MLHEP2021 Competitions

anyon2D team

anyon2D Team

Andrea di Luca
UniTn & FBK
deeppp.eu

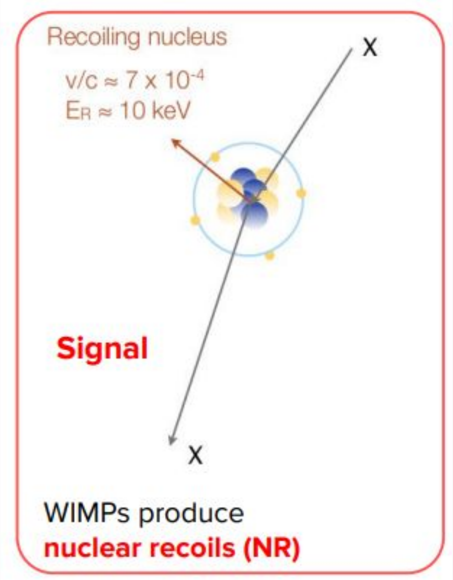
Daniela Mascione
UniTn & FBK
deeppp.eu

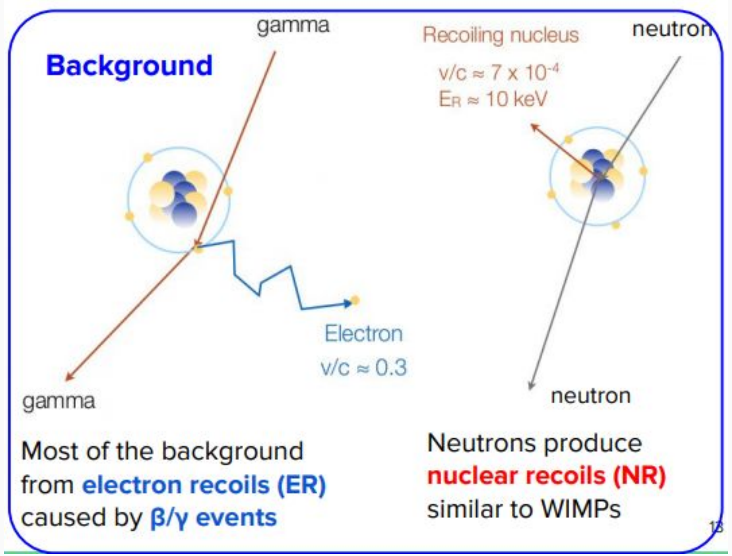
Giovanni Guerrieri
UniTS & ATLAS

Caterina Trimarelli
UnivAq & INFN-LNGS

Challenge 1

Electron recoil background rejection @ CYGNO experiment





Dark Matter particles could interact with the nuclei in ordinary matter, producing highly ionizing nuclear recoils (NR) with a kinetic energy as small as few keV. These NR would travel for hundreds to thousands of microns in gas leaving a trail of ionized atoms and free electrons.

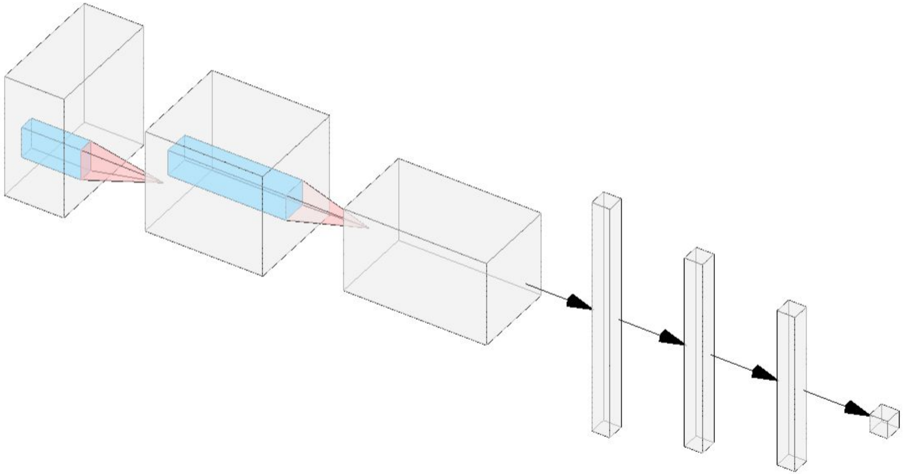
Low energy photons produced by natural radioactivity can ionize electrons from atoms and molecules in the detector, producing recoils that would represent an **important and dangerous background to Dark Matter signals**.

