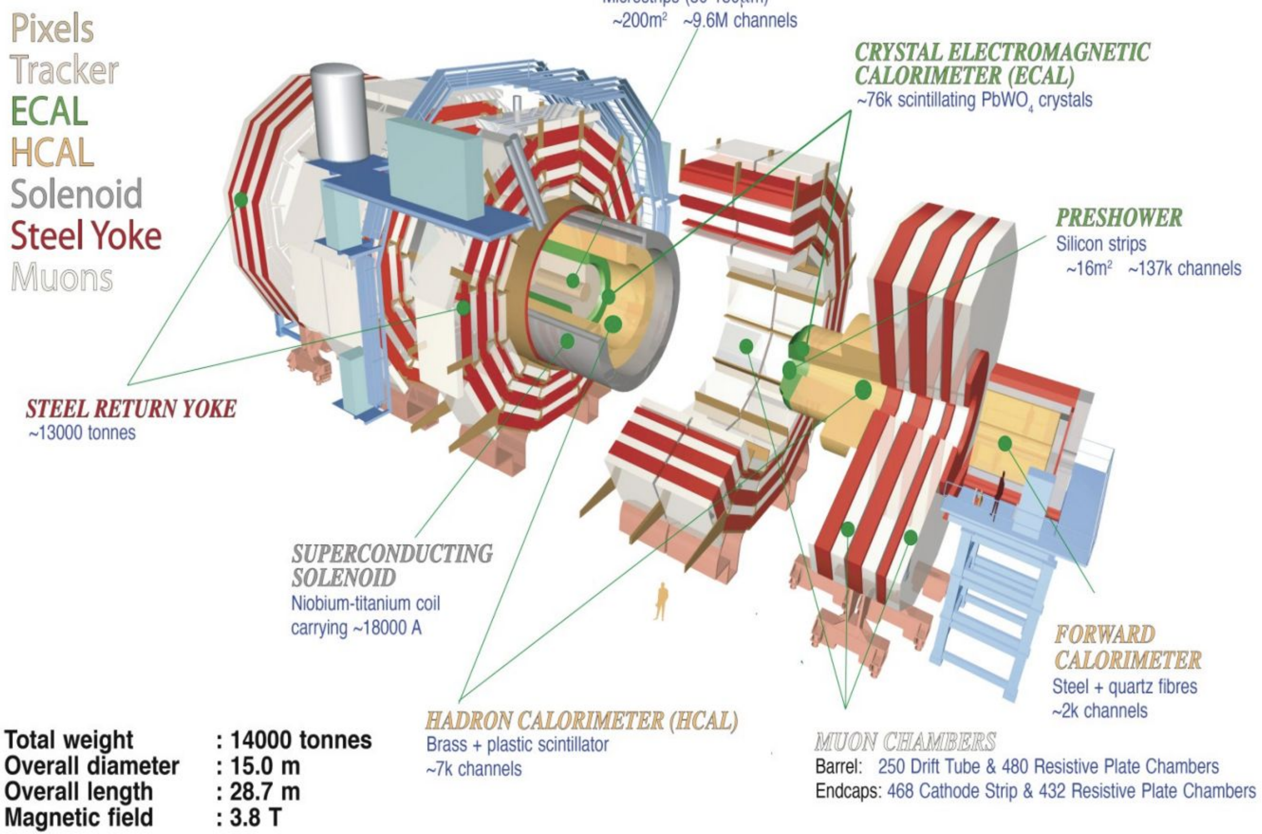
Block Practical Course: ETP Data Science

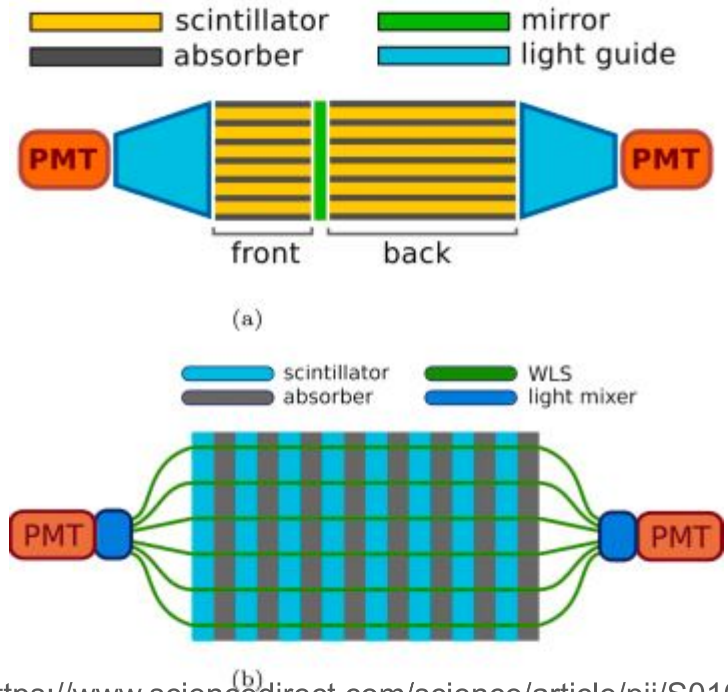
Final Presentation

CMS Detector



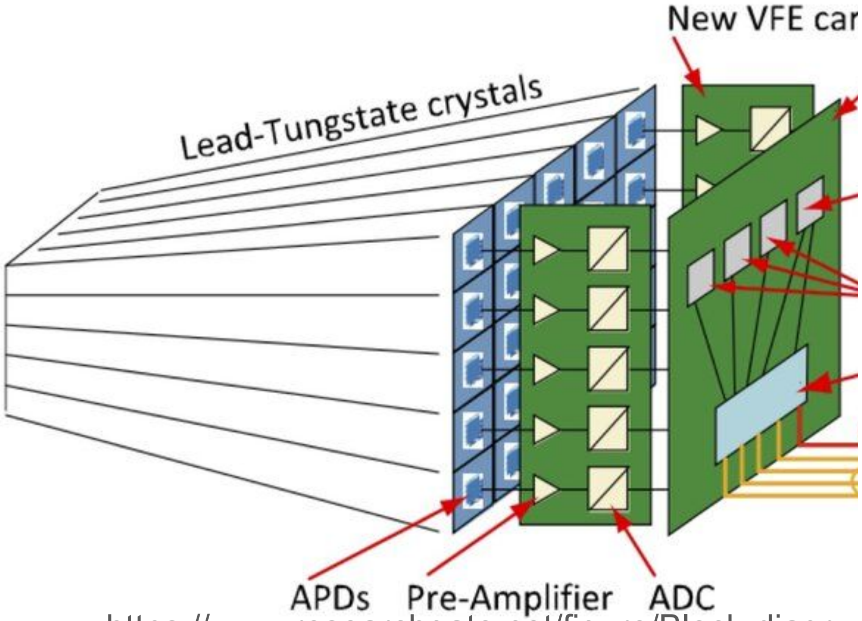
ECAL Designs

Sampling Calorimeter



https://www.sciencedirect.com/science/article/pii/S0168900225004097

Homogeneous



https://www.researchgate.net/figure/Block-diagram-for-the-upgrade-of-the-ECAL-barrel-electronics_fig1_323928662

