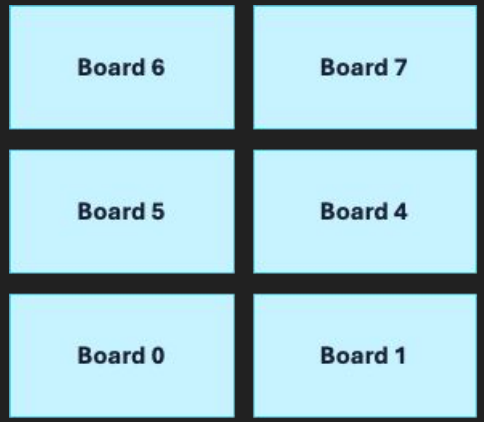
Particle Classification and Energy Regression on FoCal-H Development Data for ALICE, CERN 2026

10. juni 2026

Amanda Dyrby, Barbara Schmidt, Oliver McRoy, Rasmus Reimer

Design and Implementation

How the FoCal-H Prototype 3 data is organised and mapped into detector positions



The analysis is a translation problem

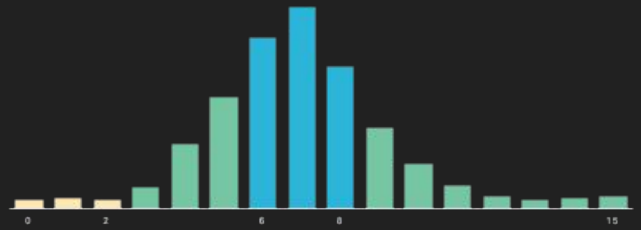
from (hardware) readout numbers → physical detector pixels

What the raw data contains

The readout is stored as event-based ADC samples. It only becomes a detector image after mapping.

one particle/shower event → VLDB 0 or 1 → ASIC 0 or 1 on each VLDB → Half 0 or 1 on each ASIC → channel physical readout channel → sample from 0–15

Each ADC number is identified by where it came from in the hardware and by when it was sampled in the waveform.



16 samples per event/channel

Board numbering

A "board" in the analysis means one unique VLDB–ASIC–Half combination.

2 VLDBs VLDB 0 and VLDB 1 → 4 ASICs 2 per VLDB → 8 half-boards Half 0 and Half 1 → board index 0 to 7

$$\text{Board\_\{index\}} = 4 \cdot \text{VLDB} + 2 \cdot \text{ASIC} + \text{Half}$$

Channel positions inside one board

Inside a board, the channels are arranged as an 8-column × 4-row map.

3	7	11	15	18	22	28	32
1	5	9	13	20	24	30	34
0	4	10	16	19	23	27	31
2	6	12	14	21	25	29	33

rows: 0–3 · columns: 0–7

bottom-left is (0,0)

move left → right then go up one row

The 180° rotation correction

Some boards are physically flipped, so their row/column coordinates must be reversed before lookup.

Board 6 rotated	**Board 7**
Board 5 rotated	**Board 4**
Board 0 rotated	**Board 1**

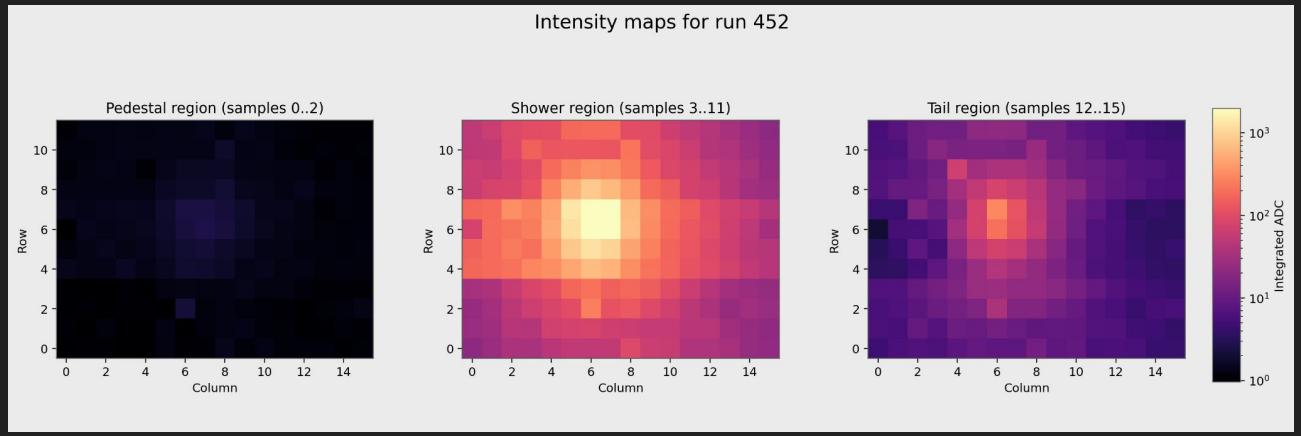
Rotated boards: 0, 5, and 6

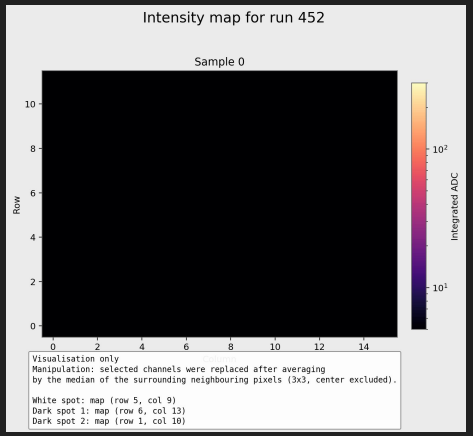
normal lookup (row, col) → **rotated lookup (row', col')**

$$row' = 3 - row$$
$$col' = 7 - col$$

From raw readout to a detector map

raw ADC value from TTree → VLDB, ASIC, Half → boardindex → board layout → board position → channel map → pixel position → detector map



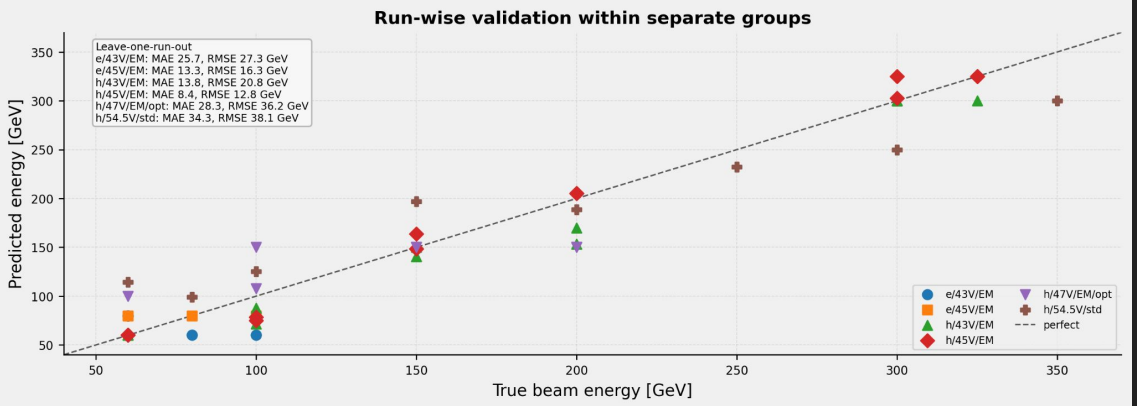


An example of implementing an intensity map for run 452

How well does the ML Model predict beam energy?

```
from sklearn.ensemble import HistGradientBoostingRegressor
```

the algorithm is a **gradient boosted decision tree regression model**



Calibration groups used by the corrected method

Group	N	Runs	E [GeV]	Status
e, 43 V, EventMatch	7	605 606 607 610 611 614 615	60 80 100	trained separately
e, 45 V, EventMatch	6	603 604 608 609 612 613	60 80 100	trained separately
h, 43 V, EventMatch	11	577 578 583 584 585 586 591 592 593 594 687	60 100 150 200 300 325	trained separately
h, 45 V, EventMatch	12	579 580 581 582 587 588 589 590 595 596 685 686	60 100 150 200 300 325	trained separately
h, 47 V, EventMatch, optical	7	653 655 656 657 658 659 662	60 100 150 200	trained separately
h, 54.5 V, standard	8	445 446 448 450 451 452 458 459	60 80 100 150 200 250 300 350	trained separately

Comparison between model architecture and ability to discover underlying relations in raw data

From boosted trees to a Convolutional Network on raw FoCal-H data

Normalized ADC directly into three architectures of rising complexity, no hand engineered features. electron vs hadron classification, then hadron energy regression.

- XGBoost
- MLP
- CNN

The data structure

`val0` branch

`(2, 64, 38)`

(3, 64, 38) for classification (bias encoded in first coordinate)

`2` detector halves (h0, h1)
`64` chips per half
`38` readout channels per chip

Classification = 7296 values / event. XGB & MLP flatten it to a vector. The CNN keeps it as a image.
Regression = 4864 values / event. XGB & MLP flatten it to a vector the CNN keeps it as a 2-channel image.

Classification: electron vs hadron

Honest test = four entire runs held out of training. The numbers below are on those unseen runs.

MLP best but all perform exceptionally well

Input size 7296 vector before the first layer.
Complexity 1.8m parameters
trains on 88k events, test on 58k from unseen runs.

Classification: Electron vs Hadron, 60 GeV

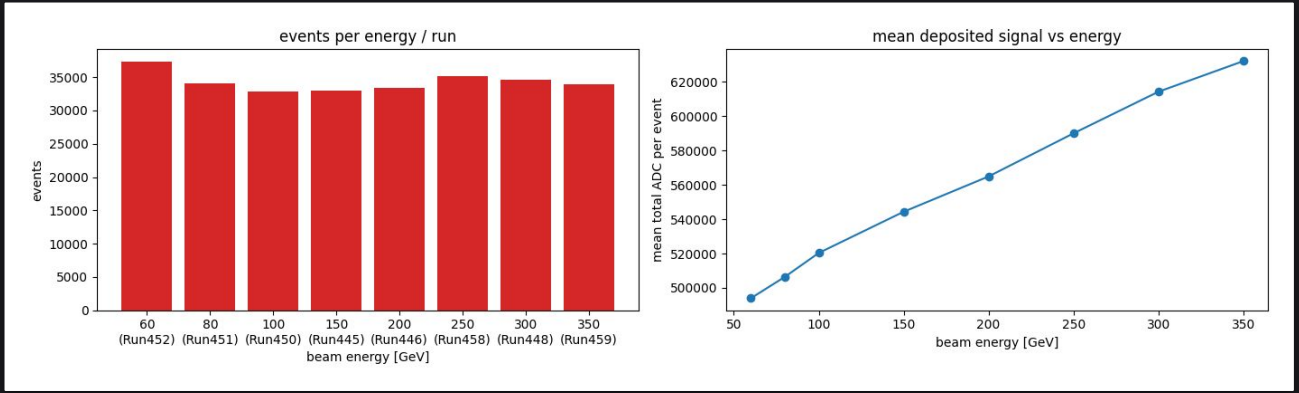
hadron 4 runs totaling approx 80k events
electron 5 runs totaling approx 66k events

per run hold out test: 4 runs held out. two hadron runs: approx 38k events
Two electron runs: aprrox 20k events

7296 values / event. XGB & MLP flatten it to a vector (no spatial notion); the CNN keeps it as a 2-channel image.

All three separate e/h easily (Acc ≥ 97.7%, AUC ≥ 0.997).
The CNN's spatial prior doesn't help here and lands it last.

