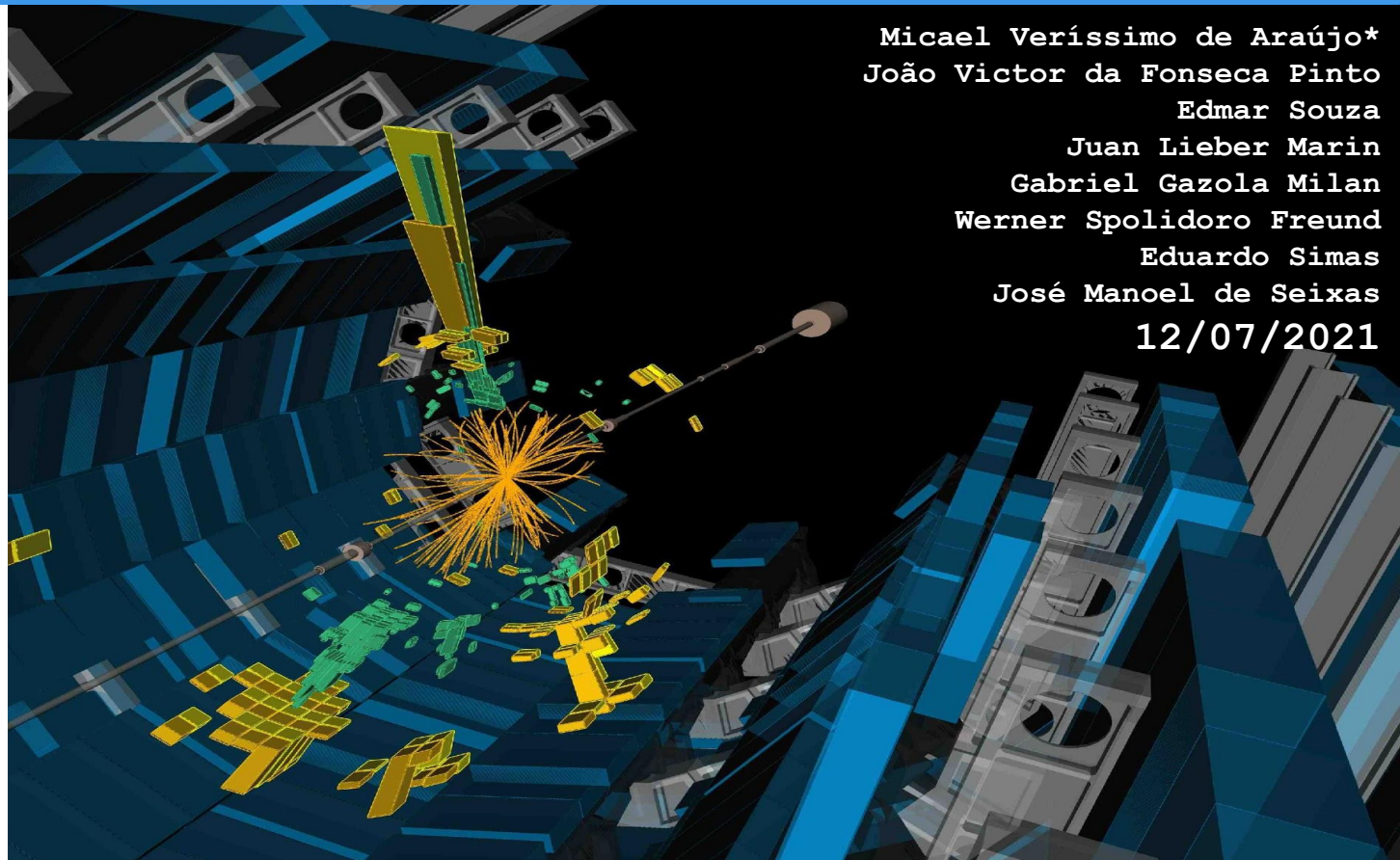
# Framework Integrado para Análise de Dados e Treinamento de Modelos de Aprendizagem de Máquina que operam no Trigger Online do Experimento ATLAS

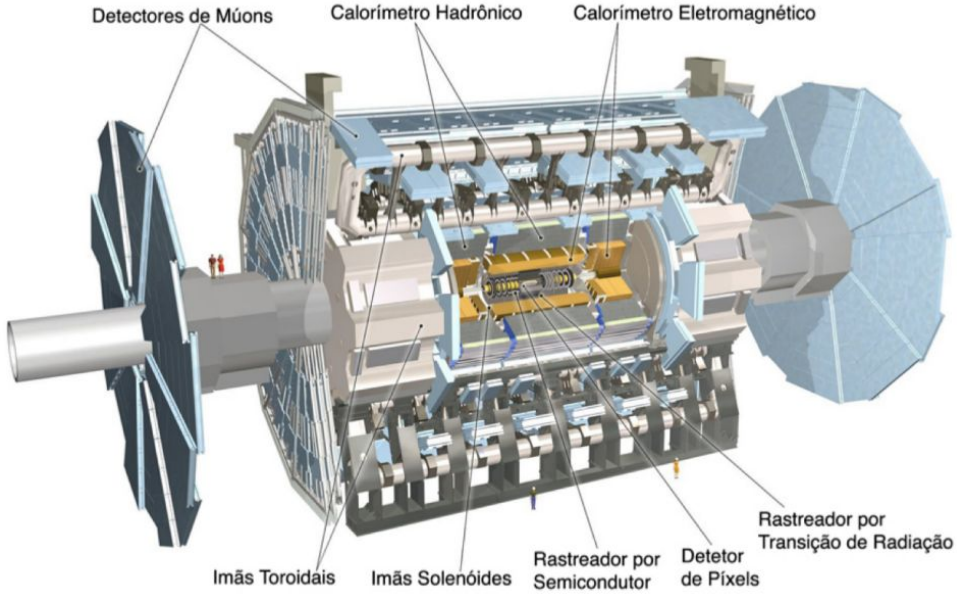


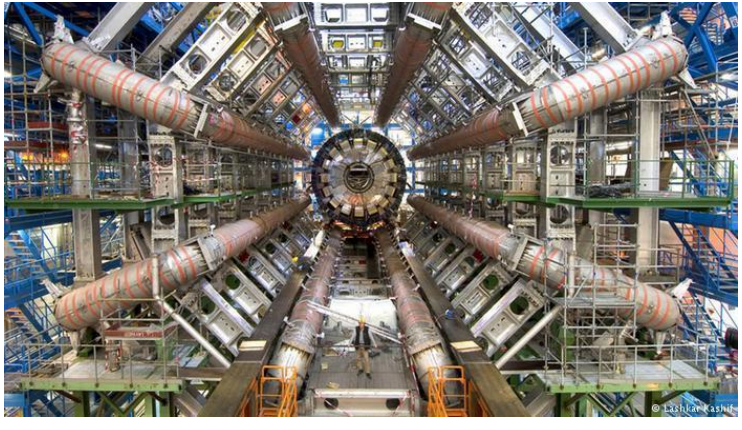

Micael Veríssimo de Araújo*
João Victor da Fonseca Pinto
Edmar Souza
Juan Lieber Marin
Gabriel Gazola Milan
Werner Spolidoro Freund
Eduardo Simas
José Manoel de Seixas


12/07/2021

# Contexto

- O ATLAS o maior experimento do LHC, medindo 44 metros de comprimento e 25 metros de largura.
- Possui 3 sistemas principais de detectores:
  - Detector Interno;
  - Sistema de Calorimetria;
  - Espectrômetro de Múons.

