12.02 easily compile now

- The PtCalculator class is an independent toolbox for the sPHENIX experiment that can reconstruct the transverse momentum pT of charged particles with only the silicon and calorimeter detectors.
- First, you need to compile PtCalculator class,Some packages are required (mainly ONNX and JSON; on the RCF cluster these are the only two missing), and you can compile the Class with mine.
- Second, after compilation, you can create a PtCalculator instance, set the required input information, and then call ComputePt() to select the appropriate method and obtain the reconstructed result.

We do not provide a compiled library here; instead, the user needs to compile it themselves, which makes it easier to modify the code of the PtCalculator class.

How to use it

1. get the git repo:

```
git clone https://github.com/Joy-whale0321/INTT-EMCAL.git
```

2. compile (Change into this directory in the repository, then just run `make`.)

```
cd  InttSeedingTrackDev/ML4Reco/Implement/src
make
```

3. Load the library of ***PtCalculator***, then follow the example in PtCalcMLTutorial.C to use the class.

```
#include "PtCalculator.h"
gSystem->Load("libPtCalc.so");
```

For more details, please refer to PtCalcMLTutorial.C on the repo and “PT Calculator document”.

to do list

- Adding η-dependence to the formula for calculating pT using the magnetic deflection angle
- previous studies were based on EMCAL clusters reconstructed with surface center; subsequently, the process will be repeated using different geometries to derive formulas for reconstruction results based on different EMCAL tower positions.

Previously Presented Materials

Calculate pt class Tutorial

Jingyu 11.10

Access my code

- https://github.com/Joy-whale0321/INTT-EMCAL/tree/main/InttSeedingTrackDev/ML4Reco/Implement (will be released on sPHENIX SiCalo repo when I finished)
- PtCalculator compute the pt
  - PtCalculator.h
  - PtCalculator.cc
- Makefile: Compilate PtCalculator class
- PtCalcMLTutorial.C: A tutorial for you

- others
  - export_to_onnx.py: turning the ML model weights and scaler to onnx and json format
  - model_MLCombined.onnx, model_MLEMD.onnx, model_MLEproj.onnx, scaler_MLEproj.json: model weights and scaler

PtCalculator Class

This class can run independently of Fun4All; it is not derived from SubsysReco. You can call it freely and multiple times as a standalone utility.

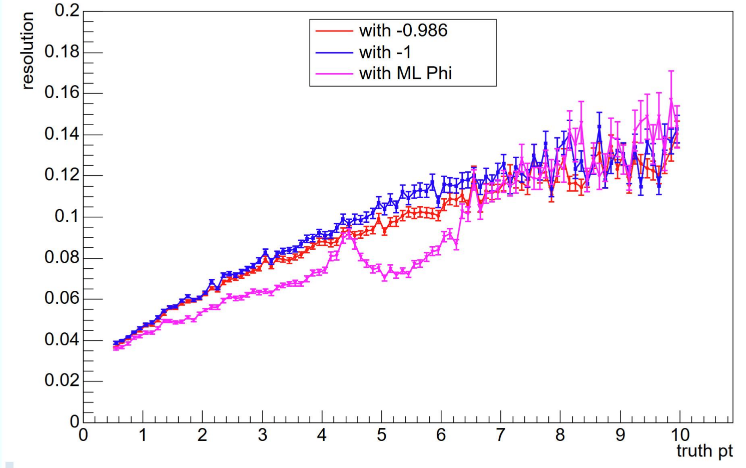
It takes a input struct (which depends on the chosen method) and directly produces an output PtResult, which contains the reconstructed pt along with some additional information.

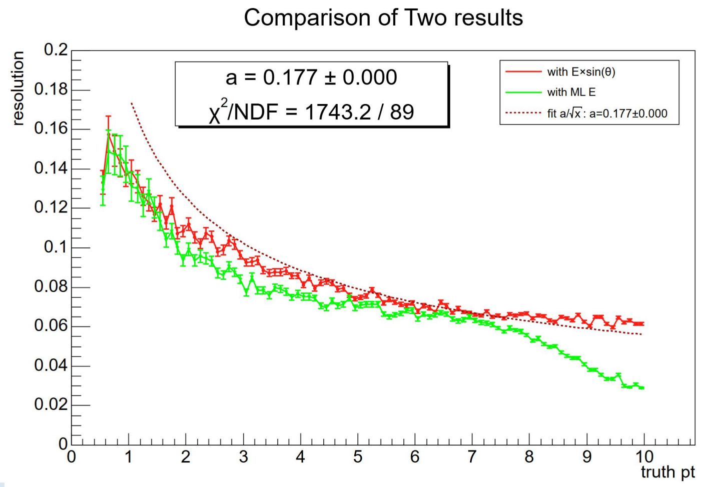
Five approaches to calculate pt, and the perform.