Datasets & signal energy relationship



Regression set (energy scan)

Regression: hadron energy scan

8 runs: 274K events total

Roughly equal size, about 35k each ± 1k

60GeV 80GeV 100GeV 150GeV
200GeV 250GeV 300GeV 350GeV

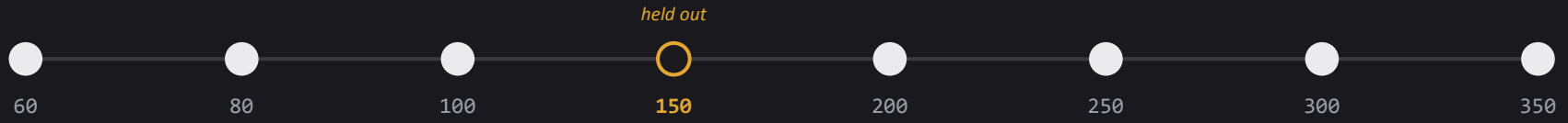
one run per energy

Mean deposited signal rises monotonically with beam energy. This near linear ADC energy map is the physics a model should learn, and what lets a good regressor interpolate to an energy it never trained on, rather than fall back on per run unique signatures.

The Discrete Trap

One run per energy ⇒ *"predict the energy" can collapse into "recognize which run this is." We need two different tests.*

8 energies, each from a single run:



TEST A · standard split

Random 70/15/15 over events. Every run appears in train AND test.

Not guaranteed to separate physics from memorization a model that just fingerprints each run scores perfectly.

TEST B · hold out a whole energy

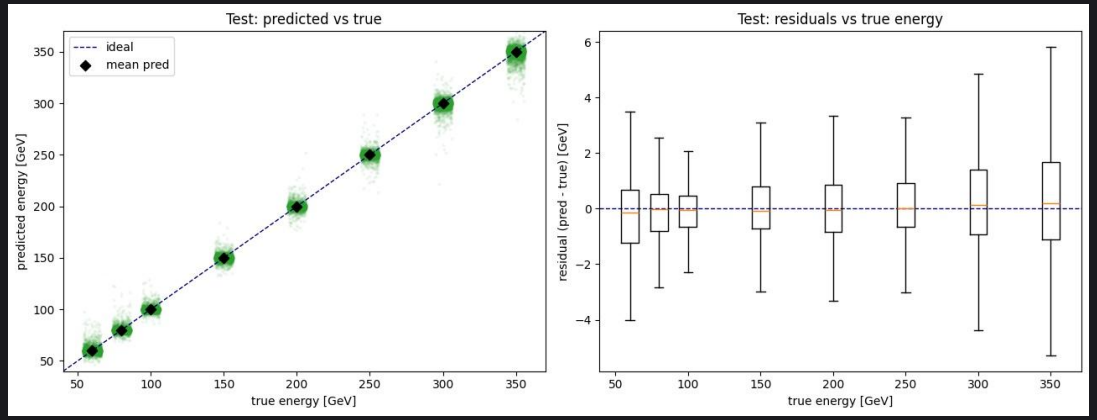
Remove all of 150 GeV from training. The model must predict an energy it has never seen.

The physics test. Land near 150 = it interpolates physics or atleast structure in the data related to energy. Drift to the seen mean (∼190 GeV) = it memorized runs.

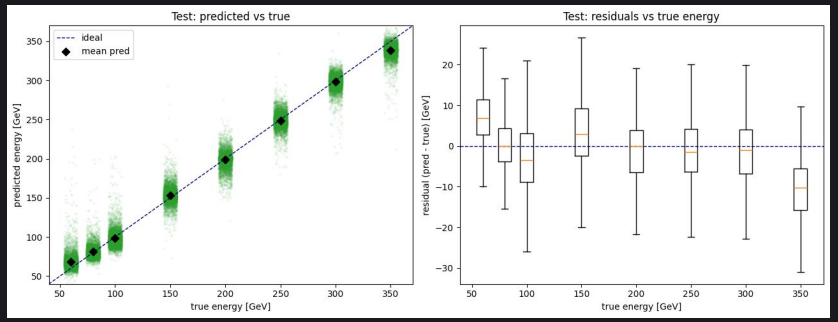
Test A: in distribution: everyone looks great

XGBoost, standard split: predictions sit on the diagonal, residuals are tight at every energy.

XGBoost test split (150 GeV present in training)



MLP test split (150 GeV present in training)

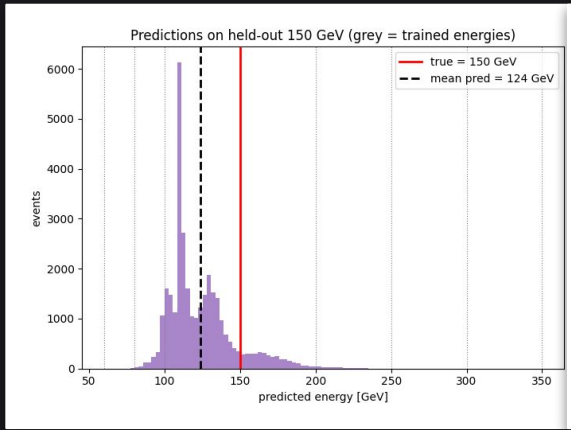


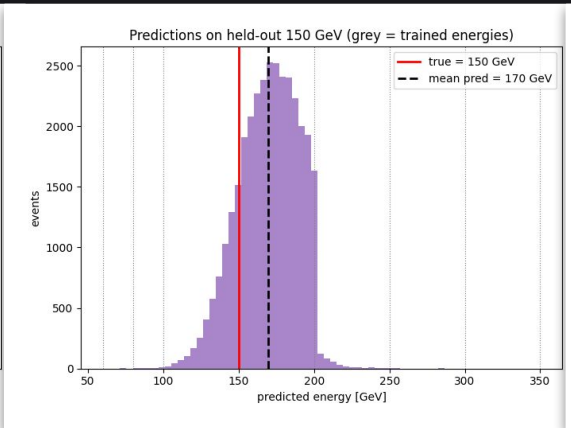
Test-split MAE

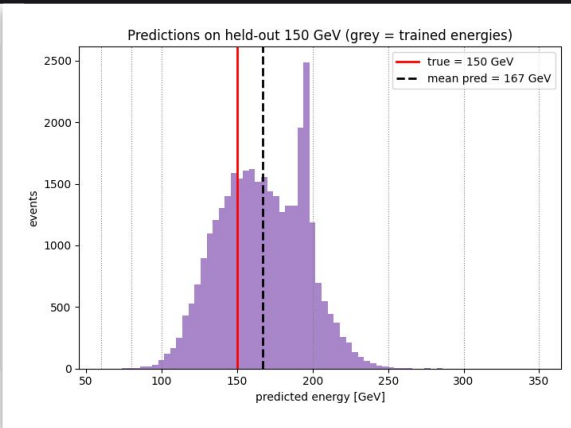
XGBoost	1.77 GeV
CNN	7.15 GeV
MLP	8.40 GeV

XGBoost:
1.77 GeV
$R^2 \approx 0.999$.
too good to be true? (Probably)

Holdout 150 GeV interpolation test







XGBoost mean 123.8 · MAE 31.2

MLP mean 169.6 · MAE 23.7

CNN mean 167.1 · MAE 26.8

XGB undershoots to 124GeV (trees can't extrapolate well).

MLP: clean, near Gaussian at 170GeV slightly skewed right side around the seen mean (190GeV).

CNN: centred near 167GeV but a spike at about 190GeV the training set mean leaking through (memorization).

Conclusion - Raw detector data in ML

01

The flattened NN wins both tasks

Best held out classification (99.1%) and best energy interpolation (23.7 GeV), but not perfect.

02

In-distribution score ≠ physics

XGBoost's 1.77 GeV on the standard split was probably run fingerprinting. Doesn’t interpolate well.

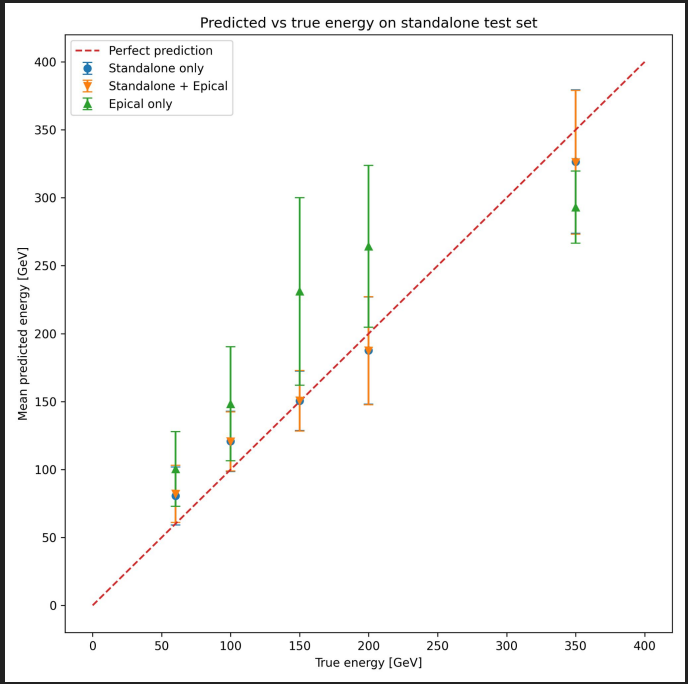
03

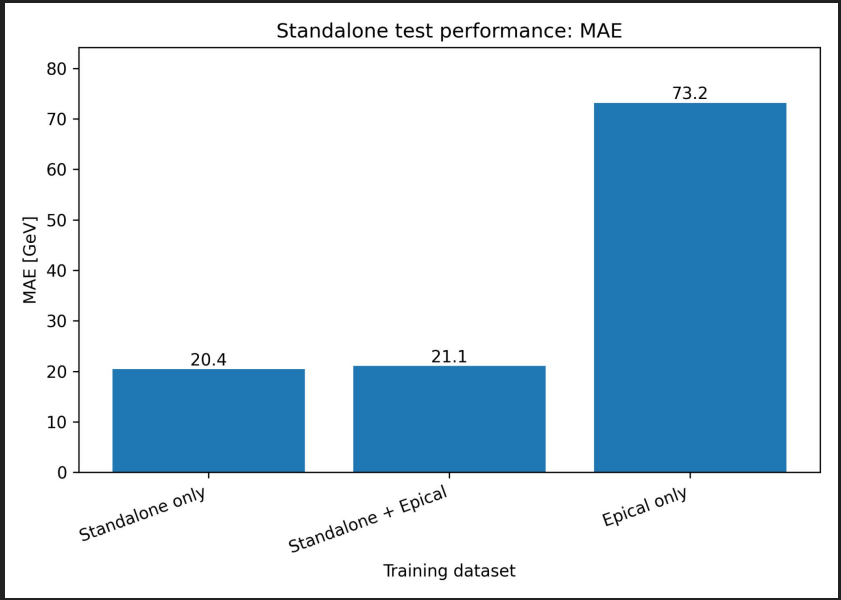
CNN shows Potential, but seems to fallback to mean when uncertain and has a larger spread

the CNN spikes at the training mean (190 GeV). Had this peak not been present, it would have been substantially closer to the true mean, as well as assume a gaussian shape, however spread would still be large compared to the MLP.
Improve with feature engineering (the proper geometry layout of the detector) should improve performance.

