# Programming at Deep Learning Field

**Tao Ruan**
**2020.11.28**

# Preliminaries

## Essential Skills

SSH(e.g., Putty, Terminal, XShell); Linux Programming(e.g., Shell); Server File Editing(e.g., Vim); Virtual Environment(e.g., Docker).

## Development Tools

### 0. Local System

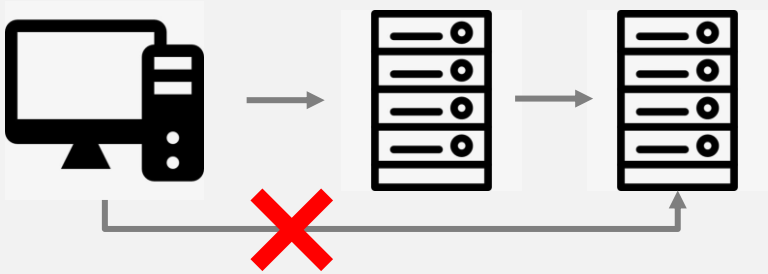
Windows, Ubuntu, CentOS, MacOS ... Whatever you like.

### 1. Remote Server + Local Pycharm

**Pros**: Rather Simple; Interactive-Friendly.

**Cons**: Practically bound with python;
Hard to handle with multi-node clusters.

- 使用PyCharm进行远程开发和调试 - 慕课网的文章 - 知乎 https://zhuanlan.zhihu.com/p/35067462
- PyCharm+Docker：打造最舒适的深度学习炼丹炉 - 刘震的文章 - 知乎 https://zhuanlan.zhihu.com/p/52827335



### 2. Remote Server + Local VS Code

**Pros**: Relatively Simple; Interactive-Friendly;
Multi-Language Supporting.

**Cons**: Hard to handle with multi-node clusters.

- 使用vscode进行远程炼丹 - 希葛格的韩少君的文章 - 知乎 https://zhuanlan.zhihu.com/p/89662757

### 3. Remote Server + Docker+ Remote Editor（e.g.，Emacs, Vim)

**Pros**: Could do whatever you want. (JOKE: Emacs is an operation system)

**Cons**: Hard to learn.

# How to Read Codes

1. Be accustomed to relying on documents.

   https://pytorch.org/docs/stable/index.html

2. Reading in a top-down, level-based manner.
3. Reading-by-debugging.
4. Try things of interest in IPython/Jupyter Notebook.

PYTORCH DOCUMENTATION

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.

