Analysis of FoCal-H Test-beam Data

by Ian Pascal, Christian Behrendorff Madsen

The Experiment

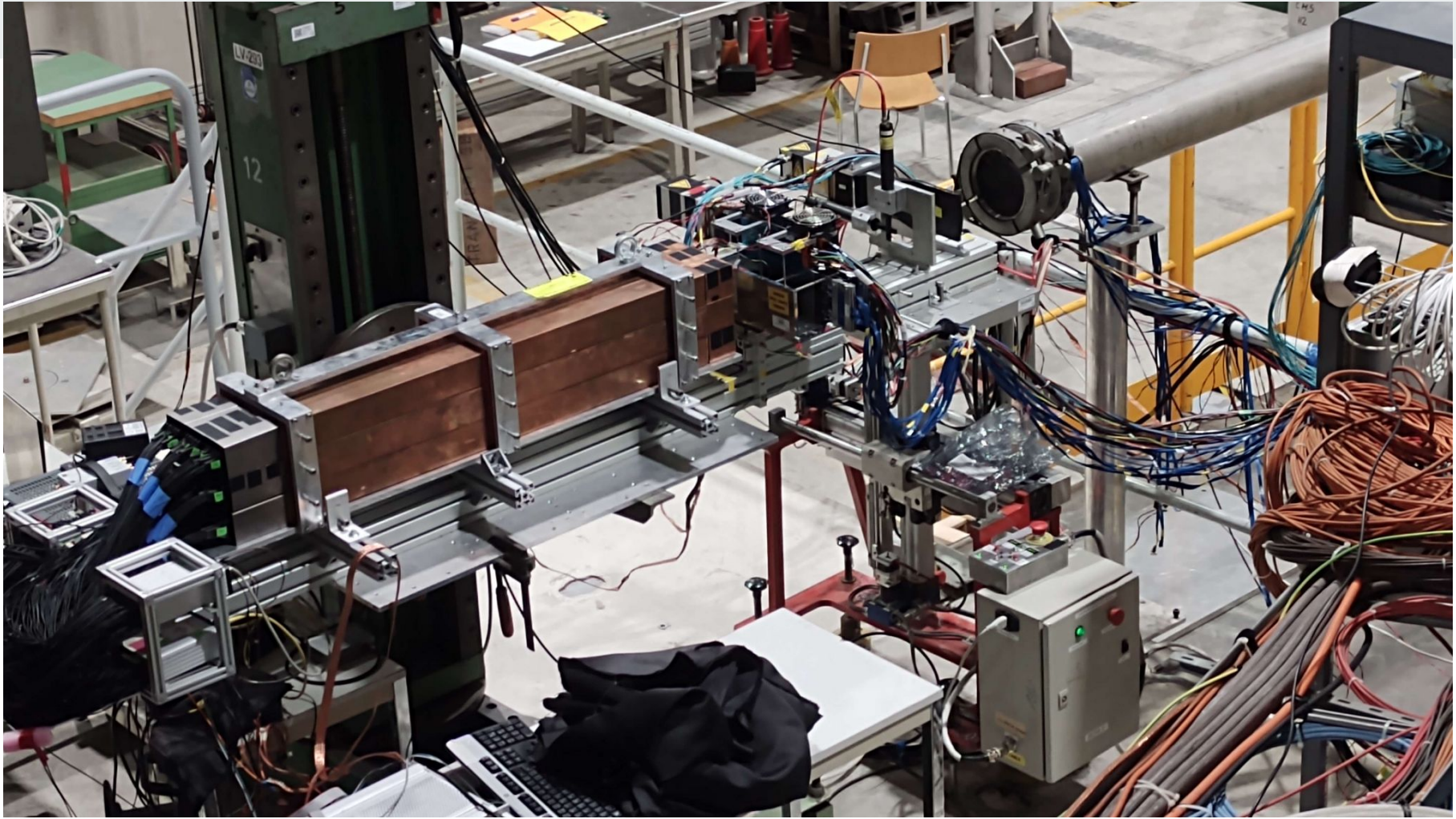
“FoCal-H”: Forward Calorimeter, Hadronic Part - an in-progress upgrade for ALICE @ CERN

A prototype device to measure the energy of particle “showers”

Essentially a long block of copper with photodetectors on the end

Has been tested on-site at CERN a few times

Last time was just a few weeks ago!



The Data

Grid-like arrangement of photodetectors give an image-like structure

One image = one event = one particle shower (usually!)

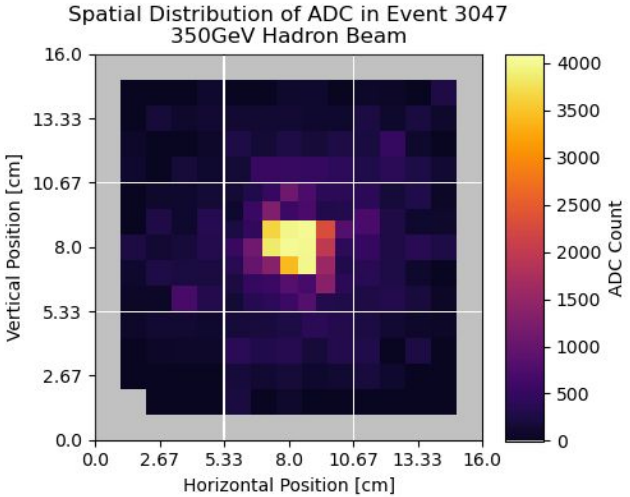
Showers typically characterized by two parameters: amplitude & width

Three types of particle showers are considered:

Hadrons: high energy, medium spread

Electrons: low energy, high spread

Muons: very high energy, very low spread



The Data

During the test-beam week, some hundreds of GB of data was taken across many individual sets of data

For this project, ~40 GB = 29 sets =~ 3.1 million events in total were used

A single data set is represented as an (N, 256)-matrix:

N events recorded, each with 256 values

The 256 values represent the recorded energy in each channel of the calorimeter

A data set contains shower data at a specific “(beam) energy”

Yes - this is actual, real, fresh and yet-analyzed experimental data!