Main Aspects of a sufficient (electromagnetic) calorimeter

- Containment (longitudinal/lateral) - detect and reconstruct all physics
- Energy Resolution - minimize noise, stochastic and constant terms

$$\frac{\sigma_E}{E} = \sqrt{\left(\frac{S}{\sqrt{E}}\right)^2 + \left(\frac{N}{E}\right)^2 + C^2}$$

- Granularity - CNN
- Response - examine all energy scales (1 - 100 GeVs in our case)

$$R(E) = \frac{\langle E_{\text{reco}} \rangle}{E_{\text{true}}}$$

EM Showers

The layer the most energy is deposited:

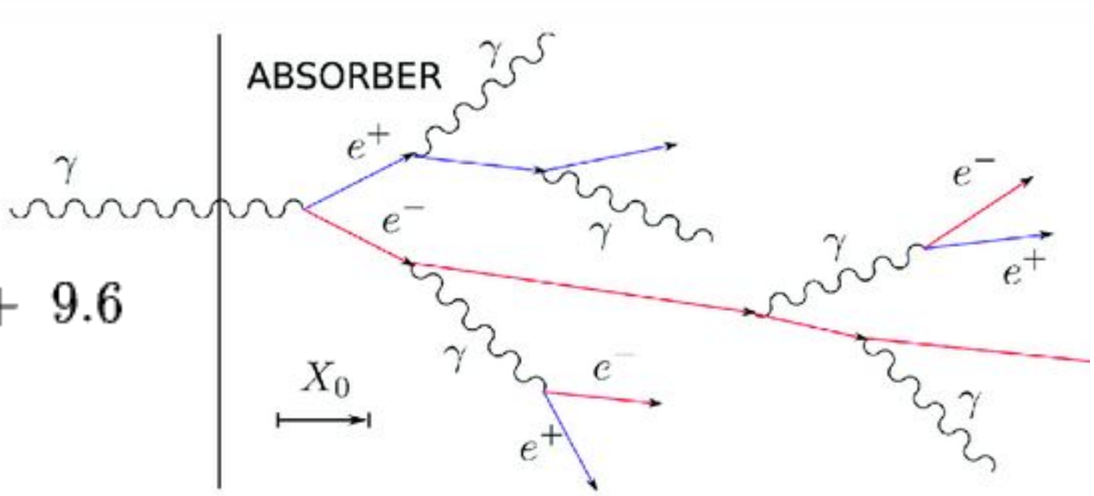
$$t_{\max} \simeq \ln\left(\frac{E}{E_c}\right)$$

$$X_0\ (\mathrm{g\,cm^{-2}}) \approx \frac{180\,A}{Z^2}\,.$$

$$L_{99\%}/X_0 \simeq t_{\max} + 0.08\,Z + 9.6$$

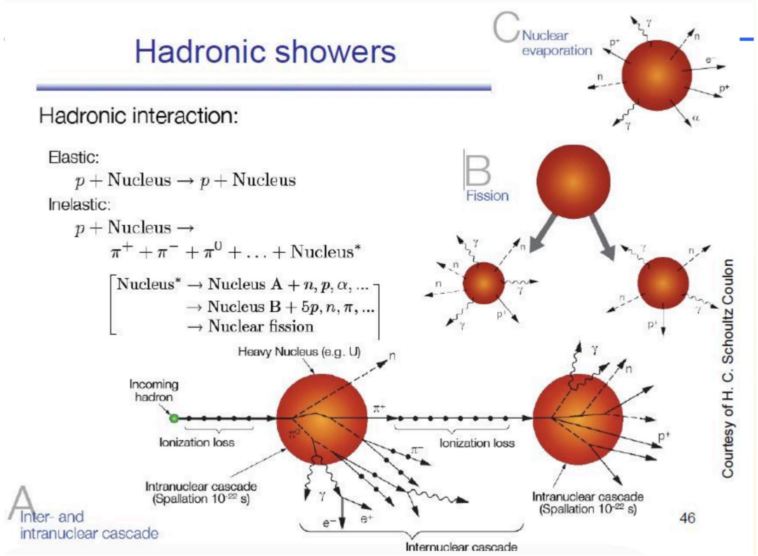
Roughly 95% is contained within 2 Moliere Radii

