Master in Computer Vision Barcelona

M5 Project: Cross-modal Retrieval

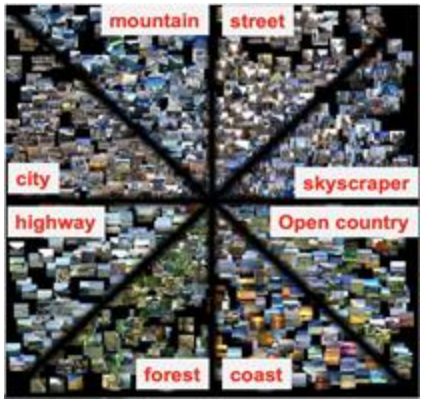
Week 1. Introduction to Pytorch

Rubèn Pérez Tito
rperez@cvc.uab.cat

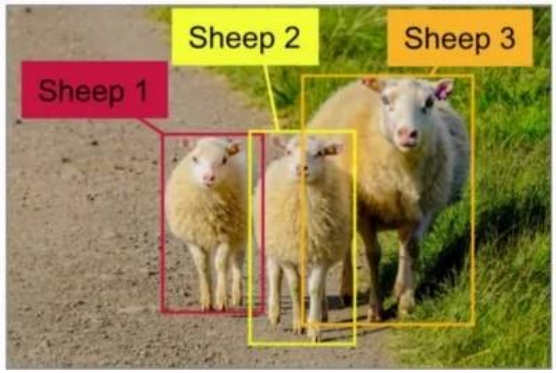
Ernest Valveny
ernest@cvc.uab.cat

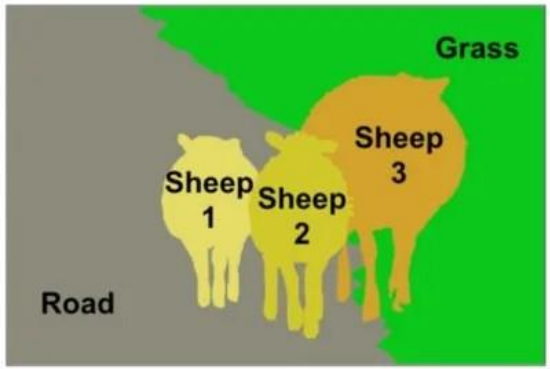
M5 Project Overview

Image classification



Object detection and segmentation





Object detection and segmentation limitations

M5 Project Overview

Image Retrieval

Query

Retrieved examples




...


M5 Project Overview

Cross-modal retrieval

Image to Text

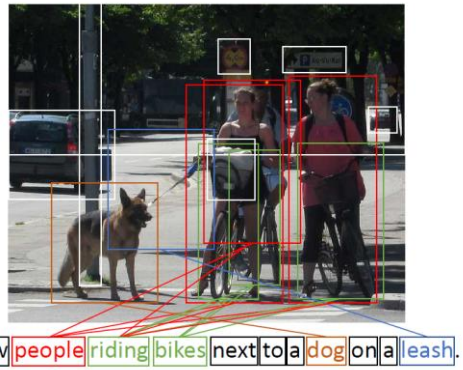
(c)

1:A female runner dressed in blue athletic wear is running in a competition , while spectators line the street . ✓
2:A lady dressed in blue running a marathon . ✓
3:A young woman is running a marathon in a light blue tank top and spandex shorts . ✓
4:A lady standing at a crosswalk . ✗
5:A woman who is running , with blue shorts . ✓

Text to Image

Query: *A man riding a motorcycle is performing a trick at a track .*

Sub-objective: **Object detection**



M5 Project Stages and Schedule

Week 1 March 6-12	**P1: Introduction to Pytorch - Image Classification**
Week 2 March 13-19	**P2 & P3: Object Detection, Recognition and Segmentation**
Week 3 Marh 20 - 26	
Week 4 March 27 – April 3	**P4: Image Retrieval**
EASTER	
Week 5 April 17 - 23	**P5: Cross-modal Retrieval** **Deliverable: Report on object Detection and Segmentation, first version**
Week 6 April 24	**Deliverable: Presentation** **Deliverable: Report on object Detection and Segmentation, final version**

M5 Project Evaluation

The **final mark** of the project will be obtained as a combination of the following items:

- 50% - Weekly submissions (code, results, discussion and analysis, ...)
- 15% - Report (30% first version, 70% final version)
- 15% - Final presentation
- 20% - Individual mark through intra-group evaluation

M5 Project Stages and Schedule

Week 1 March 6-12	**P1: Introduction to Pytorch - Image Classification**
Week 2 March 13-19	**P2 & P3: Object Detection, Recognition and Segmentation**
Week 3 Marh 20 - 26	
Week 4 March 27 – April 3	**P4: Image Retrieval**
EASTER	
Week 5 April 17 - 23	**P5: Cross-modal Retrieval** **Deliverable: Report on object Detection and Segmentation, first version**
Week 6 April 24	**Deliverable: Presentation** **Deliverable: Report on object Detection and Segmentation, final version**

M5 – P1: Introduction to Pytorch

- In M5 Project, we will use **Pytorch** framework instead of Keras for object detection and segmentation.

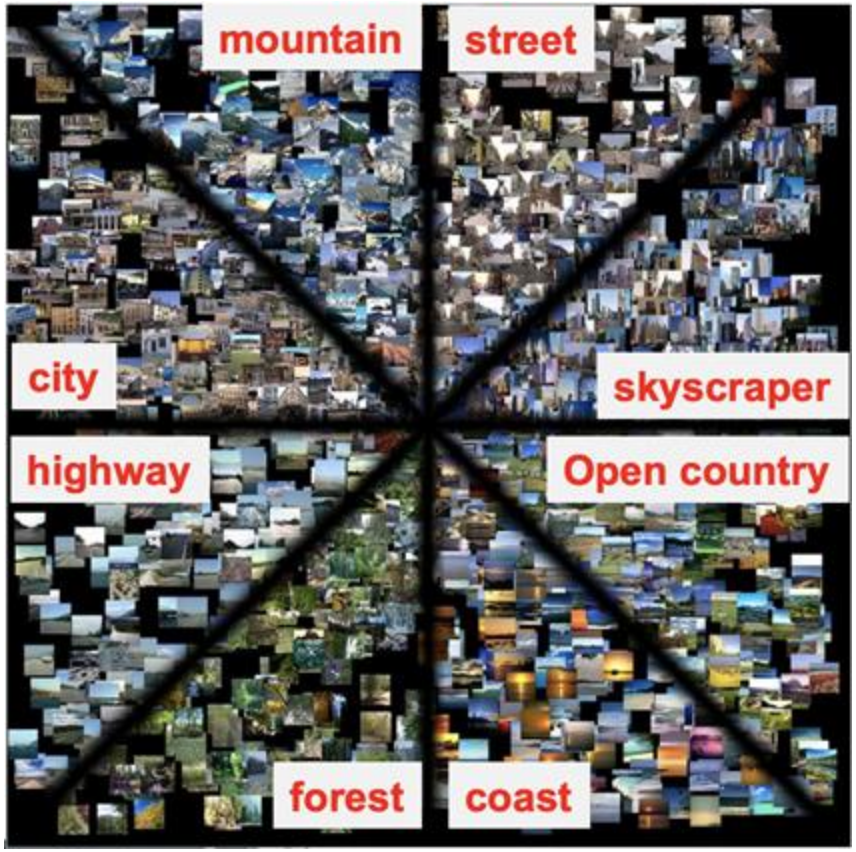
- We will see frameworks like Detectron2, which is a research platform for object detection and segmentation in Pytorch
  - https://github.com/facebookresearch/detectron2
  - More details about the project next week (W2)

- **First task**: Implementing the final model from M3 (Image Classification) in Pytorch