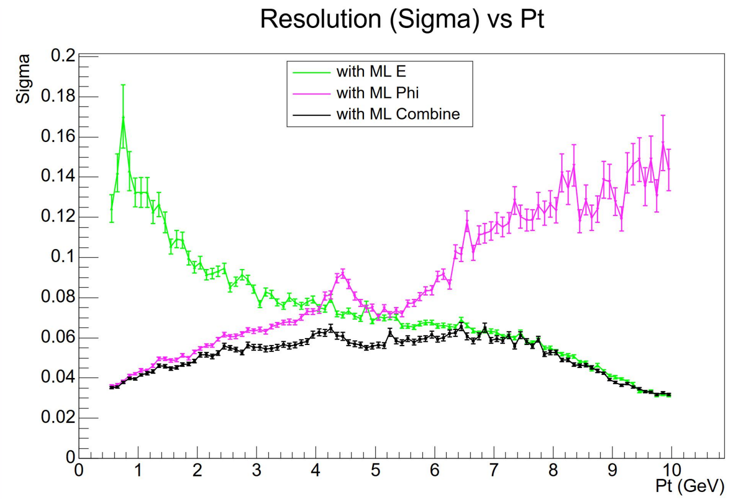
```cpp
// Five approaches enum
enum class Method
{
    MethodEMD,    // EM Deflection Method
    MethodEproj, // Energy Projection Method
    MethodMLEMD, // ML EM Deflection Method
    MethodMLEproj, // ML Energy Projection Method
    MethodMLCombined // ML Combined Method
};
```

Output Struct:

```cpp
// consistent return struct
struct PtResult
{
    float pt_reco = std::numeric_limits<float>::quiet_NaN(); // reconstructed pt
    bool ok = false; // whether successful
    std::string err; // error message if any
};
```







Note: lack of truth higher-pT data causes the model to tend to underestimate pT now

Different input struct and consistent interface

Consistent to ComputePt Func.

```
// consistent input variant for all methods
using AnyInput = std::variant<InputEMD, InputEproj, InputMLEMD, InputMLEproj, InputMLCombined>;
```

Each method has its corresponding input.

```
// Formula method input struct
struct InputEMD
{
    float EMD_Angle = 0.f;
    float EMD_Eta = 0.f;
    float EMD_Radius = 0.f;
};

struct InputEproj
{
    float Energy_Calo = 0.f;
    float Radius_Calo = 0.f;
    float Z_Calo = 0.f;
    float Radius_vertex = 0.f;
    float Z_vertex = 0.f;
};

// ML Model input struct, using vector is convenient for ONNX, length and order must be consistent with training
// Defination of length and order is in the tutorial file PtCalcMLTutorial.C
struct InputMLEMD
{
    std::vector<float> features;
};

struct InputMLEproj
{
    std::vector<float> features;
};

struct InputMLCombined
{
    std::vector<float> features;
};
```

PtResult ComputePt(Method method, const AnyInput& input) const

ComputePt receives the ***Method value(enum)*** and a ***variant AnyInput***.
Based on the selected method, it fetches the matching input struct from the variant, dispatches to the corresponding compute implementation, and returns a unified PtResult containing the reconstructed pT, a success flag, and an optional error message.

```
// general func. for Pt calculation, dispatching according to method
PtResult ComputePt(Method method, const AnyInput& input) const;

// independent interfaces for finer control
PtResult ComputeEMD(const InputEMD& in) const;
PtResult ComputeEproj(const InputEproj& in) const;
PtResult ComputeMLEMD(const InputMLEMD& in) const;
PtResult ComputeMLEproj(const InputMLEproj& in) const;
PtResult ComputeMLCombined(const InputMLCombined& in) const;
```

```
PtResult PtCalculator::ComputePt(Method method, const AnyInput& input) const
{
    try
    {
        switch (method)
        {
            case Method::MethodEMD:
                return ComputeEMD(std::get<InputEMD>(input));
            case Method::MethodEproj:
                return ComputeEproj(std::get<InputEproj>(input));
            case Method::MethodMLEMD:
                return ComputeMLEMD(std::get<InputMLEMD>(input));
            case Method::MethodMLEproj:
                return ComputeMLEproj(std::get<InputMLEproj>(input));
            case Method::MethodMLCombined:
                return ComputeMLCombined(std::get<InputMLCombined>(input));
            default:
                return PtResult{.pt_reco = NAN, .ok = false, .err = "Unknown method"};
        }
    }
    catch (const std::bad_variant_access&)
    {
        return PtResult{.pt_reco = NAN, .ok = false, .err = "Input type does not match method"};
    }
}
```