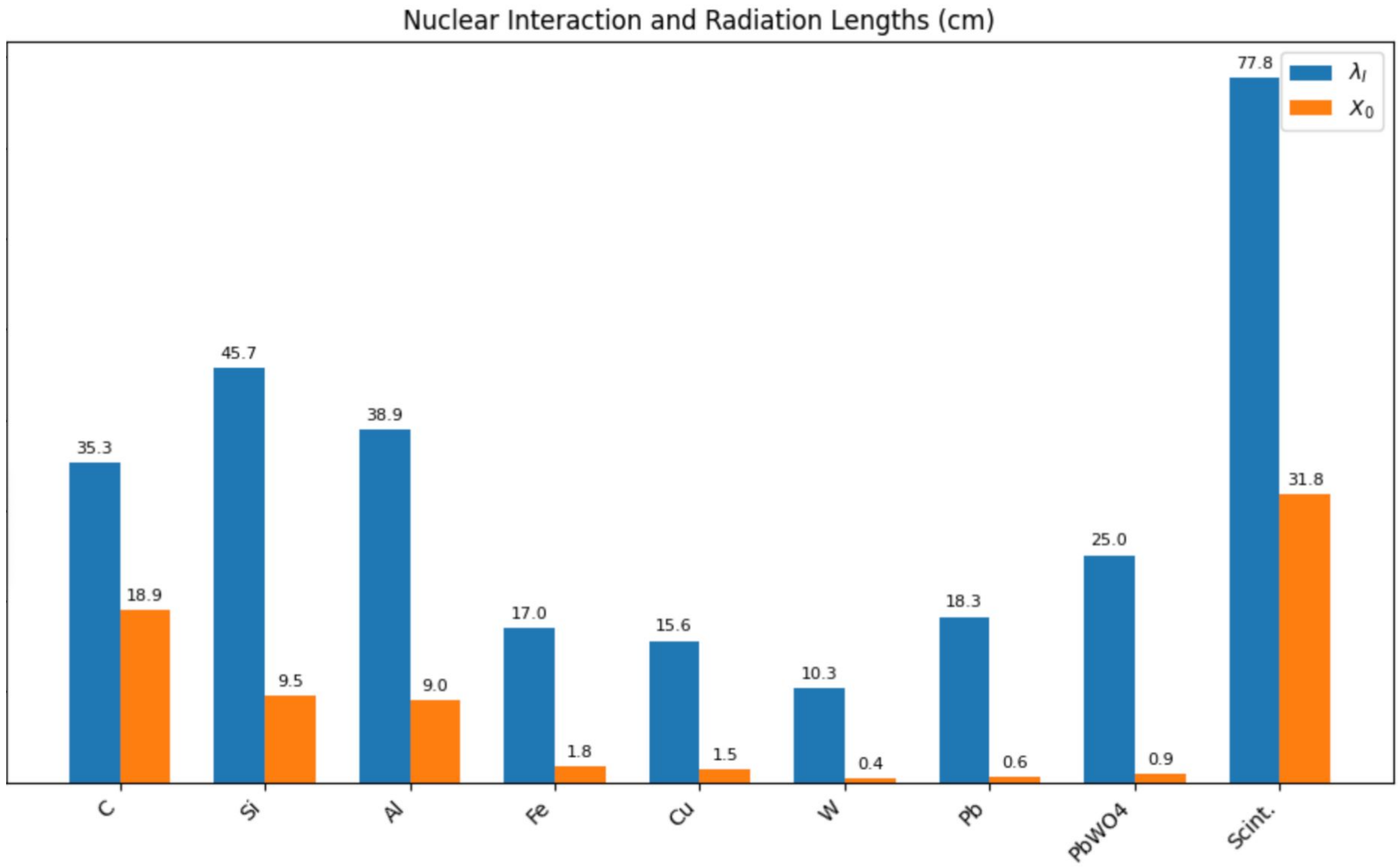
$$R_M \simeq X_0\,\frac{E_s}{E_c}, \qquad E_s \approx 21.2\ \mathrm{MeV}.$$



Hadronic Showers

- Rule of thumb: aim 7 to 10 interaction lengths for high containment at 10–100 GeV, several interaction lengths for transversal confinement $\lambda_I\,(\mathrm{cm}) = \frac{A}{\rho\, N_A\, \sigma_{\mathrm{inel}}}$.
- Nuclear binding → extra stochastic term in resolution compared with ECALs.





Nuclear Interaction and Radiation Lengths (cm)
λ_I
X_0
35.3
18.9
45.7
9.5
38.9
9.0
17.0
1.8
15.6
1.5
10.3
0.4
18.3
0.6
25.0
0.9
77.8
31.8
C
Si
Al
Fe
Cu
W
Pb
PbWO4
Scint.

Boundaries

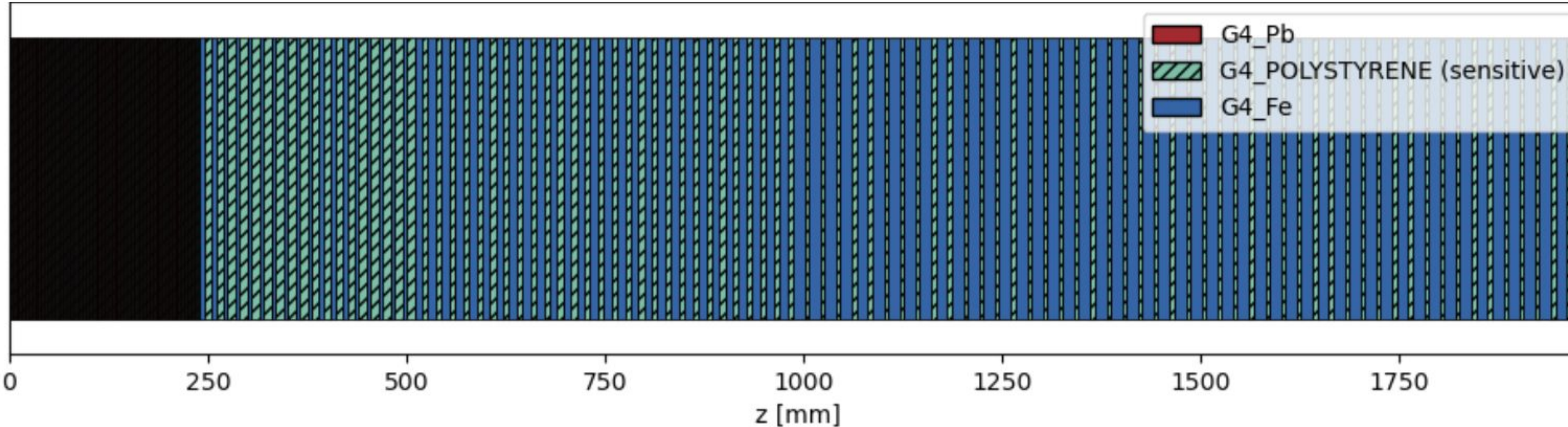
- Budget: 50k CHF
- Compact = 200 cm longitudinal
- 50cm x 50cm transversal construction

```python
def build_ecal_pbwo4_26X0_then_graded_fe_scint_hcal_200cm_v1(
    # --- ECAL ---
    total_X0: float = 26.0,                  # target depth in X0
    X0_cm: float = 0.89,                     # PbWO4 X0 in cm (≈ 0.89 cm)
    ecal_slices: int = 30,                   # slice ECAL to make nice layer histos
    ecal_material: str = "G4_PbWO4",
    # --- HCAL sections (front → mid → back), ~200 cm total length ---
    front_pairs: int = 16, fe_front: float = 0.8, sc_front: float = 0.8,   # 25.6 cm
    mid_pairs:   int = 32, fe_mid:   float = 1.2, sc_mid:   float = 0.6,   # 57.6 cm
    back_pairs:  int = 47, fe_back:  float = 1.5, sc_back:  float = 0.5,   # 94.0 cm
    absorber: str = "G4_Fe",
    active:   str = "G4_POLYSTYRENE",
    sensitive_active_only: bool = True,
    # optional cost tracker (your Trackers.CostTracker instance or None)
    cost_tracker=None,
) -> Tuple[GeometryDescriptor, Dict]:

    gd = GeometryDescriptor()
```