Notes

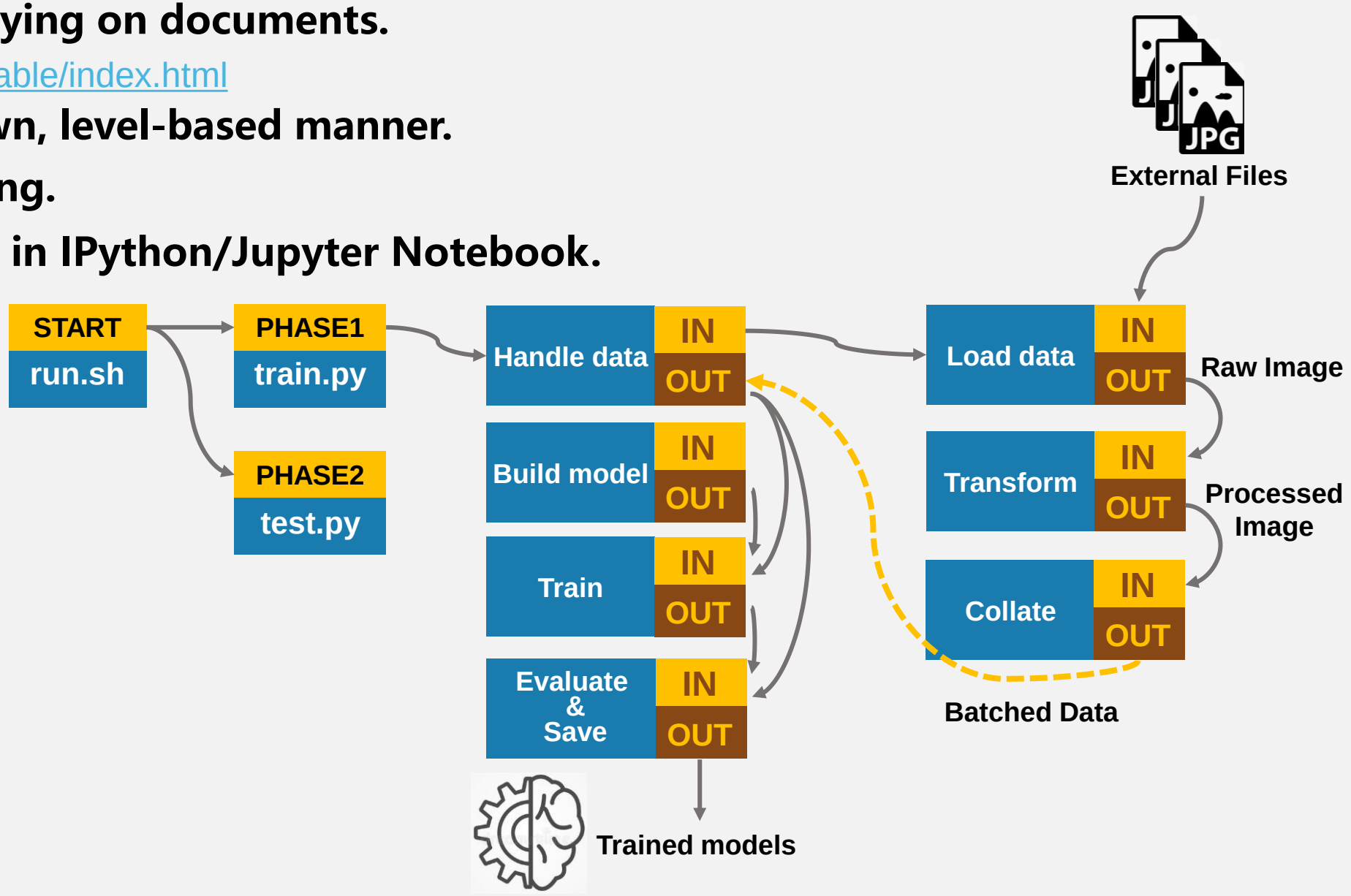
- Autograd mechanics
- Broadcasting semantics
- CPU threading and TorchScript inference
- CUDA semantics
- Extending PyTorch
- Frequently Asked Questions
- Features for large-scale deployments
- Multiprocessing best practices
- Reproducibility
- Serialization semantics
- Windows FAQ

Community

- PyTorch Contribution Guide
- PyTorch Governance
- PyTorch Governance | Persons of Interest

Package Reference

- torch
- torch.Tensor
- Tensor Attributes
- Type Info



# How to Write Codes

## [Recommend] Find a reliable project and just follow it!

https://github.com/open-mmlab/mmdetection
https://github.com/facebookresearch/detectron2
https://github.com/google-research-datasets/Objectron
……

## [Recommend] Writing comments as more as possible.

https://docs.python.org/3.8/library/pydoc.html#module-pydoc

https://www.python.org/dev/peps/pep-0008/

https://google.github.io/styleguide/pyguide.html#38-comments-and-docstrings

## [Optional] Code Styles and Design patterns.

https://google.github.io/styleguide/pyguide.html

https://github.com/faif/python-patterns

# MMDetection —— An Example



## Sources

Code Repository: https://github.com/open-mmlab/mmdetection

Documentation: https://mmdetection.readthedocs.io/

## What to do first?

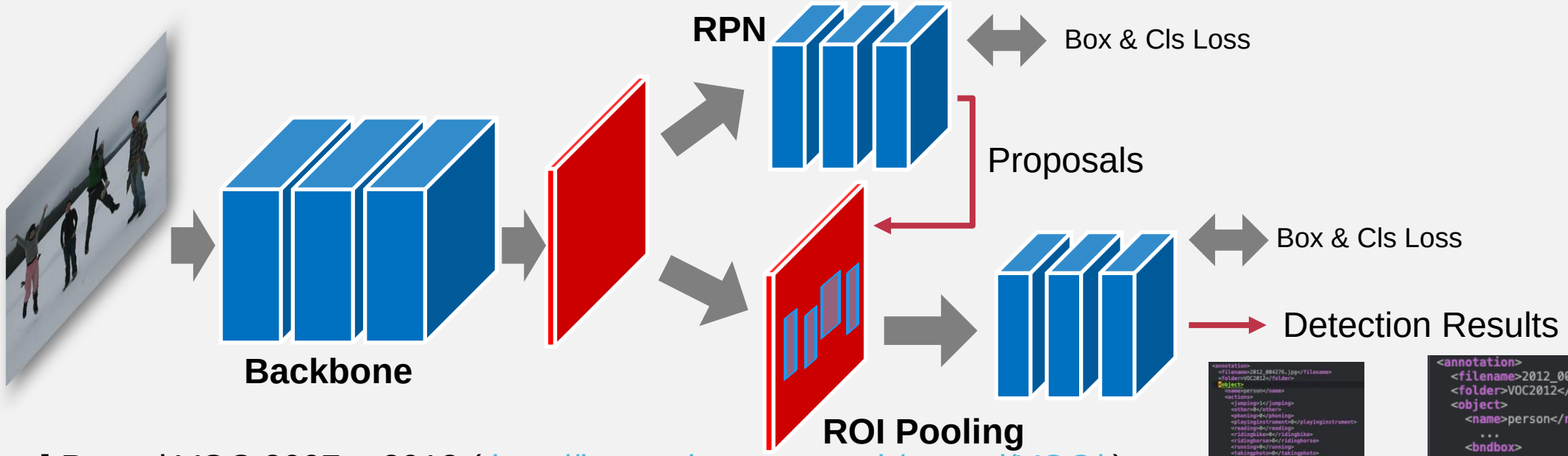
git clone https://github.com/open-mmlab/mmdetection.git