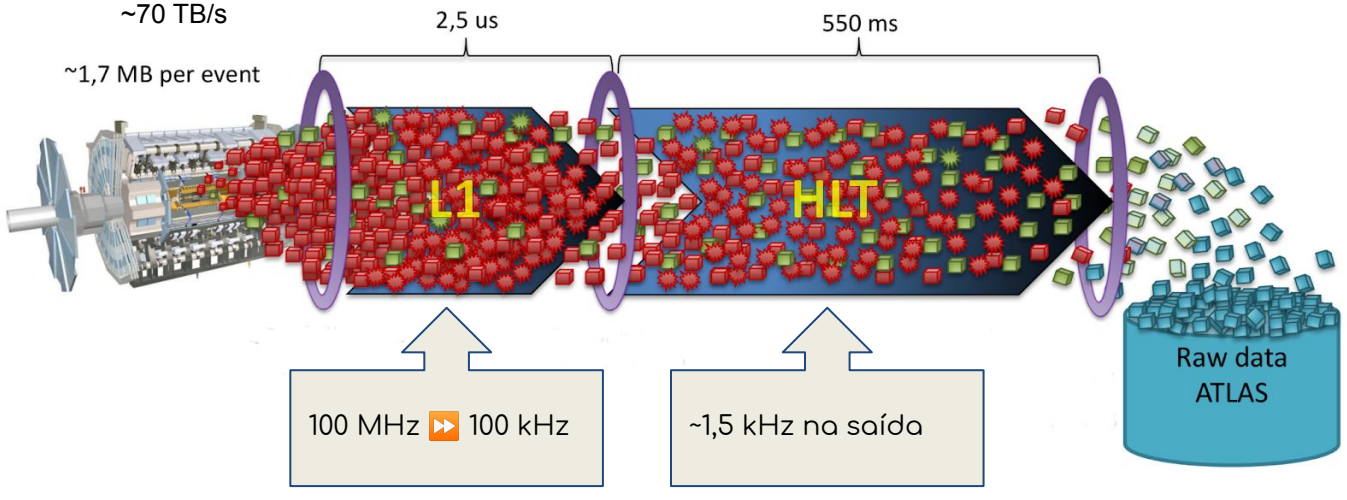


- Construído em camadas.
- Propósito geral.
- O detector é não totalmente uniforme.

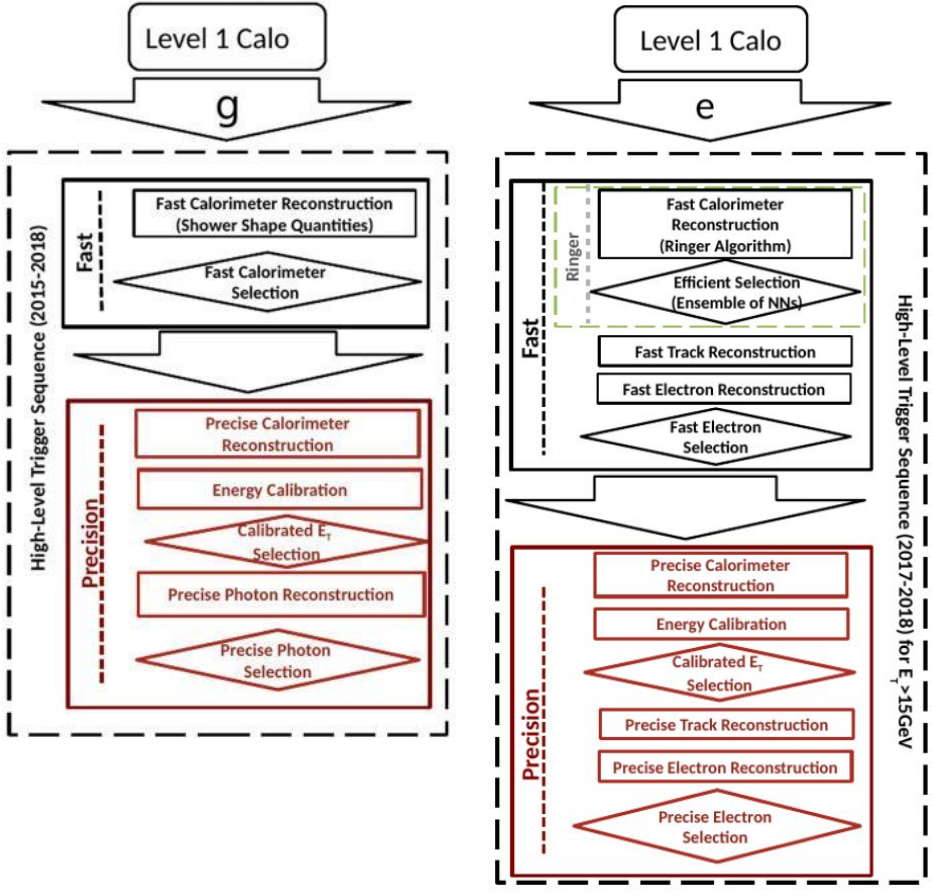
# Contexto

- O sistema de filtragem (Trigger) de elétrons do ATLAS foi desenvolvido para suportar a alta taxa de eventos, a grande quantidade de canais de leitura e o imenso ruído produzido pelas colisões;
- A atuação deste sistema se dá em duas etapas:
  - Primeiro nível de filtragem ➜ L1 implementado em hardware;
  - Trigger de alto nível (High Level Filter - HLT)➜ implementado em software