M3 GOAL REMINDER

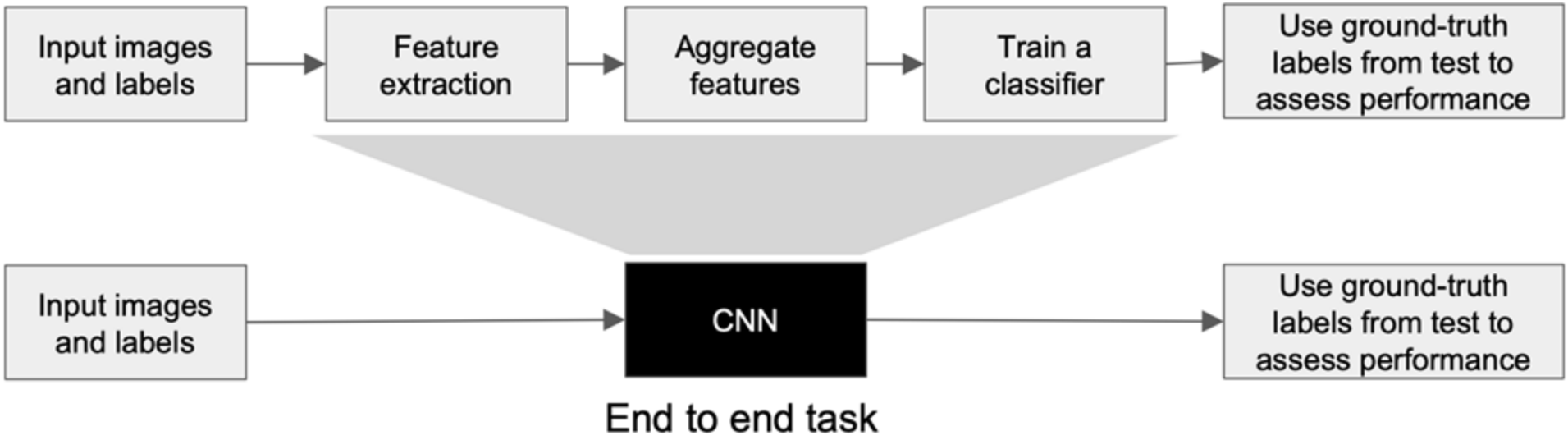
The goal of M3 was to learn the techniques for image classification:

- Handcrafted and learned features



M3 GOAL REMINDER

M3 tasks for last weeks was to train a CNN from scratch...



Machine learning for image classification:

Data driven methods: Deep Convolutional Networks: 3 sessions

From hand-crafted to learnt features

Fine tuning of pre-trained CNNs

Training a CNN from scratch

M3 GOAL REMINDER

… and you did these tasks using Keras

```
# create model
model = Sequential()
model.add(Dense(12, input_dim=8, init='uniform', activation='relu'))
model.add(Dense(8, init='uniform', activation='relu'))
```

W3-5

```
inputs = Input(shape=None))
x = Dense(12, init='uniform', activation='relu', name='fc1')(x)
x = Dense(8, init='uniform', activation='sigmoid', name= 'predictions')(x)
model = Model(inputs, x, name='example')
```