The Aim of the Project

To be able to predict the (beam) energy of a given event

- Known beam energy makes this a supervised classification problem
- We have eight beam energies: 60, 80, 100, 150, 200, 250, 300 and 350GeV

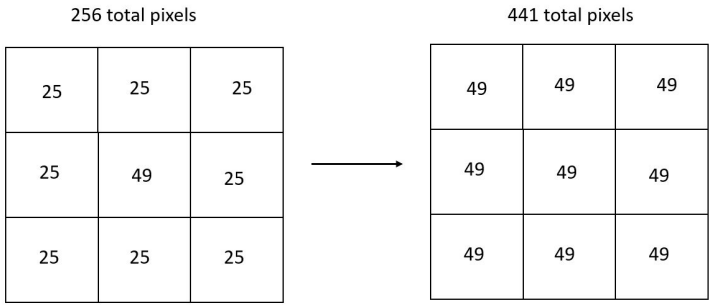
To be able to sort a data set into hadron, electron and muon showers (and remove noise/anomalies)

- Clustering of events - but it’s unsupervised: event labels do not exist
- Partially, the reason this project exists is to MAKE these labels!

First Steps - Data Preprocessing

One might think that because we have 256 channels, we have 16x16 images..

..but that is not the case!



Some parts must be up- or down-scaled to give a regular grid

We want to leave the image as unchanged as possible

Decision was to upscale each outer section from 5x5 to 7x7

Leaving the middle section (where most information is) unchanged

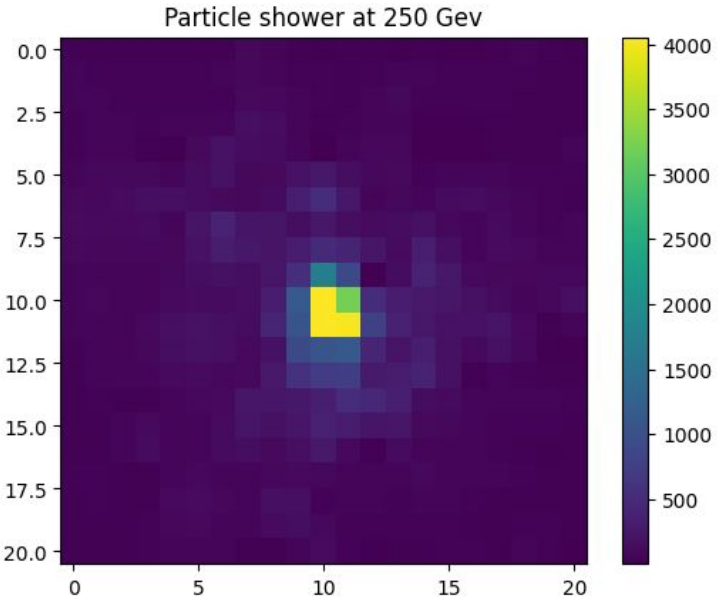
First Steps - Data Preprocessing

After image re-scaling, a data set now has the shape (N, 21, 21)

Other preprocessing that was done:

Invalid channel values zeroed (demanding 0 =< x =< 4096)

Channel values then normalized (to [0, 1])



First Steps - Deciding on Models

Some challenges had to be considered:

- A particle shower is translationally invariant
- There is considerable overlap of showers between different beam energies
- We did not know much about the contents of the data (# of a specific shower type, etc.)

So, we settled on using two types of models:

- Convolutional Neural Network, for energy classification
- Variational Auto-Encoder, for event clustering

CNN - Structure

Adapted structure from commonly seen networks

Input shape is (21, 21, 1)

Going from an image to some number of independent variables

Then compressed down to 8 variables that represent the probability of each energy

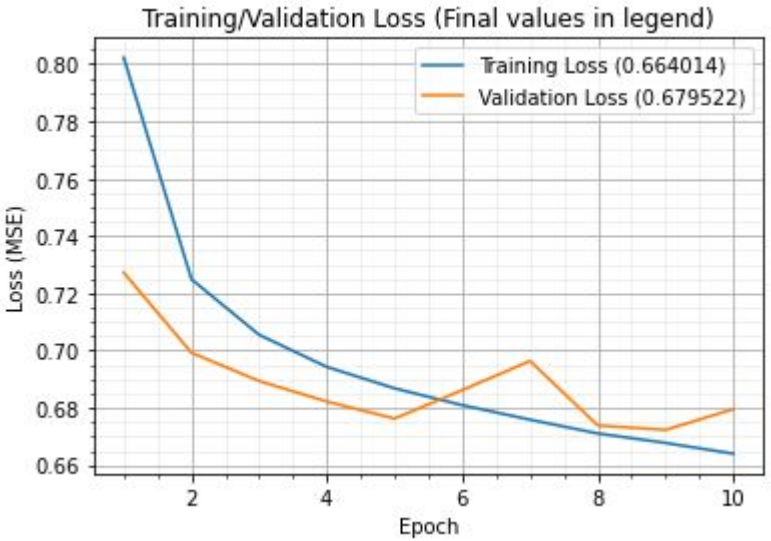
Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 21, 21, 64)	640
max_pooling2d_4 (MaxPooling 2D)	(None, 10, 10, 64)	0
conv2d_5 (Conv2D)	(None, 10, 10, 128)	73856
max_pooling2d_5 (MaxPooling 2D)	(None, 5, 5, 128)	0
conv2d_6 (Conv2D)	(None, 5, 5, 256)	295168
max_pooling2d_6 (MaxPooling 2D)	(None, 2, 2, 256)	0
conv2d_7 (Conv2D)	(None, 2, 2, 512)	1180160
max_pooling2d_7 (MaxPooling 2D)	(None, 1, 1, 512)	0
dropout_1 (Dropout)	(None, 1, 1, 512)	0
flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dense_2 (Dense)	(None, 8)	4104