Particle Energy identification - HistGradientBoosting

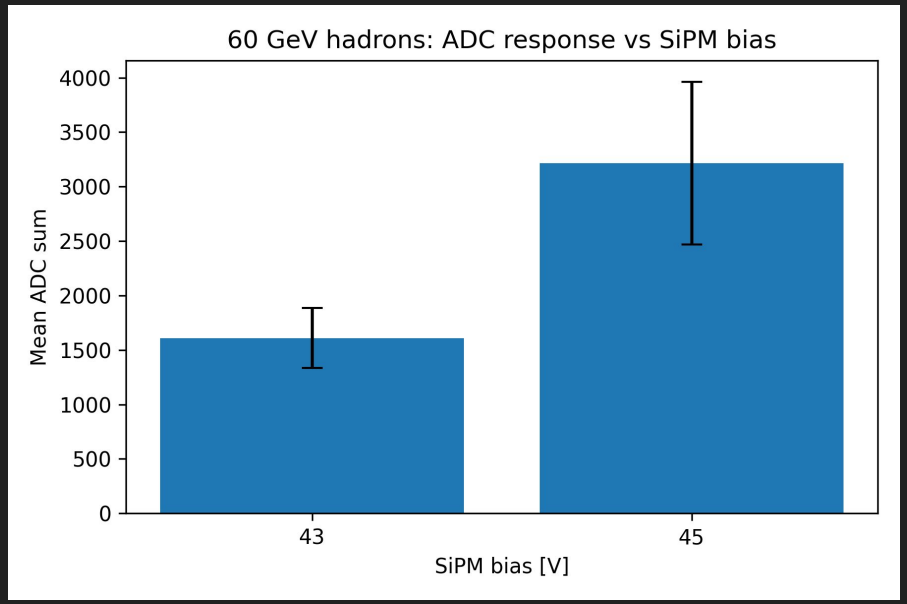
Metadata study and feature engineering -With or Without EpiCal

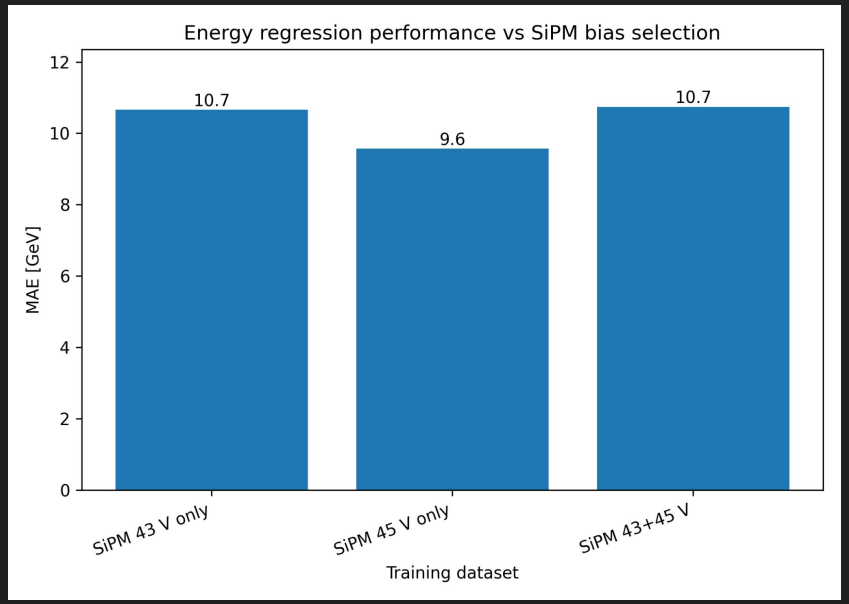




Particle Energy identification - HistGradientBoosting

Metadata study and feature engineering - SiPM

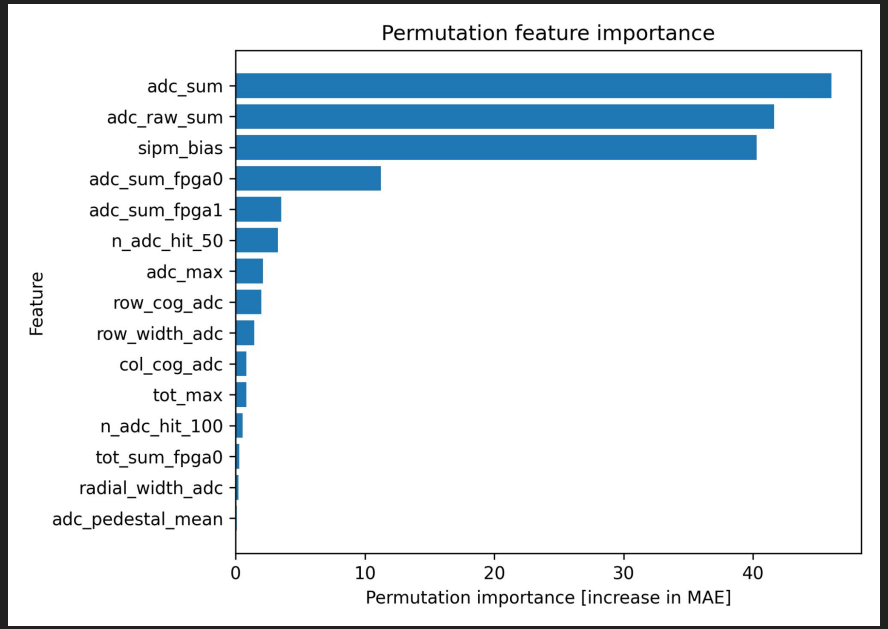




Particle Energy identification

Metadata study and feature engineering - Channel Map

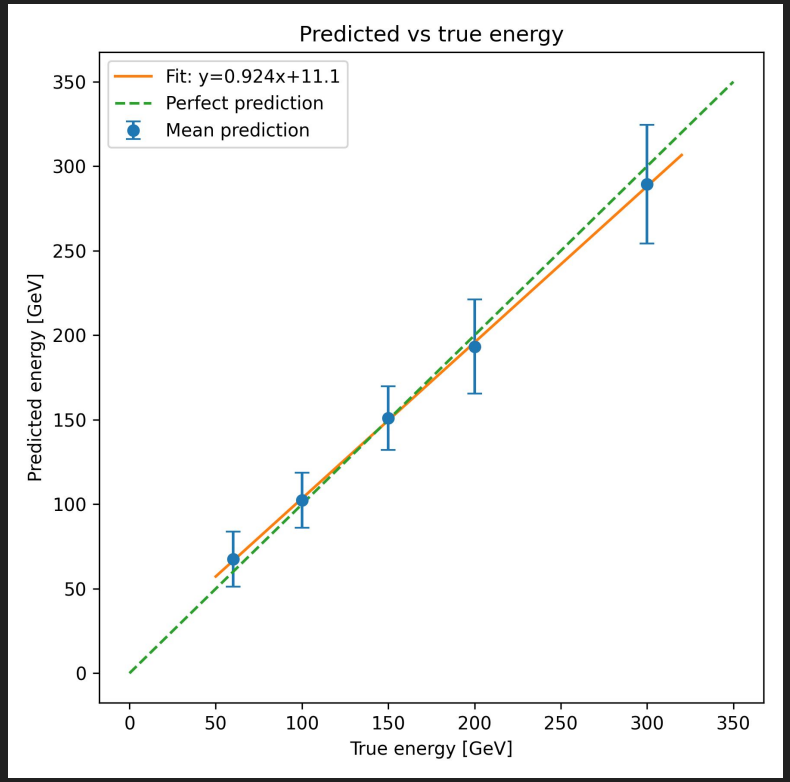
Model Version	MAE[GeV]	RMSE[GeV]
Baseline features	10.89	24.82
+ channel-map geometry	10.74	24.58

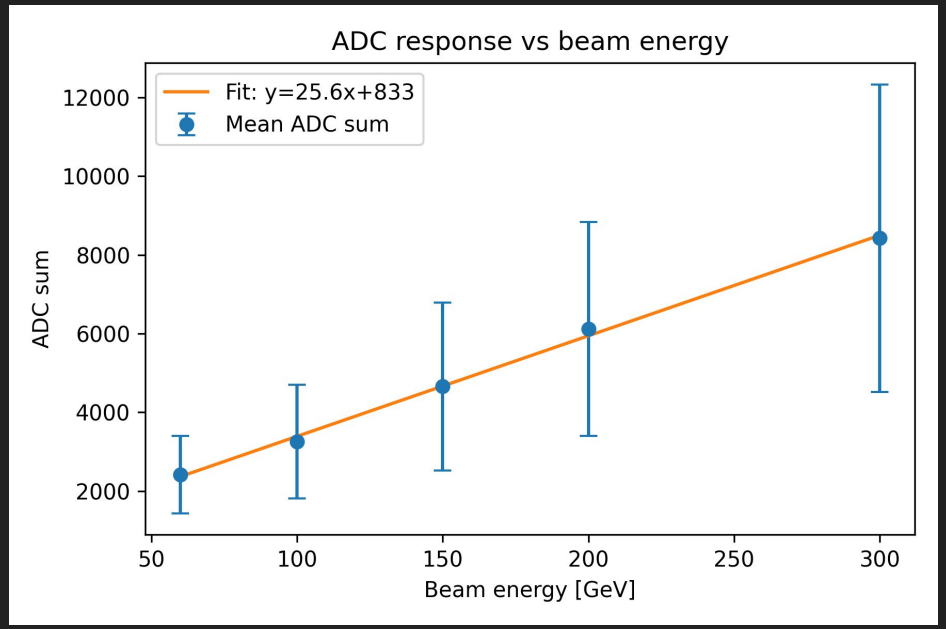


Standalone data with SiPM = 43V and 45V

Particle Energy identification

Metadata study and feature engineering



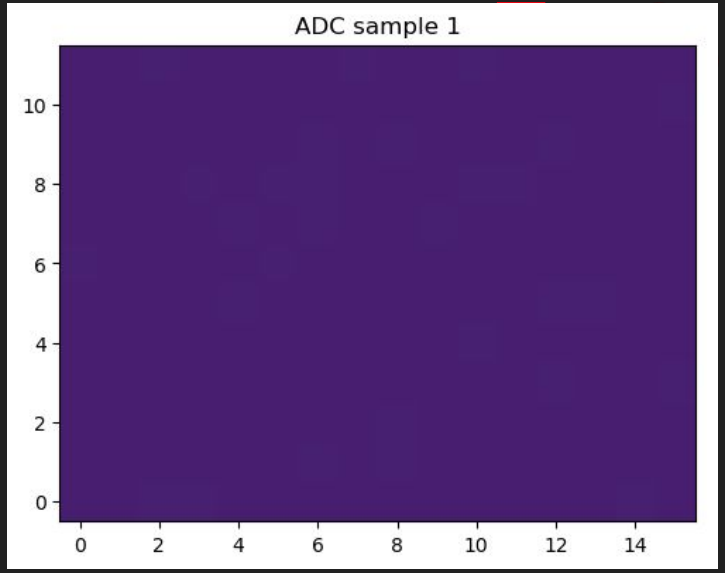


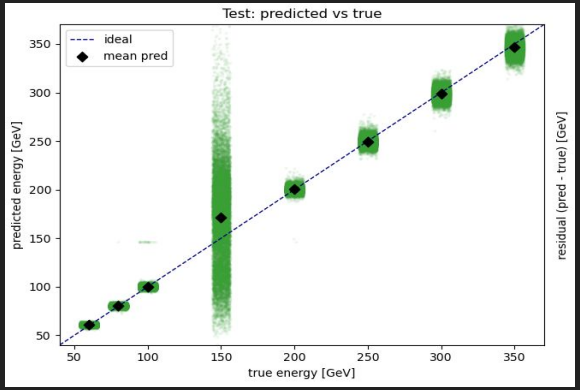
Detector images over time with 3D CNN

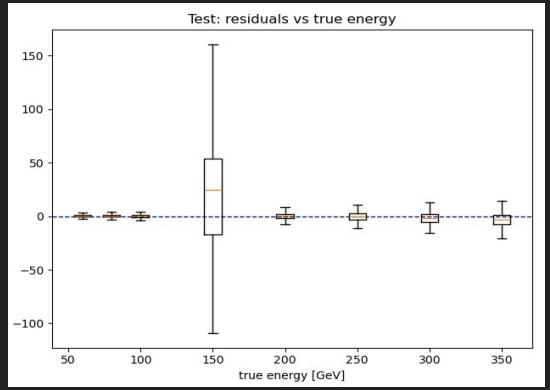
Analysis pre-processing: Mapping, pedestal subtraction, ToT

