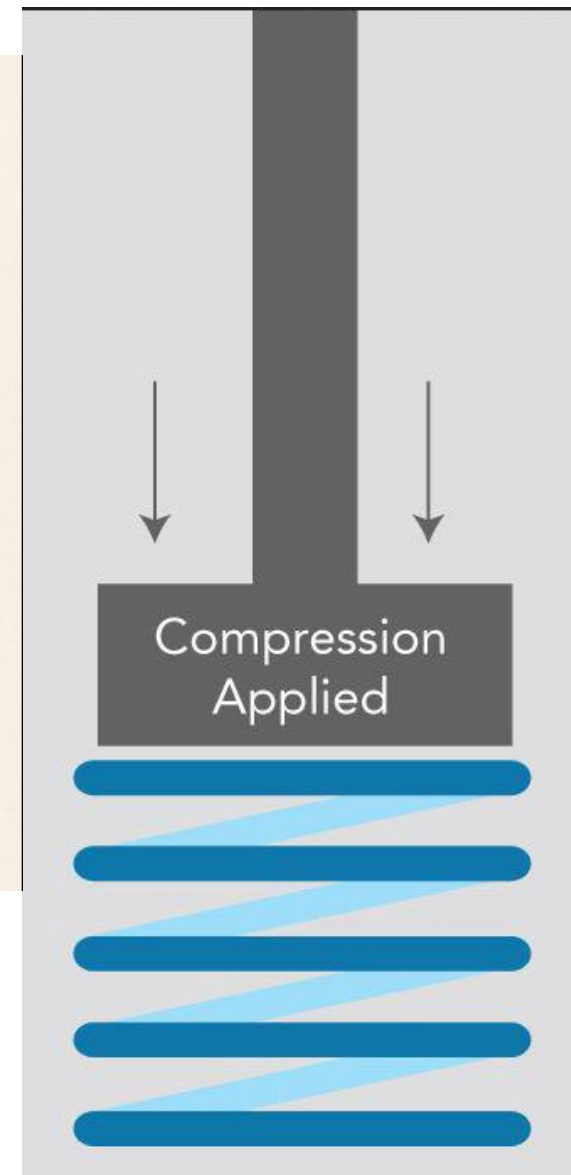
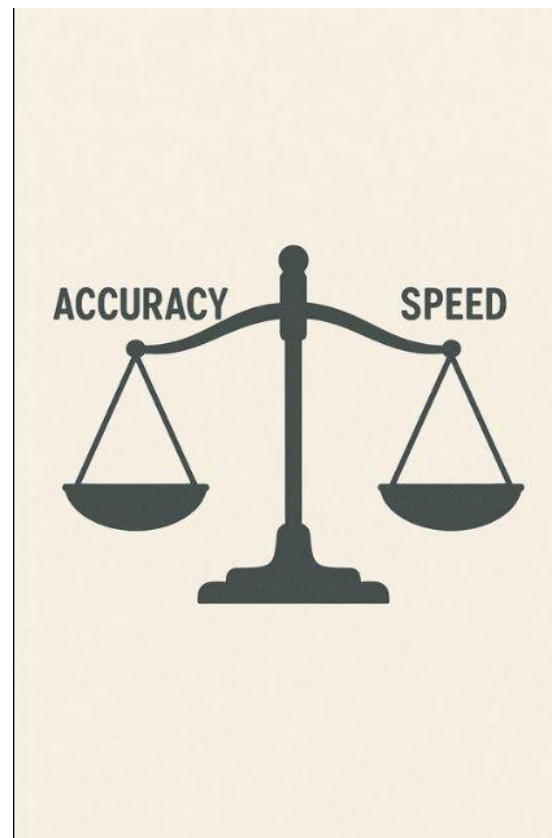
- Alussa neuroverkon tuottamat tulokset ovat täysin satunnaisia
- Opetuksen aikana neuroverkon parametrit mukautuvat jatkuvasti suuntaan, jossa lopputulos parempi



$$\begin{bmatrix} 0.1 & 0.9 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.46 & 0.56 & 0.66 & 0.76 \end{bmatrix}$$

# Kompressoidut neuroverkot

- Isot neuroverkot yleensä tehokkaampia, koska enemmän parametreja ja täten pystyvät mukautumaan monimutkaisempiin ongelmiin
- Ympäristöissä kuten trigger, joissa on vähän resursseja ja asioiden pitää tapahtua nopeasti, ei isoja neuroverkkoja voi käyttää
- Sähkönkulutus: esim. älypuhelin. Kuka haluaisi käyttää tekoälyä puhelimessa, jos joka kerta sitä käytettäessä kuluisi 5% akusta?
- Kompression avulla neuroverkon kokoa, resurssien käyttöä ja nopeutta voi optimoida



# Neuroverkkojen kompressio

Pari yleistä tapaa, muitakin on:

- 1. Rajaa parametrien tarkkuutta bitteinä (reaaliluku vs kokonaisluku)
- 2. Opeta neuroverkkoa toimimaan vähemmällä määrällä parametreja
- Esimerkki:



99	93	1	•	85	79	55	= 99 x 85 + 93 x 79 + 1 x 55 = 15817
99	93	0	•	85	79	55	= 99 x 85 + 93 x 79 + 0 x 55 = 15762

64 112 991 = 1111010010010010010101011111 = 26 bits  
25 812 = 110010011010100 = 15 bits  
54 = 110110 = 6 bits

# Tieteellisten julkaisujen tulkintaa

Esimerkki: [Wanda](#) tieteellisessä julkaisussa on sekä koodiesimerkki että matemaattiset kaavat