# Contexto

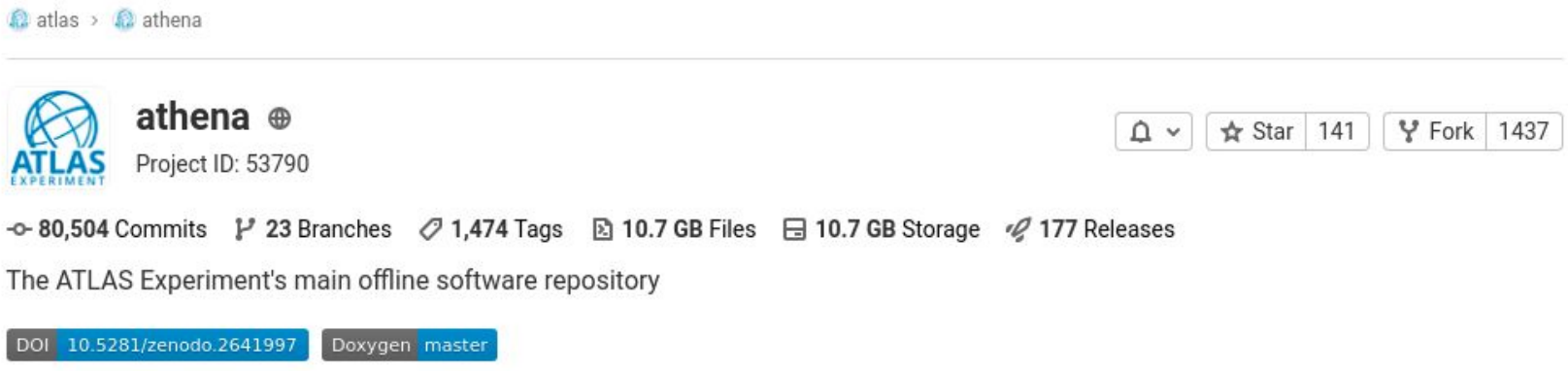
- A filtragem de e/γ no HLT se divide em duas etapas:
  - **Fast:** que utiliza os detectores com a resposta mais rápida para realizar uma seleção eficiente com uma baixa latência;
  - **Precision:** nesta etapa são utilizados algoritmos mais complexos para tomar a decisão se um candidato é ou não aprovado;
- Devido ao fato dos fótons não possuírem carga elétrica, não existe uma etapa reconstrução rápida de traços associada para as cadeias de fótons;
- Também é importante mencionar que a reconstrução rápida de calorimetria de fótons e elétrons é idêntica;



# Contexto

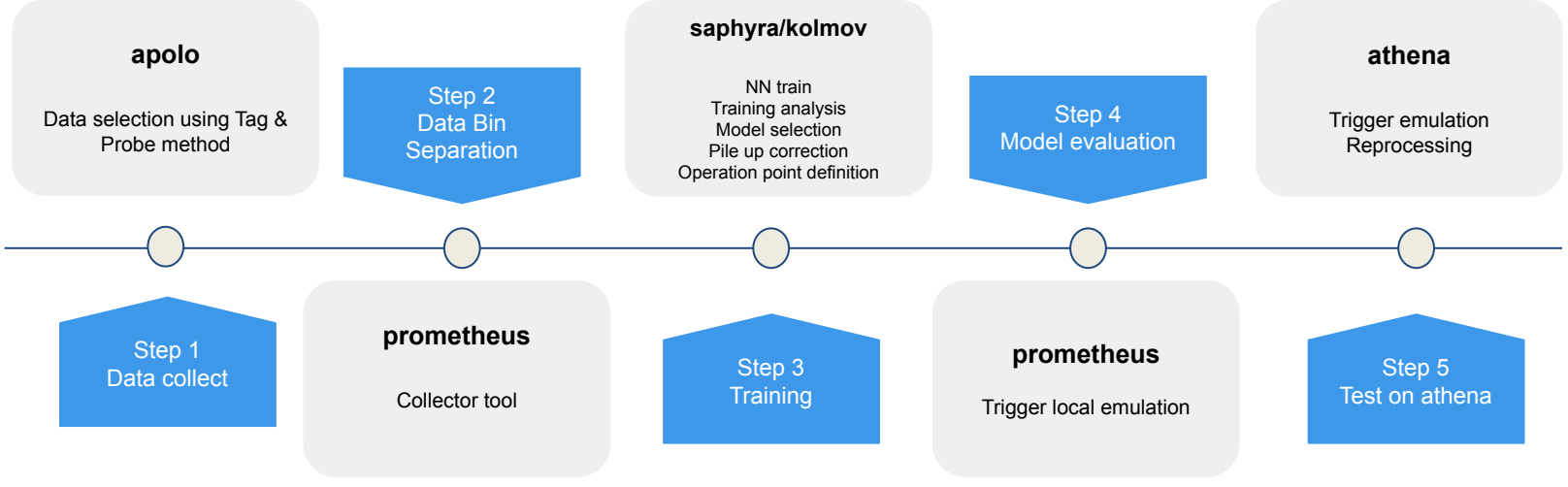
- Para lidar com as análises o ATLAS possui seu próprio *software* de análise: o Athena é responsável pela maioria das tarefas tais como:
  - Geração de eventos;
  - Simulação dos algoritmos de filtragem, monitoração e análise;
  - Reconstrução das características da física de interesse;
  - Produção de derivações (amostras para análises de assinaturas específicas).

- O Athena também é utilizado no HLT, ou seja, todos os algoritmos de filtragem devem estar implementados no Athena;

