Continuum

Simple Management of Complex Continual Learning Scenarios

```
$ pip3 install continuum
```

https://github.com/Continvvm/continuum

Arthur Douillard @Ar_Douillard
Timothée Lesort @TLesort

Data Loading for Continual Learning

Continual data loading is **complex**

Each paper requires **many different settings**

Reinventing the wheel

```
dataset = MNIST("my/data/path", download=True, train=True)
scenario = ClassIncremental(dataset, increment=2)

for task_id, taskset in enumerate(scenario):
    train_taskset, val_taskset = split_train_val(taskset, val_split=0.1)
    train_loader = DataLoader(train_taskset, batch_size=32, shuffle=True)
    val_loader = DataLoader(val_taskset, batch_size=32, shuffle=True)

    for x, y, t in train_loader:
        # Do your cool stuff here
```

UNIX Philosophy:

Minimal

One Single Goal

Modular

1. Choose a dataset,
2. Choose a scenario,
3. and use *torchvision* loaders!

```python
dataset = MNIST("my/data/path", download=True, train=True)
scenario = ClassIncremental(dataset, increment=2)

for task_id, taskset in enumerate(scenario):
    loader = DataLoader(taskset, batch_size=32, shuffle=True)

    for x, y, t in train_loader:
        # Do your cool stuff here
```

But wait there is already many libraries...

Sequoia FACIL Avalanche

But wait there is already many libraries...

Sequoia FACIL Avalanche

Continuum

Released to public and used by several researchers since **April 2020**

Light-weight with only goal in mind: data loading, no models!

Easiest to **plug in** existing codebase

Also **good synergy** with large codebase such as Sequoia!

But wait there is already many libraries...

Sequoia FACIL Avalanche

Different goal than existing libraries

Continuum

Released to public and used by several researchers since **April 2020**

Light-weight with only goal in mind: data loading, no models!

Easiest to **plug in** existing codebase

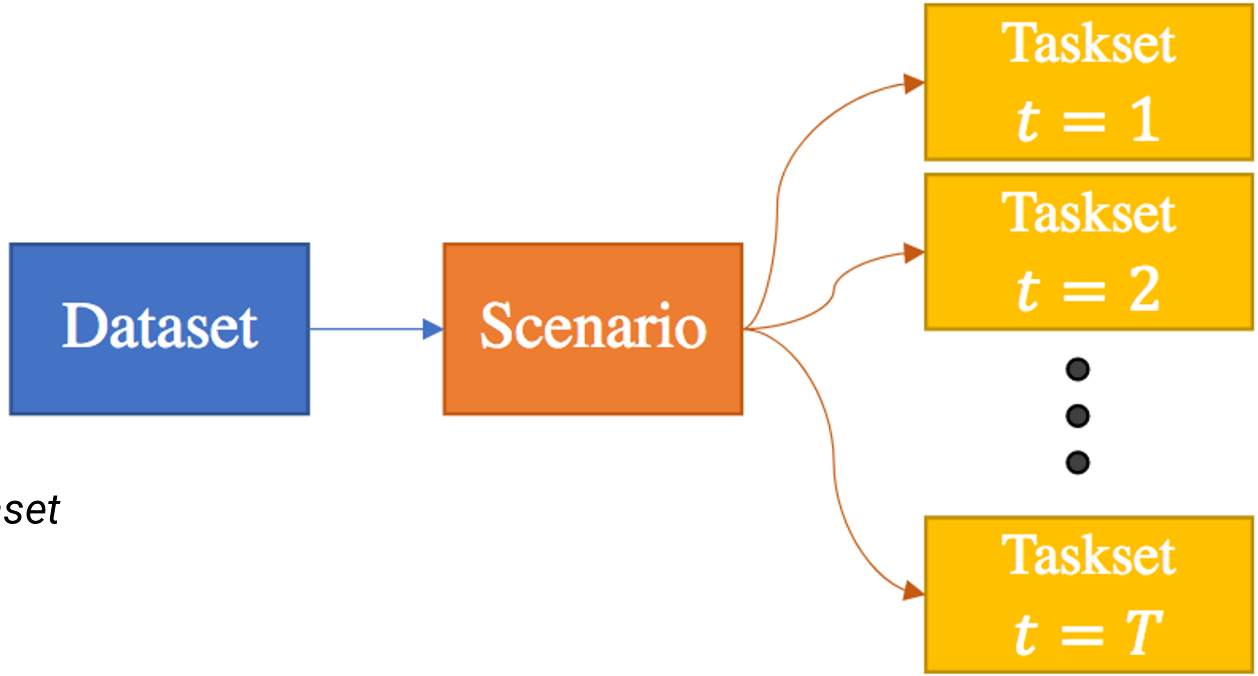
Also **good synergy** with large codebase such as Sequoia!