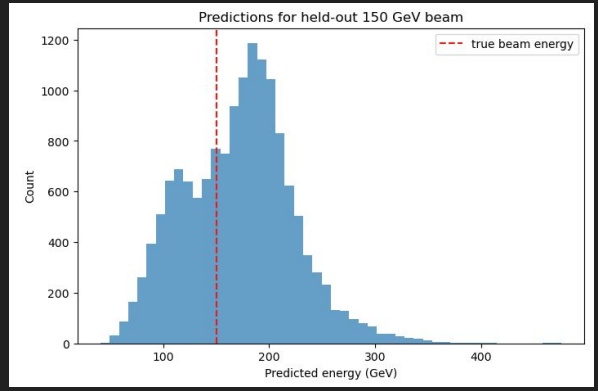
Very good MAE, from 23.7 down to 8.17 (Relative MAE of 5.07%)

Still issues with 150 GeV holdout predictions









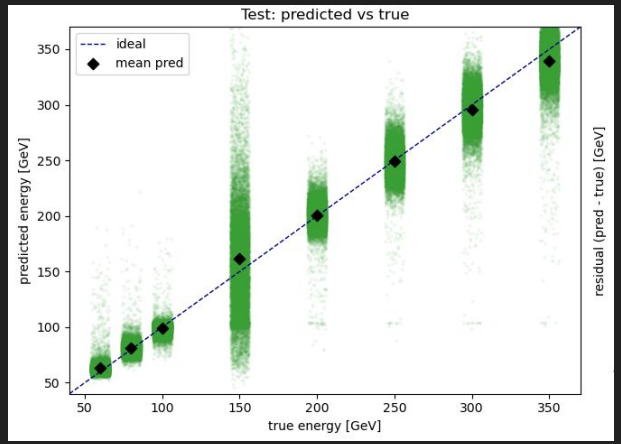
Detector event interpolation

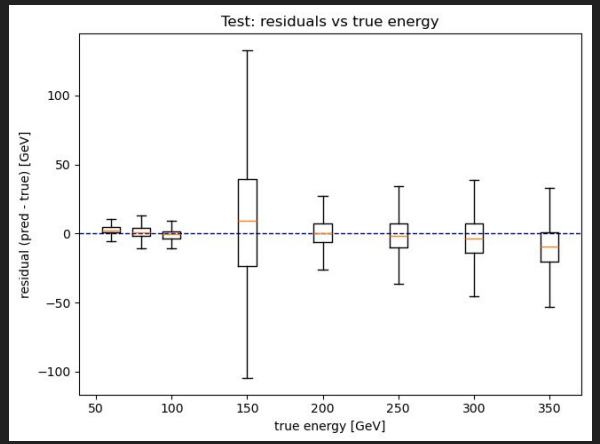
Linear interpolation between events from adjacent energies in training data

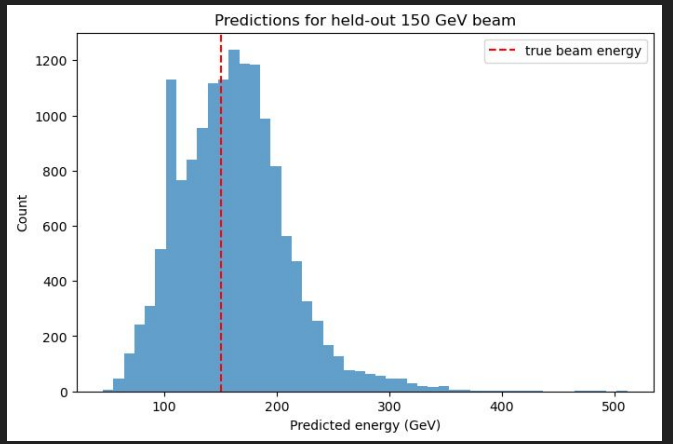
May produce unphysical events, but (hopefully) encourages model to make continuous energy predictions

Worse overall performance, however still good MAE (`MAE: 12.51`)

150 GeV prediction distribution looks *better*, with mean almost correct, but predictions still have large variance







Generating uncertainties for predictions

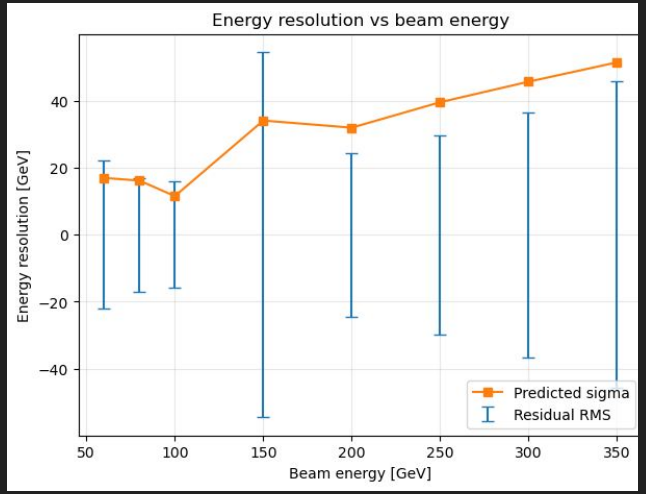
NN Model Ensemble

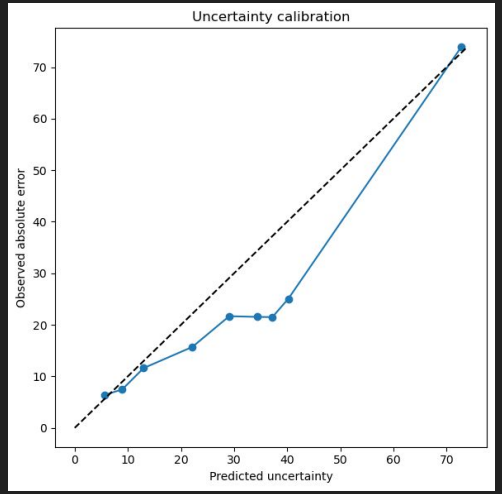
Gaussian log likelihood as a loss function

Performance is not great, worse than other loss functions tested

Predictions skewed towards the mean

$$L = \frac{(y-\mu)^2}{2\sigma^2} + \frac{1}{2}\log\sigma^2$$






Appendix

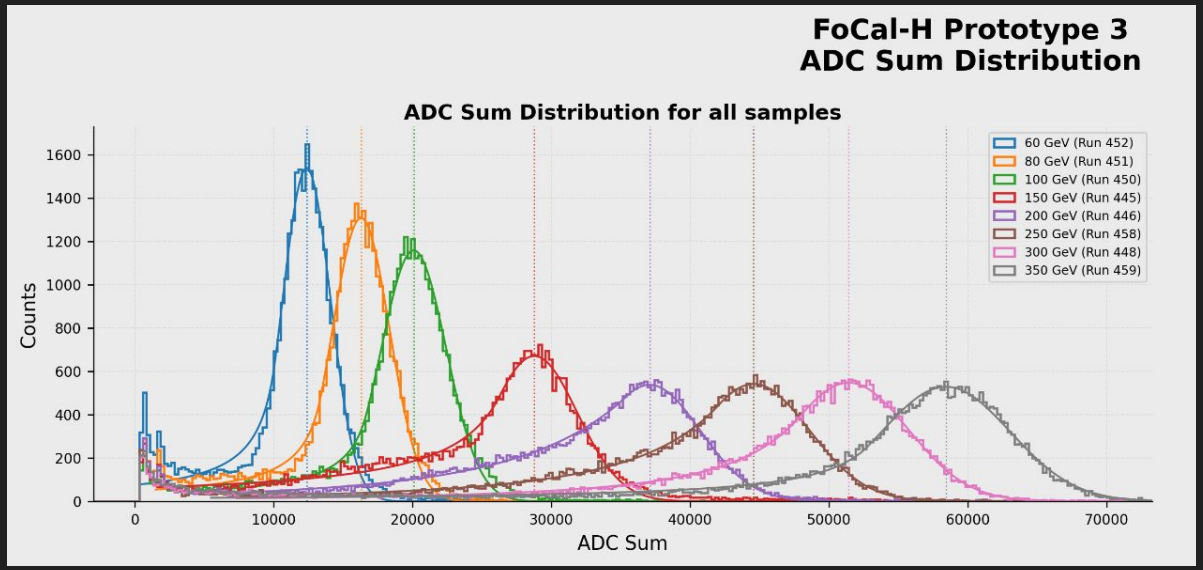
Design and Implementation

Extra informations

Unused channels are removed:

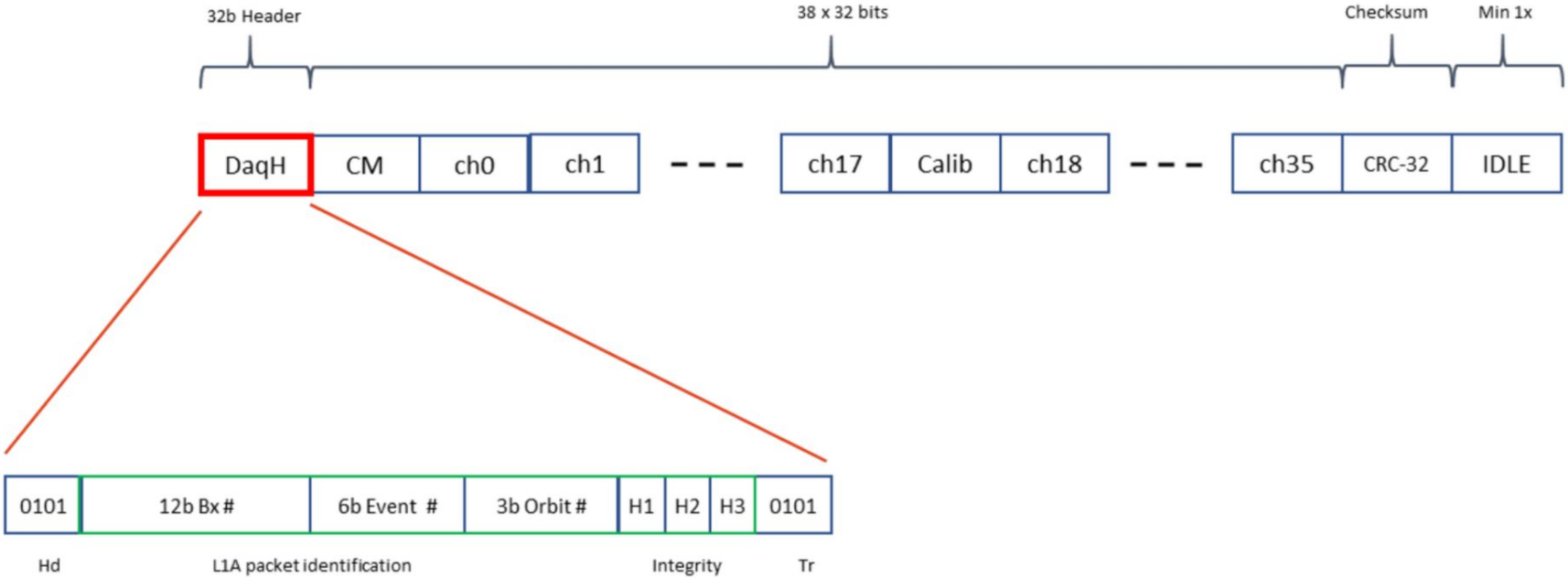
They do not correspond to active detector positions, so they should not be used in maps or sums