<build & install ([Recommend] Build with docker)>

# MMDetection —— An Example

## [Example Task] Train a Faster R-CNN model upon Pascal VOC dataset.

[Paper] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks



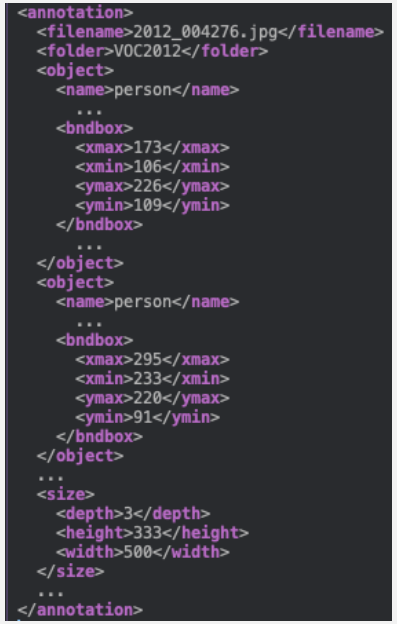

[Dataset] Pascal VOC 2007 + 2012 ( http://host.robots.ox.ac.uk/pascal/VOC/ )

- VOCdevkit
  - VOC2007
  - VOC2012
    - JPEGImages
    - Annotations
    - ImageSets

2012_004309.jpg

2012_004276.jpg

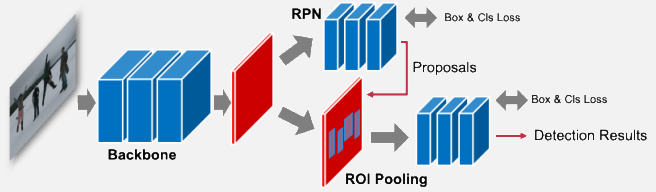
2012_004267.jpg



```
<annotation>
  <filename>2012_004276.jpg</filename>
  <folder>VOC2012</folder>
  <object>
    <name>person</name>
      ...
    <bndbox>
      <xmax>173</xmax>
      <xmin>106</xmin>
      <ymax>226</ymax>
      <ymin>109</ymin>
    </bndbox>
      ...
  </object>
  <object>
    <name>person</name>
      ...
    <bndbox>
      <xmax>295</xmax>
      <xmin>233</xmin>
      <ymax>220</ymax>
      <ymin>91</ymin>
    </bndbox>
  </object>
  ...
  <size>
    <depth>3</depth>
    <height>333</height>
    <width>500</width>
  </size>
  ...
</annotation>
```

2012_004276.xml

# MMDetection —— An Example (Faster R-CNN upon VOC)




## Find the root(entrance)

We follow [Train predefined models on standard datasets](#).

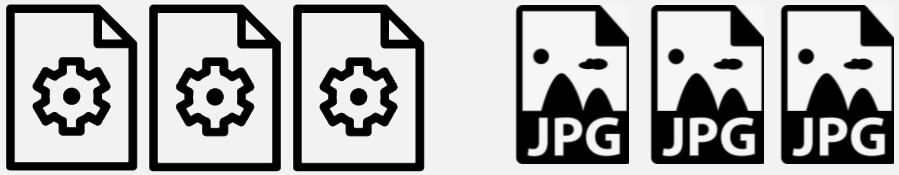
### Training on a single GPU

We provide `tools/train.py` to launch training jobs on a single GPU. The basic usage is as follows.