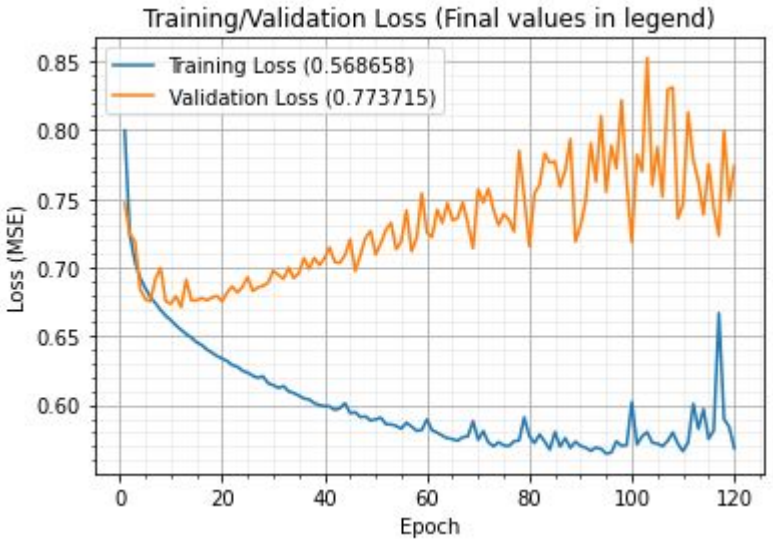
Total params: 1,816,584
Trainable params: 1,816,584
Non-trainable params: 0

CNN - Performance

Network trained on ~3 million events across 8 energies

Best results seen around epoch 10





CNN - Classification

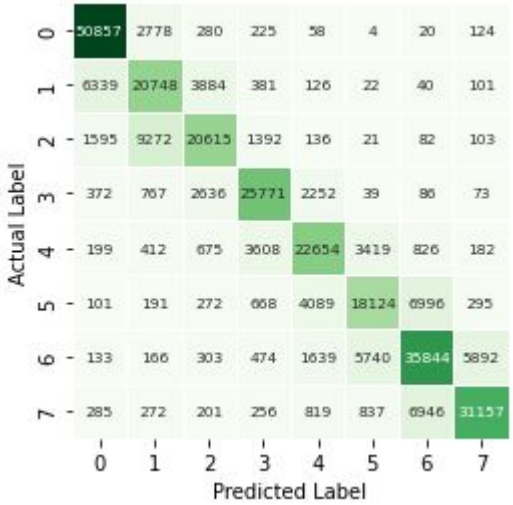
Correctly classifies ~65% of events (vs. human at ~57%)

But most of the time, a wrong classification is only slightly off:

95% correct when taking predicted label's neighbors as well

(vs. human at 87%)

So, why is this?



Relative std	80 GeV	100 GeV	150 GeV	200 GeV	250 GeV	300 Gev
Input	0.236	0.212	0.180	0.204	0.201	0.321
Output	0.288	0.239	0.213	0.187	0.181	0.151

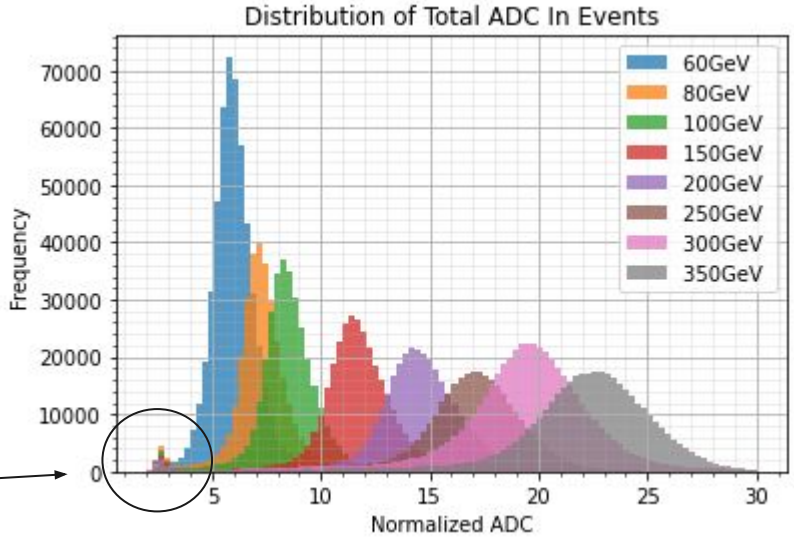
CNN - Classification

The total energy in an event is not unique to a certain beam energy (label)

