

# Deep Model Training

---

## IPEO Exercise 8

Devis Tuia  
Emanuele Dalsasso  
Manon Béchaz  
Robin Zbinden

EPFL-ECEO

November 7, 2025

In three exercises, we will focus on  
**Deep Learning with Pytorch**

- IPEO exercise 6 Introduction to Pytorch with Logistic Regression
- IPEO exercise 7 Image Classification with CNNs
- IPEO exercise 8 **Model Training and Regularization**
- IPEO exercise 9 Semantic Segmentation



Google Colaboratory

check `scitas_instructions.pdf`check `setup_gpus.pdf`you can check if you have a GPU environment with `nvidia-smi`

Fri Oct 21 11:30:56 2022

NVIDIA-SMI 460.32.03 Driver Version: 460.32.03 CUDA Version: 11.2									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	MIG	M.	
0	Tesla T4	Off	00000000:00:04.0	Off				0	
N/A	43C	P8	9W / 70W	3MiB / 15109MiB	0%	Default	N/A		

```
device = "cuda" # or "cpu"  
  
tensor = tensor.to(device)  
model = model.to(device)  
  
# back to cpu  
tensor.cpu() # or .to("cpu")
```

```
In [19]: print("cuda")
          %timeit training_step(batch, model, optimizer, device="cuda")

          print("cpu")
          %timeit training_step(batch, model, optimizer, device="cpu")

cuda
77.1 ms ± 337 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
cpu
823 ms ± 19.1 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

setup before each epoch

```
#require
model
dataloader
optimizer

# set model to training mode
model.train()

# cuda means GPU
device = "cuda"

# move model to GPU
model = model.to(device)
```

training step

```
# training step
for batch in dataloader:
    optimizer.zero_grad()

    x, y = batch

    # move data to GPU
    x = x.to(device)
    y = y.to(device)

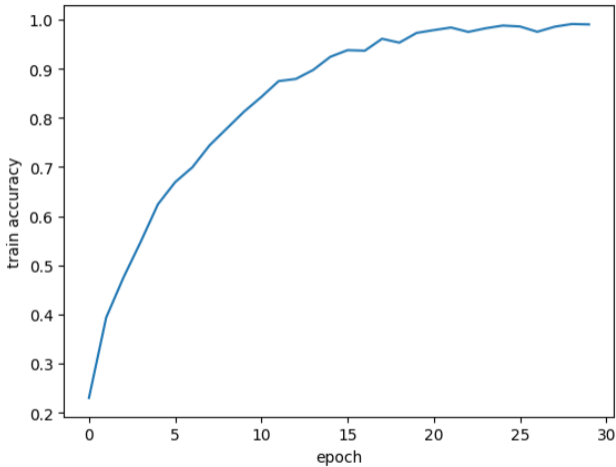
    # predict
    y_logit = model(x)

    # calculate the loss
    loss = criterion(y_logit, y)

    # backpropagation
    loss.backward()

    # model update
    optimizer.step()
```

```
epoch 0; trainloss 2.48, train accuracy 28.35%
epoch 1; trainloss 1.88, train accuracy 39.24%
epoch 2; trainloss 1.59, train accuracy 49.29%
epoch 3; trainloss 1.36, train accuracy 57.59%
epoch 4; trainloss 1.18, train accuracy 63.16%
epoch 5; trainloss 1.05, train accuracy 68.80%
epoch 6; trainloss 0.90, train accuracy 73.07%
epoch 7; trainloss 0.85, train accuracy 73.31%
epoch 8; trainloss 0.72, train accuracy 78.85%
epoch 9; trainloss 0.62, train accuracy 82.54%
epoch 10; trainloss 0.56, train accuracy 84.23%
epoch 11; trainloss 0.51, train accuracy 85.02%
epoch 12; trainloss 0.48, train accuracy 86.47%
epoch 13; trainloss 0.42, train accuracy 88.21%
epoch 14; trainloss 0.33, train accuracy 91.90%
epoch 15; trainloss 0.29, train accuracy 93.67%
epoch 16; trainloss 0.26, train accuracy 93.41%
epoch 17; trainloss 0.21, train accuracy 95.33%
epoch 18; trainloss 0.16, train accuracy 96.84%
epoch 19; trainloss 0.16, train accuracy 97.02%
epoch 20; trainloss 0.13, train accuracy 97.39%
epoch 21; trainloss 0.15, train accuracy 96.97%
epoch 22; trainloss 0.13, train accuracy 97.31%
epoch 23; trainloss 0.10, train accuracy 98.39%
epoch 24; trainloss 0.10, train accuracy 98.42%
epoch 25; trainloss 0.08, train accuracy 98.89%
epoch 26; trainloss 0.09, train accuracy 98.02%
epoch 27; trainloss 0.09, train accuracy 98.47%
epoch 28; trainloss 0.05, train accuracy 99.53%
epoch 29; trainloss 0.05, train accuracy 99.45%
```



# EPFL 30% worse validation accuracy!

```
valloss 1.43, val accuracy 66.59
```

- change the model architecture (check out other torchvision models)
- start from pre-trained weight initializations
- introduce `weight_decay` to the optimizer
- change the optimizer (e.g. to Adam)
- do data augmentation (see last exercise)

1. load UC Merced dataset module (provided in the moodle)
2. load a ResNet18 model from torchvision.models
3. train the model on the training set
4. test on the validation set
5. fight overfitting with regularization and earn your cookies

We added a pytorch lighting code at the end to train models efficiently with few lines of code.

until next week implement different regularizations and post your result in slack.

- test accuracy  $> 80\%$ , we will bring one cookie bag
- test accuracy  $> 85\%$ , we will bring two cookie bags
- test accuracy  $> 90\%$ , we will bring three cookie bags
- test accuracy  $> 95\%$ , we won't believe you

Structuring your code in multiple files is good practice to keep your code clean.

- `UCMerced_module.py`
  - `class UCMerced(Dataset)`
  - `def function ...`
  - `variable = "something"`
- `notebook.ipynb (or main.py)`
  - `from UCMerced_module import UCMerced`