CONFIG_FILE=./configs/pascal_voc/faster_rcnn_r50_fpn_1x_voc0712.py

```
python tools/train.py \
    ${CONFIG_FILE} \
    [optional arguments]
```

```
[ PublicCodes ] $ cd mmdetection/
[ mmdetection ] $ ./tools/train.py ./configs/pascal_voc/faster_rcnn_r50_fpn_1x_voc0712.py
```







START
train.py

| | |
|---|---|
| Build metas | IN / OUT |
| Build model | IN / OUT |
| Load dataset | IN / OUT |
| Run train_detector() | IN / OUT |

cfg: config instance;
meta: env infos;
logger: keep runtime logs.

model: Fast R-CNN model.

dataset: VOC data pool.

```
(Pdb) cfg
Config (path: ./configs/pascal_voc/faster_rcnn_r50_fpn_1x_voc0712.py): {'model': {'type': 'FasterRCNN', 'pretrained': '
torchvision://resnet50', 'backbone': {'type': 'ResNet', 'depth': 50, 'num_stages': 4, 'out_indices': (0, 1, 2, 3), 'fro
zen_stages': 1, 'norm_cfg': {'type': 'BN', 'requires_grad': True}, 'norm_eval': True, 'style': 'pytorch'}, 'neck': {'ty
pe': 'FPN', 'in_channels': [256, 512, 1024, 2048], 'out_channels': 256, 'num_outs': 5}, 'rpn_head': {'type': 'RPNHead',
 'in_channels': 256, 'feat_channels': 256, 'anchor_generator': {'type': 'AnchorGenerator', 'scales': [8], 'ratios': [0.
5, 1.0, 2.0], 'strides': [4, 8, 16, 32, 64]}, 'bbox_coder': {'type': 'DeltaXYWHBBoxCoder', 'target_means': [0.0, 0.0, 0
.0, 0.0], 'target_stds': [1.0, 1.0, 1.0, 1.0]}, 'loss_cls': {'type': 'CrossEntropyLoss', 'use_sigmoid': True, 'loss_wei
ght': 1.0}, 'loss_bbox': {'type': 'L1Loss', 'loss_weight': 1.0}}, 'roi_head': {'type': 'StandardRoIHead', 'bbox_roi_ext
```

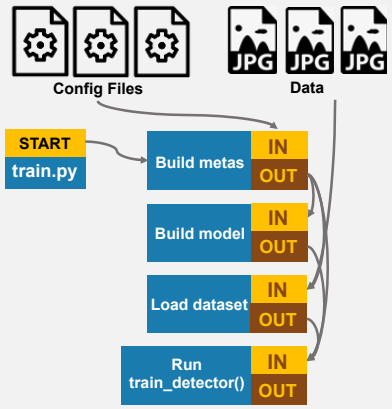
```
(Pdb) p model
FasterRCNN(
  (backbone): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (layer1): ResLayer(
      (0): Bottleneck(
        (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (downsample): Sequential(
          (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
```

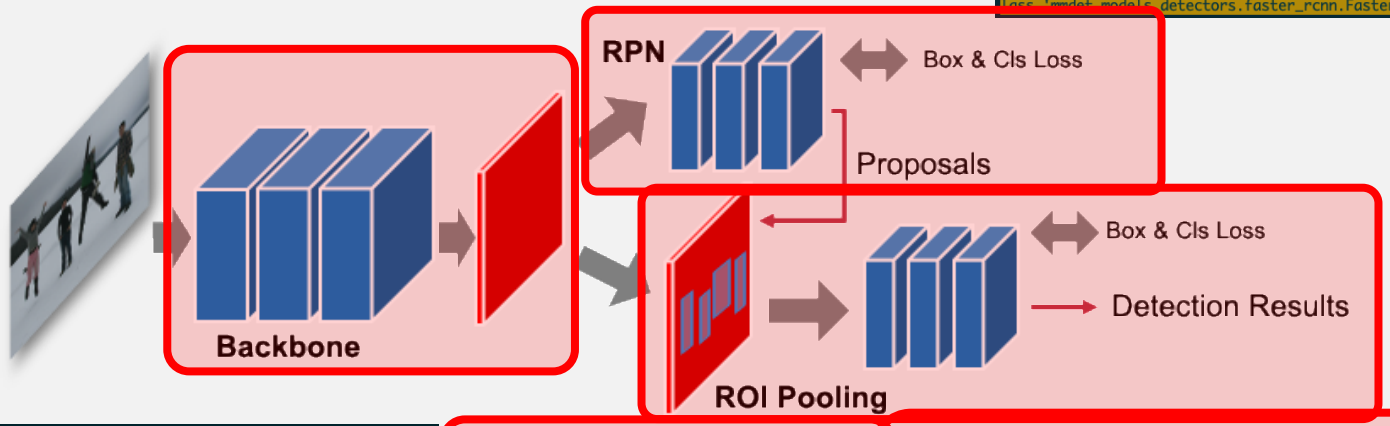
# MMDetection —— An Example (Faster R-CNN upon VOC)