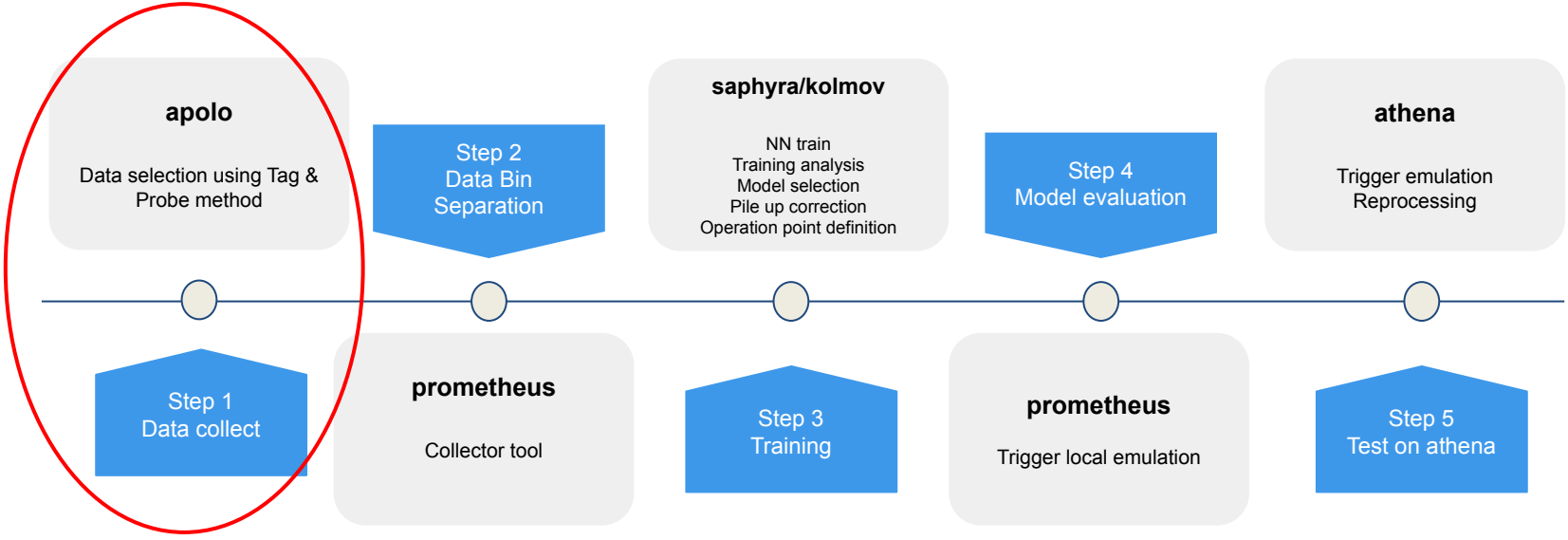


# Framework @workflow

- Para desenvolver e analisar os modelos de *Machine Learning* que vem sendo utilizados no NeuralRinger, foi desenvolvido um conjunto de pacotes que atuam nas diferentes etapas do fluxo de desenvolvimento, tais como:
  - Coleta de dados para treinamento;
  - Treinamento de modelos;
  - Avaliação e teste dos modelos.

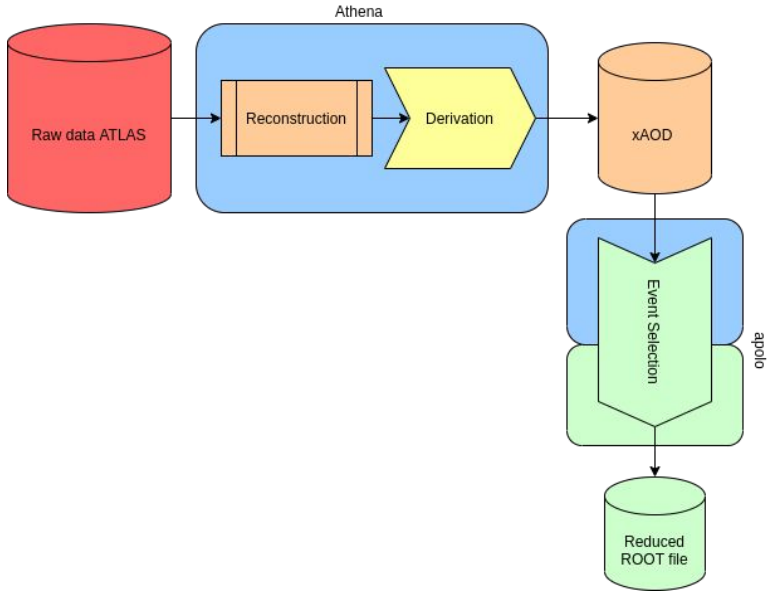


# Framework @workflow



# Framework @apolo

- O apolo é o pacote responsável por realizar a seleção e coleta das amostras que serão utilizadas para os treinamentos;
- Como input ele recebe os dados obtidos das derivações do Athena;
- Neste pacote existem dependências de subpacotes do Athena utilizados para realizar a navegação nos dados por exemplo.