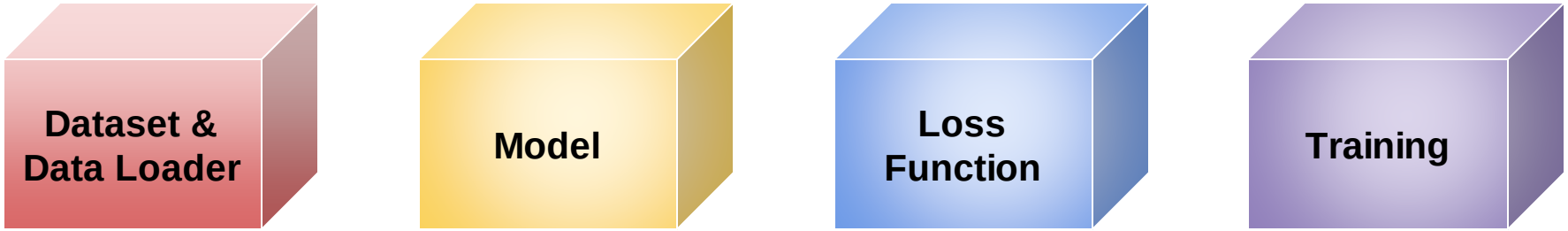
```
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Fit the model
model.fit(X, Y, nb_epoch=150, batch_size=10)
# evaluate the model
scores = model.evaluate(X, Y)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

W3-4

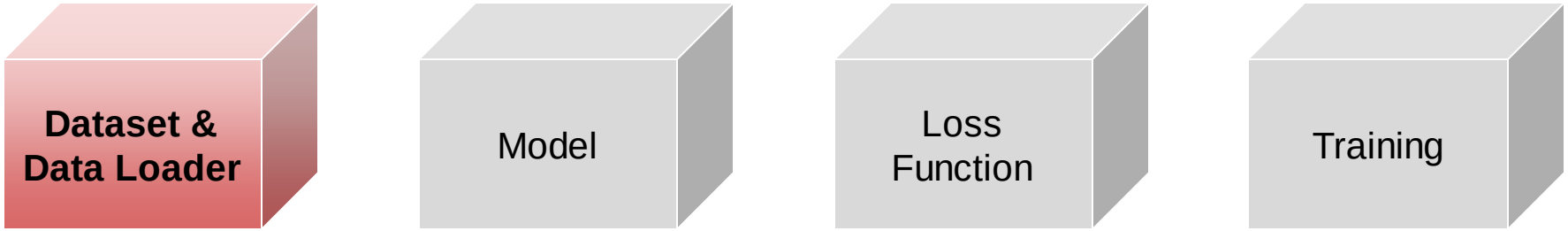
```
# predict with the model
features = model.predict(X)
```

W3-4

M5 – P1: Introduction to Pytorch



M5 – P1: Introduction to Pytorch



M5 – P1: Introduction to Pytorch

Dataset

- Abstract class representing a Dataset
- Stores the samples and their corresponding labels
- Dataset is independent to the model training*

DataLoader

- Wraps an iterable around the Dataset to enable easy access to the samples.

Pytorch Dataset

- torch.utils.data.Dataset
  - Abstract class representing a Dataset
  - class MyDataset(Dataset):
    - def __init__(self):
    - def __len__(self):
    - def __getitem__(self, index):

__init__(self):
If the dataset is small, you can load the samples/images here.
If the dataset is too big, you only prepare the path to the samples.
`self.data = …`

__len__(self):
Returns the dataset length.
Usually one line of code:
`return len(self.data)`

__getitem__(self, idx):
Returns a dataset's sample.
If the samples were not loaded before, they are loaded in this function.
Additionally, any kind of preprocessing or transformation are applied here.
`return load(self.data[idx])`

Dataloader

- Wraps an iterable around the Dataset to enable easy access to the samples.

```
DataLoader(dataset, batch_size=1, shuffle=False, sampler=None,
           batch_sampler=None, num_workers=0, collate_fn=None,
           pin_memory=False, drop_last=False, timeout=0,
           worker_init_fn=None, *, prefetch_factor=2,
           persistent_workers=False)
```

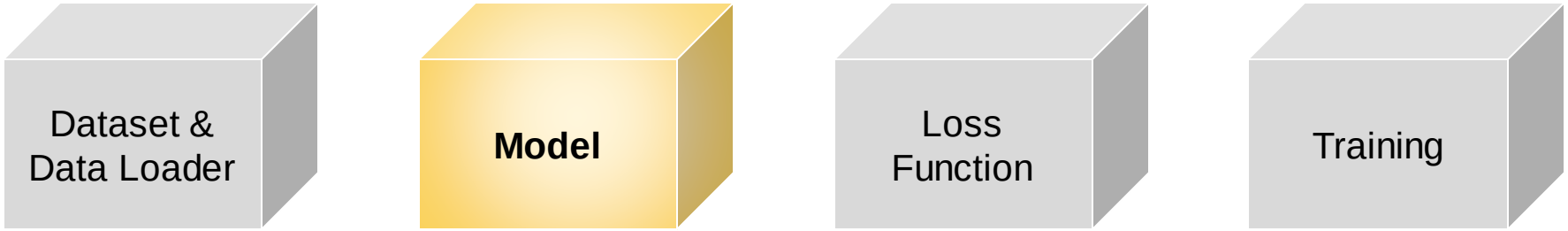
Torchvision

Package which consists of popular datasets, model architectures, and common image transformations for computer vision.

- MODEL ZOO: AlexNet, VGG, ResNet, Inception v3, …
- Transforms: CenterCrop, Normalize, RandomCrop, Flip, VerticalFlip, etc.
  - You can "**append**" them together with Compose.