Mitä Wanda tekee käytännössä? Visualisointi auttaa

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2$$

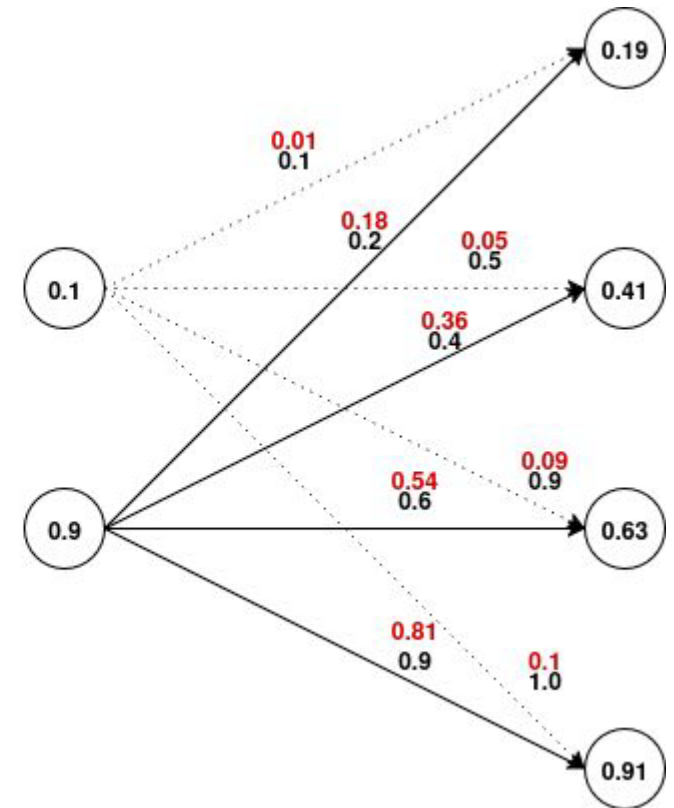
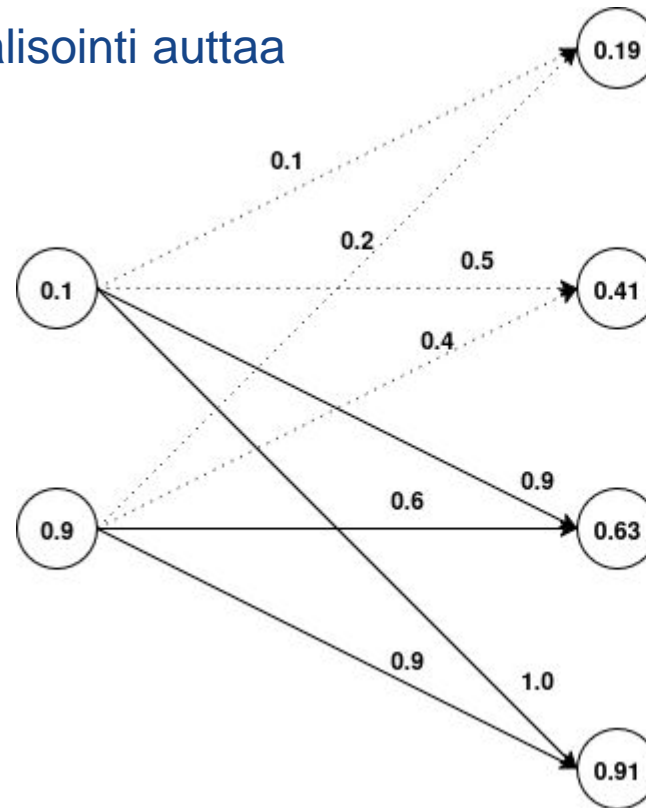
**Algorithm 1** PyTorch code for Wanda

```
# W: weight matrix (C_out, C_in);
# X: input matrix (N * L, C_in);
# s: desired sparsity, between 0 and 1;

def prune(W, X, s):
    metric = W.abs() * X.norm(p=2, dim=0)

    _, sorted_idx = torch.sort(metric, dim=1)
    pruned_idx = sorted_idx[:, :int(C_in * s)]
    W.scatter_(dim=1, index=pruned_idx, src=0)

    return W
```



# Neuroverkkojen optimointi

- Hyperparameterit: joukko parametreja jotka määritetään ennen koulutusta. Kertovat MITEN neuroverkko koulutetaan, tai mistä kaikesta neuroverkko koostuu. Vähän kuin resepti kokkauksessa
- Neuroverkot ovat musta laatikko, sisälle ei näe. Koulutus vie paljon aikaa ja vasta lopussa näkee miten meni nykyisellä joukolla hyperparametreja. Vrt. kokkaus
- Hyperparametrikombinaatioita on todella suuri määrä, miten löytää paras kombo? Käymällä kaikki läpi?
  - Satunnainen valinta jo tehokkaampaa
  - Hyödynnä etukäteistietoa ja koulutuksen aikana kertyvää tietoa kaventamaan hakuvaihtoehtojen määrää