## Build Model

```
model = build_detector(
    cfg.model, train_cfg=cfg.train_cfg, test_cfg=cfg.test_cfg)
```

```
-> return build(cfg, DETECTORS, dict(train_cfg=train_cfg, test_cfg=test_cfg))
(Pdb) p DETECTORS
Registry(name=detector, items={'SingleStageDetector': <class 'mmdet.models.detectors.single_stage.SingleStageDetector'>
, 'ATSS': <class 'mmdet.models.detectors.atss.ATSS'>, 'TwoStageDetector': <class 'mmdet.models.detectors.two_stage.TwoS
tageDetector'>, 'CascadeRCNN': <class 'mmdet.models.detectors.cascade_rcnn.CascadeRCNN'>, 'CornerNet': <class 'mmdet.mo
dels.detectors.cornernet.CornerNet'>, 'FastRCNN': <class 'mmdet.models.detectors.fast_rcnn.FastRCNN'>, 'FasterRCNN': <c
lass 'mmdet.models.detectors.faster_rcnn.FasterRCNN'>, 'FCOS': <class 'mmdet.models.detectors.fcos.FCOS'>, 'FOVEA': <cl
```




```
x = self.extract_feat(img)

losses = dict()

# RPN forward and loss
if self.with_rpn:
    proposal_cfg = self.train_cfg.get('rpn_proposal',
                                      self.test_cfg.rpn)
    rpn_losses, proposal_list = self.rpn_head.forward_train(
        x,
        img_metas,
        gt_bboxes,
        gt_labels=None,
        gt_bboxes_ignore=gt_bboxes_ignore,
        proposal_cfg=proposal_cfg)
    losses.update(rpn_losses)
else:
    proposal_list = proposals

roi_losses = self.roi_head.forward_train(x, img_metas, proposal_list,
                                         gt_bboxes, gt_labels,
                                         gt_bboxes_ignore, gt_masks,
                                         **kwargs)
losses.update(roi_losses)

return losses
```

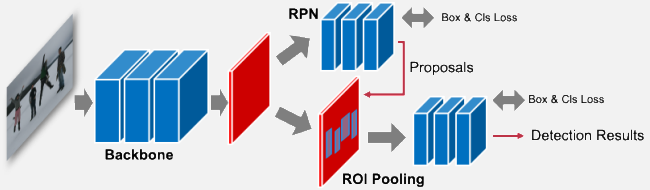
```
(Pdb) p type(x)
<class 'tuple'>
(Pdb) p len(x)
5
(Pdb) for i in x: print(i.shape)
torch.Size([2, 256, 152, 200])
torch.Size([2, 256, 76, 100])
torch.Size([2, 256, 38, 50])
torch.Size([2, 256, 19, 25])
torch.Size([2, 256, 10, 13])
```

```
(Pdb) p rpn_losses
{'loss_rpn_cls': [tensor(0.5092, device='cuda:0', grad_fn=<MulBackward0>), tensor(0.1363, device='cuda:0', grad_fn=<Mul
Backward0>), tensor(0.0256, device='cuda:0', grad_fn=<MulBackward0>), tensor(0.0149, device='cuda:0', grad_fn=<MulBackw
ard0>), tensor(0.0081, device='cuda:0', grad_fn=<MulBackward0>)], 'loss_rpn_bbox': [tensor(0., device='cuda:0', grad_fn
=<MulBackward0>), tensor(0., device='cuda:0', grad_fn=<MulBackward0>), tensor(0., device='cuda:0', grad_fn=<MulBackward
0>), tensor(0.0097, device='cuda:0', grad_fn=<MulBackward0>), tensor(0.0038, device='cuda:0', grad_fn=<MulBackward0>)]}
```