```
transforms = torch.nn.Compose(
    transforms.CenterCrop(10),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
)
```

M5 – P1: Introduction to Pytorch



Model architecture

```
class MLP(torch.nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(MLP, self).__init__()
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.num_classes = num_classes
        self.fc1 = torch.nn.Linear(self.input_size, self.hidden_size)
        self.relu = torch.nn.ReLU()
        self.fc2 = torch.nn.Linear(self.hidden_size, self.num_classes)
        self.softmax = torch.nn.Softmax(dim=1)

    def forward(self, x):
        hidden = self.fc1(x)
        relu = self.relu(hidden)
        output = self.fc2(relu)
        output = self.softmax(output)
        return output
```

Using GPU

- Setting your GPU device
  - torch.cuda.set_device(device=gpu_id)

- Converting your model to CUDA tensors:
  - model.cuda()

- Converting your inputs and targets to CUDA tensors:
  - inputs = inputs.cuda()
  - targets = targets.cuda()

Grey notes indicates that you won't need this when you work in the MCV cluster.
However, it's important to take into account when working in other environments.

Using GPU

- Setting your GPU device
  - device = 'cpu' || 'cuda' (default) || 'cuda:0' (gpu 0)

- Converting your model to CUDA tensors:
  - model.to(device)

- Converting your inputs and targets to CUDA tensors:
  - inputs = inputs.to(device)
  - targets = targets.to(device)

Grey notes indicates that you won't need this when you work in the MCV cluster.
However, it's important to take into account when working in other environments.

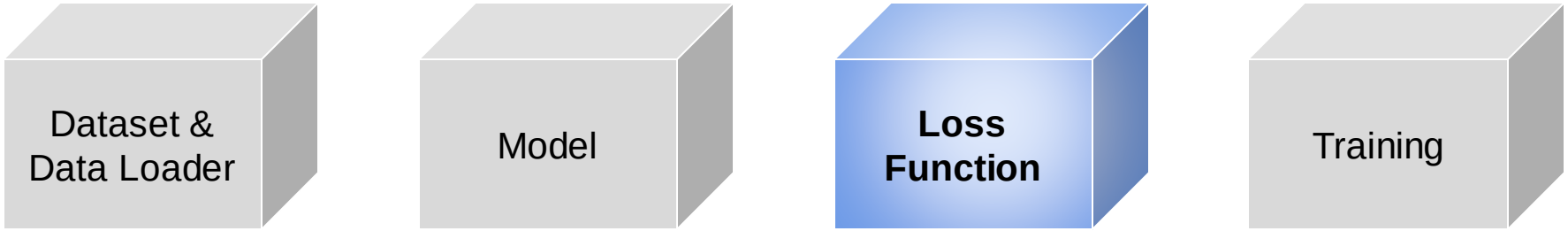
Using GPU

- Checking GPU is available
  - torch.cuda.is_available()

- Numpy cannot work on CUDA Tensors, you need to send them to CPU before performing any operation on numpy:
  - var_cpu = var_gpu.cpu()

- Once all required operations on numpy have been done, remember to transform your data again to CUDA Tensors:
  - var_gpu = var_cpu.cuda()

M5 – P1: Introduction to Pytorch



M5 – P1: Introduction to Pytorch

Loss functions

- Pytorch loss functions
  - MSE
  - CrossEntropyLoss
  - BCELoss
  - …

Optimizer

- Pytorch optimizers
  - SGD
  - Adam
  - …

```
import torch.nn as nn
import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