Instead, these follow a Gaussian distribution, with significant overlap

Muons also creating an extra peak

Small relative energy resolution (18%-33%) on the input



Muon peak
Same for all energies

VAE - Structure

Layers were set up similarly to the CNN

Final output of encoder (left) is the 16-dimensional latent space

Decoder (right) takes latent space back into an image

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 11, 11, 32)	320
conv2d_1 (Conv2D)	(None, 6, 6, 64)	18496
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 16)	18448
re_lu (ReLU)	(None, 16)	0

Total params: 111,120
Trainable params: 111,120
Non-trainable params: 0

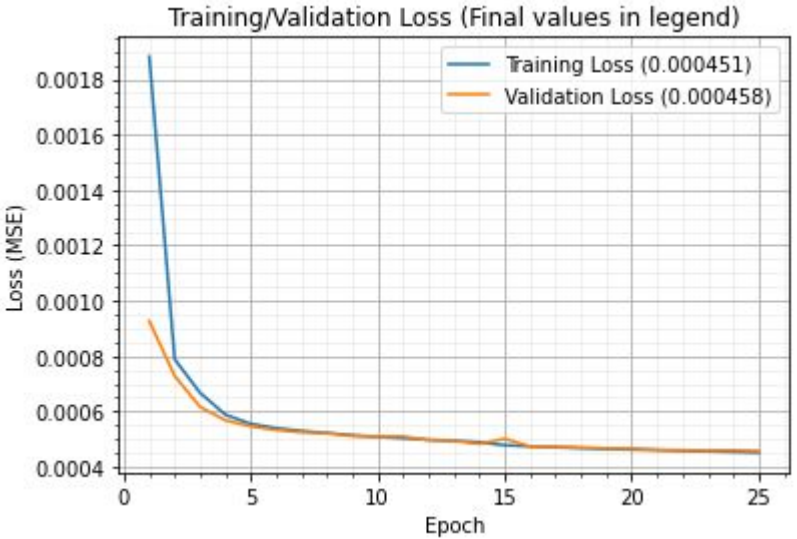
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1152)	19584
reshape (Reshape)	(None, 3, 3, 128)	0
conv2d_transpose (Conv2DTranspose)	(None, 6, 6, 64)	73792
conv2d_transpose_1 (Conv2DTranspose)	(None, 12, 12, 32)	18464
conv2d_transpose_2 (Conv2DTranspose)	(None, 24, 24, 1)	289
resizing (Resizing)	(None, 21, 21, 1)	0

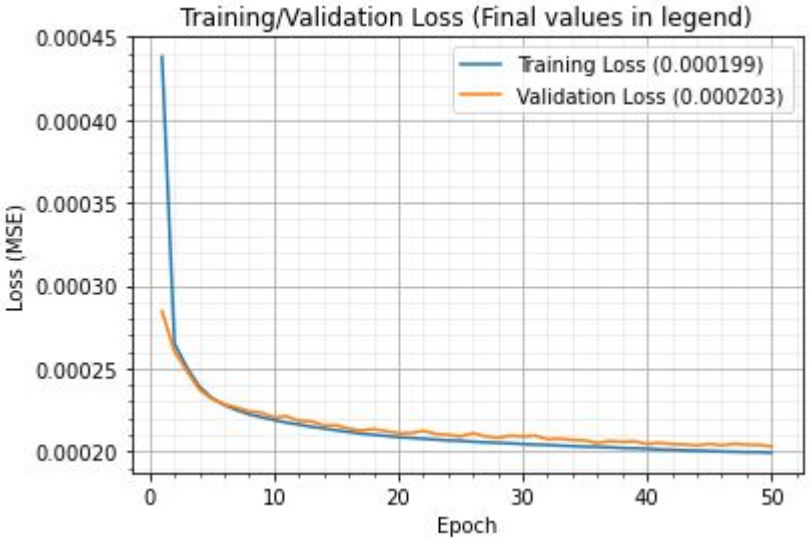
Total params: 112,129
Trainable params: 112,129
Non-trainable params: 0

VAE - Performance

Trained on fewer events: ~100k at same beam energy (left), and ~800k at various energies (right)