Joao Victor Da Fonseca Pinto › apolo

A

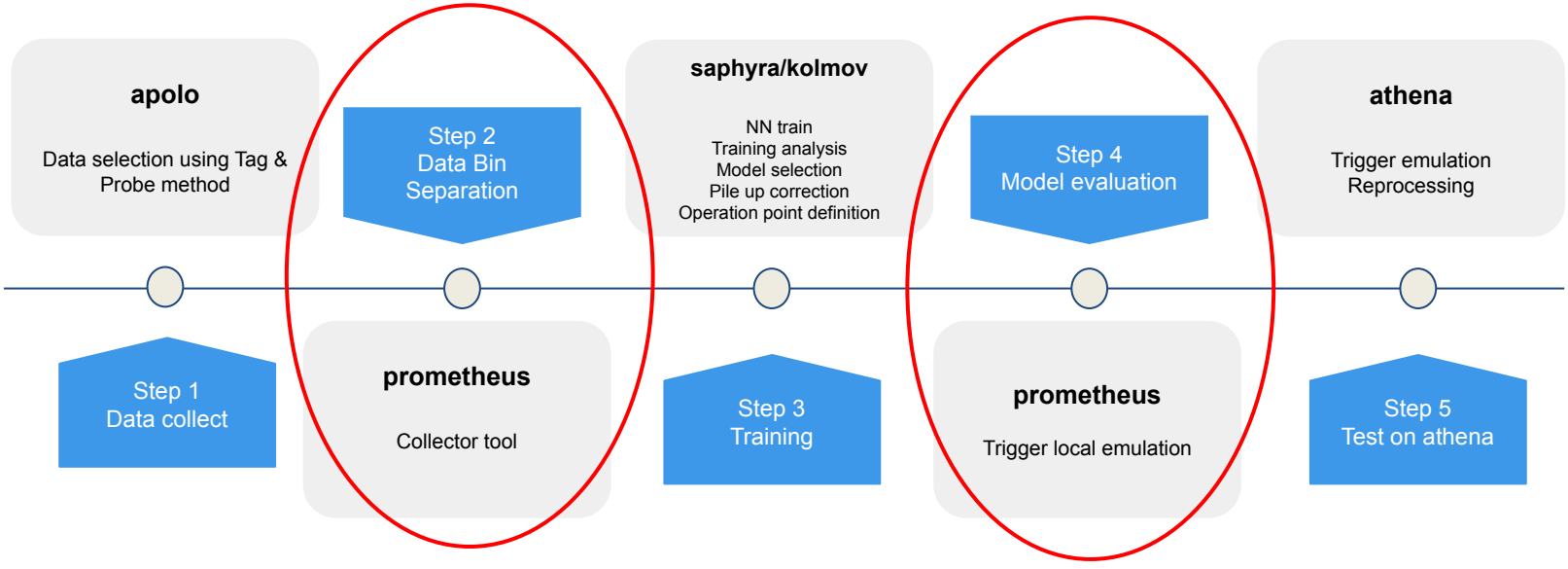
**apolo**

Project ID: 98874 Leave project

Star 0 Fork 0

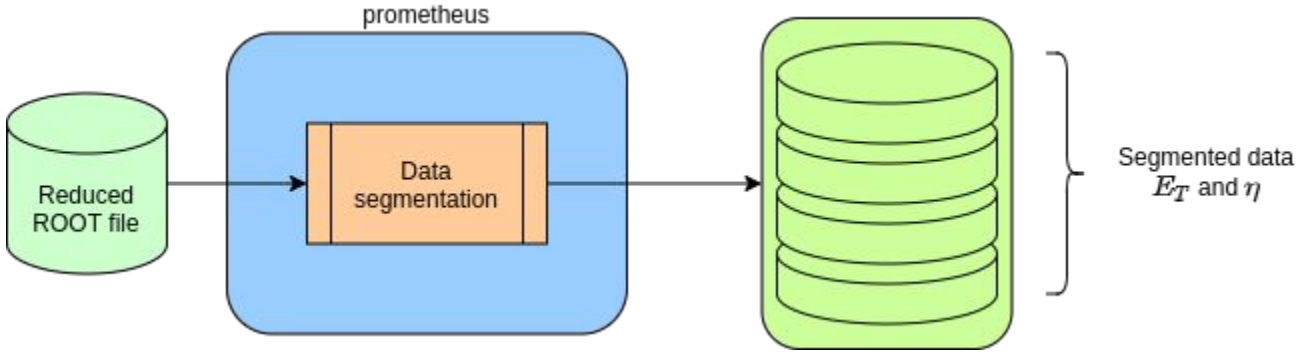
**54** Commits **2** Branches **0** Tags **1.3 MB** Files **5 MB** Storage

# Framework @workflow



# Framework @prometheus

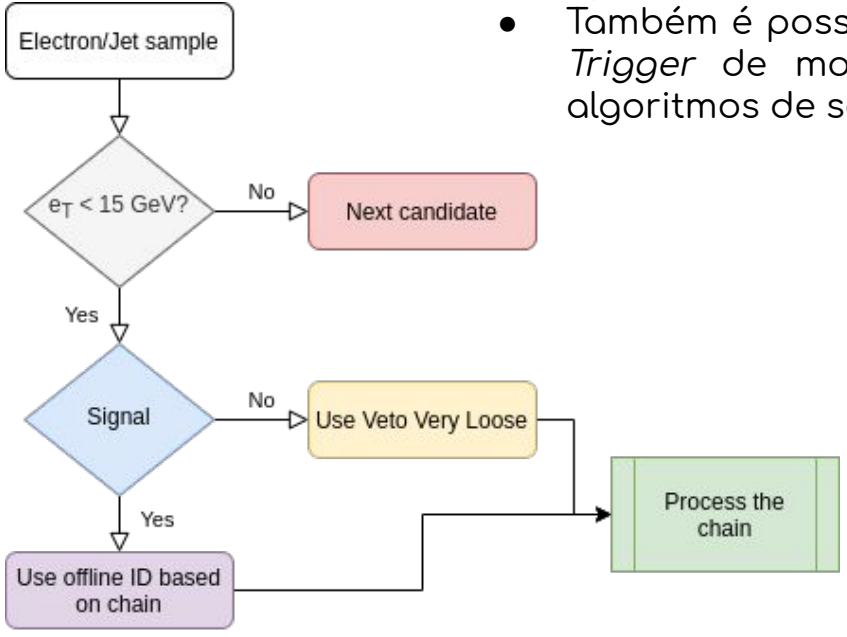
- O prometheus é um pacote que não possui nenhuma dependência do Athena.
- Entretanto ele foi projetado para emular o Athena permitindo realizar diversas análises que visando a operação do *Trigger*;
- No prometheus é onde realiza-se a segmentação dos dados em regiões de energia ($E_T$) e pseudorapidez ($\eta$);
- Atualmente o prometheus está hospedado no github: [prometheus]

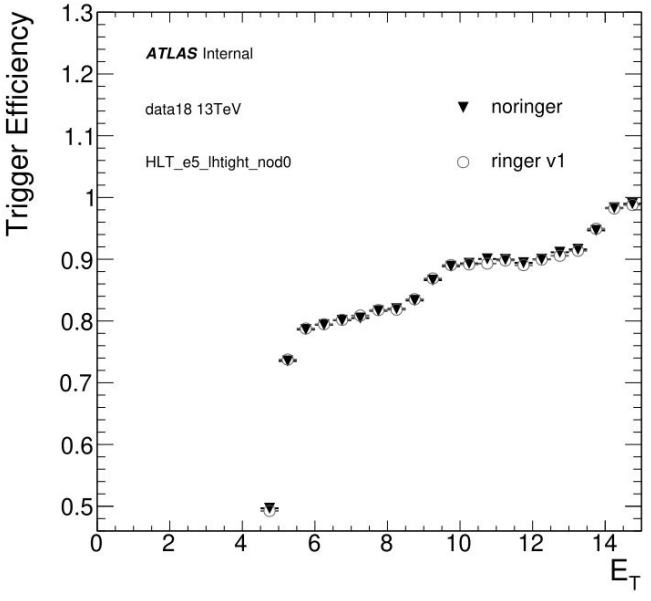



Os dados são obtidos em formato .npz

# Framework @prometheus

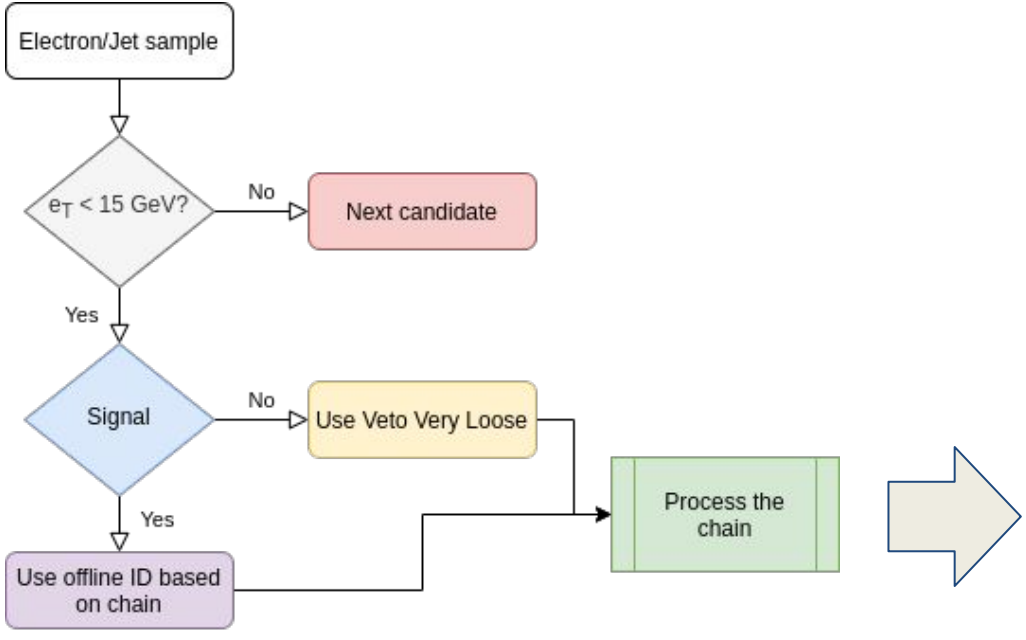
- No prometheus também é possível filtrar os candidatos sendo possível avaliar amostras específicas dentre os dados filtrados pelo apolo;
- Também é possível realizar todo o processamento de uma cadeia de *Trigger* de modo a emular seu comportamento com diferentes algoritmos de seleção;