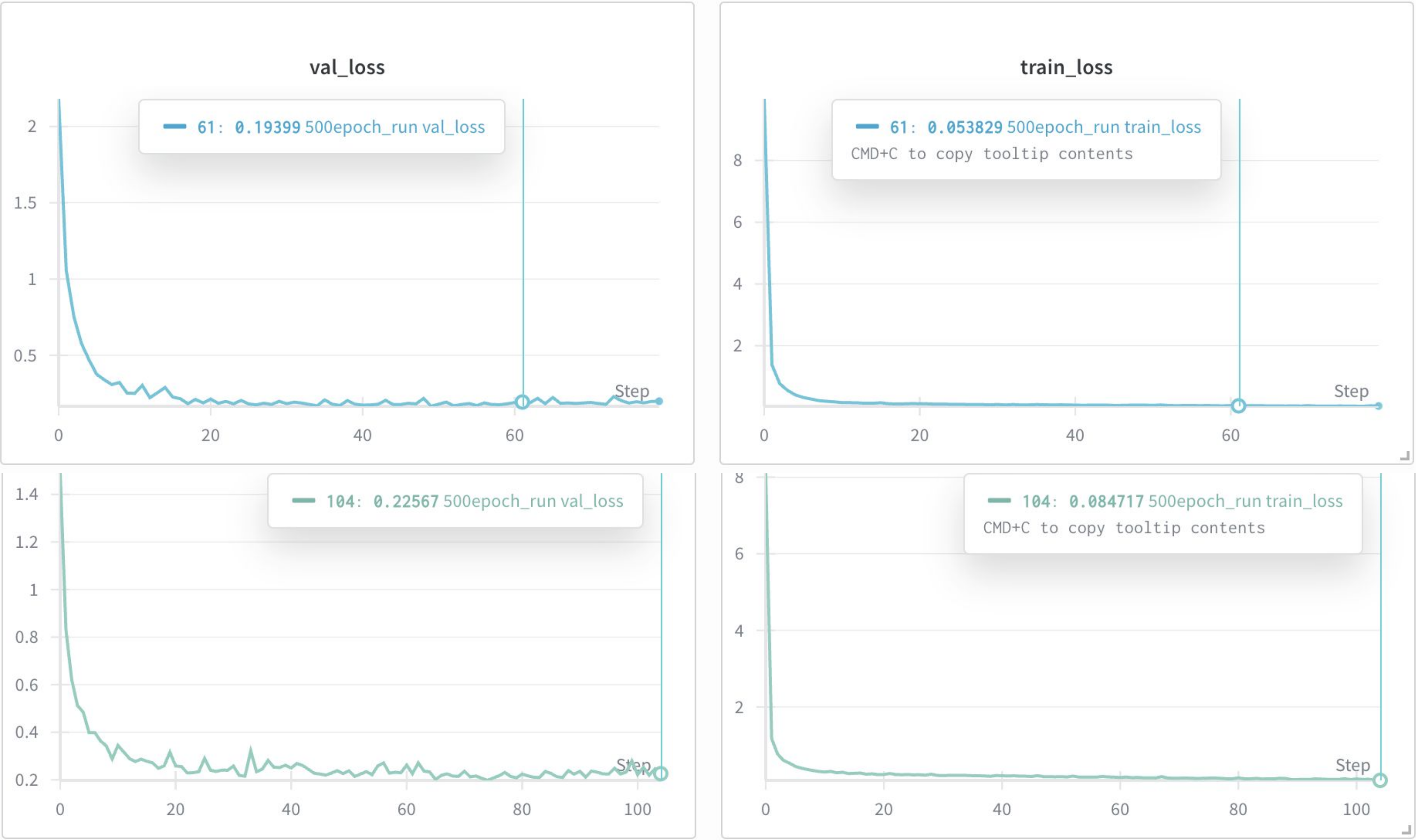
```python
    ecal_len_cm  = float(total_X0) * float(X0_cm)     # ~ 26 * 0.89 = 23.14 cm
    ecal_slice   = ecal_len_cm / int(ecal_slices)

    for _ in range(int(ecal_slices)):
        gd.addLayer(ecal_slice, ecal_material, True)
        if cost_tracker is not None:
            cost_tracker.add(ecal_material, ecal_slice)


    def add_section(n_pairs: int, t_abs_cm: float, t_act_cm: float) -> None:
add a breakpoint range(int(n_pairs)):
            gd.addLayer(float(t_abs_cm), absorber, False)
            gd.addLayer(float(t_act_cm), active,   bool(sensitive_active_only))
            if cost_tracker is not None:
                cost_tracker.add(absorber, t_abs_cm)
                cost_tracker.add(active,   t_act_cm)

    add_section(front_pairs, fe_front, sc_front)
    add_section(mid_pairs,   fe_mid,   sc_mid)
    add_section(back_pairs,  fe_back,  sc_back)
```

```python
def build_pb_scint_ecal_then_graded_fe_scint_hcal_200cm_v2(
    # --- ECAL: Pb/Scint sampling (fine sampling, ~24 cm total, ~21–22 X0)
    ecal_pairs: int = 60,          # 60 × (0.20 Pb + 0.20 Sc) → ~24.0 cm ECAL
    pb_per_pair_cm: float = 0.20,  # lead per period (cm)
    sc_per_pair_cm: float = 0.20,  # scint per period (cm)
    ecal_absorber: str = "G4_Pb",
    ecal_active:   str = "G4_POLYSTYRENE",
    # --- HCAL: graded Fe/Scint (front transition → mid → back)
    front_pairs: int = 18, fe_front: float = 0.5, sc_front: float = 1.0,   # 27.0 cm
    mid_pairs:   int = 28, fe_mid:   float = 1.0, sc_mid:   float = 0.7,   # 47.6 cm
    back_pairs:  int = 50, fe_back:  float = 1.5, sc_back:  float = 0.5,   # 100.0 cm
    absorber: str = "G4_Fe",
    active:   str = "G4_POLYSTYRENE",
    sensitive_active_only: bool = True,
```

```
ECAL length: 24.00 cm, HCAL length: 174.60 cm
Total length: 198.60 cm
Total cost:   2300 CHF
```