Dimension of latent space was chosen to be 16

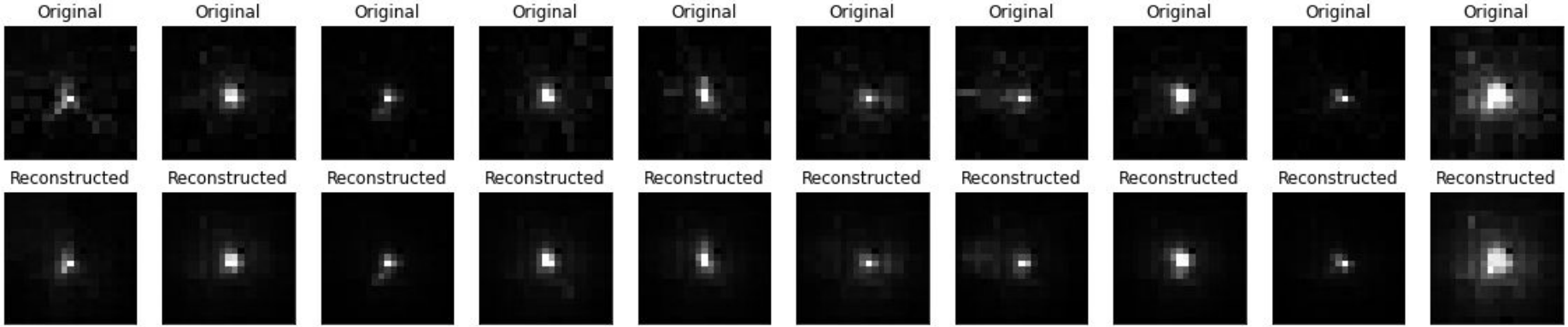




VAE - Performance

The network is able to recreate events very well

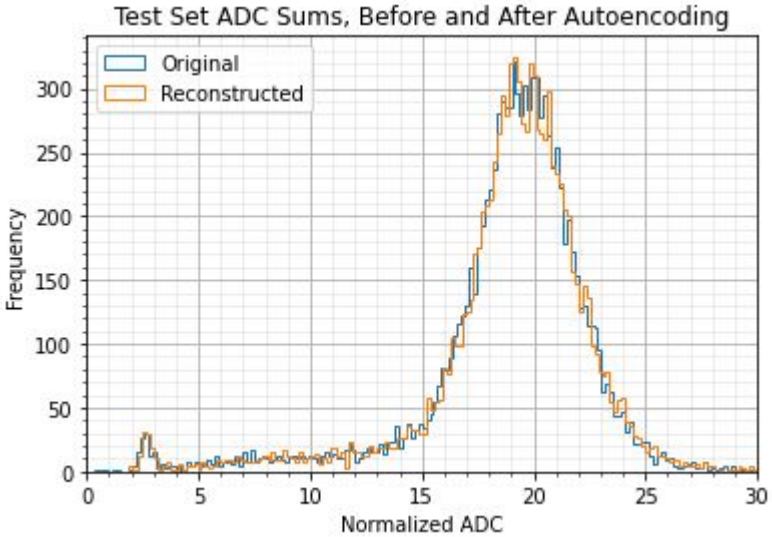
Seems to de-noise / remove background

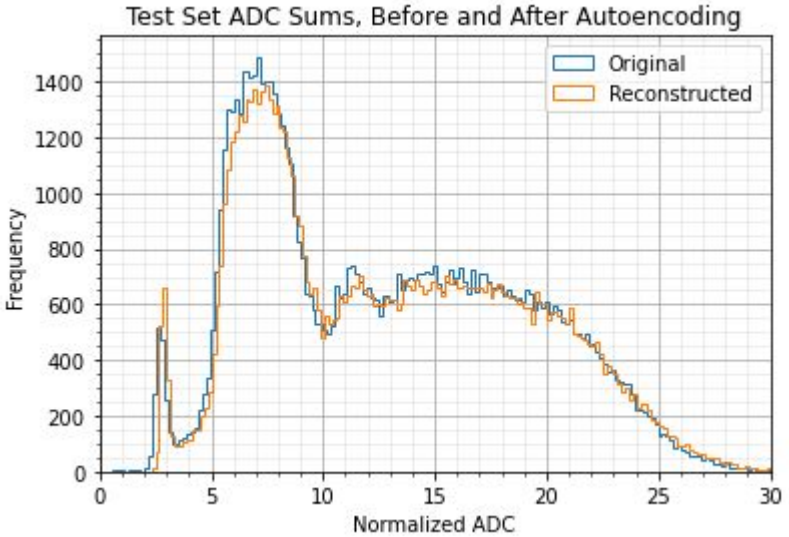


VAE - Performance

The network is able to recreate events very well

And doesn't change the energy distribution much (left: 300GeV; right: all energies)





VAE - Clustering

We perform clustering on the 16 latent variables

Multiple different algorithms were tried & tested

- For clustering of events according to type, BIRCH seems most effective
- For clustering of events according to energy, KMeans is much more effective

Clustering was also attempted on a reduced-dimensionality set of the latent variables

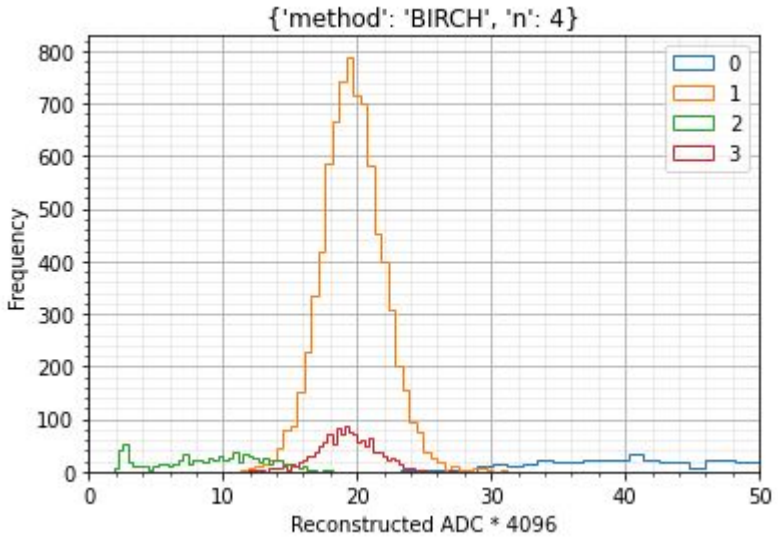
- Using UMAP to do this, then testing clustering algorithms
- Results decent, but never better than clustering latent variables directly

VAE - Clustering

Separates the full spectrum into multiple, Gaussian distributions

This is how we know it works!