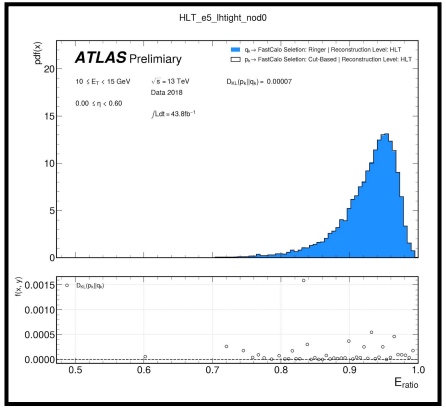


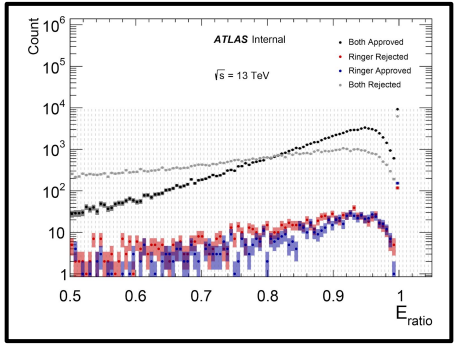



# Framework @prometheus

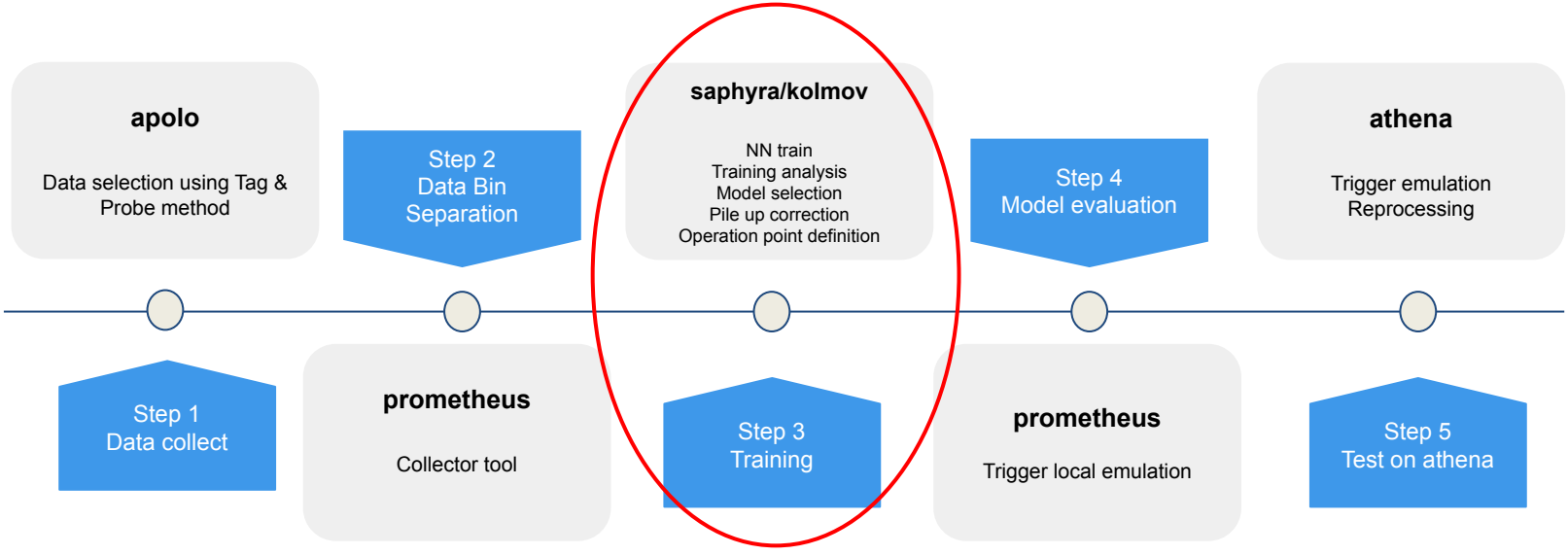
- Também é possível realizar comparações entre os diferentes algoritmos de seleção:
  - Análise de quadrante;
  - Análise de impacto.