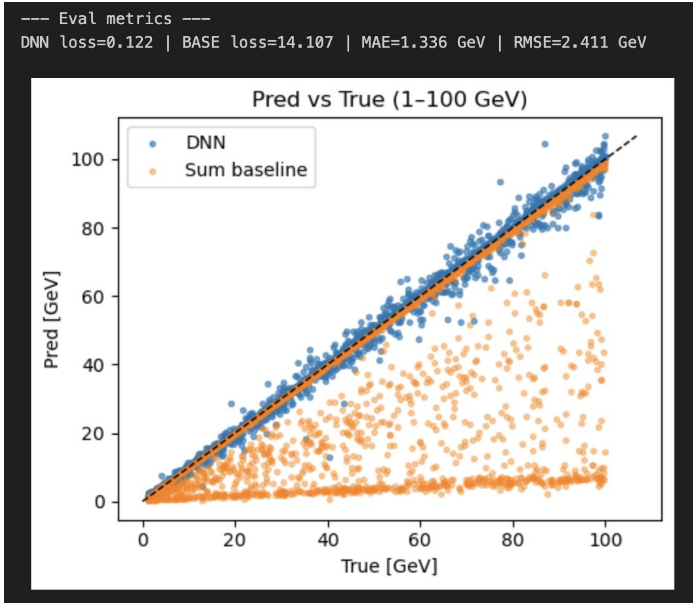


```python
# ---------- 3) Model / Optim / Loss ----------
input_dim = X_train.shape[1]

model = nn.Sequential(
    nn.Linear(input_dim, 256), nn.GELU(),
    nn.Linear(256, 128), nn.GELU(),
    nn.Linear(128, 1), nn.Softplus()   # keeps predictions positive
).to(device)

opt = torch.optim.AdamW(model.parameters(), lr=1e-3, weight_decay=1e-5)

def relE_loss(pred, y):
    # ((E_true - E_pred)^2)/E_true
    y_safe = torch.clamp(y, min=1e-6)
    return torch.mean(((y - pred)**2) / y_safe)
```

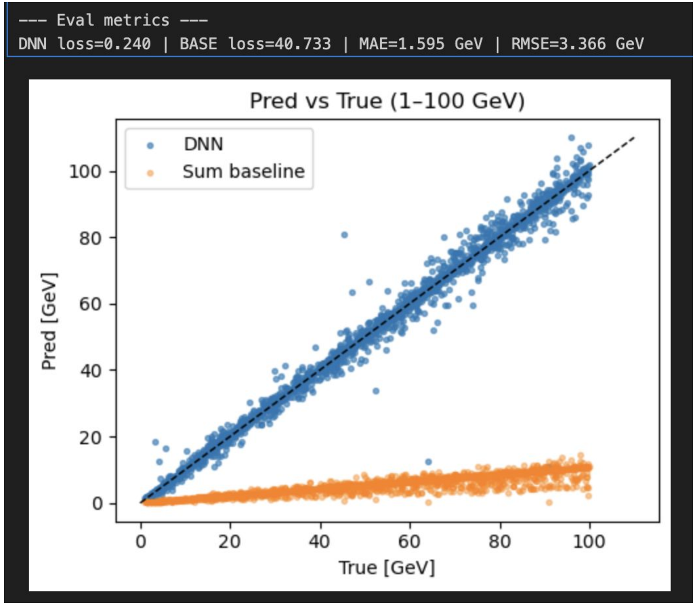

val_loss
61: 0.19399 500epoch_run val_loss
Step
train_loss
61: 0.053829 500epoch_run train_loss
CMD+C to copy tooltip contents
Step
104: 0.22567 500epoch_run val_loss
Step
104: 0.084717 500epoch_run train_loss
CMD+C to copy tooltip contents
Step

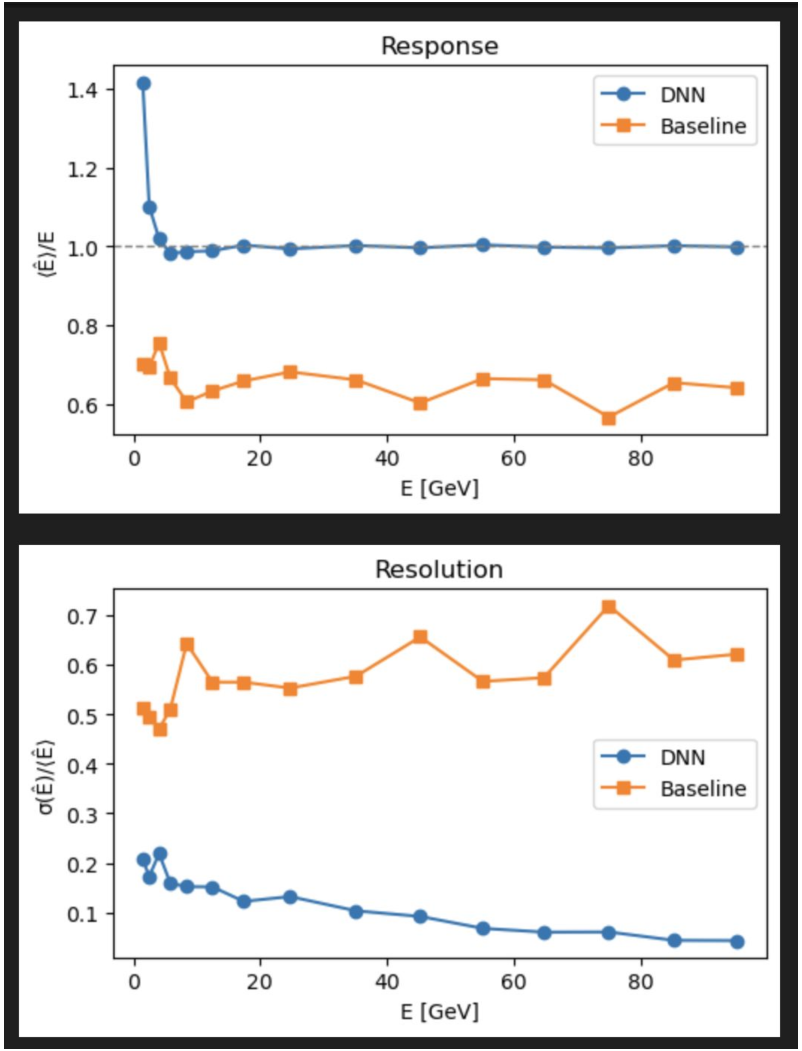
ECAL Comparison:

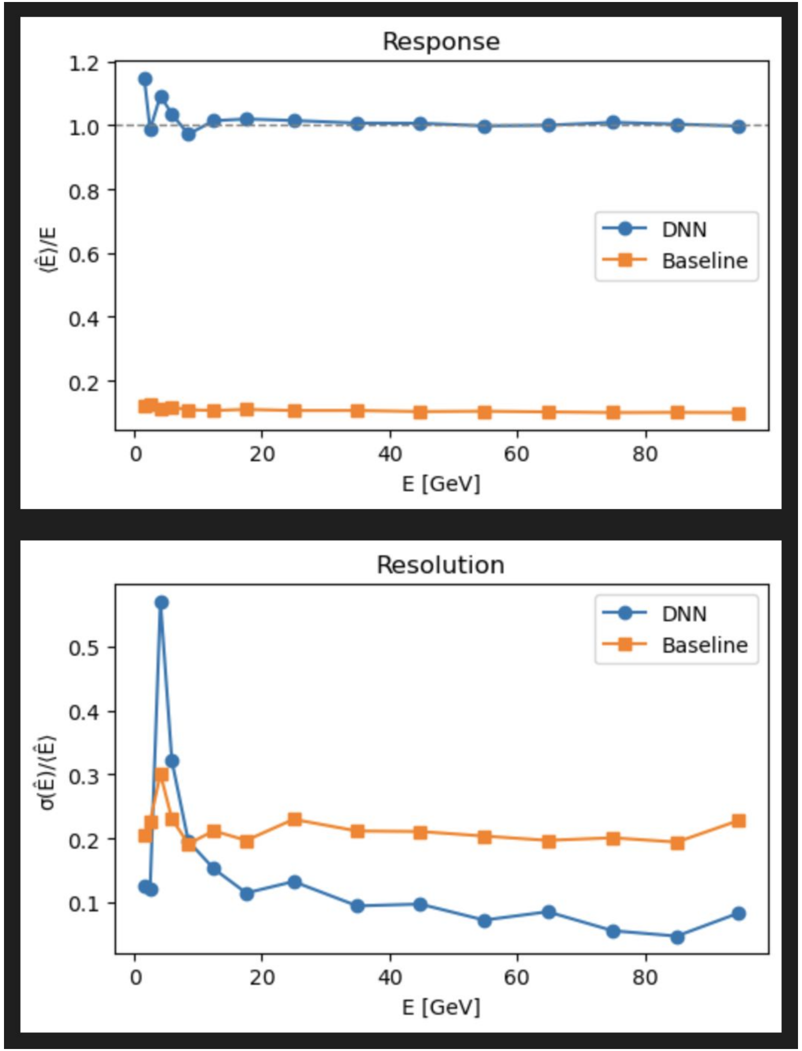
Homogeneous: PbWO4

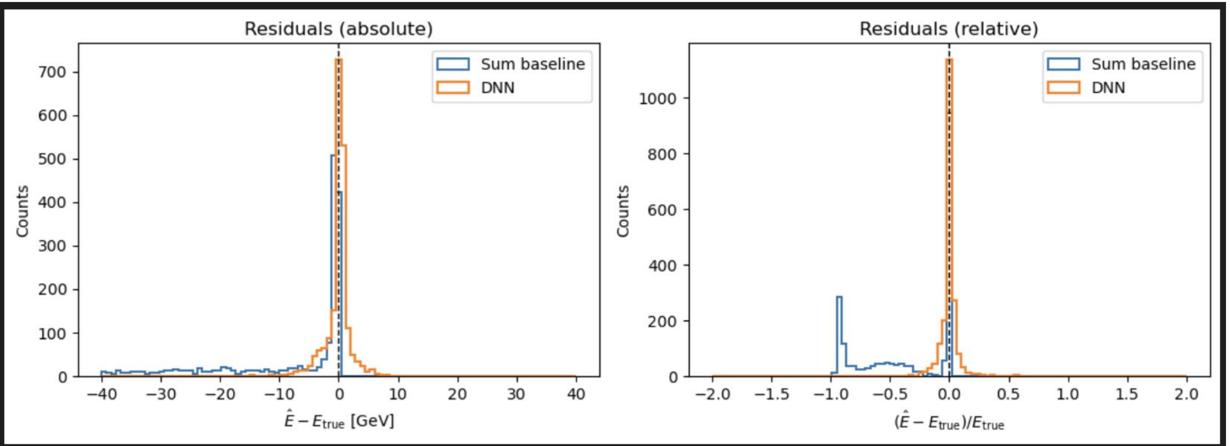


Sampling: Pb + Scint.

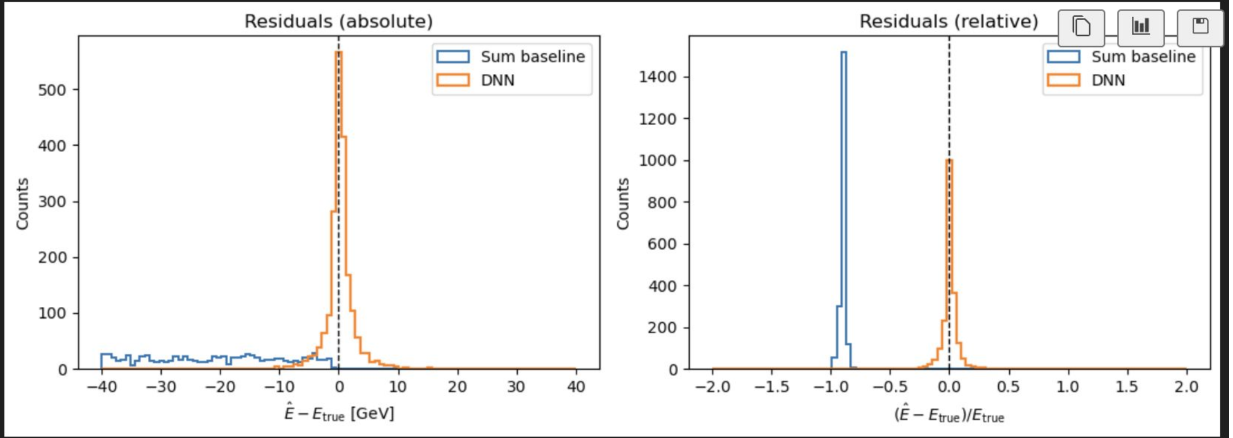



Response
⟨Ê⟩/E
DNN
Baseline
E [GeV]
Resolution
σ(Ê)/⟨Ê⟩
DNN
Baseline
E [GeV]


Response
⟨Ê⟩/E
DNN
Baseline
E [GeV]
Resolution
σ(Ê)/⟨Ê⟩
DNN
Baseline
E [GeV]