ADC Sum Distribution (No ML)



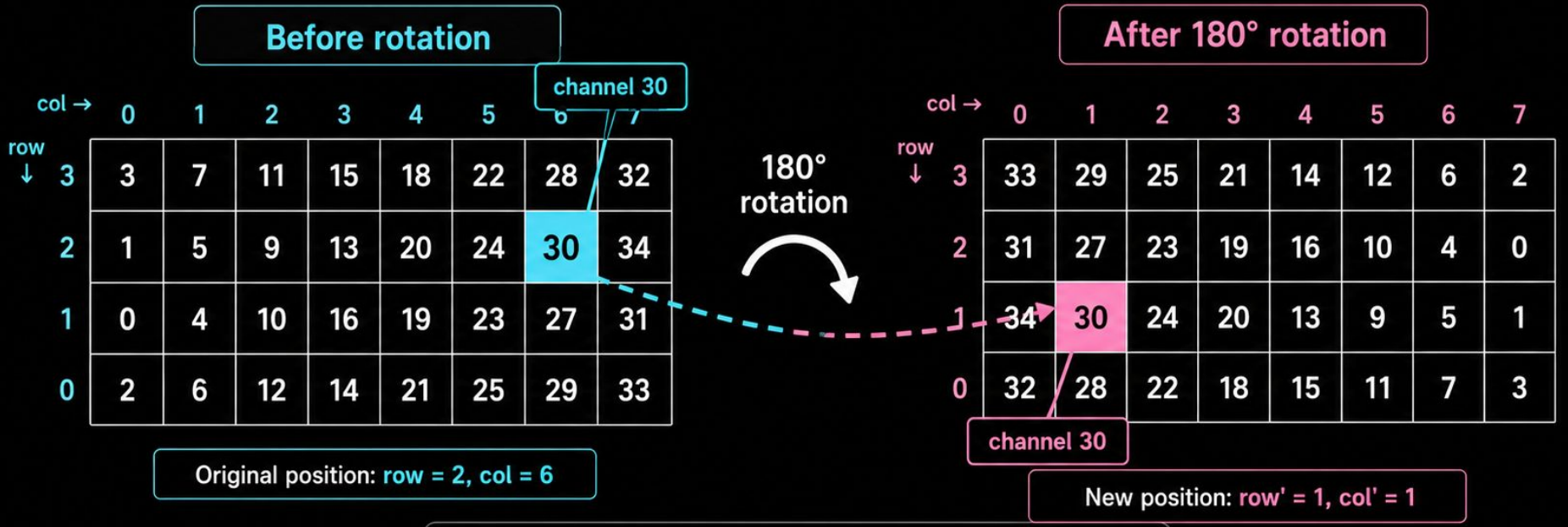
Shows a large overlap between energies, which makes it hard to perfectly differentiate between particle energy.

The hardware readout



How channel 30 moves during 180° rotation

Example of how the channel position changes when the board is rotated.



Rotation rule	For channel 30:
row' = 3 − row	row' = 3 − 2 = 1
col' = 7 − col	col' = 7 − 6 = 1

Appendix

Comparison between model architecture and ability to discover underlying relations in raw data

The Input

Raw data, with no feature engineering done

The structure

`val0` branch

`(2, 64, 38)`

`2` detector halves (h0, h1)
`64` chips per half
`38` readout channels per chip

Classification = 7296 values / event. XGB & MLP flatten it to a vector. The CNN keeps it as a image. Bias is Encoded as -1, 1, as it varies on the classification run, leading to the extra values (3x64x38).
Regression = 4864 values / event. XGB & MLP flatten it to a vector (no spatial notion); the CNN keeps it as a 2-channel image.

SELECTED RUNS

Classification: Electron vs Hadron, 60 GeV

hadron 4 runs totaling approx 80k events
electron 5 runs totaling approx 66k events

per run hold out test: 4 runs held out. two hadron runs: approx 38k events
Two electron runs: aprrox 20k events

Regression: hadron energy scan

8 runs: 274K events total

Roughly equal size, about 35k each ± 1k

`60GeV` `80GeV` `100GeV` `150GeV`
`200GeV` `250GeV` `300GeV` `350GeV`

one run per energy

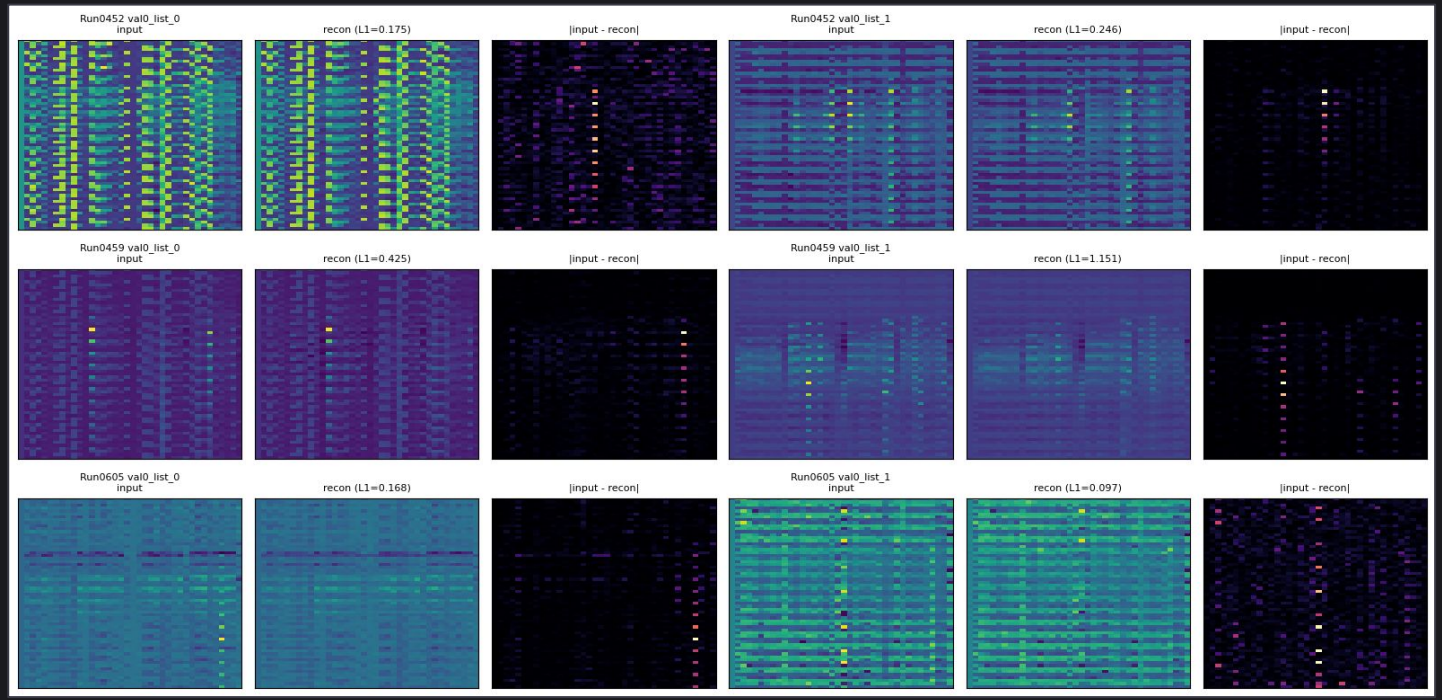
ConvAE - Convolutional Autoencoder is there structure in the raw data?

```
ENCODER
Conv  2 ->16 (3x3, pad 1)              -> ReLU   (64x38 kept)
Conv 16 ->32 (3x3, stripe 2, pad 1)    -> ReLU   (64x38 -> 32x19)
Conv 32 ->16 (3x3, stripe 2, pad 1)    -> ReLU   (32x19 -> 16x10)

LATENT
z = 16 channels @ 16x10   (spatial bottleneck, no flatten) = 2560 values when flattened for umap

DECODER
ConvT 16 ->32 (3x3, stripe 2, pad 1, outpad (1,0)) -> ReLU   (16x10 -> 32x19)
ConvT 32 ->16 (3x3, stripe 2, pad 1, outpad (1,1)) -> ReLU   (32x19 -> 64x38)
Conv  16 ->2  (3x3, pad 1) (reconstruction)
19,122 params

forward -> returns (reconstruction, z)

LOSS
L1  (switched from MSE; keeps sparse hits sharper)

OPTIMIZER
Adam, lr 1e-3, 25 epochs
```

```
h = Hadron run
e = Elektron run

Run0445: 150 GeV h
Run0446: 200 GeV h
Run0448: 300 GeV h
Run0450: 100 GeV h
Run0451: 80 GeV  h
Run0452: 60 GeV  h
Run0458: 250 GeV h
Run0459: 350 GeV h
Run0577: 60 GeV  h
Run0578: 60 GeV  h
Run0579: 60 GeV  h
Run0580: 60 GeV  h
Run0603: 60 GeV  e
Run0604: 60 GeV  e
Run0605: 60 GeV  e
Run0606: 60 GeV  e
Run0607: 60 GeV  e
```

Can the autoencoder reconstruct event reliably?