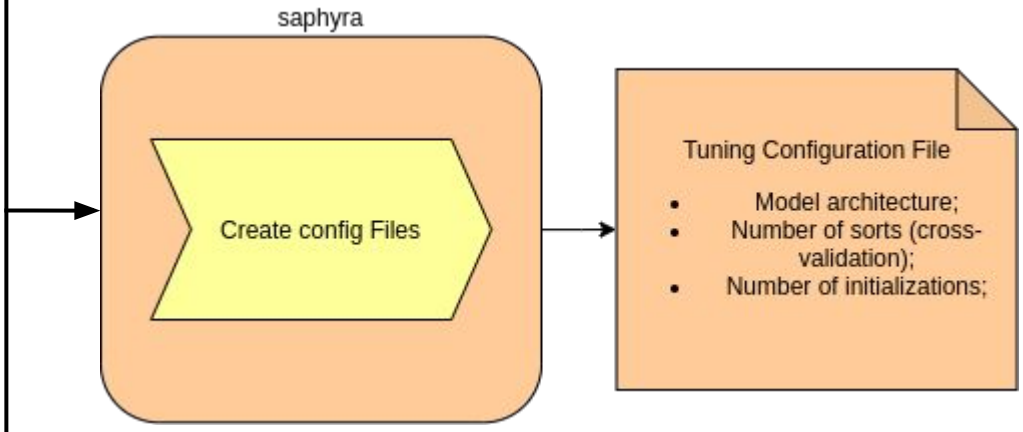
# Framework @workflow



# Framework @saphyra

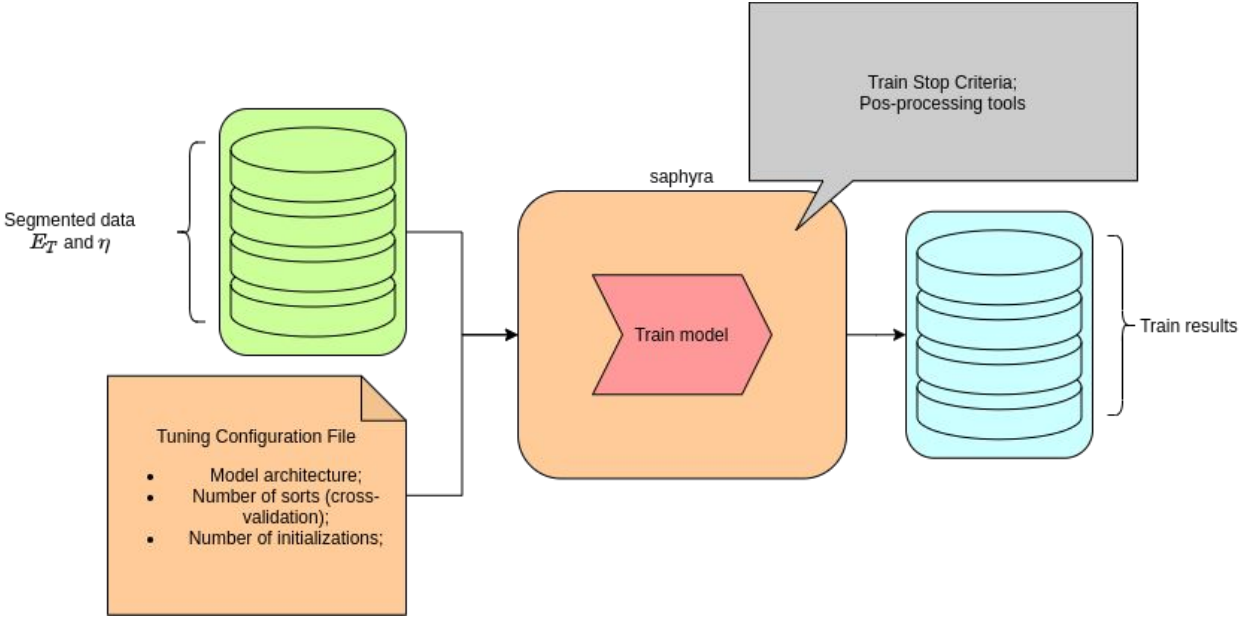
- O pacote saphyra é o responsável por realizar os treinamentos coletando as medidas necessárias para o processo de avaliação dos posterior dos modelos;
- Ele foi projetado para suportar modelos feitos com Keras (TensorFlow);
- No saphyra é possível executar de forma automatizada diferentes modelos de aprendizado de máquina;
- As configurações de treinamento são recebidas através de um arquivo de configuração;

```python
from saphyra import *


def get_model( ):
  modelCol = []
  from tensorflow.keras.models import Sequential
  from tensorflow.keras.layers import Dense, Dropout, Activation, Conv1D, Flatten
  for n in range(2,5+1):
    model = Sequential()
    model.add(Dense(n, input_shape=(100,), activation='tanh', name='dense_layer'))
    model.add(Dense(1, activation='linear', name='output_for_inference'))
    model.add(Activation('tanh', name='output_for_training'))
    modelCol.append(model)
  return modelCol



create_jobs( models = get_model(),
        nInits        = 5,
        nInitsPerJob  = 1,
        sortBounds    = 10,
        nSortsPerJob  = 1,
        nModelsPerJob = 5,
        outputFolder  = 'job_config.Jpsiee_v1.n2to5.10sorts.5inits' )
```



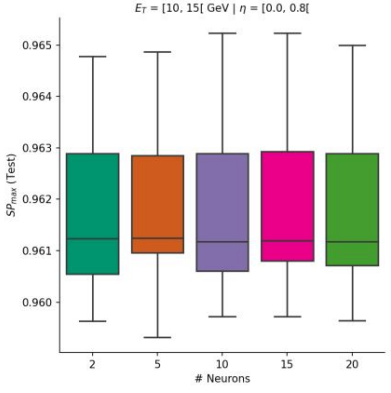
# Framework @saphyra

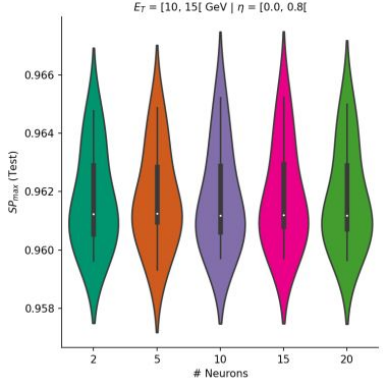
- Uma vez criadas as configurações do treinamento o saphyra irá utilizar os dados segmentados para executar os treinamentos seguindo as instruções obtidas dos arquivos de configuração;
- Após o treinamento é possível executar outras ferramentas tais como:
  - Ajuste de operação por referência dada;
  - Cálculo de curvas ROC;



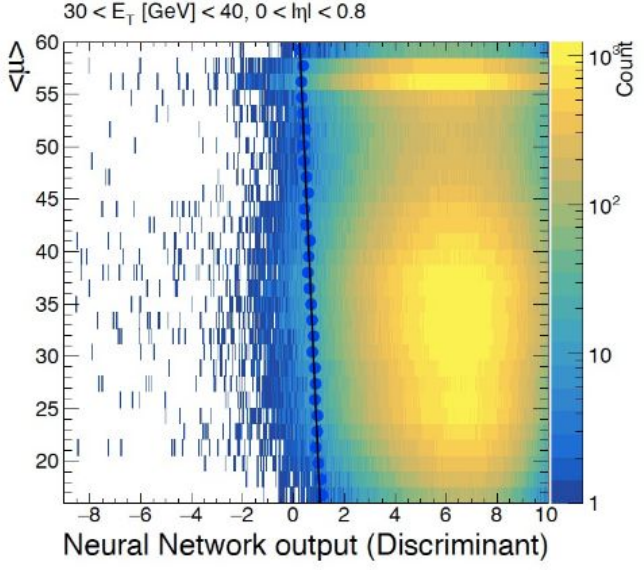
# Framework @kolmov

- No pacote kolmov é possível realizar a análise inicial dos modelos treinados pelo saphyra;
- O pacote é foi totalmente implementado em python;
- Nele, é possível escolher o melhor modelo obtido pelo processo de validação cruzada.
- A análise é feita em termos de probabilidade de detecção e probabilidade de falso alarme para cada região do espaço de fase definido no treinamento.
- Ademais, o framework provê a correção do modelo por valores de empilhamento de cada amostra.