M5 – P1: Introduction to Pytorch

Loss functions

- Pytorch loss functions
  - MSE
  - CrossEntropyLoss
  - BCELoss
  - …

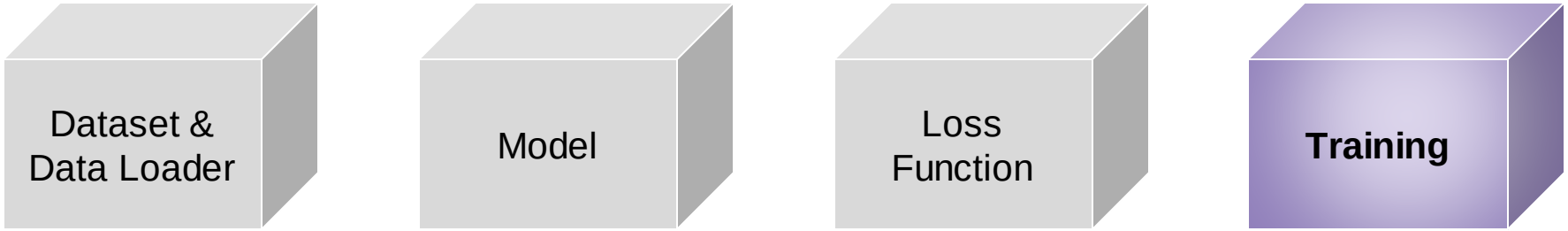
Optimizer

- Pytorch optimizers
  - SGD
  - Adam
  - …

```
for input, target in dataset:
    optimizer.zero_grad()
    output = model(input)
    loss = loss_fn(output, target)
    loss.backward()
    optimizer.step()
```

net = model
criterion = loss_fn

M5 – P1: Introduction to Pytorch



Training your network

```
for epoch in range(2):  # loop over the dataset multiple times
    net.train()
    running_loss = 0.0
    for i, data in enumerate(trainloader):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

Testing your network

```
net.eval()
with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs.data, 1)
```

M5 – P1: Introduction to Pytorch

Monitoring your training:

- With Tensorboard
  - Pytorch Tensorboard
  - Installation in server:
    - conda install -c conda-forge tensorboardx
  - How to run it:
    - tensorboard --logdir=/path/to/summary/file --port XXXX (--bind_all)

Monitoring your training:

- With Tensorboard

```
from torch.utils.tensorboard import SummaryWriter
import numpy as np

writer = SummaryWriter()

for n_iter in range(100):
    writer.add_scalar('Loss/train', np.random.random(), n_iter)
    writer.add_scalar('Loss/test', np.random.random(), n_iter)
    writer.add_scalar('Accuracy/train', np.random.random(), n_iter)
    writer.add_scalar('Accuracy/test', np.random.random(), n_iter)
```

Monitoring your training:

- With Weights and Bias (WandB)

```
pip install wandb
```

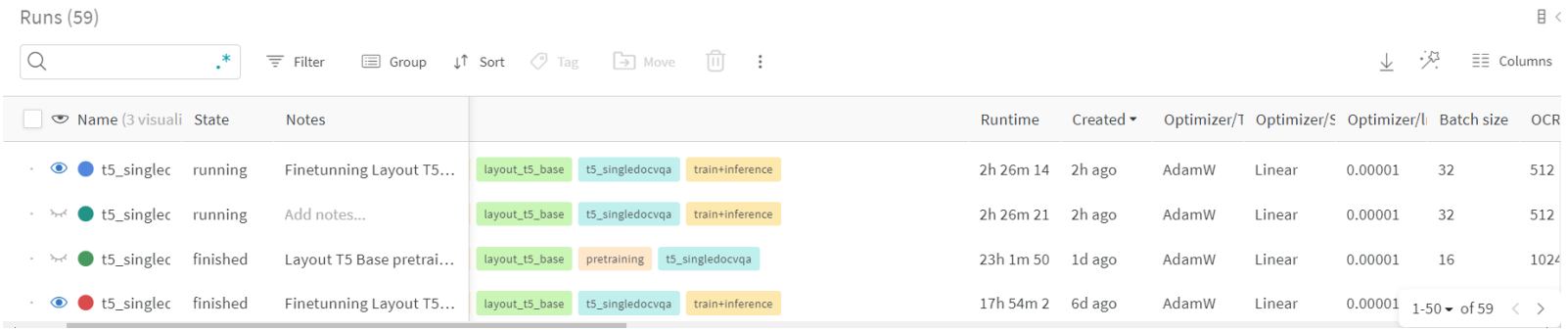
```
import wandb as wb

writer = wb.init(name="My experiment 1", project="P1",
                 config={"bs": 2, "lr": 1e-5}, entity="G1")