No big diagram

More than 20 datasets

5 different scenarios

Tasksets are *torchvision.Dataset*



Smooth integration in Pytorch

```python
from torch.utils.data import DataLoader
from continuum import ClassIncremental
from continuum.datasets import MNIST

dataset = MNIST("/data", train=True)
scenario = ClassIncremental(dataset, increment=2)

for task_id, train_taskset in enumerate(scenario):
    train_loader = DataLoader(train_taskset, batch_size=32, shuffle=True)

    for x, y, t in train_loader:
        # x --> data
        # y --> labels
        # t --> task ids
```

Class-Incremental vs Task-Incremental

Use task ids, or not.

Your choice!

```
from torch.utils.data import DataLoader
from continuum import ClassIncremental
from continuum.datasets import MNIST

dataset = MNIST("/data", train=False)
scenario = ClassIncremental(dataset, increment=2)

for task_id, test_taskset in enumerate(scenario):
    test_loader = DataLoader(test_taskset, batch_size=32)

    for x, y, t in test_loader:
        # x --> data
        # y --> labels
        # t --> task ids
```

Tasks Flexibility

```python
dataset = MNIST("/data", train=False)
scenario = ClassIncremental(dataset, increment=2)

third_taskset = scenario[2]
all_seen_tasksets = scenario[:3]
```

All usual python fancy indexing for scenario

Split MNIST

```python
from continuum import ClassIncremental
from continuum.datasets import MNIST

dataset = MNIST("/path", train=True)
scenario = ClassIncremental(dataset, increment=2)
```





Split MNIST

```python
from continuum import ClassIncremental
from continuum.datasets import MNIST

dataset = MNIST("/path", train=True)
scenario = ClassIncremental(dataset, increment=[4, 4, 2])
```

New Instances

```python
from continuum import InstanceIncremental
from continuum.datasets import Core50v2_79

dataset = Core50v2_79("/path", train=True, download=True)
scenario = InstanceIncremental(dataset)
```

...

Segmentation

```python
dataset = PascalVOC2012("/path", train=True, download=True)
scenario = SegmentationClassIncremental(
    dataset,
    increment=[19, 1],  # Learning 19 classes then 1
    nb_classes=20,
    mode="overlap",
    transformations=[Resize((512, 512)), ToTensor()]
)

for taskset in scenario:
    loader = DataLoader(taskset, batch_size=12)
    for x, y, t in loaders:
        # Same interface, your model goes here
```

Incremental Learning Techniques for Semantic Segmentation, Michielli and Zanuttight, ICCV-W 2019
Modeling the Background for Incremental Learning in Semantic Segmentation, Cermelli et al., CVPR 2020

Transformations Scenario

```python
from torchvision.transforms import RandomAffine

dataset = MNIST("/path", train=True)
scenario = TransformationIncremental(
    dataset,
    [
        [RandomAffine(degrees=0)],
        [RandomAffine(degrees=45)],
        [RandomAffine(degrees=90)]
    ]
)
```

Task 0: 0° degree

Task 1: 45° degree

Task 2: 90° degree

Rehearsal Learning

Handle rehearsal memory as you want

Use memory in the taskset you want

Oversample with **native** pytorch samplers

```
from torch.utils.data import DataLoader
from continuum import ClassIncremental
from continuum.datasets import MNIST

dataset = MNIST("/path", train=True)
scenario = ClassIncremental(dataset, increment=2)

taskset = scenario[2]
# Add rehearsal memory to current task
taskset.add_samples(memory_x, memory_y)

loader = DataLoader(
    taskset,
    sampler=my_pytorch_sampler
)
```

🚀 Future

🚫 We don’t want to increase complexity:

- No addition of tons of options, attributes, etc.
- No models, no losses, etc.

✅ We want to offer a large choice of simple interfaces:

- More datasets (more segmentation, more NLP, etc.)
- More scenarios

Don’t use Continuum

- if you wants a whole ecosystem,
- many models,
- and a strict way to do continual learning

Do use Continuum

- if you wants a **light-weight** library,
- easily **pluggable** in any codebase, no matter how small or large it is
- with the most **pytorch*nic*** interface

Want more?

📖 Documentation: continuum.readthedocs.io

💻 Github: github.com/Continvvm/continuum

⚙️ Colab tutorial: colab.research.google.com

```
$ pip3 install continuum
```