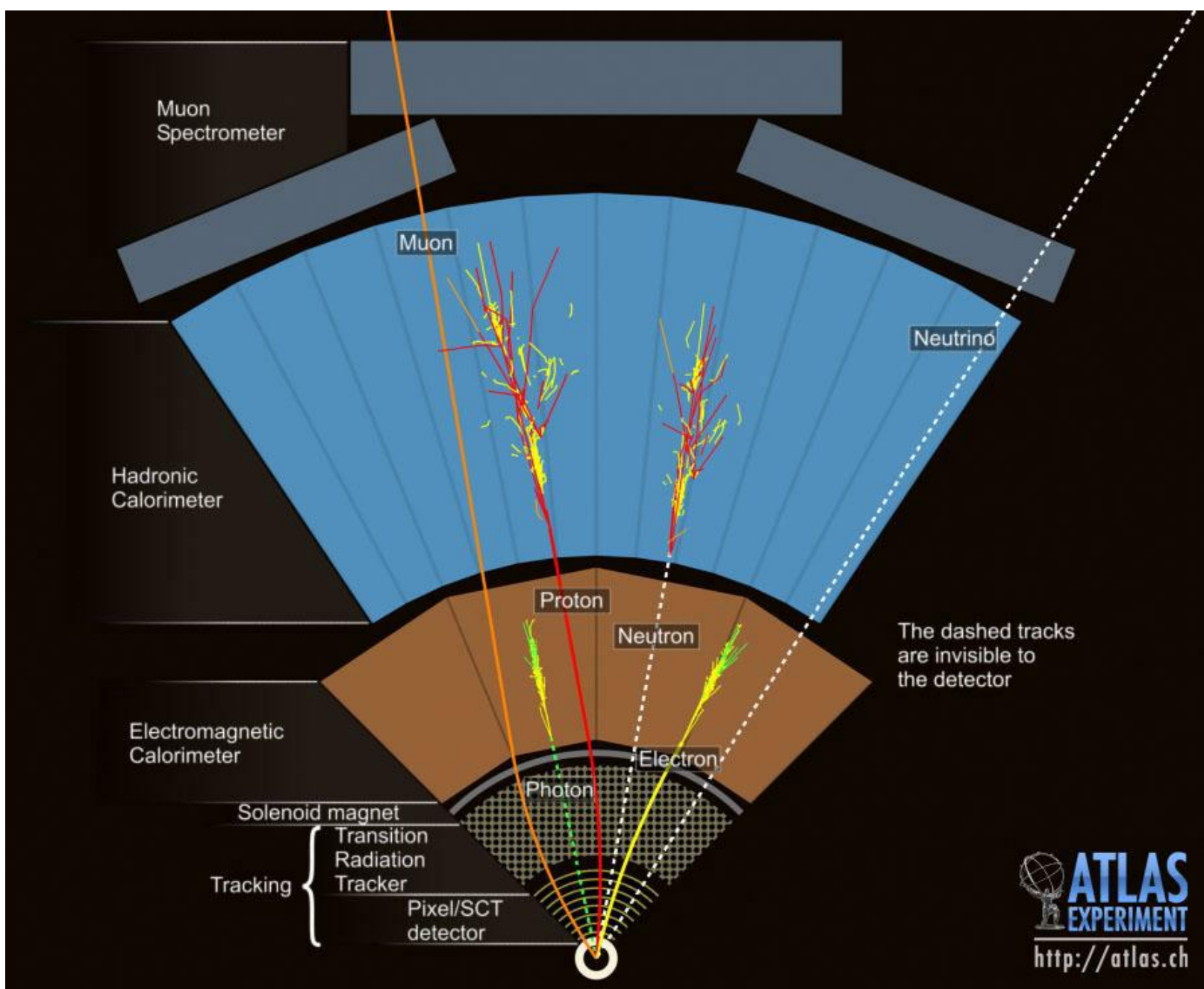
# Tekoälyn käyttökohteita CERNissä



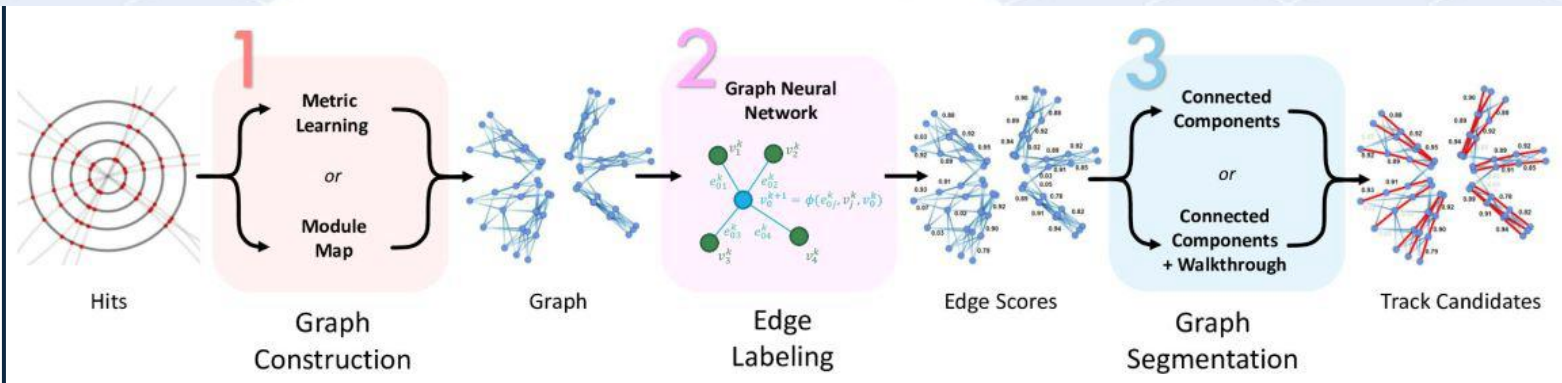
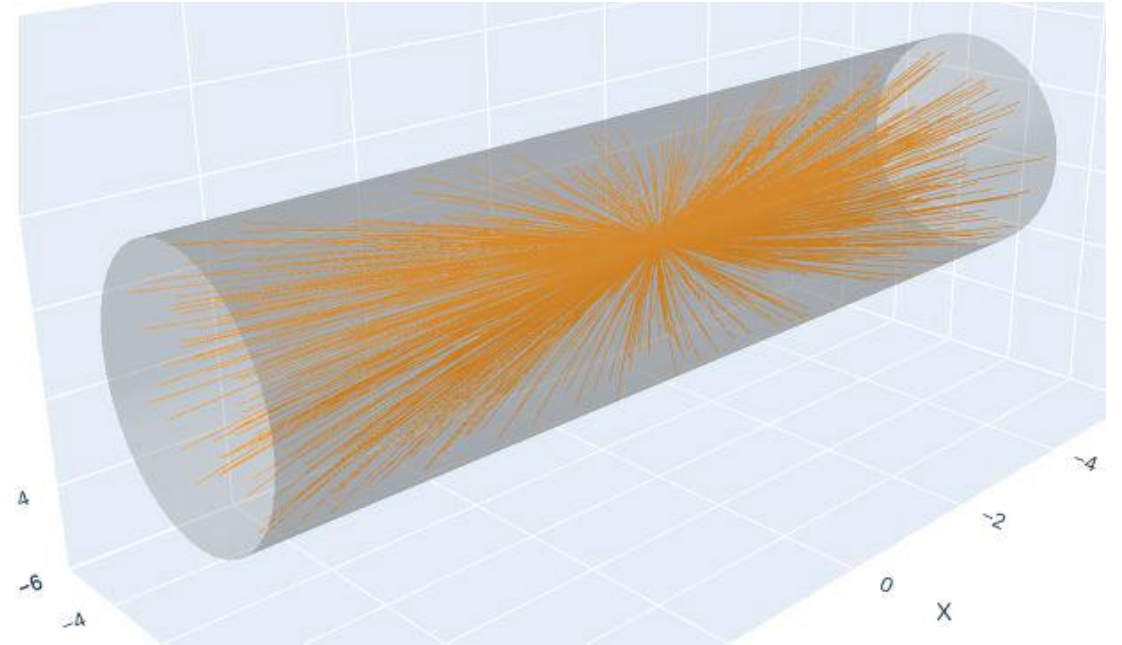
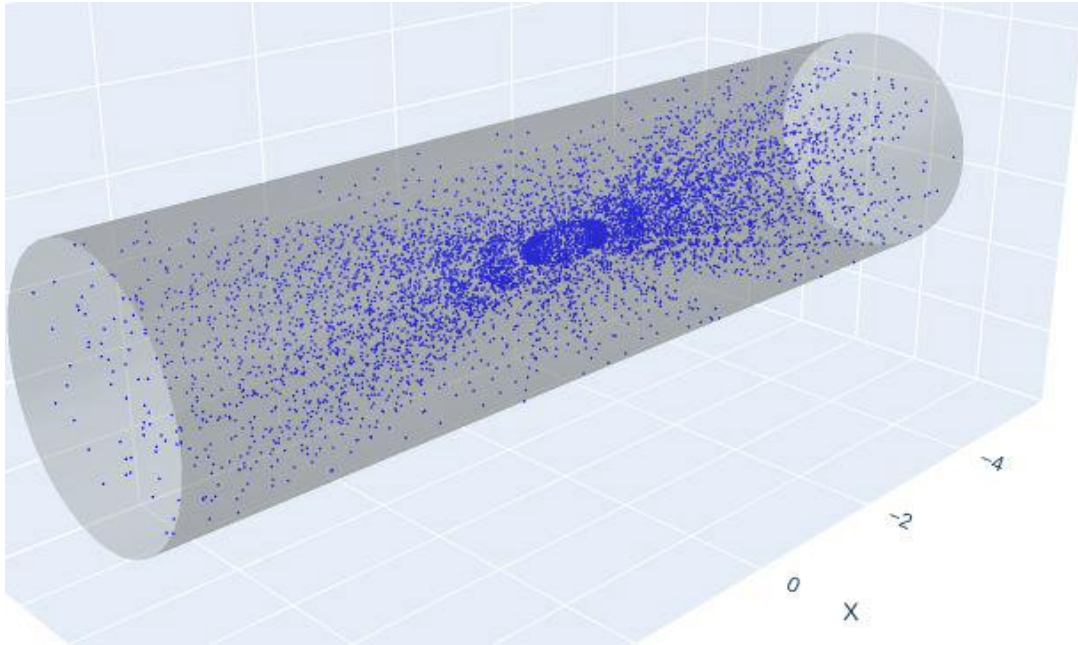
**NexTGen**  
Next Generation Triggers