G. D'Imperio presentation

Classifier architecture

```python
class Classifier(pl.LightningModule):
    def __init__(self, mode: ["classification", "regression"] = "classification"):
        super().__init__()
        self.mode = mode
        self.layer1 = nn.Sequential(
                        nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=2),
                        nn.BatchNorm2d(6),
                        nn.ReLU(),
                        nn.MaxPool2d(kernel_size=2),
                        nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=2),
                        nn.BatchNorm2d(16),
                        nn.ReLU(),
                        nn.MaxPool2d(kernel_size=2),
                        nn.Flatten(),
                )

        self.drop_out = nn.Dropout()
        self.selu = nn.SELU()
        self.fc1 = nn.Linear(14400, 1800)
        self.fc7 = nn.Linear(1800, 500)
        self.fc2 = nn.Linear(500, 2)  # for classification
        self.fc3 = nn.Linear(500, 1)  # for regression
        self.stem = nn.Sequential(
            self.layer1, self.drop_out,
            self.fc1,self.selu,self.drop_out,
            self.fc7,self.selu,self.drop_out,
            )
        if self.mode == "classification":
            self.classification = nn.Sequential(self.stem, self.fc2)
        else:
            self.regression = nn.Sequential(self.stem, self.fc3)

        self.train_acc = pl.metrics.Accuracy()
        self.valid_acc = pl.metrics.Accuracy()
        self.test_acc = pl.metrics.Accuracy()
```

Code soon on GitHub



*produced with *NN SVG*

Training strategy

Dataset
6761 ER images
6649 NR images
576 x 576 pixel images

Training dataset
10k images

Validation dataset
~3k images

Image preprocessing
Crop image to center (120 pixel)

Optimizer

Stochastic gradient descent
- Learning rate:10^{-2}
- Weight decay: 10^{-5}
- Momentum: 0.95

Regularization layers

- **Dropout** (p=0.5) between fully connected layers
- **BatchNorm** between convolutional layers

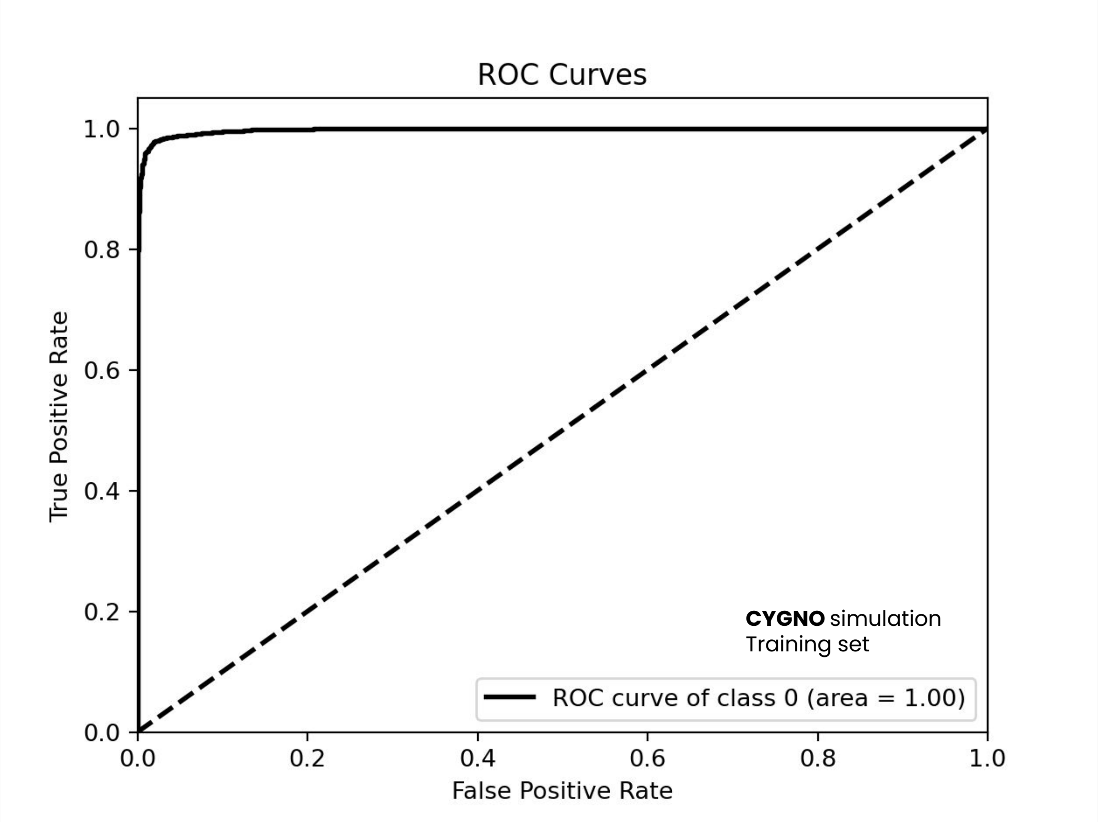
Model is chosen looking at the **best epoch** after training.