Clusters can be further split by clustering again

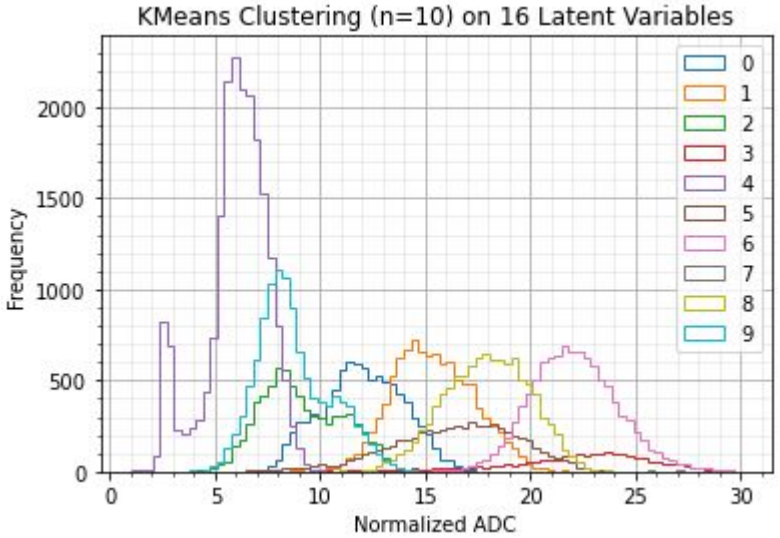


VAE - Clustering

Can also fairly well separate different beam energies

But no. of clusters has to be tuned slightly

Still not perfect, but good outlook



Hyperparameter Optimization

Learning rate of models was briefly explored

Optimization of networks was mostly kept to the topology

- Number and types of layers, filters, etc.
- Finding the right latent dim. size

Various loss functions were used too

- Mean squared error semt best

Summary and Future Work

Both CNN and VAE performed very well - beyond expectations!

- Accuracy of CNN is limited by similarity of events
  - But shower features are learnable and distinguishable
  - Maybe a regressor instead?
- VAE enables separation of events to a high degree
  - But it is very dependent on the clustering algorithm
  - Could also function to de-noise events

Thank you for your attention!

It is now question time!

Appendix

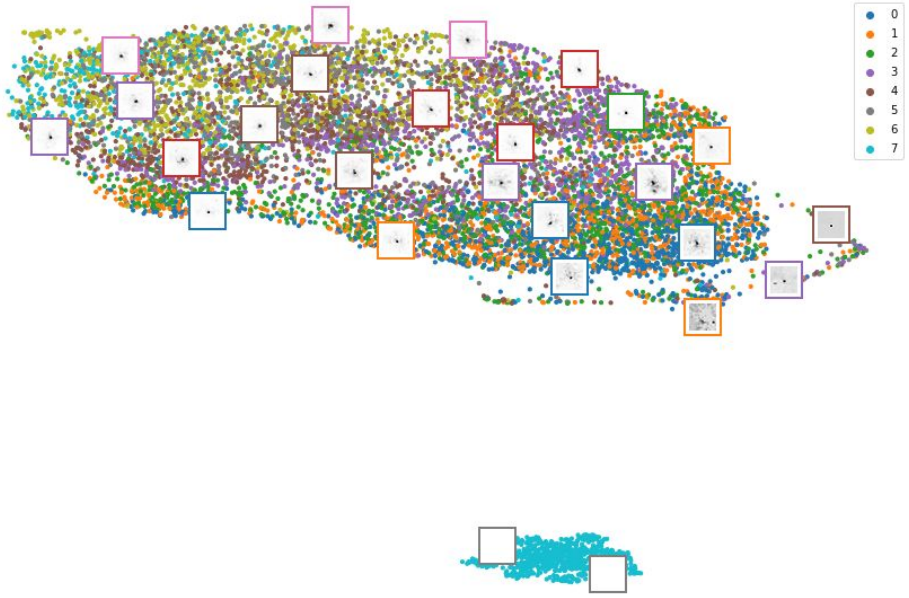
UMAP On VAE Latent Space

Outliers (empty events, etc.) easily removed

Different beam energies have large overlap

But are distinguishable (notice the “bands”)

(also, colors of image borders are irrelevant here)



Event Images from VAE Clusters

Four clusters obtained using BIRCH on a data set at a specific beam energy (see slide 18)

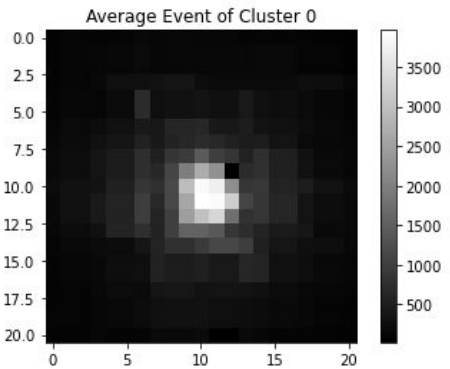
Each cluster has an “average event” (i.e. sum of all events, then divided by number of events)

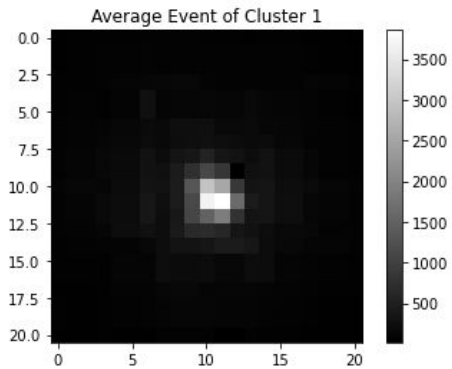
0: unknown

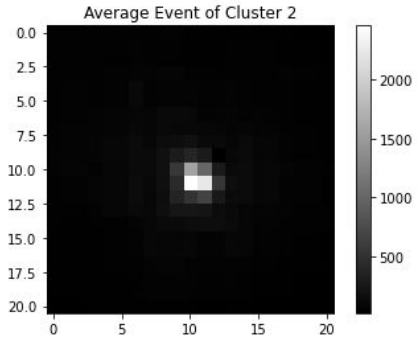
1: hadron showers

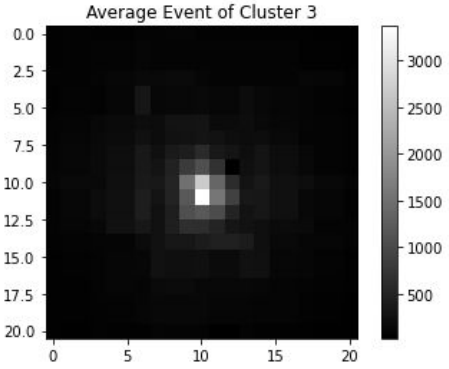
2: muon events (+ something else; have to cluster this separately!)