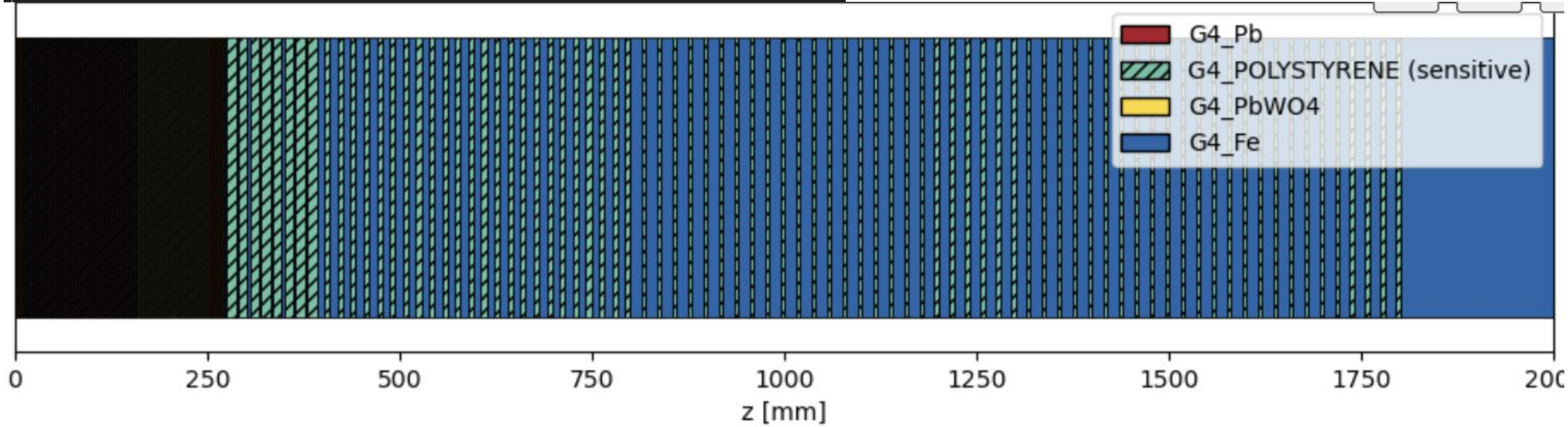

```
DNN       | μ=-0.080 GeV, σ=2.410 GeV | μ_rel=+0.29%, σ_rel=8.33%
Baseline  | μ=-18.074 GeV, σ=25.131 GeV | μ_rel=-35.41%, σ_rel=38.08%
```




```
DNN       | μ=+0.206 GeV, σ=3.361 GeV | μ_rel=+0.96%, σ_rel=14.13%
Baseline  | μ=-45.351 GeV, σ=25.387 GeV | μ_rel=-89.56%, σ_rel=2.12%
```

```python
def build_triple_ecal_then_fe_scint_hcal_2m_v4_2(
    # --- ECAL triple period: [Pb, Scint(sens), PbWO4]
    ecal_pairs: int = 60,
    pb_cm: float = 0.15,
    sc_cm: float = 0.20,
    pbwo4_cm: float = 0.10,
    add_scint_endcap: bool = True,          # adds one extra scint layer aft
    ecal_pb: str = "G4_Pb",
    ecal_scint: str = "G4_POLYSTYRENE",
    ecal_pbwo4: str = "G4_PbWO4",

    # --- HCAL: transition → nominal
    trans_pairs: int = 8,   fe_trans: float = 0.30, sc_trans: float = 1.20,
    mid_pairs:   int = 24,  fe_mid:   float = 1.00, sc_mid:   float = 0.70,
    back_pairs:  int = 50,  fe_back:  float = 1.50, sc_back:  float = 0.50,
    hcal_absorber: str = "G4_Fe",
    hcal_active:   str = "G4_POLYSTYRENE",
    sensitive_active_only: bool = True,
```

```
Total length: 200.00 cm
Total cost:   47192 CHF
```