Best epoch metric
Accuracy on validation dataset

Classifier performance

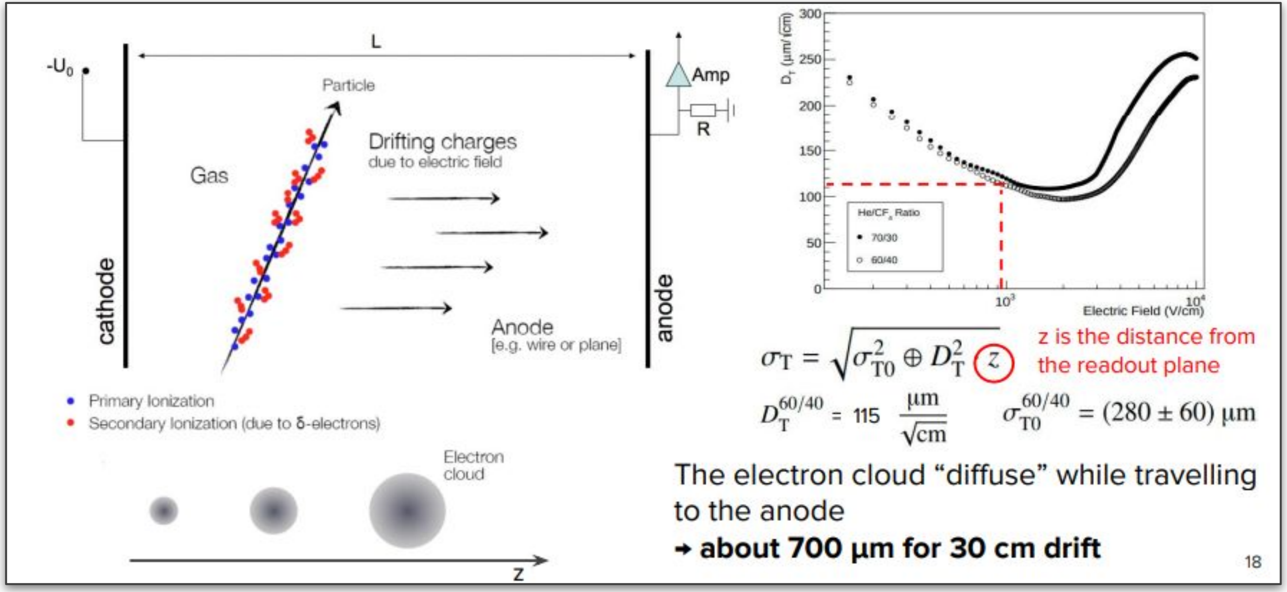
99.4%

ROC AUC
evaluated on test set



Challenge 2

Kinetic energy regression @ CYGNO experiment



While traveling in gas, **ionized electrons** are subject to **diffusion effects** that **blur final images**. Moreover, optical sensors used to collect the photons have a small, yet not negligible, electronic noise that adds up to physical signals.

