MLHEP21: Bronze Medal Solution to the 2nd Competition

Santeri Laurila (CERN)

MLHEP 2021
30 July 2021

A Close Race

- Three components of my solution (tweaking the baseline model):
  - Making the model "deeper": more parameters and more nonlinearities
  - Improved training
  - A bit of luck :-)
- Code available at GitHub: https://github.com/slaurila/mlhep-2021-baseline/tree/final_submission_santeri

#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	—	Polina Simkina			0.36406	12	2d
2	▲1	anyon2D(*D)			0.81665	10	1d
3	▼1	anyon2D(*A)			0.82910	19	2d
4	—	anyon2D(*C)			1.48251	10	2d
5	▲1	Santeri Laurila			1.48825	2	1d
6	▲2	BlueWatermelon			1.51353	5	2d
7	▼2	CosmosDong			1.53151	12	2d
8	▼1	DM_hunters			1.54290	16	2d
9	—	AKA	anyon2D(*A)		1.56053	7	1d

Model

```python
class SimpleConv(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.ReLU(),
                    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.drop_out, self.fc1,
            )
```

```python
class SanteriConv(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.ReLU(),
                    nn.MaxPool2d(kernel_size=2, stride=2),
                    nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1),
                    nn.BatchNorm2d(32),
                    nn.ReLU(),
                    nn.MaxPool2d(kernel_size=2, stride=2),
                    nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
                    nn.BatchNorm2d(64),
                    nn.ReLU(),
                    nn.MaxPool2d(kernel_size=2, stride=2),
                    nn.Flatten(),
                )

        self.drop_out = nn.Dropout(p=0.3)

        self.fc1 = nn.Linear(14400, 1000)
        self.fc2 = nn.Linear(1000, 2)  # for classification
        self.fc3 = nn.Linear(1000, 1)  # for regression
        self.relu = nn.ReLU()

        self.stem = nn.Sequential(
            self.layer1, self.drop_out, self.fc1, self.relu,
            )
```

- https://github.com/slaurila/mlhep-2021-baseline/blob/final_submission_santeri/idao/model.py#L160

Training

```
[DATA]
DatasetPath = /home/user/share/competition
Extension = png

[TRAINING]
ClassificationEpochs = 2
RegressionEpochs = 2
UseGPU = True
ModelParamsSavePath = ./checkpoints
BatchSize = 128

```

```
[DATA]
DatasetPath = /home/user/share/competition
Extension = png

[TRAINING]
ClassificationEpochs = 5
RegressionEpochs = 5
UseGPU = True
ModelParamsSavePath = ./checkpoints
BatchSize = 64

```

- https://github.com/slaurila/mlhep-2021-baseline/blob/final_submission_santeri/config.ini
- Increased the **number of epochs** to squeeze everything out of the model (3-4 might have been enough)
- Decreased the **batch size** (more "noisy" gradient estimates) to avoid overfitting and ensure that the model generalizes to the new data set
- Used the automatic **learning rate** finder provided by PyTorch Lightning, instead of constant rate of 1e-3
  - trainer = pl.Trainer(auto_lr_find=True) [code]
- Did not optimize everything systematically due to time constraints – expert insights are welcome!