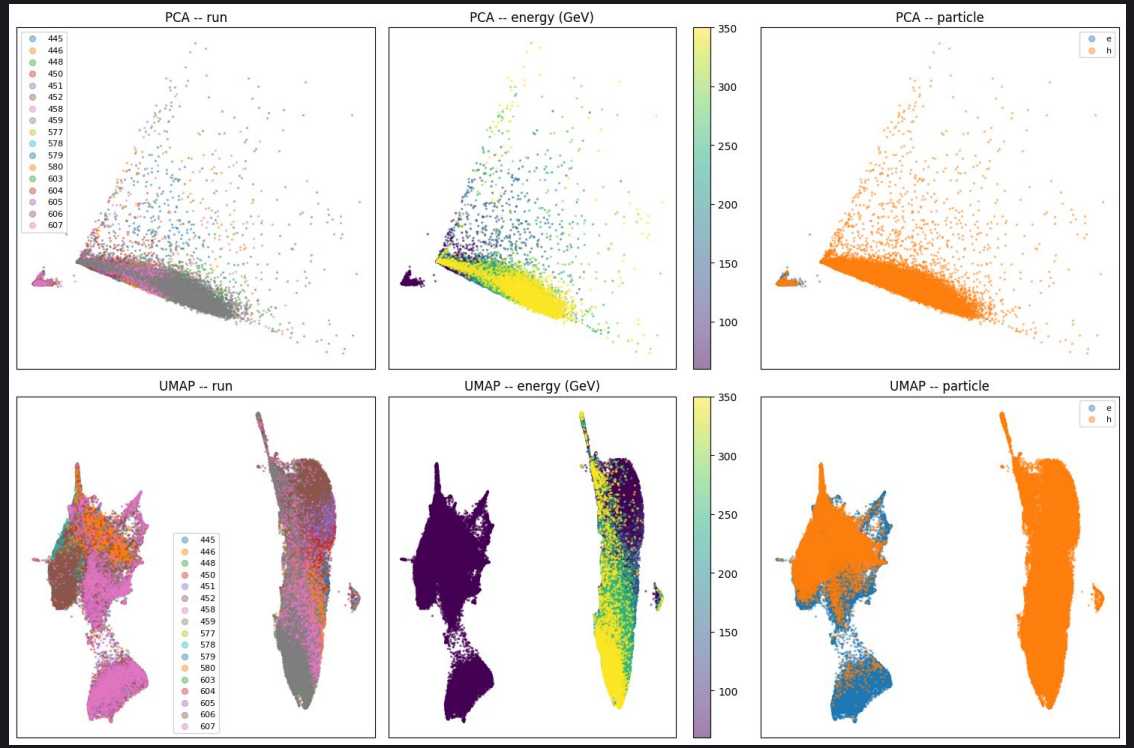
Clearly the raw data is not the correct representation of the geometry of the detector
However it is still able to reconstruct the events fairly well.
Therefore there must be learnable structure in the raw data.



h = Hadron run
e = Elektron run

Run0445: 150 GeV h
Run0446: 200 GeV h
Run0448: 300 GeV h
Run0450: 100 GeV h
Run0451: 80 GeV h
Run0452: 60 GeV h
Run0458: 250 GeV h
Run0459: 350 GeV h
Run0577: 60 GeV h
Run0578: 60 GeV h
Run0579: 60 GeV h
Run0580: 60 GeV h
Run0603: 60 GeV e
Run0604: 60 GeV e
Run0605: 60 GeV e
Run0606: 60 GeV e
Run0607: 60 GeV e

The input is fed to Umap, is there a clear difference between H and E runs?



h = Hadron run
e = Elektron run

Run0445: 150 GeV h
Run0446: 200 GeV h
Run0448: 300 GeV h
Run0450: 100 GeV h
Run0451: 80 GeV h
Run0452: 60 GeV h
Run0458: 250 GeV h
Run0459: 350 GeV h
Run0577: 60 GeV h
Run0578: 60 GeV h
Run0579: 60 GeV h
Run0580: 60 GeV h
Run0603: 60 GeV e
Run0604: 60 GeV e
Run0605: 60 GeV e
Run0606: 60 GeV e
Run0607: 60 GeV e

DATA

Pedestal normalization (per run)

1

Pick quiet events

Per run, take the quietest 50% of events by total ADC: mostly noise / pedestal, little deposited signal.

2

Per-channel stats

From those quiet events compute a mean and std for every one of the 2×38×64 channels.

3

z-score every event

Normalize all events of that run: (x − mean) / (std + ε). Removes perchannel pedestal offsets & gain.

Leakage control (classification holdout)

Pedestal statistics are computed from training pool events only, then applied to every event including the heldout runs using those training derived numbers.

So the model's input distribution carries no information from the hold-out set. Verified: training pool mean ≈ 0, holdout mean ≈ 0.09 (small, as expected).

PART 1 · CLASSIFICATION

Model architectures (e vs h)

Input: 3 × 38 × 64 = 7296 (the third channel encodes bias, 43/45 V).
MLP and CNN trained on GPU m5 (MPS) XGBoost on CPU in parallel

Classification: Electron vs Hadron, 60 GeV

hadron 4 runs totaling approx 80k events
electron 5 runs totaling approx 66k events

per run hold out test: 4 runs held out. two hadron runs: approx 38k events
Two electron runs: aprrox 20k events

XGBoost - CPU train time: 12m 40s best iteration: 1998

```
ARCHITECTURE
binary:logistic
eval_metric  logloss
input  7296-vec (flat, 3x38x64)

HYPERPARAMETERS
n_estimators       2000
learning_rate      0.05
max_depth          6
subsample          0.8
colsample_bytree   0.5
min_child_weight   5
reg_lambda         1.0
tree_method        hist

TRAINING
early_stopping  30 rounds on val logloss
```

MLP - GPU train time: 0m 22s best epoch: 4

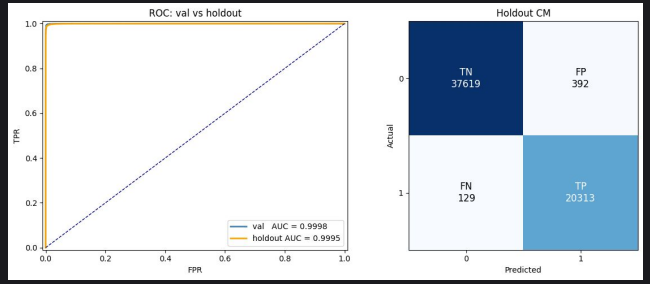
```
ARCHITECTURE
Flatten 3x38x64 -> 7296
Linear 7296->256 -> BatchNorm1d -> ReLU
Dropout 0.3
Linear 256->64 -> BatchNorm1d -> ReLU
Dropout 0.3
Linear 64->1  (single logit)
1.89M params

LOSS
BCEWithLogits

OPTIMIZER + SCHEDULER
AdamW  lr 1e-3  weight_decay 1e-4
ReduceLROnPlateau  factor 0.5
patience 4  min_lr 1e-6
batch 256
early_stopping 12 rounds on val loss
```

CNN - GPU train time: 4m 30s best epoch 30

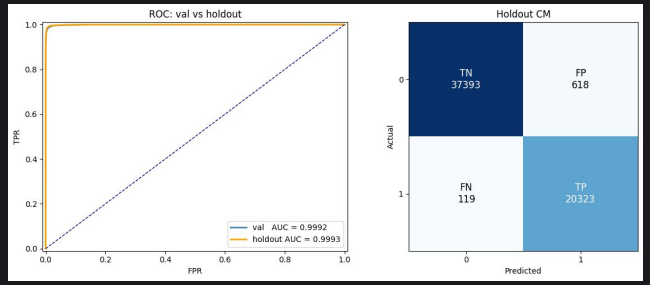
```
ARCHITECTURE
Conv 3->16 (3x3, pad 1) -> BN -> ReLU
Conv 16->32 (3x3, s2)   -> BN -> ReLU
(38x64 -> 19x32)
Conv 32->64 (3x3, s2)   -> BN -> ReLU
(19x32 -> 10x16)
AdaptiveAvgPool -> 1
Dropout 0.3
Linear 64->32 -> ReLU
Dropout 0.3
Linear 32->2  (class logits)
26k params

LOSS
CrossEntropy

OPTIMIZER + SCHEDULER
Same as MLP
```

ROC (val vs hold-out) + holdout confusion matrix

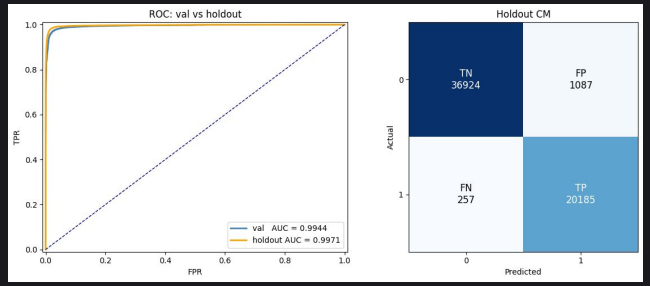
MLP



XGB



CNN



Holdout

MLP **99.11% Accuracy,** AUC .9995, Recall 99.37%

XGB **98.74% Accuracy,** AUC .9993, Recall 99.42 %

CNN **97.70% Accuracy,** AUC .9971, Recall 98.74 %

All models does extremely well, no difference between val loss and the test result on runs which werent present in training.
FP seems to be the hardest for all the models, but really clean result all around.

holdout gap is tiny in every case (≤0.004).

Model architectures (hadron energy regression)

Input: 2 × 38 × 64 = 4864. Target standardized on the train split (mean 185.6, std 100.6 GeV) and inverted back to GeV for every reported metric. MLP and CNN trained on GPU m5 (MPS) XGBoost on CPU in parallel

Regression: hadron energy scan

8 runs: 274K events total

Roughly equal size, about 35k each ± 1k

60GeV 80GeV 100GeV 150GeV
200GeV 250GeV 300GeV 350GeV

one run per energy

XGBoost - CPU train time: 15m 40s best iteration 2000

ARCHITECTURE
reg:pseudohubererror
eval_metric mae

HYPERPARAMETERS

n_estimators	2000
learning_rate	0.05
max_depth	6
subsample	0.8
colsample_bytree	0.5
min_child_weight	5
reg_lambda	1.0
tree_method	hist

(didnt trigger early stopping so still room for more learning, but very diminishing returns after it 1500.)

[1000] val_loss:0.01761
[1500] val_loss:0.01689
[1999] val_loss:0.01647

MLP - GPU train time: 3m 26s best epoch: 56

ARCHITECTURE
Flatten 2x38x64 -> 4864
Linear 4864->256 -> BatchNorm1d -> ReLU
Dropout 0.3
Linear 256->64 -> BatchNorm1d -> ReLU
Dropout 0.3
Linear 64->1
1.26M params

LOSS
Huber (delta 1.0) on z-scored target

OPTIMIZER + SCHEDULER
AdamW
lr 1e-3 weight_decay 1e-4
ReduceLROnPlateau factor 0.5
patience 4 min_lr 1e-6
batch 256

CNN - GPU train time: 26m 0s best epoch: 66

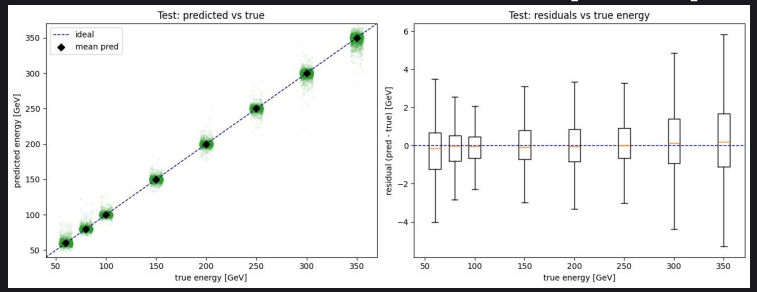
ARCHITECTURE
Conv 2->16 (3x3, pad 1) -> BN -> ReLU
Conv 16->32 (3x3, stride 2) -> BN -> ReLU 38x64 -> 19x32
Conv 32->64 (3x3, stride 2) -> BN -> ReLU 19x32 -> 10x16
AdaptiveAvgPool -> 1
Dropout 0.3
Linear 64->32 -> ReLU
Dropout 0.3
Linear 32->1
26k params