others func. of PtCalculator class

- Please first refer to the comments provided in the code.

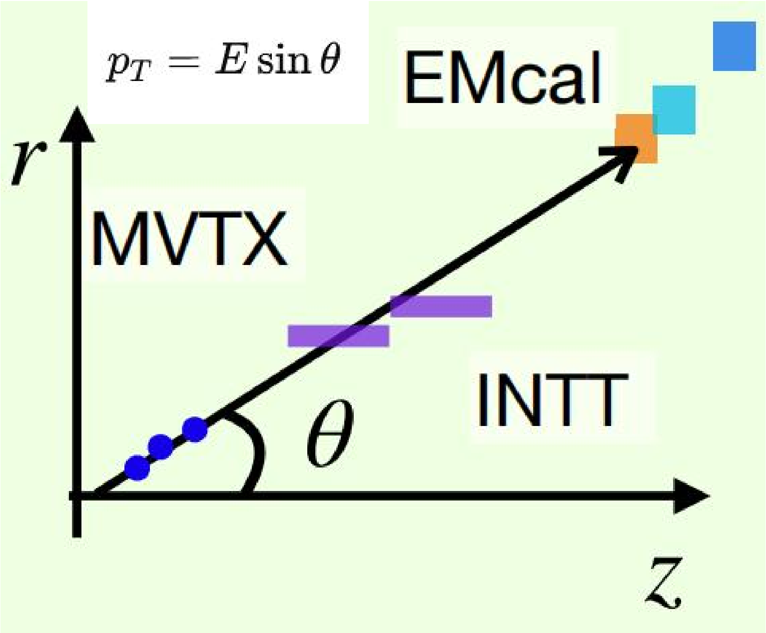
A tutorial PtCalcMLTutorial.C

- remember change the path of model weight and scaler to yours

```
jingyu@Jingyu7000:/mnt/e/sphenix/INTT-EMCAL/InttSeedingTrackDev/ML4Reco/Implement/src$ root -b -q PtCalcMLTutorial.C
   ------------------------------------------------------------------
  | Welcome to ROOT 6.28/04                        https://root.cern |
  | (c) 1995-2022, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on May 08 2023, 02:44:07                 |
  | From tags/v6-28-04@v6-28-04                                      |
  | With c++ (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0                   |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------


Processing PtCalcMLTutorial.C...
[init] OK
[EMD-analytic] ok=1  pt=8  err=""
[Eproj-analytic] ok=1  pt=1.8  err=""
[MLEMD-2D] ok=1  pt=2.9584  err=""
[MLEproj-7D] ok=1  pt=5.18625  err=""
[MLCombined] ok=1  pt=9.36326  err=""
jingyu@Jingyu7000:/mnt/e/sphenix/INTT-EMCAL/InttSeedingTrackDev/ML4Reco/Implement/src$
```