writer.log({'Loss/train': net_loss, 'Accuracy/train': net_acc}, n_iter)
```

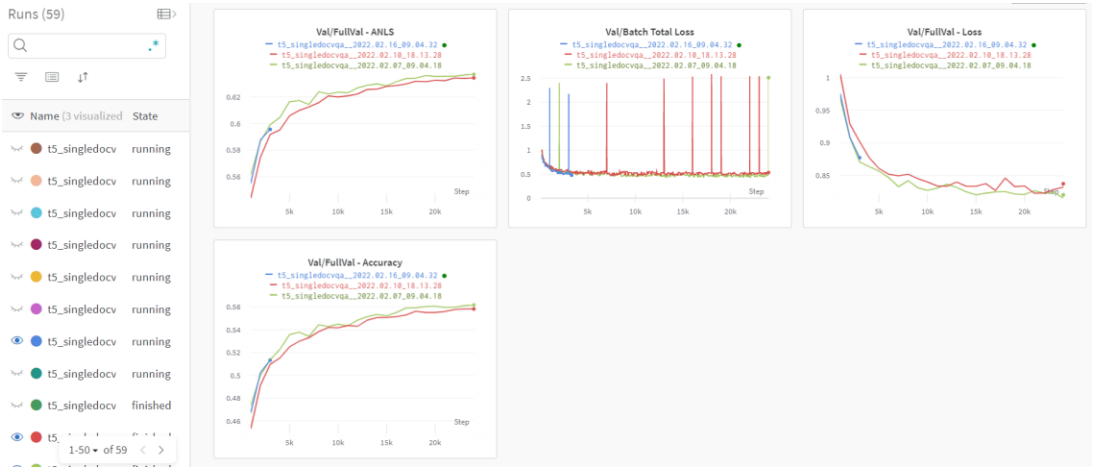
```
writer = wb.init(name="My experiment 1", project="P1",
                 config={"bs": 2, "lr": 1e-5}, entity="G1",
                 sync_tensorboard=True)
```

M5 – P1: Introduction to Pytorch

Monitoring your training:

- With Weights and Bias (WandB)





M5 – P1: Introduction to Pytorch

GPU Cluster use:

- Cluster information
  - Host: 158.109.75.50
  - SSH Port: 22
  - Username: group{01,02,...,08}
  - Passwd: {01,02,...,08}group

- Code edition/management:
  - Edit directly in the server. Connecting with MobaXterm you can right-click and edit with local editor.
  - Edit in local, send the new files to the server.
  - Edit in local, push/pull with GitHub

M5 – P1: Introduction to Pytorch

GPU Cluster use:

- Job management:
  - SLURM
  - /home/example/Graphics_DCC_Cluster_User's_Guide_for_MCV_V2.pdf
    - Cluster information is deprecated (IP, port, etc.)
  - /home/example/ mtgpulow.sh | tgpu.sh | mtgpuhigh.sh
  - Partition (-p) and QOS (-q) especially important. Check documentation. Use case C is recommended.
  - **IMPORTANT**: Save temporal results (weights) during training every N iterations/epochs. Always can happen something that breaks, kills your process.

GPU Cluster use:

- When the job is launched:
  - Check constantly during the first ~2 minutes everything is OK.
    - sinfo, squeue, tensorboard/wandb logger
    - watch –n **2** nvidia-smi

Details on tasks, deliverables, and marks for this week

Week 1. Introduction to Python

M5 Project Stages and Schedule

Week 1 March 6-12	**P1: Introduction to Pytorch - Image Classification**
Week 2 March 13-19	**P2 & P3: Object Detection, Recognition and Segmentation**
Week 3 Marh 20 - 26	
Week 4 March 27 – April 3	**P4: Image Retrieval**
EASTER	
Week 5 April 17 - 23	**P5: Cross-modal Retrieval** **Deliverable: Report on object Detection and Segmentation, first version**
Week 6 April 24	**Deliverable: Presentation** **Deliverable: Report on object Detection and Segmentation, final version**

M5 – P1: Introduction to Pytorch – Tasks

Week 1: Introduction to Pytorch

Tasks

(a) Form groups.
(b) Install and setup the development framework.
(c) Set Up collaborative tools.
(d) Understand Pytorch framework.
(e) Implement Image Classification network from M3 in Pytorch
(f) Compute loss graphs and compare them with yours from Keras
(g) Compute accuracy graphs and compare them with yours from Keras
(h) Extras

Deliverable (for next week)

- **Github** repository with readme.md (Members of the group, code explanation & instructions)
- Presentation with all items listed in the tasks under the **Project presentation** title.
- **One summary slide** at the end of your presentation.

M5 – P1: Introduction to Pytorch – Tasks

Task (a): **Form teams**

- There will be a maximum of 8 groups.
- No more than 4 people per group.
- Follow the link in Campus Virtual to fill in information about the group.

Important

Each member of the team must contribute equally to each assignment.

We will ask you who did what.

M5 – P1: Introduction to Pytorch – Tasks

Task (b): **Install and setup the development framework**

- If you use the master GPU cluster everything should be already installed (basic software and datasets). **Check it!**
  - Host: 158.109.75.50
  - SSH Port: 22
  - Username: group{01,02,...,10}
  - Passwd: *****

- Browse images in the dataset directory in the GPU cluster
  - /home/mcv/datasets/MIT_split/
  - This is the same dataset as used in M3

M5 – P1: Introduction to Pytorch – Tasks

Task (c): **Setup collaboration tools**

- **GitHub** repository for the code management.
  - Create your own github repository (one per group)