| | $P_D$[%] | $P_F$[%] |
|---|---|---|
| Ref. | 97.78 | 36.91 |
| v1.data17.mlp2 | 97.78±0.01 | 10.30±0.30 |
| v1.data17.mlp5 | 97.79±0.01 | 10.30±0.38 |
| v1.data17.mlp10 | 97.78±0.01 | 10.27±0.32 |



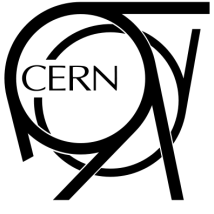
# Framework @Infraestrutura

- Os pacotes foram escritos utilizando majoritariamente python e C++
- Todo o framework foi encapsulado utilizando Docker:
  - Com suporte a treinamentos utilizando CPU's e GPU's;
  - Sendo possível realizar o uso stand alone sem necessidade de conexão remota com o CERN ou com a UFRJ
- Atualmente estes pacotes tem sido utilizados por pesquisadores ligados aos desenvolvimentos no Trigger:
  - Rio de Janeiro ➜ COPPE/UFRJ;
  - Minas Gerais ➜ UFJF;
  - Bahia ➜ UFBA;
  - Paris ➜ Sorbonne Université;
  - Berna ➜ Universität Bern.
- Tendo integração com os computadores do Laboratório de Processamento de Sinais da COPPE/UFRJ;
- Embora tenham sido planejados para trabalhos relativos ao ATLAS os pacotes podem ser adaptados para outros formatos de dados, sendo possível inclusive tratar de outros tipos de problemas de classificação:
  - Ex. Classificação de Sinais de Sonar;

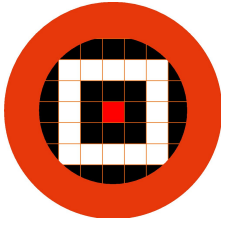
# Projeto

| | |
|---|---|
| Projeto | Física Experimental de Altas Energias e Tecnologias Associadas |
| Instituições | Brasil: UFRJ/COPPE; UFBA; UFJF; UERJ<br>Internacional: CERN, Sorbonne Université; Universität Bern; Universidade Nacional de La Plata; Indiana University Bloomington; |
| Timeline | 2021-2: início do processo de tuning dos modelos para Run 3 |
| Custos e fontes de recursos | Aquisição de computação distribuída (Lobo Carneiro e Santos Dummont); Agências de fomento: CAPES, CNPQ, FAPERJ, FAPESB, FAPEMIG |
| Possíveis sinergias | Uso intensivo de ML em sistemas de HEP; |
| Possíveis spin-offs | 6 empresas nascidas no LPS e podem ser inseridas no CERN |

# Obrigado!


CERN


ATLAS
EXPERIMENT

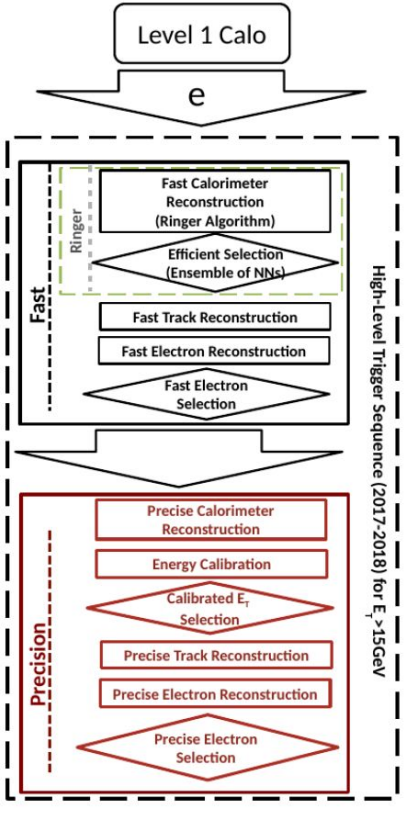


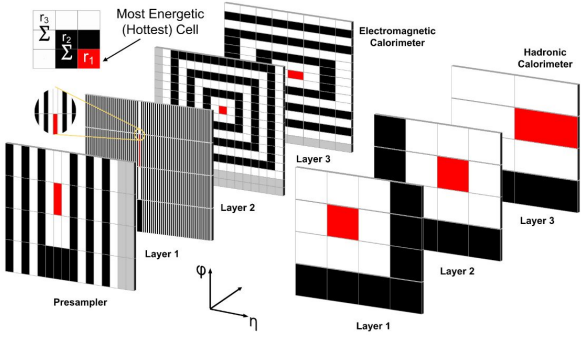

lps
COPPE/Poli/UFRJ
Laboratório de Processamento de Sinais
Inteligência Computacional. Inovação

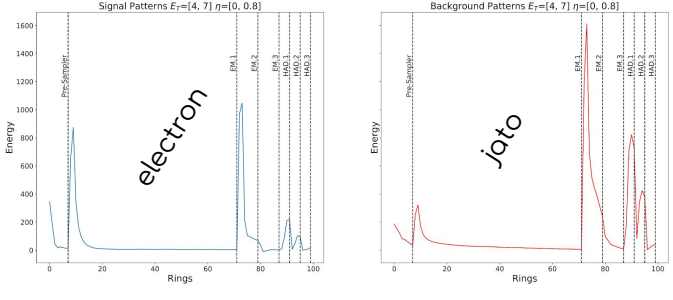
# Backup Slides

# Neural Ringer

- NeuralRinger é um dos algoritmos que atua na etapa rápida de calorimetria do Trigger de elétrons;
- Ele combina as propriedades dos chuveiros eletromagnéticos e Machine Learning para realizar uma seleção eficiente utilizando apenas a informação advinda do sistema de calorimetria;
- A informação do sistema de calorimetria é compactada em 'anéis' de energia que por sua vez são utilizados para construir um ensemble de redes neurais para tomada de decisão;
- São construídos um total de 100 anéis e o ensemble é composto atualmente por 25 redes neurais que atuam por regiões de energia ($E_T$) e pseudorapidez ($\eta$);
- Na Run3 o ensemble irá aumentar para 40 redes neurais de modo a englobar também as cadeias com $E_T$ < 15 GeV;




| Total number of Rings per layer (covering 0.4 x 0.4 region in η x φ) | | | | | | |
|---|---|---|---|---|---|---|
| PS | EM1 | EM2 | EM3 | HAD1 | HAD2 | HAD3 |
| 8 | 64 | 8 | 8 | 4 | 4 | 4 |