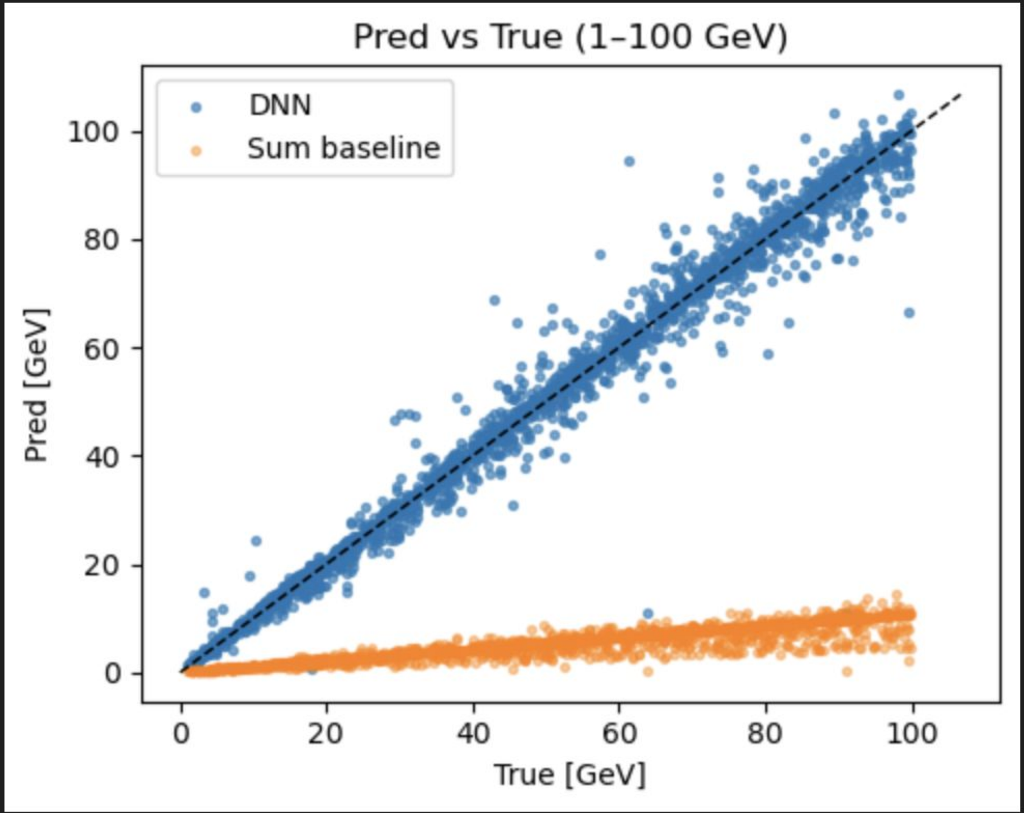


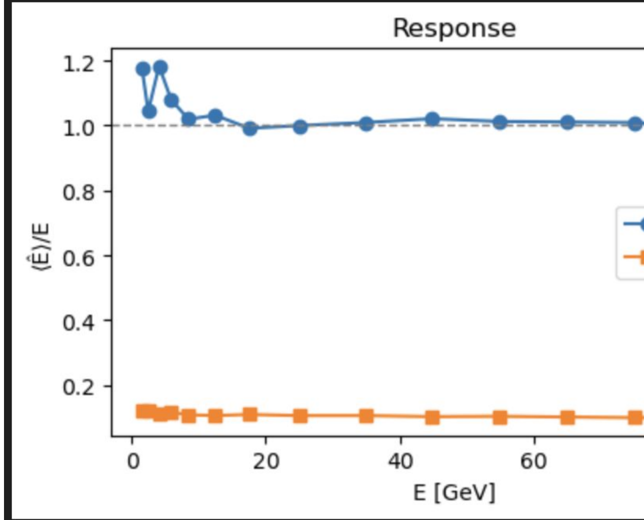
```python
model = nn.Sequential(
    nn.Linear(input_dim, 256), nn.GELU(),
    nn.Dropout(0.20),              # was 0.10
    nn.Linear(256, 128), nn.GELU(),
    nn.Dropout(0.20),
    nn.Linear(128, 64),  nn.GELU(),
    nn.Linear(64, 1),
    nn.Softplus()
)

opt = torch.optim.AdamW(model.parameters(), lr=INIT_LR, weight_decay=
```

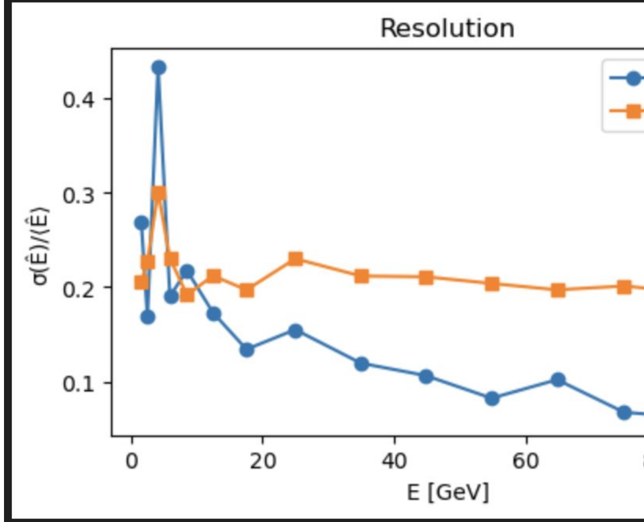
```python
VAL_SPLIT = 0.1
BATCH = 256
EPOCHS     = 500
WEIGHT_DECAY = 2e-4  # was 1e-5
PATIENCE   = 25     # early stopping
```

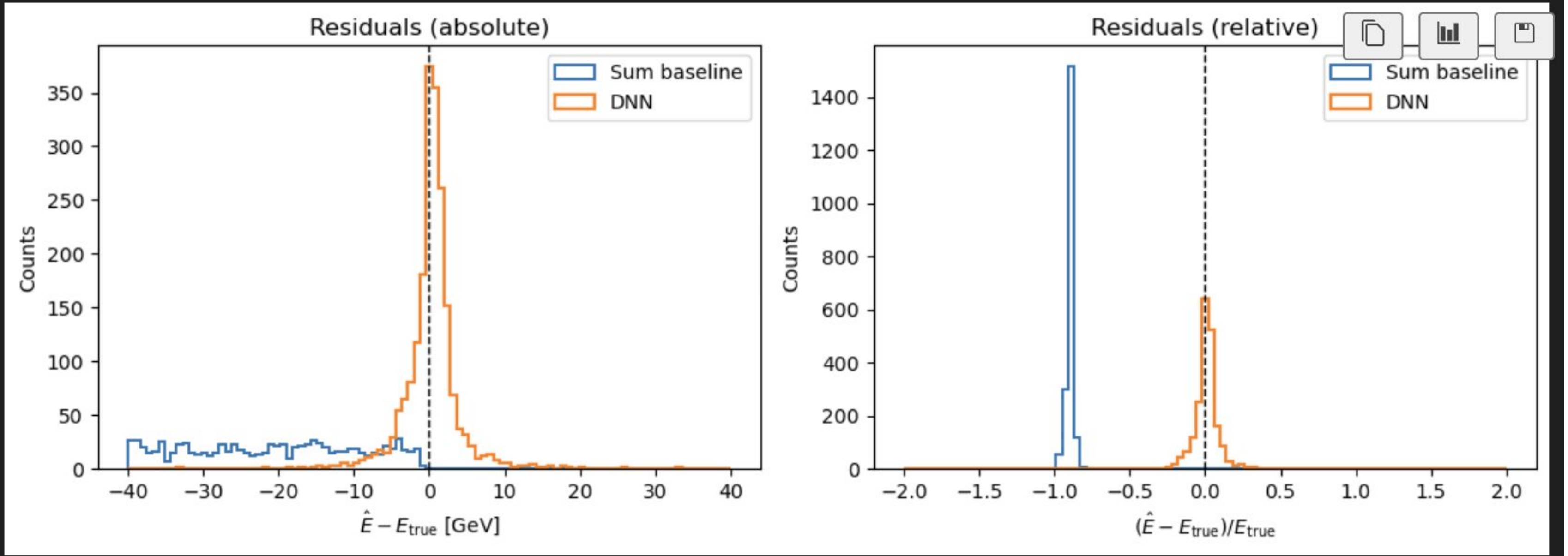
```python
for epoch in range(1, EPOCHS + 1):
    # train
    model.train()
    tr_sum, n_seen = 0.0, 0
    for xb, yb in tr_loader:
        opt.zero_grad(set_to_none=True)
        pred = model(xb)
        loss = relE_loss(pred, yb)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), MAX_GRAD_NORM)
        opt.step()
        tr_sum += loss.item() * xb.size(0)
        n_seen += xb.size(0)
    train_loss = tr_sum / max(1, n_seen)

    # val
    model.eval()
    va_sum, v_seen = 0.0, 0
    with torch.no_grad():
        for xb, yb in va_loader:
            va_sum += relE_loss(model(xb), yb).item() * xb.size(0)
            v_seen += xb.size(0)
    val_loss = va_sum / max(1, v_seen)

    scheduler.step()

    # early stopping
    if val_loss < best_val - 1e-6:
        best_val = val_loss
        best_state = {k: v.detach().cpu().clone() for k, v in model.state_dict().items()}
        since_best = 0
    else:
        since_best += 1
```

```
--- Eval metrics (matches training loss) ---
DNN loss=0.354 | BASE loss=40.733 | MAE=2.332 GeV | RMSE=4.269 GeV
```










```
DNN        | μ=+0.244 GeV, σ=4.263 GeV | μ_rel=+1.49%, σ_rel=13.68%
Baseline   | μ=-45.351 GeV, σ=25.387 GeV | μ_rel=-89.56%, σ_rel=2.12%
```