3: electron showers







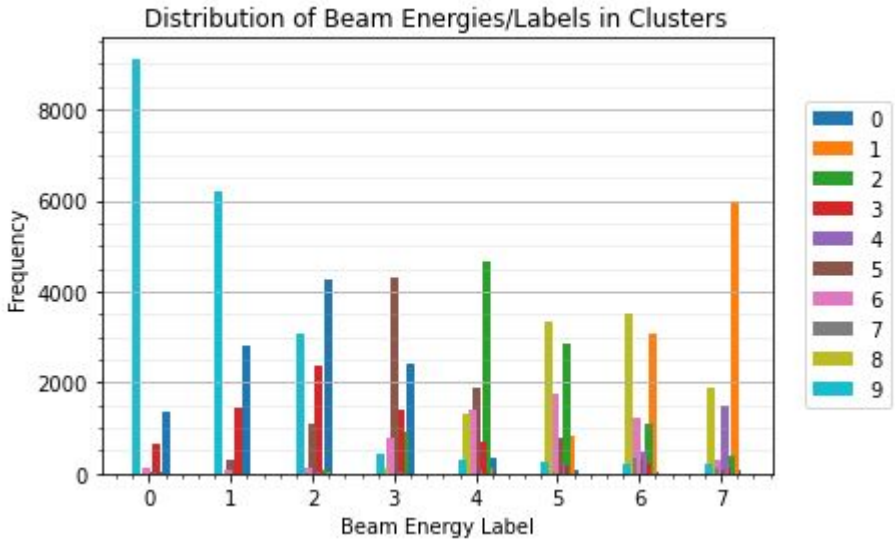


Distribution of Beam Energies in VAE KMeans Clusters

This graph shows the number of each beam energy label (horizontal axis) that is present in each of the 10 found clusters from slide 19.

The distribution is much akin to the CNN's predictions of beam energies

But not perfect, due to clustering not being the best

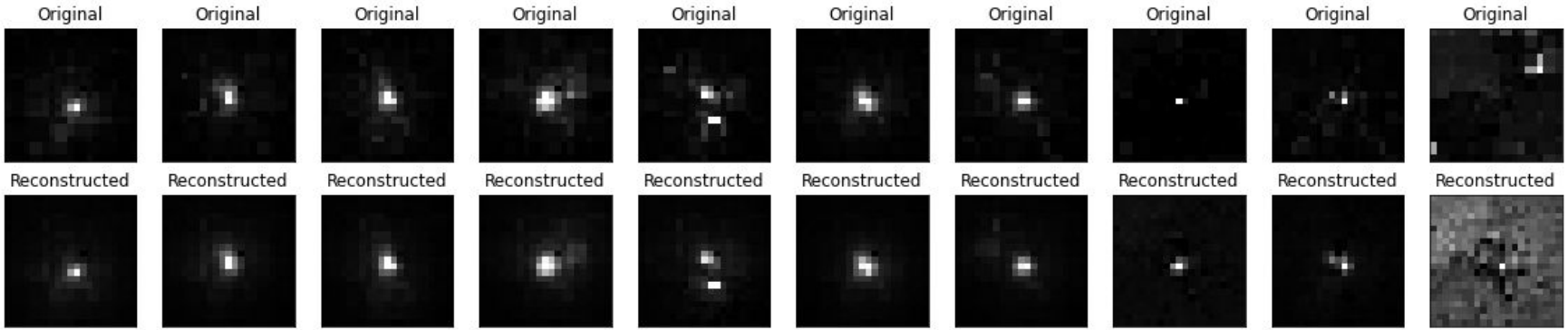


Potential for VAE to Serve as Anomaly Detector/Removal

In the reconstructed images below, we see that regular events are mostly unchanged

Including strange, but legitimate events (5th)

But obviously anomalous events (last image) are very different, thus easily detected