# Foundation model



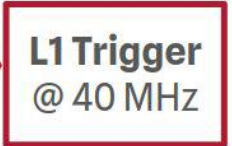
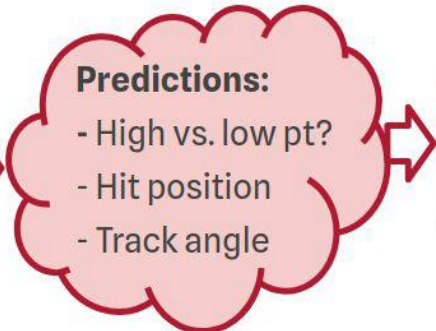
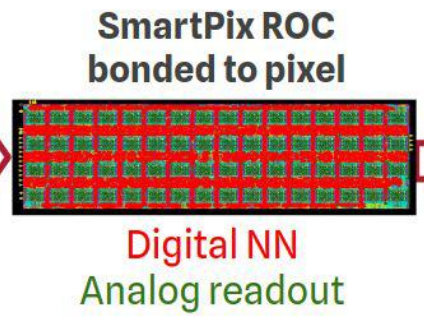
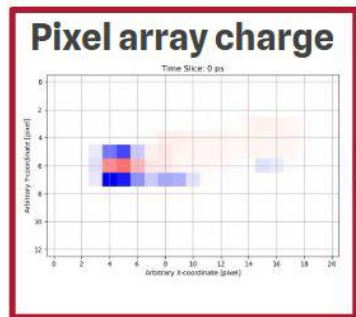
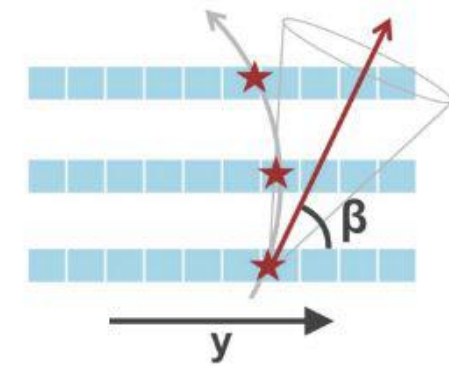
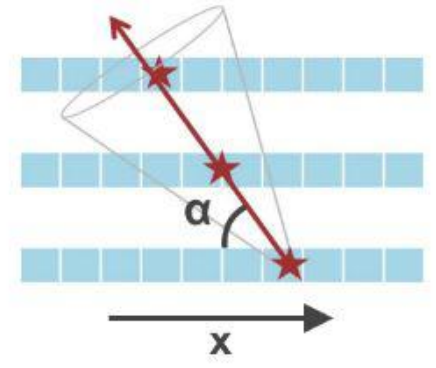
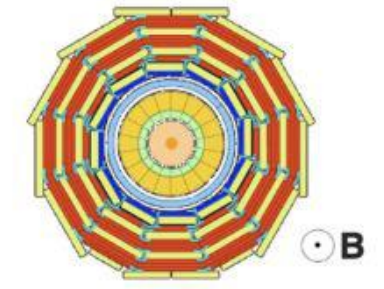
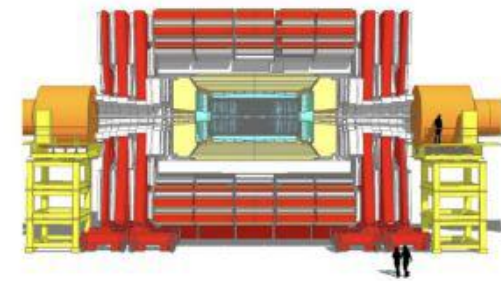
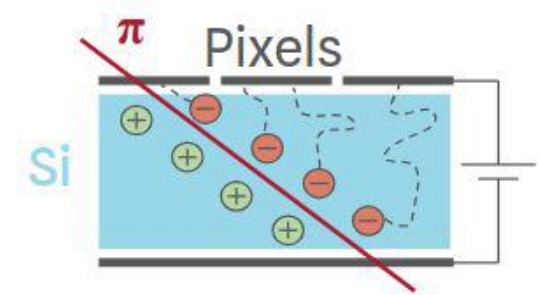


