Common Task 1: Electron/photon classification (PyTorch)

In [1]:

```
!pip install Lightning -q
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packag
e manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

In [2]:

```
# Import necessary packages
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
import matplotlib.pyplot as plt
import lightning.pytorch as pl
import h5py
import urllib
import math
from tqdm.auto import tqdm
```

In [3]:

```
DATA_DIR = '/kaggle/input/electron-vs-photons-ml4sci'
```

In [4]:

```
# Define the filenames and batch size
electrons_filename = f'{DATA_DIR}/SingleElectronPt50_IMGCROPS_n249k_RHv1.hdf5'
photons_filename = f'{DATA_DIR}/SinglePhotonPt50_IMGCROPS_n249k_RHv1.hdf5'
batch_size = 64
num_features = 32


class Dataset(torch.utils.data.Dataset):
    def __init__(self, electrons_filename, photons_filename, batch_size, frac=0.032):
        self.batch_size = batch_size
        self.frac = frac
        self.electrons_file = h5py.File(electrons_filename, 'r')
        self.photons_file = h5py.File(photons_filename, 'r')
        num_electrons = self.electrons_file['X'].shape[0]
        num_photons = self.photons_file['X'].shape[0]
        self.num_batches = max(num_electrons, num_photons) // self.batch_size
```

```python
    def __len__(self):
        dataset_size = math.ceil((self.electrons_file['X'].shape[0])/self.batch_size)
        train_size = math.ceil(dataset_size * self.frac)
        return train_size

    def __getitem__(self, idx):
        # Load a batch of electrons and photons
        electrons_x = self.electrons_file['X'][idx*self.batch_size:(idx+1)*self.batch_size]
        electrons_y = self.electrons_file['y'][idx*self.batch_size:(idx+1)*self.batch_size]

        photons_x = self.photons_file['X'][idx*self.batch_size:(idx+1)*self.batch_size]
        photons_y = self.photons_file['y'][idx*self.batch_size:(idx+1)*self.batch_size]

        # Combine the data
        batch_x = np.concatenate([electrons_x, photons_x])
        batch_y = np.concatenate([electrons_y, photons_y])

        # expand dims of batch_y
        batch_y = np.expand_dims(batch_y, axis=1)

        # shuffle it
        perm = np.random.permutation(len(batch_x))
        batch_x = batch_x[perm]
        batch_y = batch_y[perm]

        # Convert the data to pytorch tensors and yield it
        return torch.tensor(batch_x, dtype=torch.float32), torch.tensor(batch_y, dtype=torch.int32)


train_dataset = Dataset(electrons_filename, photons_filename, batch_size//2)
train_loader = torch.utils.data.DataLoader(train_dataset, shuffle=True)
```

In [5]:

```python
class Model(pl.LightningModule):

    def __init__(self):
        super().__init__()
        self.classifier = nn.Sequential(
            nn.Conv2d(2, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(64, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(64, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Flatten(),
            nn.Linear(1024, 64),
            nn.ReLU(),
            nn.Linear(64, 2),
        )

    def forward(self, x):
        return self.classifier(x)

    def training_step(self, batch, batch_idx):
        x, y = batch
        x = x.squeeze(0).permute(0, 3, 1, 2)
        y = y.squeeze(0).squeeze(-1).type(torch.long)
        y_hat = self(x)
        loss = F.cross_entropy(y_hat, y)
        return loss

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters())
```

In [6]:

```
trainer = pl.Trainer(max_epochs=10, accelerator="gpu")
model = Model()

trainer.fit(model, train_dataloaders=train_loader)
```

```
INFO: GPU available: True (cuda), used: True
INFO: TPU available: False, using: 0 TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO: HPU available: False, using: 0 HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:
  | Name       | Type       | Params
------------------------------------------
0 | classifier | Sequential | 195 K
------------------------------------------
195 K     Trainable params
0         Non-trainable params
```

```
195 K     Total params
0.782     Total estimated model params size (MB)

INFO: `Trainer.fit` stopped: `max_epochs=10` reached.
```