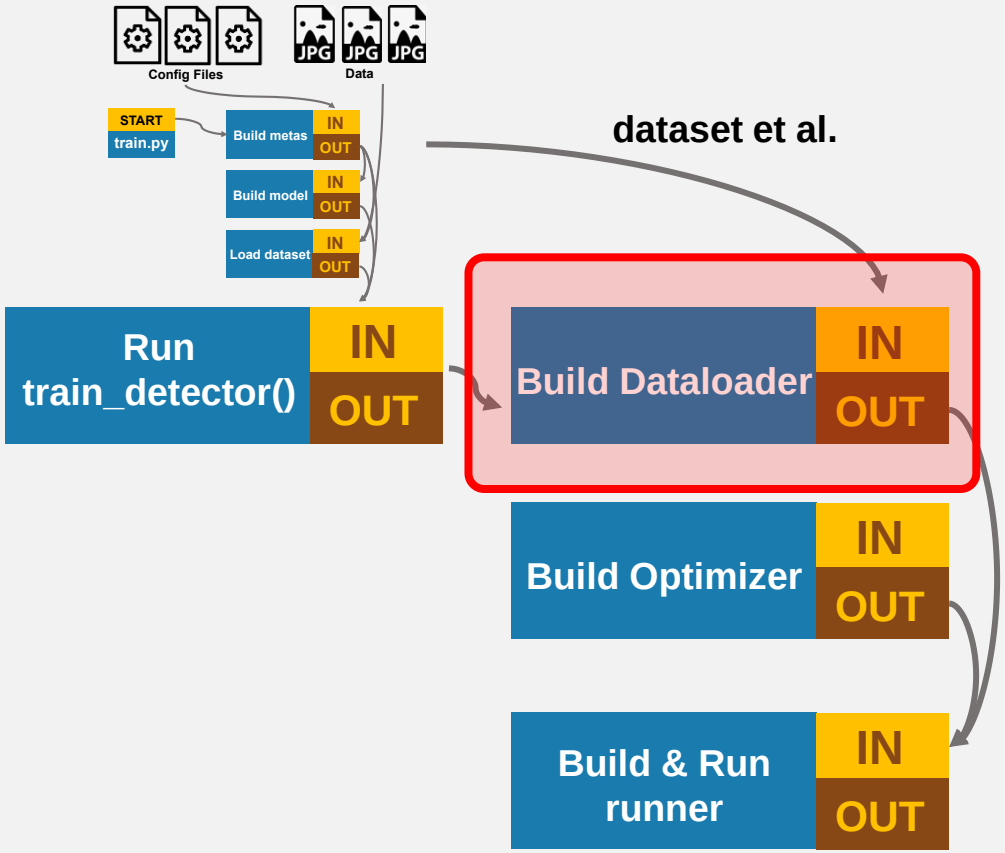
```
(Pdb) p len(proposal_list)
2
(Pdb) p len(proposal_list[0])
1000
(Pdb) p len(proposal_list[1])
1000
(Pdb) p proposal_list[0].shape
torch.Size([1000, 5])
(Pdb) p proposal_list[0][:5]
tensor([[2.3610e+02, 6.5041e+02, 2.6127e+02, 6.9722e+02, 5.5740e-01],
        [2.3196e+02, 6.4960e+02, 2.5661e+02, 6.9620e+02, 5.5687e-01],
        [2.0488e+02, 6.2922e+02, 2.3009e+02, 6.7398e+02, 5.5538e-01],
        [1.8551e+02, 6.3057e+02, 2.1067e+02, 6.7473e+02, 5.5388e-01],
        [1.8095e+02, 6.2932e+02, 2.0609e+02, 6.7423e+02, 5.5363e-01]],
       device='cuda:0')
```

```
(Pdb) p roi_losses
{'loss_cls': tensor(2.8783, device='cuda:0', grad_fn=<MulBackward0>), 'acc': tensor([8.6914], device='cuda:0'), 'loss_b
box': tensor(0.0058, device='cuda:0', grad_fn=<MulBackward0>)}
```

# MMDetection —— An Example (Faster R-CNN upon VOC)




## Running the Training




**Set breakpoint**

```
    import pdb; pdb.set_trace()
    runner.run(data_loaders, cfg.workflow, cfg.total_epochs)
```

**Dataloader list**

```
(Pdb) p data_loaders
[<torch.utils.data.dataloader.DataLoader object at 0x7f7f3dcc6d90>]
```

**Visualize a batch**

```
(Pdb) for b in data_loaders[0]: aaaa = b; break
```

```
(Pdb) p aaaa.keys()
dict_keys(['img_metas', 'img', 'gt_bboxes', 'gt_labels'])
```