G. D'Imperio presentation

Model architecture & Training strategy

```python
class Regressor(pl.LightningModule):
    def __init__(self, mode: ["classification", "regression"] = "classification"):
        super().__init__()
        self.mode = mode
        self.layer1 = nn.Sequential(
                        nn.Conv2d(1, 16, kernel_size=5, stride=1, padding=2),
                        nn.BatchNorm2d(16),
                        nn.SiLU(),
                        nn.MaxPool2d(kernel_size=19, stride=7),
                        nn.Flatten(),
                    )

        #self.drop_out = nn.Dropout()

        self.fc1 = nn.Linear(3600, 500)
        self.fc2 = nn.Linear(500, 2)  # for classification
        self.fc3 = nn.Linear(500, 1)  # for regression


        self.stem = nn.Sequential(
            self.layer1, self.fc1
            )
        if self.mode == "classification":
            self.classification = nn.Sequential(self.stem, self.fc2)
        else:
            self.regression = nn.Sequential(self.stem, self.fc3)

        self.train_acc = pl.metrics.Accuracy()
        self.valid_acc = pl.metrics.Accuracy()
        self.test_acc = pl.metrics.Accuracy()
```

The Dropout has been removed in order to improve the validation loss

BATCH SIZE = 128

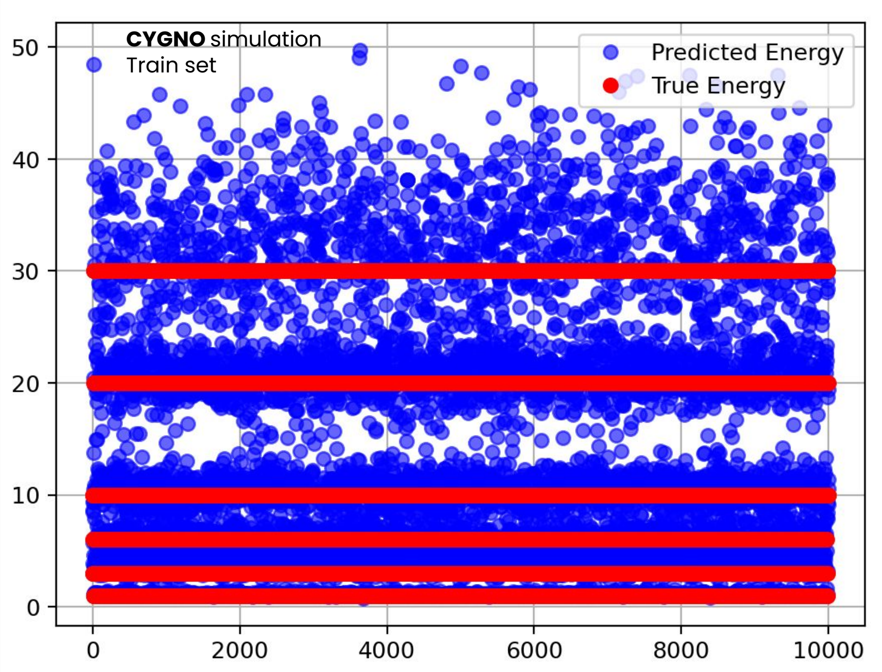
```python
Option (1)
reg_loss = F.mse_loss(reg_pred,
reg_target.float().view(-1, 1))

optimizer =
torch.optim.Adam(self.parameters(), lr=1e-3,
weight_decay=0.0001)
```

BATCH SIZE = 64

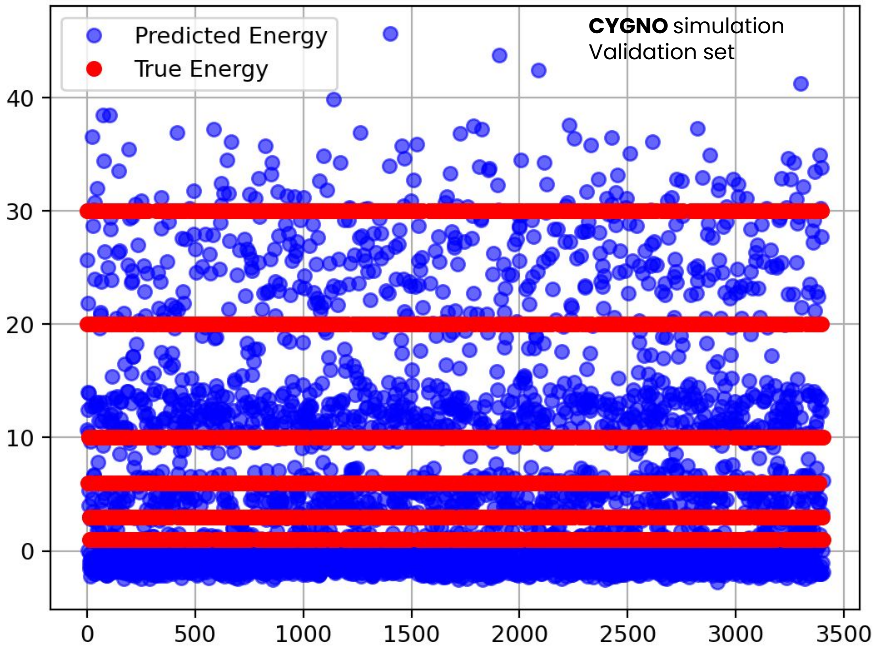
```python
Option (2)
reg_loss = torch.sum(torch.abs(reg_pred -
reg_target.float().view(-1, 1)) /
reg_target.float().view(-1, 1))

 optimizer =
torch.optim.Adam(self.parameters(), lr=1e-2,
weight_decay=0.0001)
```