LOSS
Huber (delta 1.0) on z-scored target

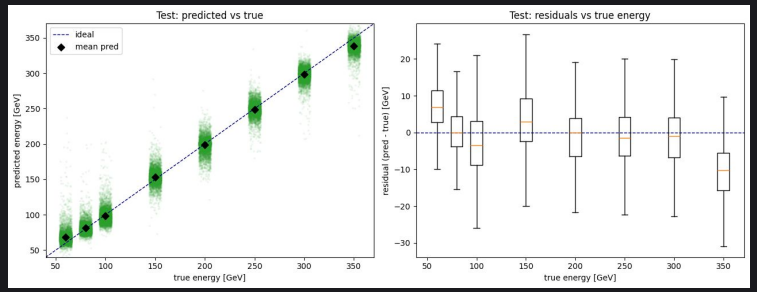
OPTIMIZER + SCHEDULER
Same as MLP

Standard train / val / test split: predicted vs true & residuals

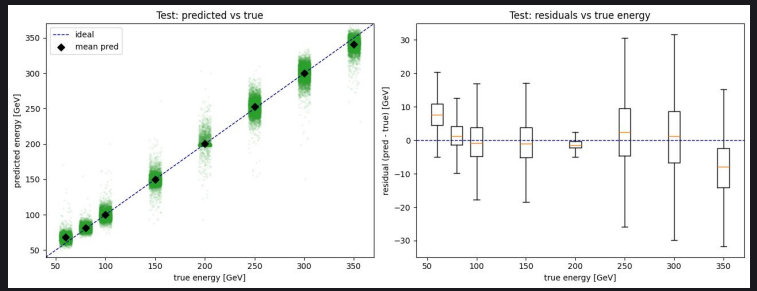
XGB



MLP



CNN



Test split

XGB
MAE 1.77 GeV

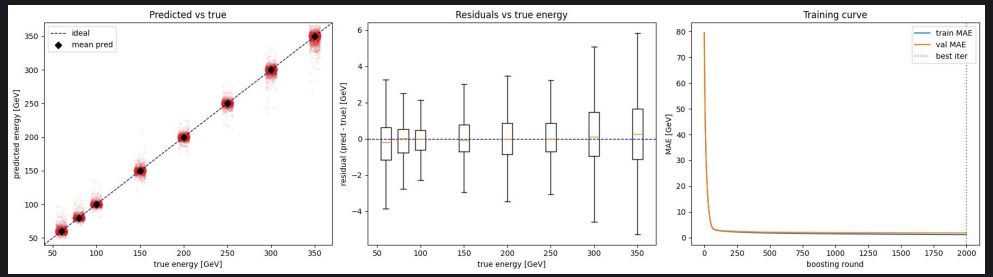
CNN
MAE 7.15 GeV

MLP
MAE 8.40 GeV

XGB near perfect in distribution

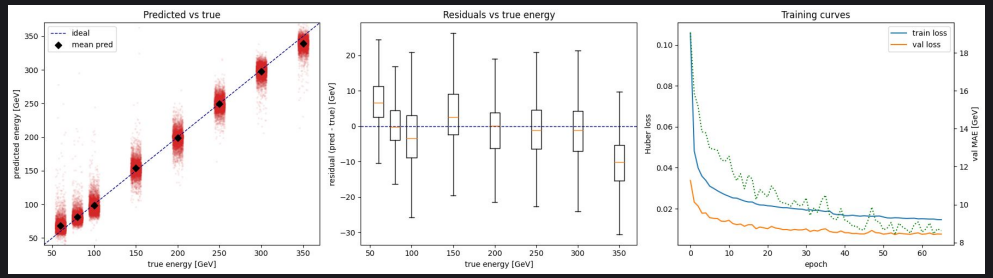
Validation split + training curves

XGB



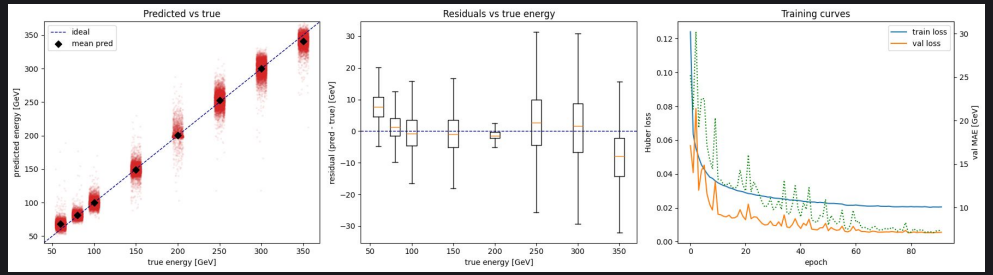
Very small gains on val after round 100 but early stopping doesnt trigger meaning there is a very slight increase in performance on the val loss until training forcibly stop at it 2000. Training time could be cut by a factor 10 with little to no real generalization performance loss by setting max iterations to 200.

MLP



Model converges quickly, could probably had gotten away with a smaller first hidden layer (256) perhaps moving it down to 128 or 64 to cut complexity and training time. However the mlp was already the fastest to train, so i didn't see much need in making that compromise.

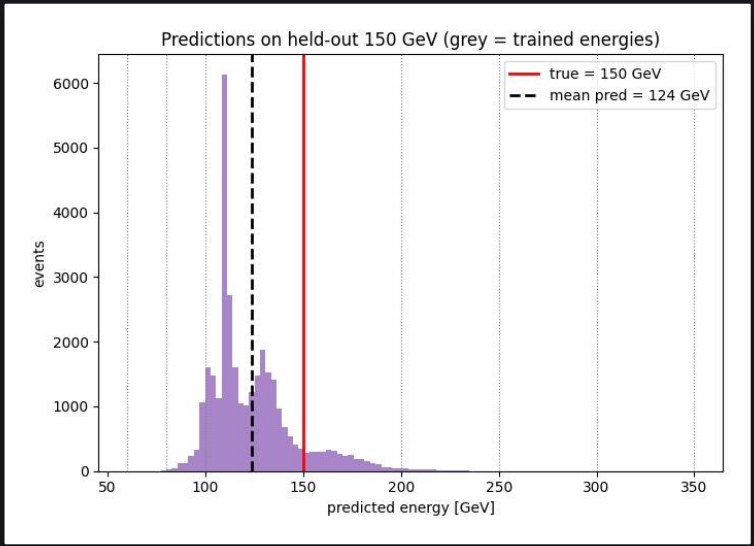
CNN



When looking at the residuals and predicted vs true energy we clearly see that the model often converges to the mean of the data around 190Gev instead of learning real energy / adc relations.

Test B: interpolation: the ranking flips

150 GeV removed from training entirely.



XGBoost on held-out 150 GeV mean prediction 124, undershooting badly.

Heldout 150 GeV · true = 150, seen mean = 190

MLP mean **169.6** MAE **23.7 GeV**

best mostly clean, near Gaussian, overshoots a touch

CNN mean **167.1** MAE **26.8 GeV**

close mean, but a spike at around 190 (training mean)

XGBoost mean **123.8** MAE **31.2 GeV**

worst doesn't extrapolate well, undershoots

Best in distribution (XGB) The worst at interpolation. The flattened NN generalizes best.

REFERENCE

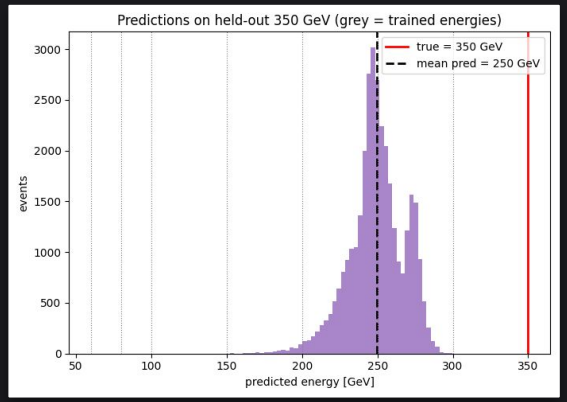
All metrics in one place

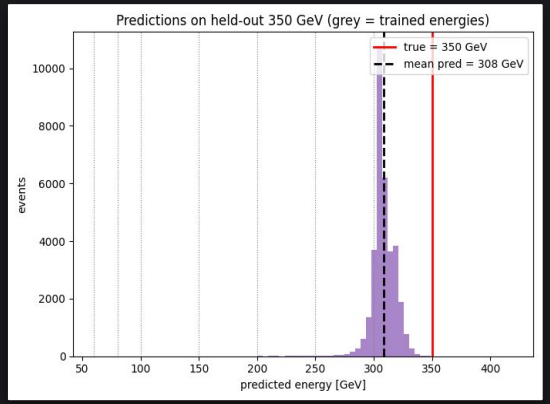
Model	Class. hold-out acc	Class. AUC	Reg. test MAE	Reg. test R^2	Hold-out 150: mean	Hold-out 150: MAE
MLP / NN	99.11%	0.9995	8.40 GeV	0.985	169.6 GeV	23.7 GeV
XGBoost	98.74%	0.9993	1.77 GeV	0.999	123.8 GeV	31.2 GeV
CNN	97.70%	0.9971	7.15 GeV	0.990	167.1 GeV	26.8 GeV

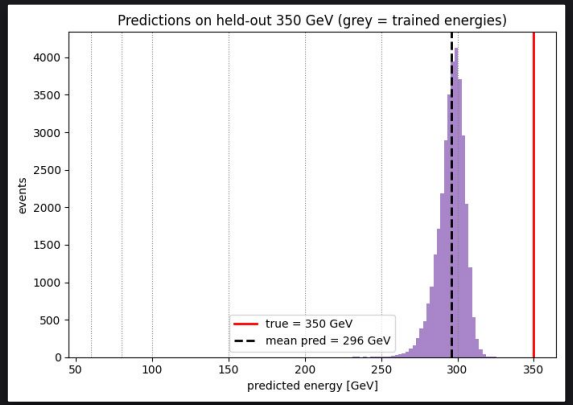
color cell = best in that column. Seen energy mean (holdout reference) = 190 GeV.

Inversion: XGBoost is best in distribution (1.77 GeV MAE) yet worst at interpolation (31.2 GeV MAE); the flattened MLP is the most reliable generalizing to an unseen energy (it must learn some underlying relations in the data).

Holdout 350 GeV extrapolation test (Highest energy the models saw in training was 300GeV)







XGBoost mean 249.5 GeV · MAE 100.5 GeV

MLP mean 308.4 GeV · MAE 41.6 GeV

CNN mean 295.8 GeV · MAE 54.2 GeV

XGB converges more or less to the seen 250 GeV data, very poor performance, unusable.

MLP: Good shape but under shoots a lot 41.6 GeV best by comparison but unusable by most metrics.

CNN: Good shape, also undershoots even undershoots the largest seen energy, 300GeV. poor performance, unusable.

Interestingly same ranking as interpolation, but much worse performance by all models approximately 1.5x MAE for the Neural Networks, more than 3x MAE for the XGB. Expected as trees really arent made for extrapolation by design limitations.

Narrow figures meaning models are more consistent than they were with interpolation.

Conclusion - Raw detector data in ML

A flat NN was the most reliable model in both tasks

Best held out classification (99.1%) and best energy extrapolation (23.7 GeV), despite discarding all spatial structure at the input.
Computationally cheaper to train than the CNN by a factor of approx 8 (3min vs 25min for cnn)

In distribution accuracy is not evidence of physics

With one run per energy, a standard split rewards run fingerprinting. XGBoost's 1.77 GeV looked best but failed the when an energy was held out.
Makes sense and is expected given the Decision tree architecture

The CNN's spatial prior didn't pay off here

It trailed in classification and leaked memorization (190 GeV spike) in extrapolation. Likely need feature engineering to properly utilize the spatial architecture to outperform the MLP

Best Generalization is the goal.