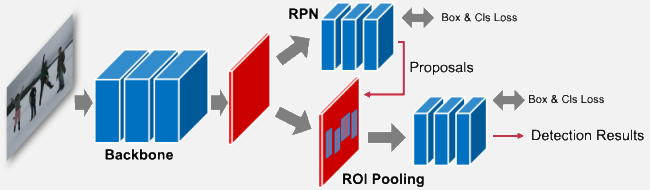
**Image infos**

```
(Pdb) p type(aaaa['img'])
<class 'mmcv.parallel.data_container.DataContainer'>
(Pdb) p type(aaaa['img'].data)
<class 'list'>
(Pdb) p len(aaaa['img'].data)
1
(Pdb) p aaaa['img'].data[0].shape
torch.Size([2, 3, 608, 832])
```

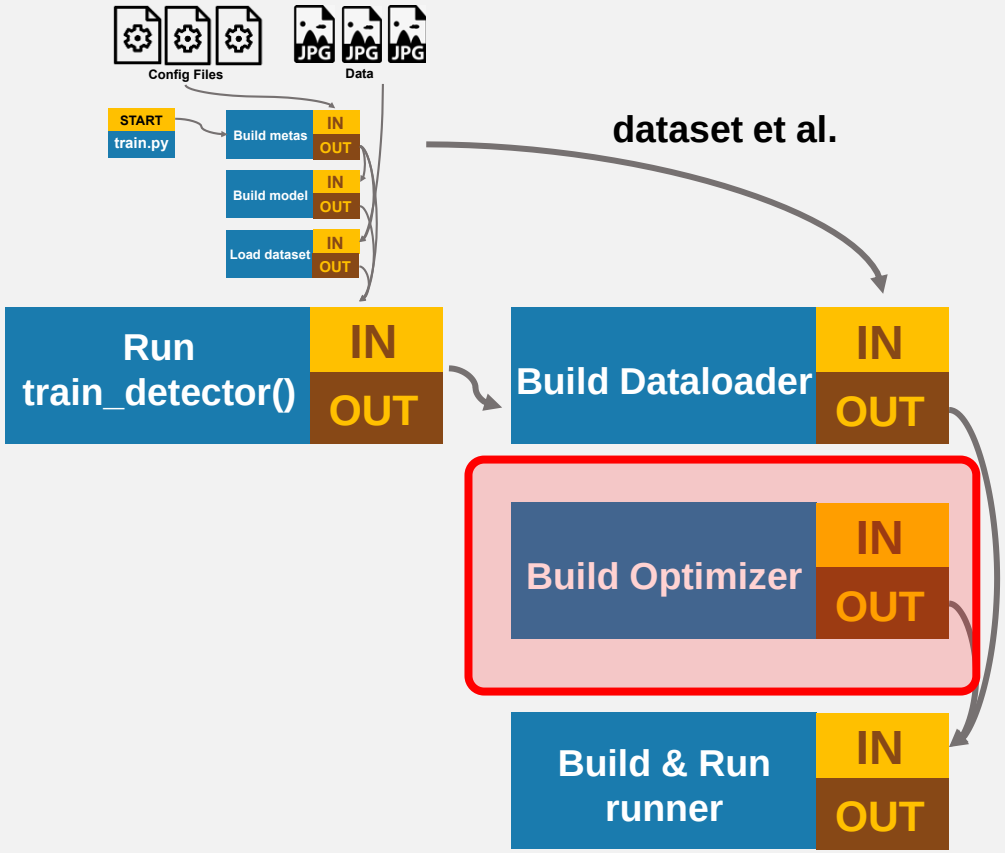
**Boxes & class labels**

```
(Pdb) p aaaa['gt_bboxes'].data[0]
[tensor([[147.2000, 491.2000, 696.0000, 596.8000],
        [107.2000, 270.4000, 128.0000, 316.8000],
        [414.4000, 409.6000, 462.4000, 563.2000],
        [476.8000, 513.6000, 504.0000, 598.4000],
        [436.8000, 502.4000, 465.6000, 598.4000],
        [ 70.4000, 304.0000, 243.2000, 444.8000],
        [140.8000, 305.6000, 238.4000, 456.0000],
        [ 17.6000, 313.6000, 134.4000, 460.8000],
        [  4.8000, 350.4000,  54.4000, 556.8000]]), tensor([[582.2520, 256.6845, 800.3960, 450.8022],
        [ 28.8720, 263.1016, 522.9040, 439.5722],
        [  0.0000, 421.9251, 344.8600, 598.3958],
        [670.4720, 352.9412, 784.3560, 598.3958],
        [407.4160, 352.9412, 545.3600, 598.3958],
        [526.1120, 333.6898, 561.4000, 455.6150],
        [486.0120, 335.2941, 532.5280, 455.6150],
        [203.7080, 367.3797, 288.7200, 513.3690],
        [174.8360, 352.9412, 227.7680, 436.3636],
        [102.6560, 354.5455, 168.4200, 433.1551],
        [  0.0000, 367.3797, 333.6320, 476.4706]])]
(Pdb) p aaaa['gt_labels'].data[0]
[tensor([10,  4,  4,  4,  4, 10,  8,  8,  8]), tensor([ 5,  5,  6, 14, 14, 14, 14, 14, 14, 14,  6])]
```

