

PyTorch

Training a classifier

Setting the environment

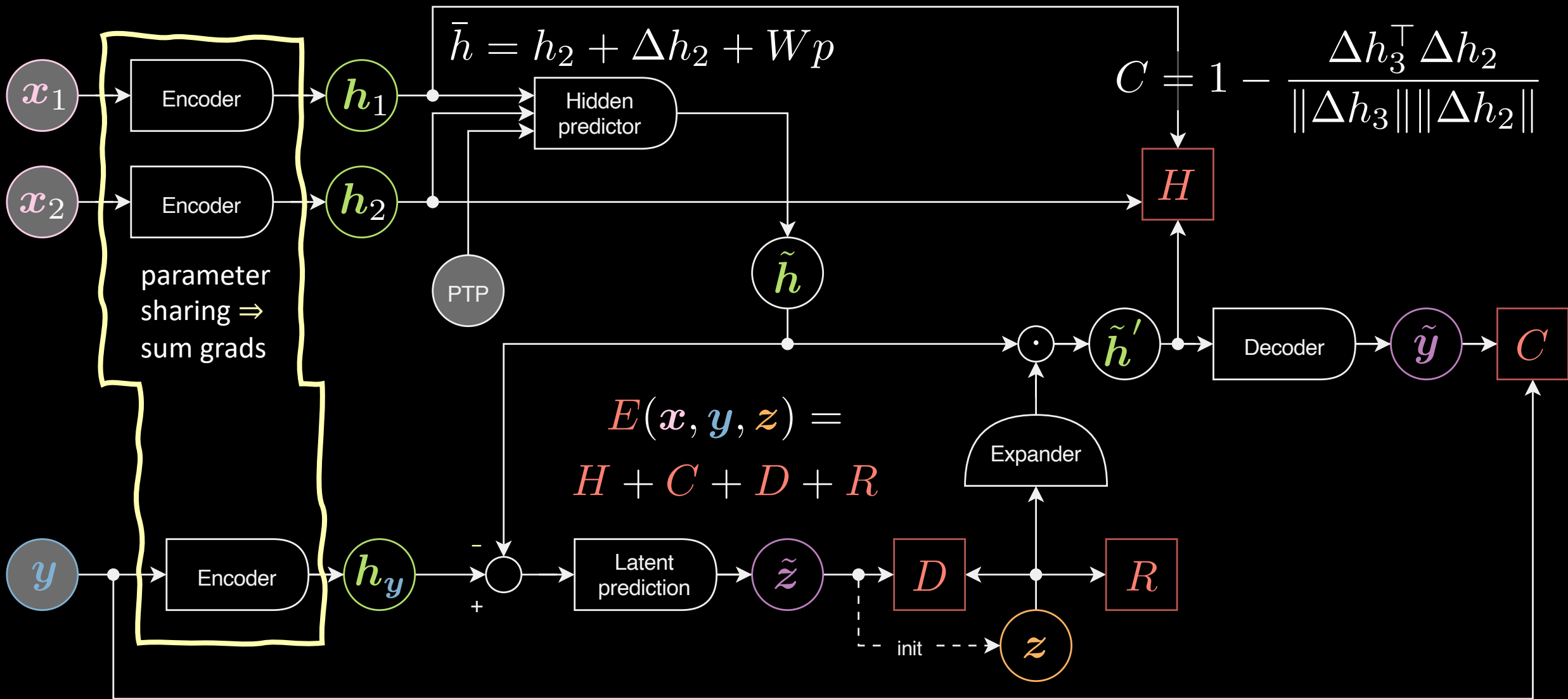
- `import torch`
- `from torch import nn, optim`
- `device = torch.device(...) # CPU or GPU or TPU`
- `model = nn.Sequential(...).to(device)`
- `C = nn.CrossEntropyLoss() # cost definition`
- `optimiser = optim.SGD(model.parameters())`

Training loop 5 steps

for (x, y) in dataset:

1. $\tilde{y} = \text{model}(x)$ # generate a prediction
2. $L = F = C(\tilde{y}, y)$ # compute the loss
3. `optimiser.zero_grad()` # zero ∇params
4. `L.backward()` # compute & accumulate ∇params
5. `optimiser.step()` # step in towards $-\nabla \text{params}$
... # logging

Why backward accumulates ∇params (I)



Why backward accumulates ∇params (II)

```
optimiser.zero_grad()
```

```
 $\tilde{y}_1$  = model( $x_1$ )
```

```
L = F = C( $\tilde{y}_1$ ,  $y_1$ )
```

```
L.backward() # compute & accumulate  $\nabla \text{params}_1$ 
```

```
 $\tilde{y}_2$  = model( $x_2$ )
```

```
L = F = C( $\tilde{y}_2$ ,  $y_2$ )
```

```
L.backward() # compute & accumulate  $\nabla \text{params}_2$ 
```

```
optimiser.step() # step towards  $-(\nabla_1 + \nabla_2)$ 
```

Node and sum modules