Energy comparison

Training

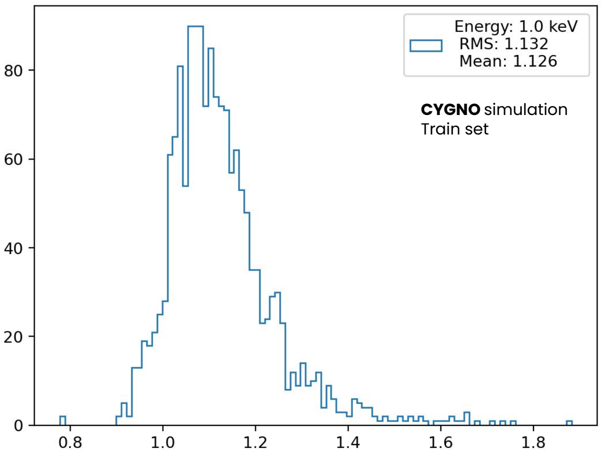


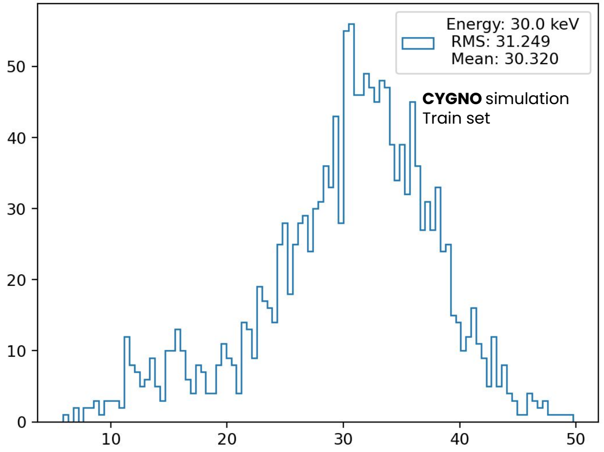
Validation

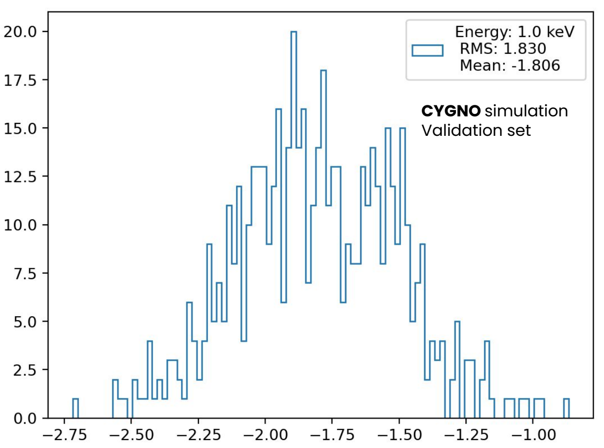


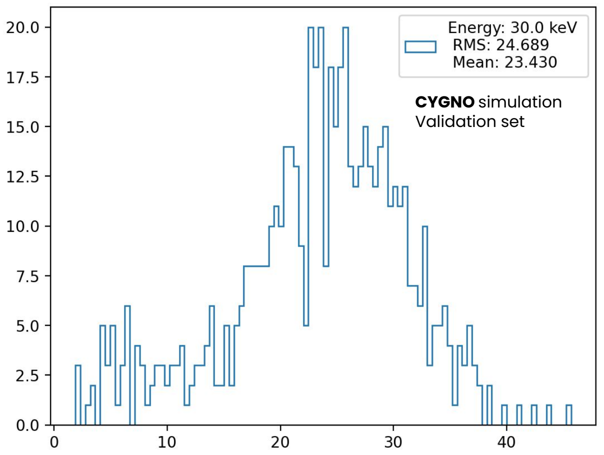
Energy histograms

Training







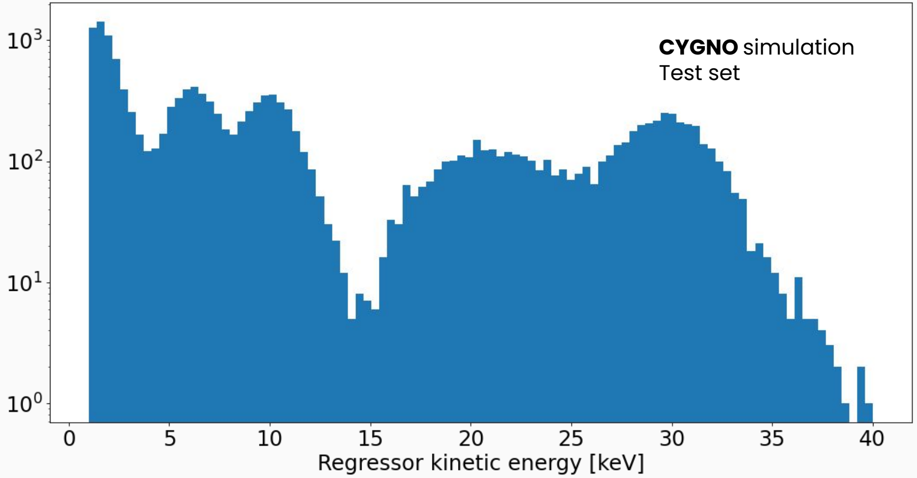


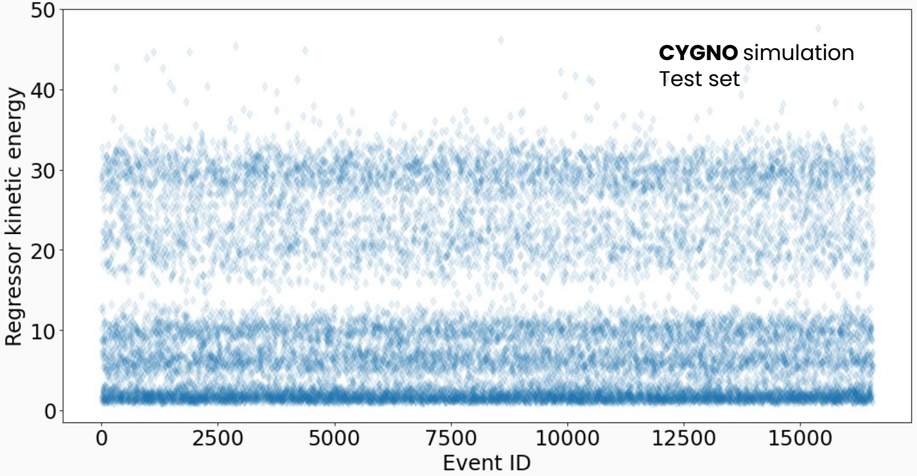
Validation

Regressor performance

MAE=1.48

evaluated on test*





*already second best results on Kaggle leaderboard

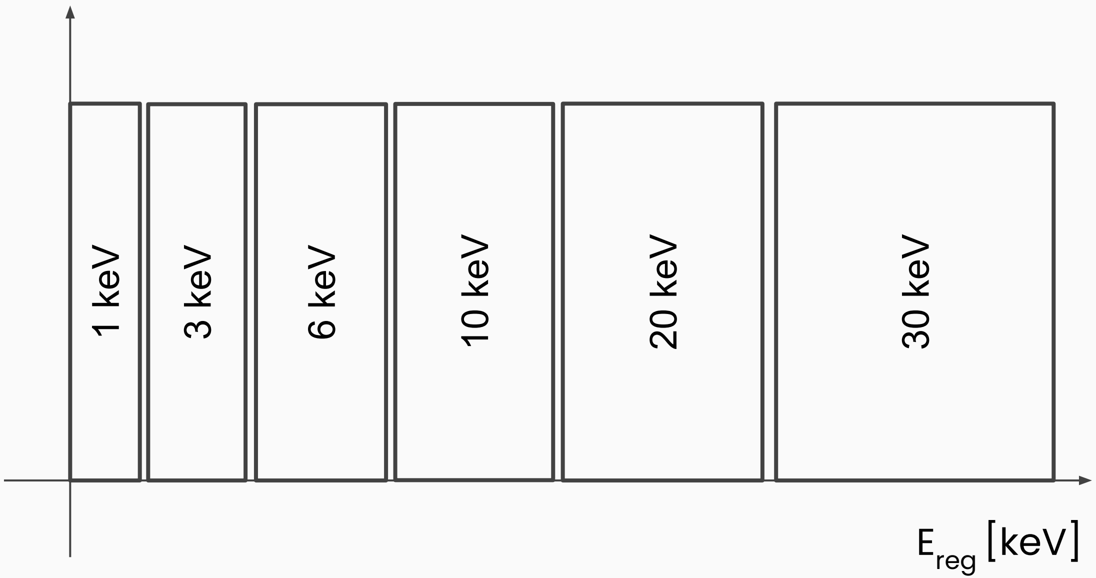
Introducing discrete energy

MAE=0.84

evaluated on test

To **absorb any bias** at certain energies due to **regressor saturation**, we applied a **discrete binning on the kinetic energy range**.

Each bin is **symmetric** and **centered** on the kinetic energy values we expect.



Major doubts after the challenge

Data augmentation

Random image rotation

```
transform=transforms.Compose(
    [transforms.ToTensor(),
    transforms.CenterCrop(120),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomRotation(degrees=(-90, 90)),
    transforms.RandomVerticalFlip(p=0.5),
    ]
```

We tested different transformation but always obtained **worst** performances.