  - Structure the github according to weeks.

- **Overleaf**: Project for the reports (next week tasks)
  - You can use CVPR paper format

- **PPT / Google Slides:** Project for the presentations.

M5 – P1: Introduction to Pytorch – Tasks

- **All the deliverables should be accessible through your group GitHub project.**
- We recommend you to make public your GitHub project so you can share your results with other groups.
- If you decide to make it private, invite Rubèn and Ernest (rubenpt91, evalveny) as contributors.
- Add the link to your GitHub project in the link with information about groups in Campus Virtual

M5 – P1: Introduction to Pytorch – Tasks

Your GitHub should contain a README.md file with:

- Title of the project.
- Name of the group.
- Name and contact email of all the team members.
- Link to the Overleaf article (Non-editable link) at the moment no content yet.
- Links to the presentations with the summary of your weekly work.

M5 – P1: Introduction to Pytorch – Tasks

Task (d): **Understand Pytorch framework**

- **Check Pytorch documentation**
  - Deep learning with Pytorch:
    - Deep learning 60min blitz tutorial
    - Training a classifier: CIFAR 10 tutorial
    - Torchvision documentation main page

- **Install Pytorch framework**
  - Follow instructions from /home/mcv/installing_m5.txt

- **Project presentation:**
  - Problems that you find when learning Pytorch or interesting features that can't be found in Keras.
    (1 slide)

M5 – P1: Introduction to Pytorch – Tasks

Task (e): **Implement M3 Image Classification network in Pytorch**

- **GitHub** repository for the code management.
  - Include your implementation in your github repository
- **Project presentation**
  - Include the description of your network architecture in your presentation (diagrams are welcome).
  - Details, hyperparameters and logger tool used are also welcome.
  - Provide the results of the network (with analysis).

M5 – P1: Introduction to Pytorch – Tasks

Task (f): **Compute losses graphs and compare them with yours from Keras**

- **Project presentation**:
  - Include your training and validation losses graphs from Keras and Pytorch in your presentation.

M5 – P1: Introduction to Pytorch – Tasks

Task (g): **Compute accuracy graphs and compare them with yours from Keras**

- **Project presentation**.
  - Include your accuracy graphs from Keras and Pytorch in your presentation.

M5 – P1: Introduction to Pytorch – Tasks

Task (e, f, g): **Results and graphics**

- When we ask for the accuracy results and graphics, **we expect analysis**.
  - Do both training and validation loss have the same behavior?
  - If you achieve better results with one framework or the other, are the loss functions coherent with this?
  - Seems to be overfitting?
  - Is the accuracy the same for all the classes in the dataset?
- Graphics are tools that helps to **understand** what is happening in your network / training experiment.

M5 – P1: Introduction to Pytorch – Tasks

Project presentation

- We won't consider any experiment you have done and is not included into the presentation.
- Differently of common presentations, we will check this one offline, so a little bit of text describing diagrams, plots, figures, tables and results is expected.

M5 – P1: Introduction to Pytorch – Tasks

Project presentation

- Include **one** summary slide at the end of your presentation with 2 items:
  - Main difficulties and problems (if any)
  - Main results and conclusions

- One member of the group members will have to present this slide in **1 minute** during the follow-up session next week.

M5 – P1: Introduction to Pytorch – Deadline

Due date

13th of March, Monday, before 10:00 AM