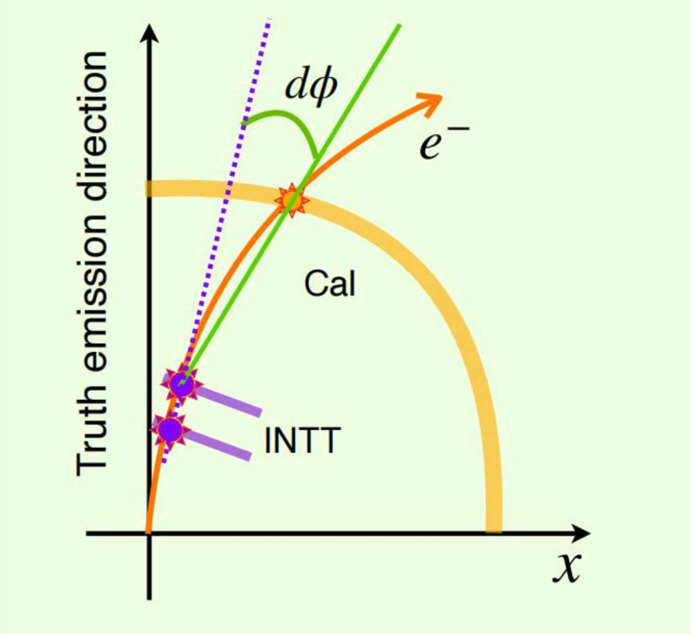
MethodEproj




```
// ======================= Eproj Formula =========================
{
    SiCaloPt::InputEproj in;
    in.Energy_Calo    = 1.8;    // EMCal Cluster energy in GeV
    in.Radius_Calo    = 93.5;   // EMCal Cluster radius in cm
    in.Z_Calo         = 0.0;    // EMCal Cluster z in cm
    in.Radius_vertex  = 0.0;    // Vertex radius in cm
    in.Z_vertex       = 0.0;    // Vertex z in cm

    auto r = calcTutorial.ComputePt(SiCaloPt::Method::MethodEproj, SiCaloPt::AnyInput{in});
    std::cout << "[Eproj-analytic] ok=" << r.ok
              << "  pt=" << r.pt_reco
              << "  err=\"" << r.err << "\"\n";
}
```

MethodEMD



pT = par_Ceta·(Δϕ^par_Power)

Func. setParCeta() and setParPower() to setting parameters
default: par_Ceta = 0.2, par_Power = -1 not consider eta-dependent now

```
// ======================== EMD Formula ==========================
{
    SiCaloPt::InputEMD in;
    in.EMD_Angle  = 0.025;  // delta_phi - EM Deflection angle in rad
    in.EMD_Eta    = 0.00;   // EMCal Cluster eta
    in.EMD_Radius = 93.5;   // EMCal Cluster radius in cm

    calcTutorial.setParCeta(0.2);    // set C_eta parameter if needed
    calcTutorial.setParPower(-1.0); // set Power parameter if needed

    auto r = calcTutorial.ComputePt(SiCaloPt::Method::MethodEMD, SiCaloPt::AnyInput{in});
    std::cout << "[EMD-analytic] ok=" << r.ok
              << "  pt=" << r.pt_reco
              << "  err=\"" << r.err << "\"\n";
}
```

MethodMLEproj

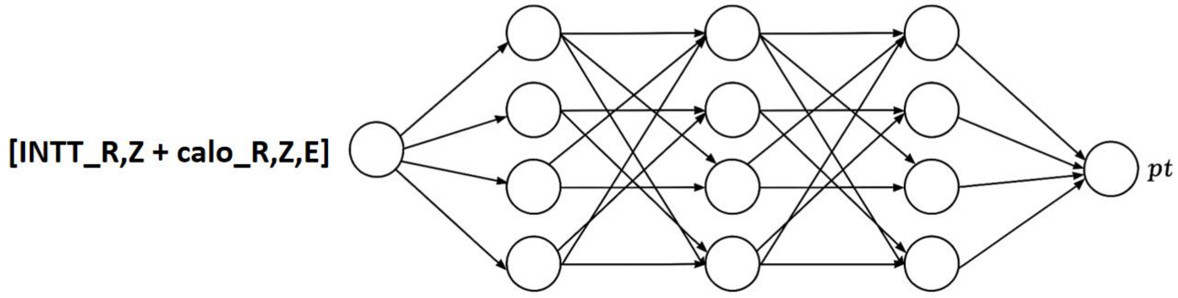
Train_model1 - [INTT_R,Z + calo_R,Z,E]

Hyper parameters:
- batch_size=1024, epochs=200, lr=5e-5, val_ratio=0.3

Loss function:
- pt_reso = (yb - pred) / (yb + 1e-6) — **Relative resolution, 1e-6 to prevent division by zero.**
- weights = (pt_reso.abs() < 0.2).float() * 2.0 + 1.0 — **Increase the weight inside the peak, reduce the influence of outlier data points**
- loss = ((pt_reso) ** 2 * weights).mean() — **Use squared values instead of absolute values to make the peak position closer to zero.**