# Track reconstruction



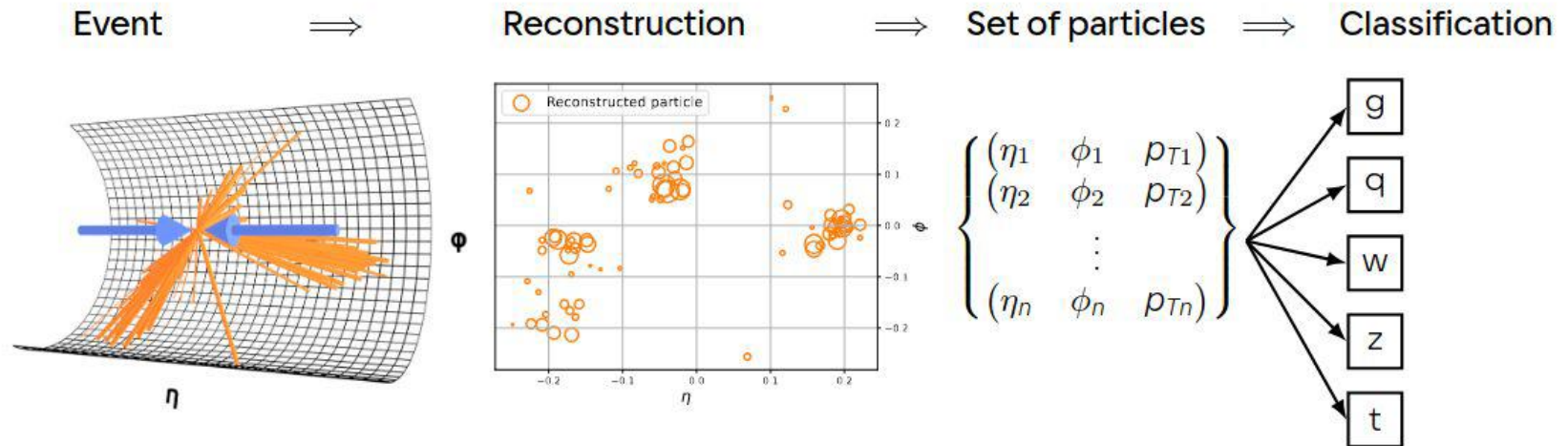
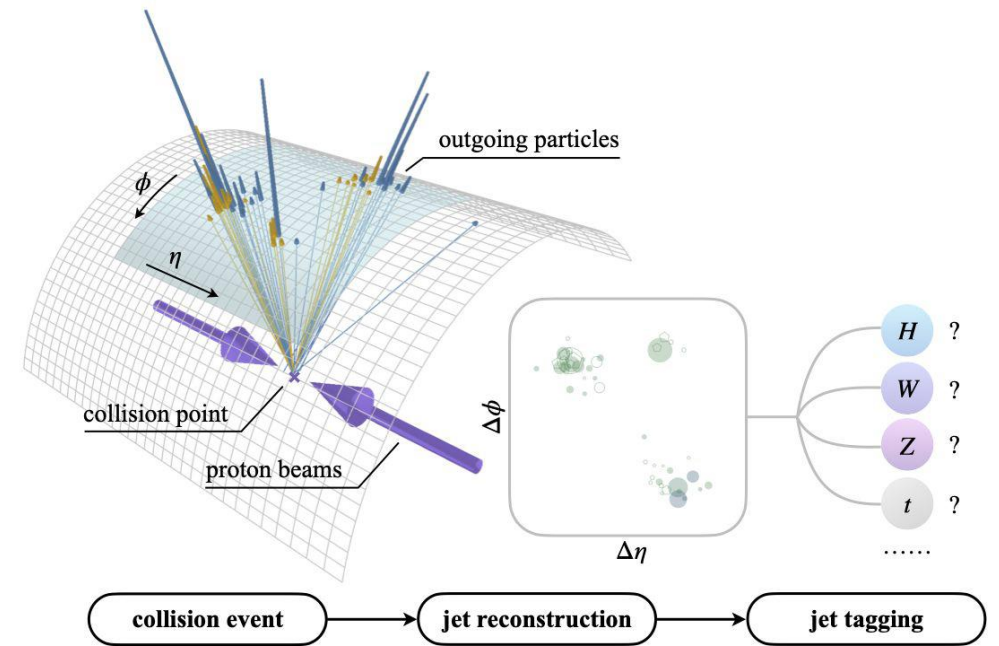
# SmartPixels

- Mitä jos pystyttäisiin valikoimaan mielenkiintoisia tapahtumia jo inner detectorissa? Käytä AI:ta löytämään suuren liikemäärän omaavat tapahtumat suoraan reaaliajassa.
- Karsi matalan liikemäärän omaavat tapahtumat suoraan automaattisesti