The decision tree did great in validation and test on seen energy, but generalizes poorly to unseen energy. All models do poorly in extrapolation (holdout of 350Gev, the ranking remains the same MLP > CNN > XGBoost

Appendix

Particle Energy identification

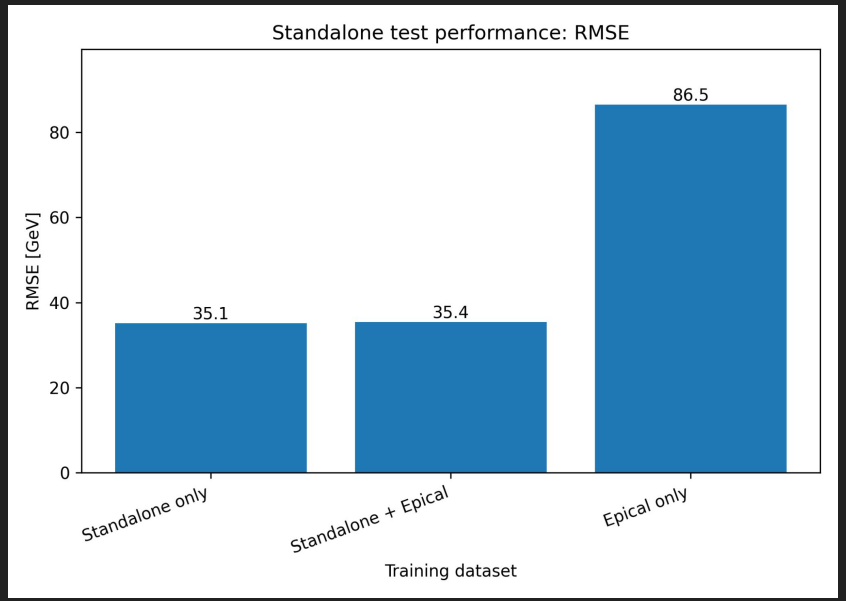
Metadata study and feature engineering

Particle Energy identification

Metadata study and feature engineering - With or Without EpiCal

- Demonstrates that including the EpiCal detector did not improve the performance, and training on EpiCal alone lead to significantly worse results.
- We accidentally grouped the data so almost all EpiCal data was with a locked phase, while the the standalone data was with unlocked phase. That meant that we could not differentiate between those two features.
- EpiCal had less data than standalone, and epical data was with SiPM = 47V while standalone was with SiPM = 45V, 45V

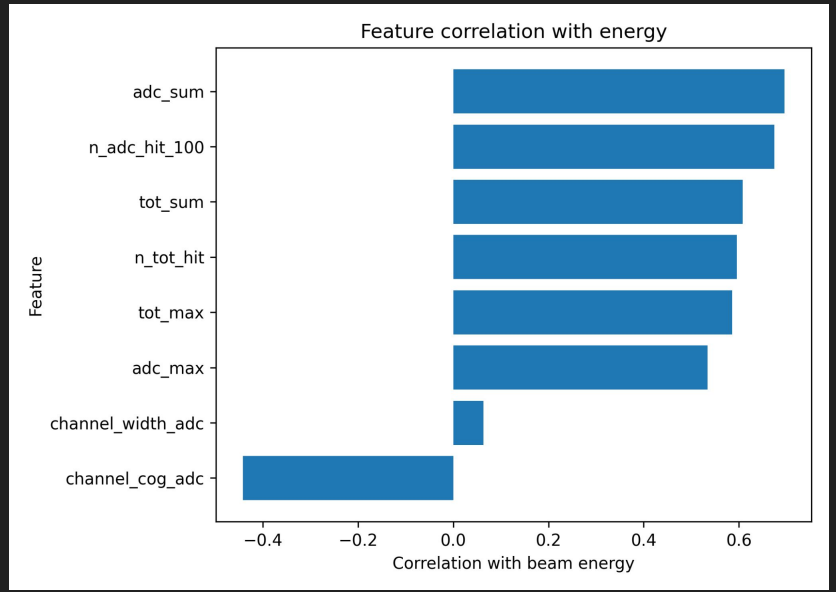
Standalone RMS after including Channel Map: 24.5 GeV



Particle Energy identification

Metadata study and feature engineering - All Data

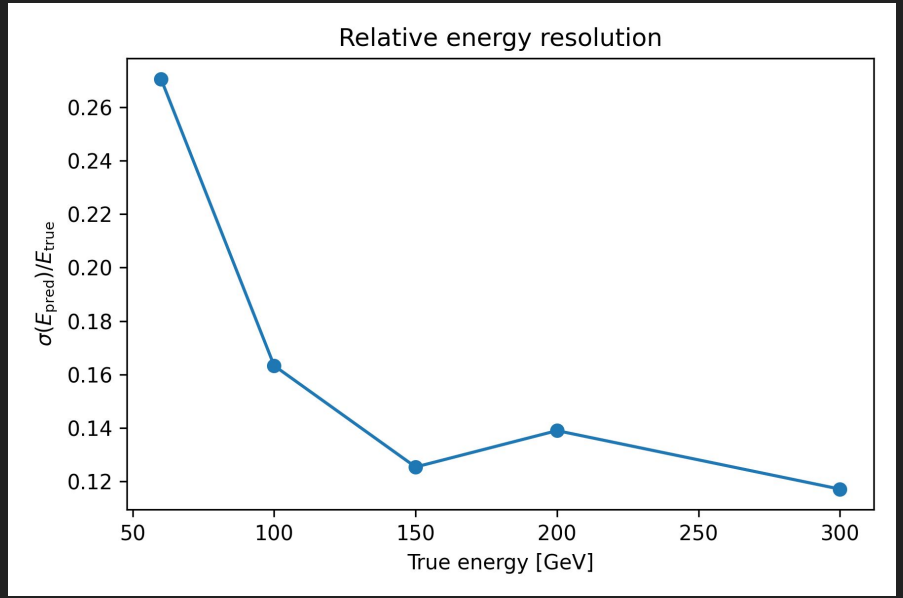
- Shows Pearson correlation coefficient between each feature and beam energy
  - If this variable increases, does energy tend to increase or decrease?
  - Not equivalent to feature importance
- Before channelmap is introduced, so channel_width and channel_cog are nonsensical



Particle Energy identification

Metadata study and feature engineering

- Prediction improve with increased beam energy
- This is as expected as higher energy showers gives larger signals, which gives higher signal to noise ratio

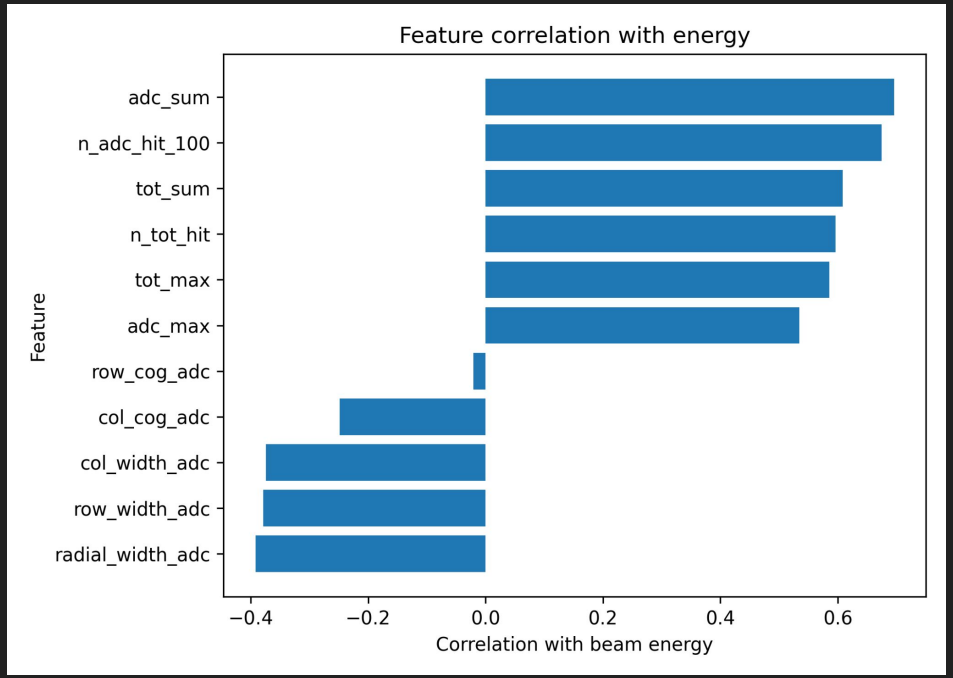


Particle Energy identification

Metadata study and feature engineering

If this variable increases, does energy tend to increase or decrease?

adc_sum	- Total ADC signal in event
n_adc_hit_100	- Number of channels with ADC > 100
tot_sum	- Total ToT signal in event
adc_max	- Maximum ADC signal in a single channel
tot_max	- Maximum ToT signal in a single channel
n_tot_hit	- Number of channels with non-zero ToT
row_width_adc	- Shower width in detector row direction
col_width_adc	- Shower width in detector column direction
radial_width_adc	- Overall shower transverse width
row_cog_adc	- ADC-weighted shower row position
col_cog_adc	- ADC-weighted shower column position



Appendix

3D Convolutional Network

PART 2 · REGRESSION

Model architectures (hadron energy regression)

3D CNN

Input: 2 × 12 × 16 × 16 (ADC, ToT, detector rows, detector cols, time). Energy truth labels were log transformed and all data was normalized before being fed into CNNs. 150 GeV events held out of training data. Trained on RTX 4050m

Regression: hadron energy scan

8 runs: 274K events total

Roughly equal size, about 35k each ± 1k

60GeV 80GeV 100GeV 150GeV
200GeV 250GeV 300GeV 350GeV

one run per energy

3D CNN - GPU train time: 3h 32m 21s
best epoch: 289

```
Loss:Huber
eval_metric: MAE

ARCHITECTURE
Conv3D 2->16 (3×3×3, pad 1) -> ReLU
Conv3D 16->32 (3×3×3, pad 1) -> ReLU
MaxPool3D (1×2×2)      12×16×16 -> 12×8×8
Conv3D 32->64 (3×3×3, pad 1) -> ReLU
MaxPool3D (2×2×2)      12×8×8 -> 6×4×4
Conv3D 64->128 (3×3×3, pad 1) -> ReLU
AdaptiveAvgPool3D -> 1×1×1
Flatten
Linear 128->64 -> ReLU
Dropout 0.2
Linear 64->1
≈ 332k parameters
OPTIMIZER + SCHEDULER
AdamW
lr 1e-3  weight_decay 1e-4
ReduceLROnPlateau  factor 0.5
patience 10  min_lr 1e-6
batch 256
```

3D CNN - GPU train time: 26m 0s
best epoch: 91

```
Loss:Heteroscedastic (gaussian log likelihood)
eval_metric: MAE

ARCHITECTURE
Conv3D 2->16 (3×3×3, pad 1) -> ReLU
Conv3D 16->32 (3×3×3, pad 1) -> ReLU
MaxPool3D (1×2×2)      12×16×16 -> 12×8×8
Conv3D 32->64 (3×3×3, pad 1) -> ReLU
MaxPool3D (2×2×2)      12×8×8 -> 6×4×4
Conv3D 64->128 (3×3×3, pad 1) -> ReLU
AdaptiveAvgPool3D -> 1×1×1

Flatten
Linear 128->64 -> ReLU
Dropout 0.2
Linear 64->1
≈ 332k parameters

OPTIMIZER + SCHEDULER
AdamW
lr 1e-3  weight_decay 1e-4
ReduceLROnPlateau  factor 0.5
patience 8  min_lr 1e-6
batch 256
```