Do we expect to lose any spatial information by rotating images?

Image normalization

Max pixel level in train: 244

transforms.Lambda(lambda x : x/244)

We didn’t observe any improvement during training and we obtained **worst** performances.

Do we expect that keeping the absolute pixel levels should help the model?

Different metrics values during training/validation/test

```
Global seed set to 666
Epoch 0:  95%|                    | 200/211 [00:30<00:01,  6.64it/s, loss=0.0743, v_num=116M
etric val_regression_loss improved. New best score: 0.389                    | 54/54 [00:08<00:00,  6.30it/s]
Epoch 1:  95%|                    | 200/211 [00:30<00:01,  6.60it/s, loss=0.0608, v_num=116M
etric val_regression_loss improved by 0.161 >= min_delta = 0.0. New best score: 0.228                    | 54/54 [00:07<00:00,  7.13it/s]
Epoch 2:  57%|                    | 120/211 [00:19<00:14,  6.20it/s, loss=0.0529, v_num=116
```

In this case we launched a training with **L1 loss function**.

One order of magnitude between L1 loss values on training and test sample.
Fixed reducing learning rate.

```
tric val_regression_loss improved by 0.031 >= min_delta = 0.0. New best score: 0.241                    | 54/54 [00:07<00:00,  7.56it/s]
poch 2:  95%|                    | 200/211 [00:29<00:01,  6.74it/s, loss=0.229, v_num=118M
tric val_regression_loss improved by 0.008 >= min_delta = 0.0. New best score: 0.233                    | 54/54 [00:07<00:00,  7.36it/s]
poch 8:  95%|                    | 200/211 [00:29<00:01,  6.68it/s, loss=0.145, v_num=118M
tric val_regression_loss improved by 0.007 >= min_delta = 0.0. New best score: 0.226                    | 54/54 [00:07<00:00,  7.69it/s]
poch 9:  95%|                    | 200/211 [00:30<00:01,  6.49it/s, loss=0.127, v_num=118M
tric val_regression_loss improved by 0.006 >= min_delta = 0.0. New best score: 0.220                    | 54/54 [00:07<00:00,  7.29it/s]
poch 10:  95%|                    | 200/211 [00:29<00:01,  6.79it/s, loss=0.118, v_num=118M
tric val_regression_loss improved by 0.107 >= min_delta = 0.0. New best score: 0.113                    | 54/54 [00:07<00:00,  7.57it/s]
```

L1 loss values on validation sample should, in some way (?), be related to the expected MAE value on the test set.