- if val_loss < best_val_loss: Saved best model

10



MLP1: 3hidden layer, hidden_dim=256

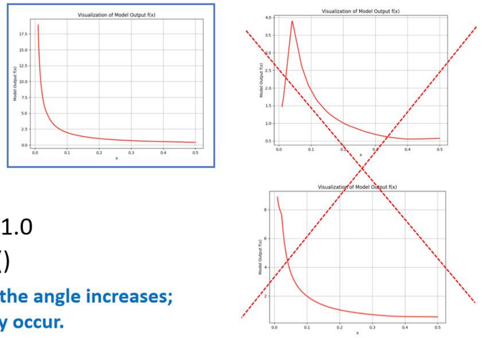
```cpp
// ====== ML: MLEproj (7-d input: INTT 3/4 layer R,Z, INTT 5/4 layer R,Z, Calo R,Z,Energy) ======
{
    // 7-d input features:{ INTT 3/4 layer R, INTT 3/4 layer Z, INTT 5/4 layer R, INTT 5/4 layer Z, Calo R, Calo Z, Calo Energy }
    std::vector<float> featsMLEproj = { 10.0,  5.0,   // INTT 3/4 layer
                                        15.0,  7.5,   // INTT 5/4 layer
                                        100.0, 50.0, 8.0 }; // Calo R,Z,Energy

    SiCaloPt::InputMLEproj in{featsMLEproj};
    auto r = calcTutorial.ComputePt(SiCaloPt::Method::MethodMLEproj, SiCaloPt::AnyInput{in});
    std::cout << "[MLEproj-7D] ok=" << r.ok
            << "  pt=" << r.pt_reco
            << "  err=\"" << r.err << "\"\n";
}
```

MethodMLEMD

Train_model2 - [1/ΔΦ]

model visualization



epochs=500.

Loss function:

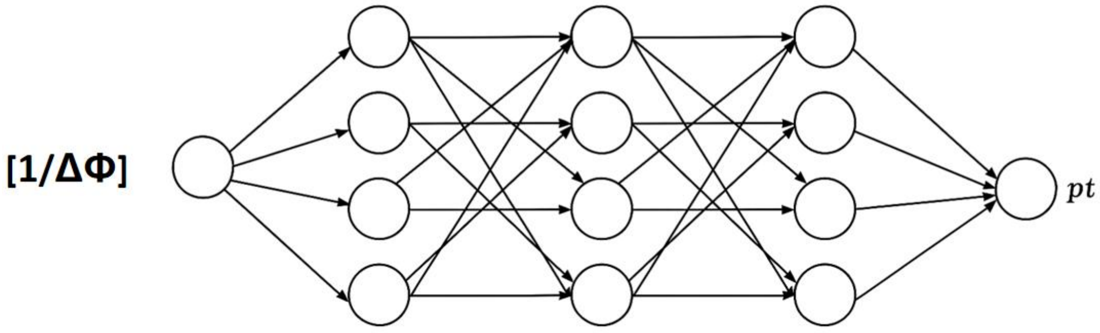
- pt_reso = (yb - pred) / (yb + 1e-6)
- weights = (pt_reso.abs() < 0.2).float() * 2.0 + 1.0
- main_loss = ((pt_reso) ** 2 * weights).mean()
- monotonic_loss
  - lambda_mono = 0.3

requires pt to decrease as the angle increases; otherwise, oscillations may occur.

- boundary_loss
  - [0.5, 1, 2, 10, 15, 25, 50, 100, 200] ->[0.0961, 0.1922, 0.3844, 1.922, 2.883, 4.805, 9.61, 19.22, 38.44] :
  - lambda_boundary = min(0.005 * epoch, 0.2)
- loss = main_loss
- + lambda_mono * monotonic_loss
- + lambda_boundary * boundary_loss

Add boundary conditions outside the data range to ensure that the pt estimate is sufficiently elevated at small angles.
Dynamic weighting to prevent affecting data learning in the early stages.