# Jet tagging (with transformers)

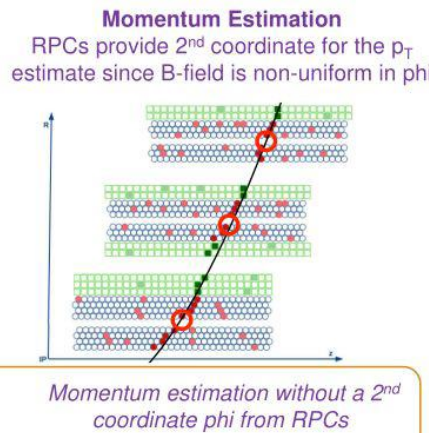
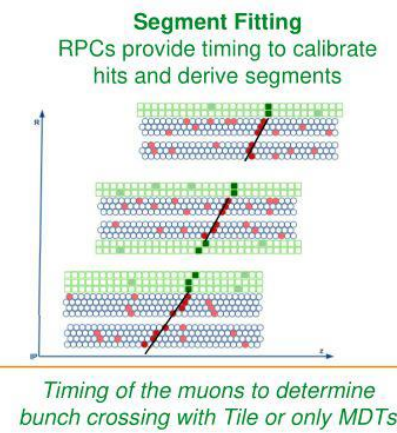
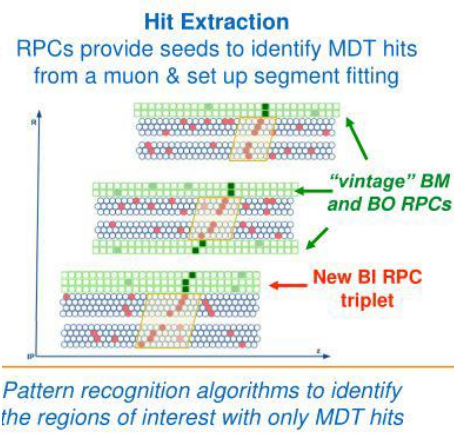
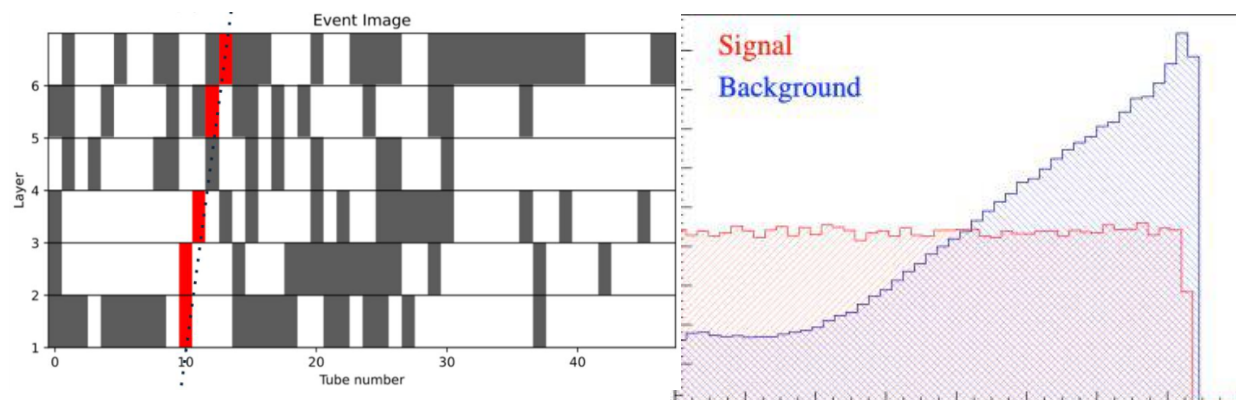
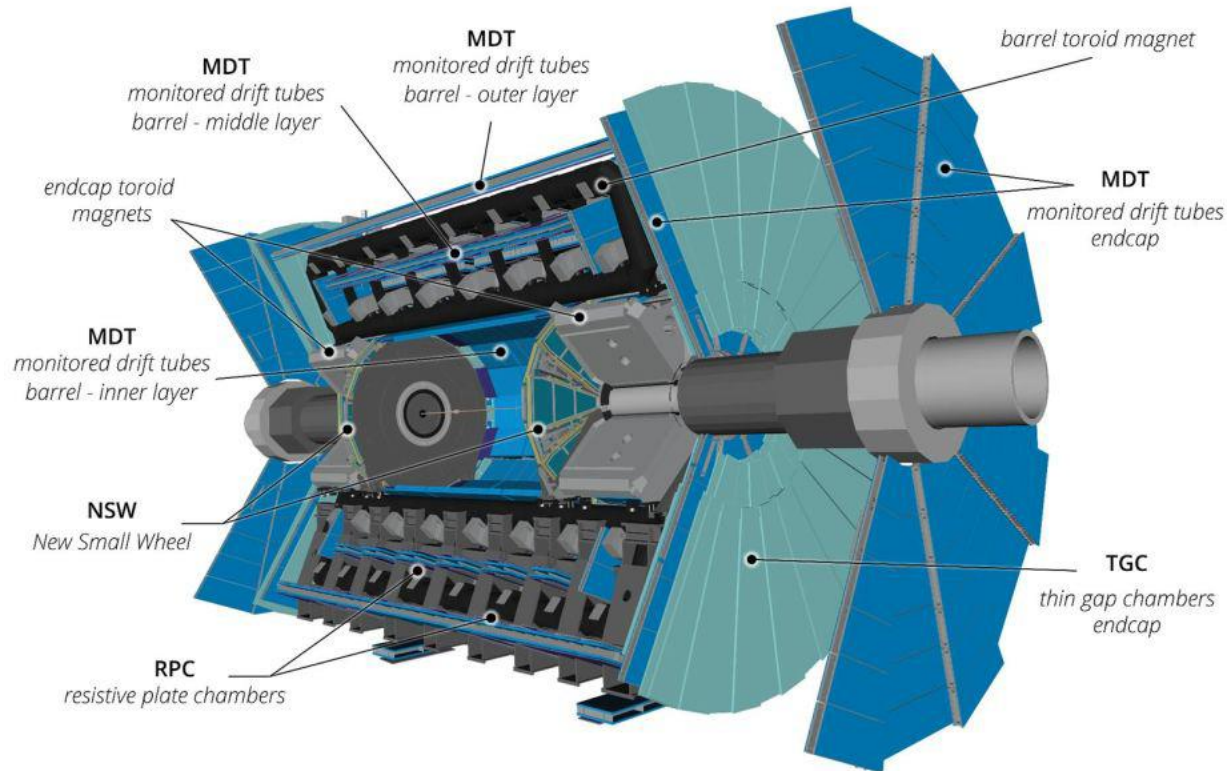
- Tunnista hiukkanen josta hiukkassuihku lähti aluille
- Käytä mitattuja hiukkasten ominaisuuksia ja hiukkasparien keskenäisiä vuorovaikutuksia tunnistamaan hiukkastyypin neuroverkoilla