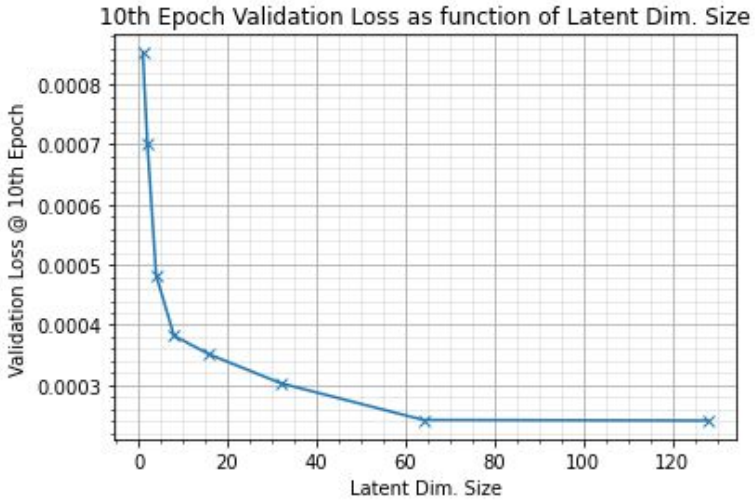


Search for Optimal Latent Space Dimension

VAE functions well down to about 8 latent variables

Does not become much better above 64

Group decided on 16 to save computational power on clustering

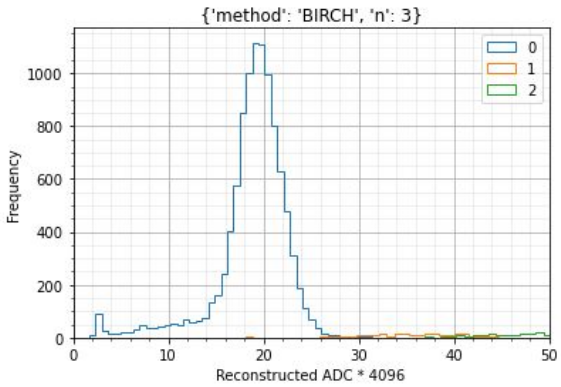


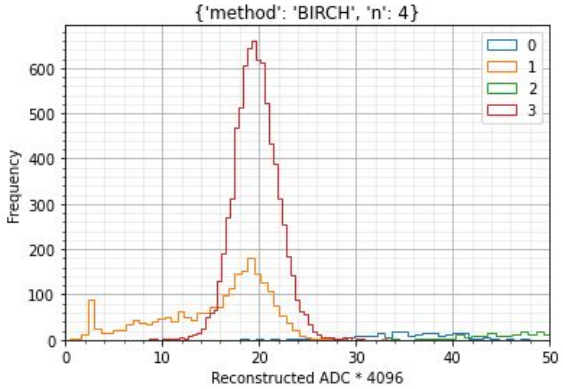
Examples of BIRCH Clustering on VAE Output

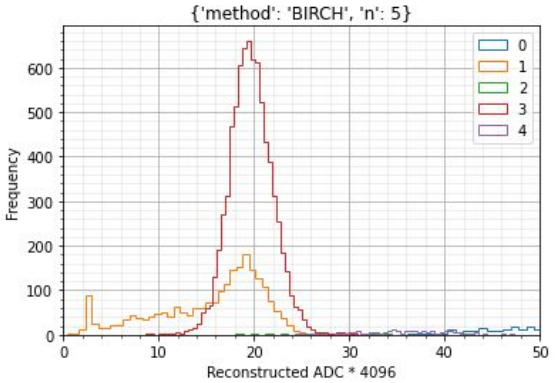
The graphs below highlight the importance of adjusting clustering as needed

Left-most does not separate the primary peak, while the other two does

However, also no new information is gained from the additional group in the right graph

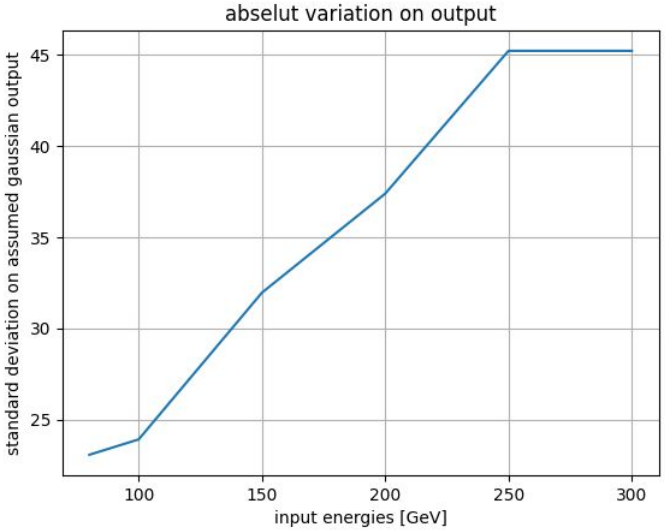


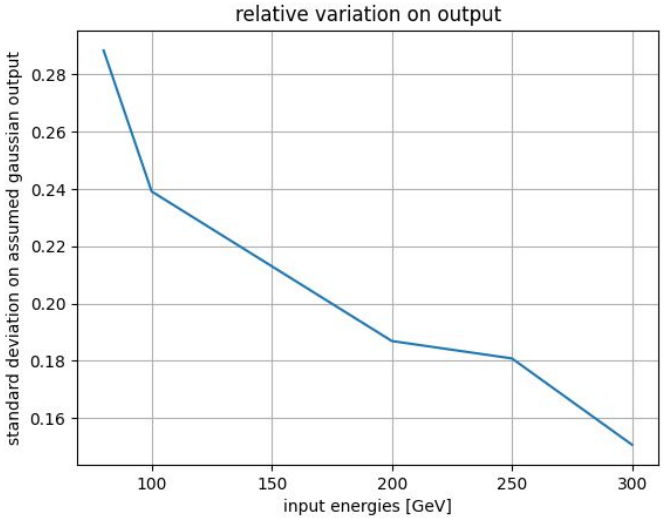




Variations of the energy in the CNN outputs

The absolute and relative variation of energy output as a function of input energies





Descriptions of Used .py Scripts

VAE.py - Script for the autoencoder network & related outputs

CNN.py - Script for the convolutional neural network & related outputs

HCalImageBuilder.py - Takes a data file and converts it to an (N, 21, 21)-matrix for use here

ImageDataGrouper.py - Takes M (N, 21, 21)-matrices and combines them into one data set with labels

AnalysisLibrary.py - A library of various functions used for analysis of FoCal-H data. Only the channel mapping is used here.