11



MLP2: 3hidden layer, hidden_dim=256

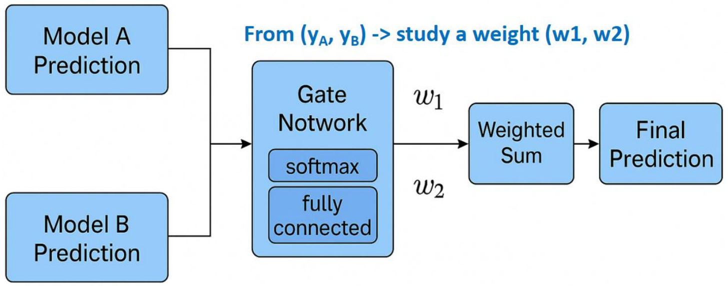
```
// ============= ML: MLEMD (2-d input: 1/dphi_EMD, eta_track = 0 now) ==============
{
    // 2-d input features:{ dphi_EMD, eta_track }
    std::vector<float> featsMLEMD = {15, 0};

    SiCaloPt::InputMLEMD in{featsMLEMD};
    auto r = calcTutorial.ComputePt(SiCaloPt::Method::MethodMLEMD, SiCaloPt::AnyInput{in});
    std::cout << "[MLEMD-2D] ok=" << r.ok
            << "  pt=" << r.pt_reco
            << "  err=\"" << r.err << "\"\n";
}
```

MethodMLCombined

Gated Fusion Model

Aiming to combine the prediction results of two existing models to achieve higher accuracy.
Using a Gate Network to dynamically adjust the weights based on the input.
This Gate Network takes the prediction values from both models as input and learns, through training, the optimal weighting strategy under different input conditions.
After training, the network dynamically generates two weights, w1 and w2, for each event to combined the predictions from 2 model



$$\hat{y} = w_1 \cdot y_A + w_2 \cdot y_B, \quad w_1 + w_2 = 1$$

39

```cpp
// =========== ML: Combined/Gate (2-d input: pt_from_MLEMD, pt_from_MLEproj) ==================
{
    // 2-d input features:{ pt_from_MLEMD, pt_from_MLEproj }
    std::vector<float> featsMLCombined = {8.0, 9.5};

    SiCaloPt::InputMLCombined in{featsMLCombined};
    auto r = calcTutorial.ComputePt(SiCaloPt::Method::MethodMLCombined, SiCaloPt::AnyInput{in});
    std::cout << "[MLCombined] ok=" << r.ok
              << "  pt=" << r.pt_reco
              << "  err=\"" << r.err << "\"\n";
}
```

worked on rcf, have not test on condor now

- ***get onnx on rcf sphnxuser05***

cd ~/Downloads

wget https://github.com/microsoft/onnxruntime/releases/download/v1.19.2/onnxruntime-linux-x64-1.19.2.tgz

tar -xf onnxruntime-linux-x64-1.19.2.tgz

mkdir -p ~/opt/onnxruntime

cp -r onnxruntime-linux-x64-1.19.2/include ~/opt/onnxruntime/

cp -r onnxruntime-linux-x64-1.19.2/lib   ~/opt/onnxruntime/

- ***get json on rcf sphnxuser05***

mkdir -p ~/opt/nlohmann/include/nlohmann

wget -O ~/opt/nlohmann/include/nlohmann/json.hpp \

  https://raw.githubusercontent.com/nlohmann/json/v3.11.3/single_include/nlohmann/json.hpp

Change the Path on Makefile:

ORT_PREFIX=/sphenix/u/jzhang1/opt/onnxruntime

JSON_INC=/sphenix/u/jzhang1/opt/nlohmann/include

```
jzhang1@sphnxuser05 /sphenix/user/jzhang1/testcode4all/INTT-EMCAL/InttSeedingTrackDev/ML4Reco/Implement/src$ root -b -q PtCalcMLTutorial.C
   ------------------------------------------------------------------
  | Welcome to ROOT 6.32.06                        https://root.cern |
  | (c) 1995-2024, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Dec 13 2024, 17:23:48                 |
  | From tags/6-32-06@6-32-06                                        |
  | With g++ (GCC) 14.2.0                                            |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------


Processing PtCalcMLTutorial.C...
[init] OK
[EMD-analytic] ok=1  pt=8  err=""
[Eproj-analytic] ok=1  pt=1.8  err=""
[MLEMD-2D] ok=1  pt=2.9584  err=""
[MLEproj-7D] ok=1  pt=5.18625  err=""
[MLCombined] ok=1  pt=9.36326  err=""
jzhang1@sphnxuser05 /sphenix/user/jzhang1/testcode4all/INTT-EMCAL/InttSeedingTrackDev/ML4Reco/Implement/src$
```

Thanks for your attention