# L0 Myoni trigger

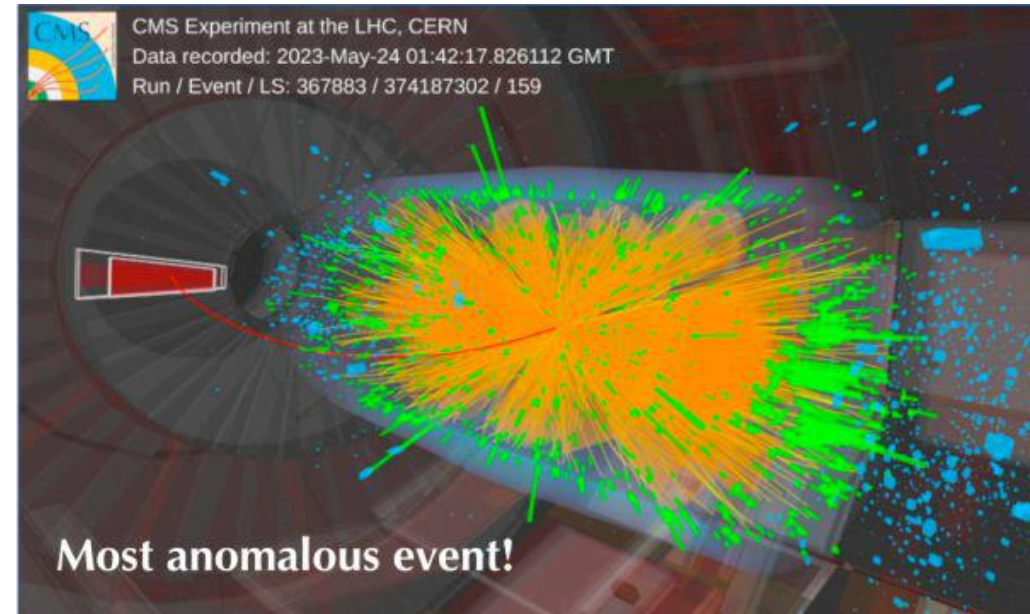
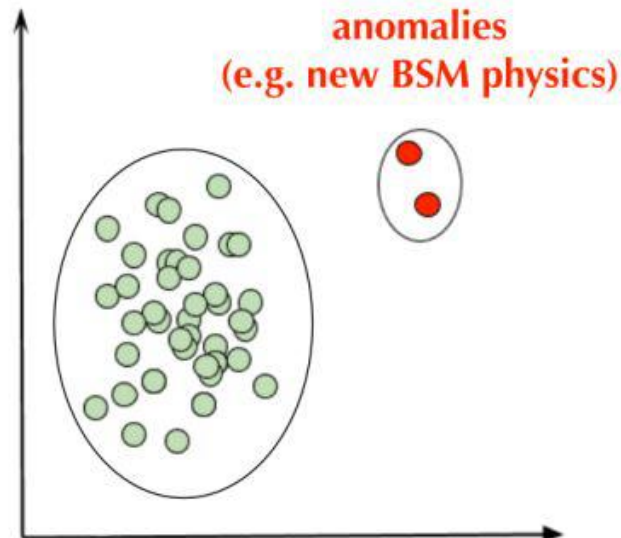
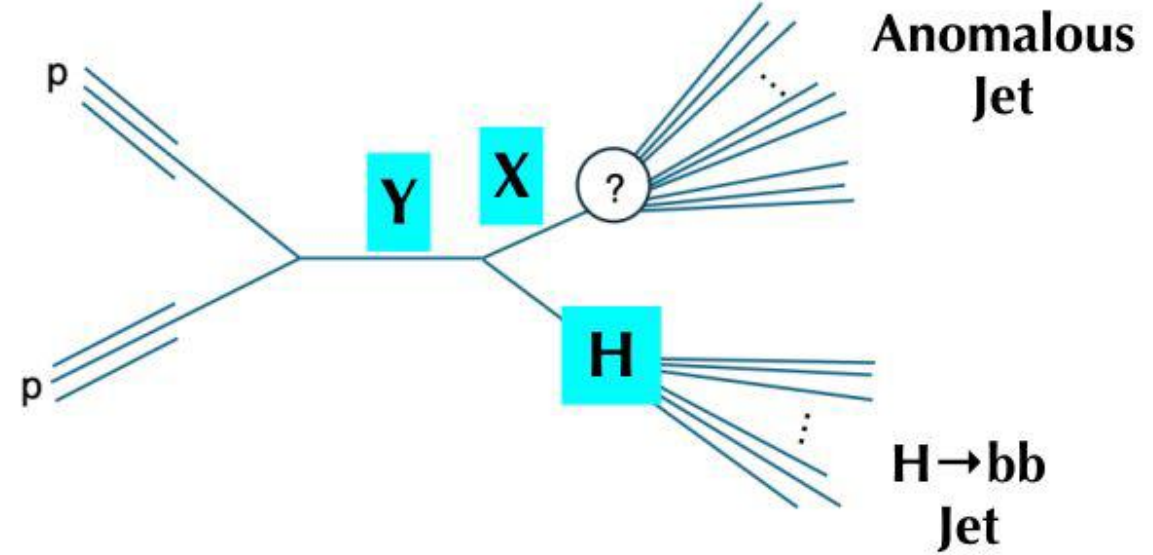
Tapautumiin jossa heikentynyt RPC:

- Löydä myoni taustakohinasta
- Estimoi liikemäärä



# Poikkeaman havainnointi

- Tekoäly oppii vain opittua dataa
- Jos triggerin AI koulutetaan nykytiedon varassa, mitä jos se hylkää ns. uutta fysiikkaa?
- Hyödyntäen kalorimetri ja myoni trigger dataa, löydä poikkeamat



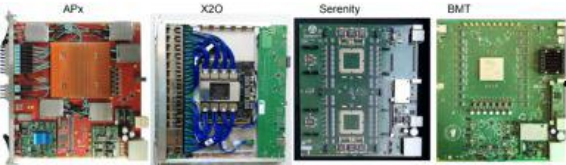
# Ylimääräistä



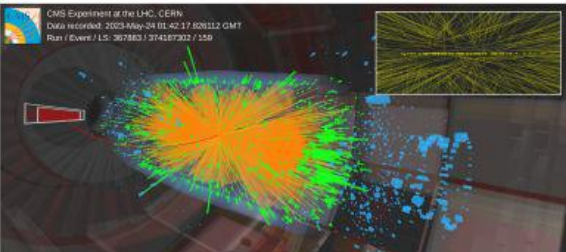
**NexTGen**  
Next Generation Triggers

# Tekoälyä missä?

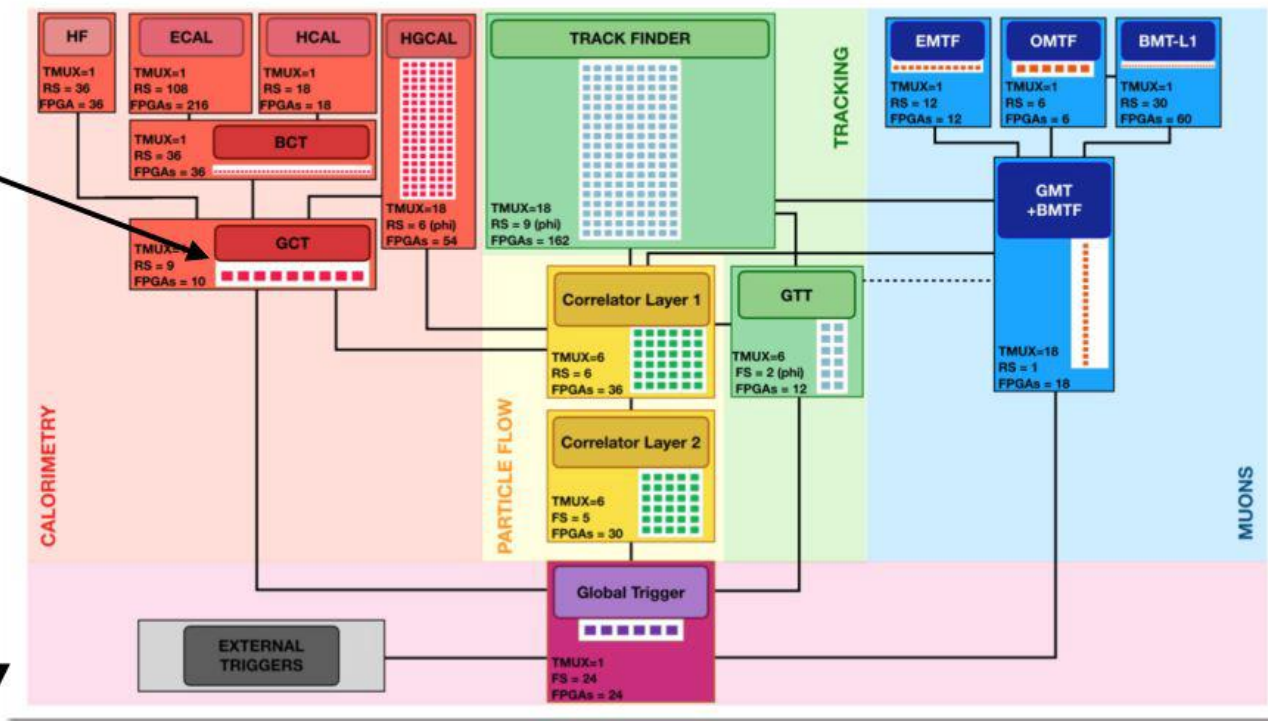
$t = 0$



1 small box = 1 FPGA board with AMD VU13P FPGA



$t < 12.5 \mu\text{s}$



0  $\mu\text{s}$

Detector hits

5  $\mu\text{s}$

Clusters & Tracks

6  $\mu\text{s}$

Particles

7  $\mu\text{s}$

Event Categorisation

8  $\mu\text{s}$

1 bit: keep / discard

# AI matematiikka tiivistettynä

- Lukion kurseista derivointi, integrointi, todennäköisyyslaskenta, vektorit auttavat jo pitkälle. Yliopistossa lineaarialgebra ja matriisilaskenta, koska kaikki tapahtuu AI:ssa matriiseilla. Tilastotieteessä jakaumien ominaisuuksien ymmärtäminen auttaa paljon, sillä jakaumia tuijotetaan paljon
- Tekoäly on käytännössä vain suuri joukko matriisikertolaskuja (paljon pistetuloja) ja funktion optimoimista, ja jakaumien tulkintaa. Toki siellä välissä voi olla kaikkea muuta siistiä
- Tärkeää on miten malli koulutetaan. Datan pitää olla hyvää, tarkoituksen hyvin määritelty, koulutustekniikoiden pitää olla hyvää

# Sources

---

- <https://indico.cern.ch/event/1421629/timetable/#all.detailed>
- <https://indico.cern.ch/event/1496673/timetable/#all.detailed>
- <https://cds.cern.ch/record/2939326>