# MMDetection —— An Example (Faster R-CNN upon VOC)




## Running the Training

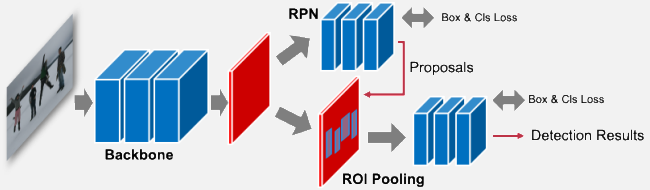



```
(Pdb) p cfg.optimizer
{'type': 'SGD', 'lr': 0.01, 'momentum': 0.9, 'weight_decay': 0.0001}
```

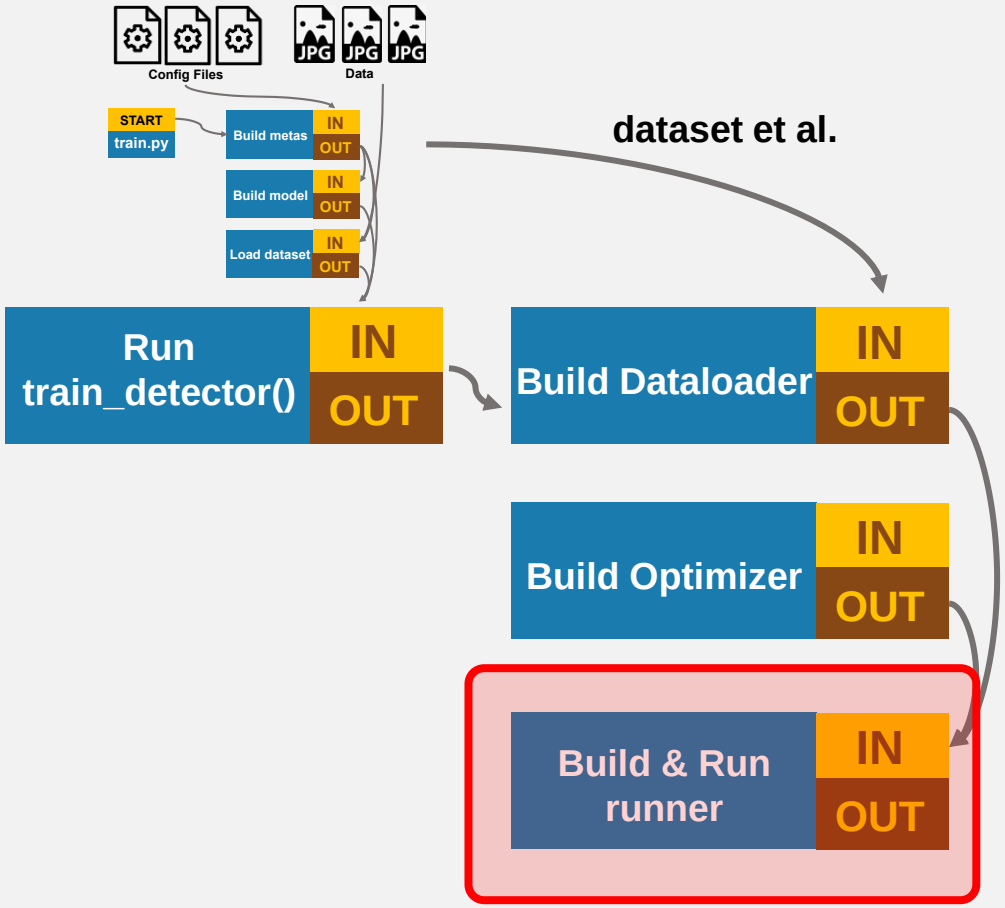
```
optimizer = build_optimizer(model, cfg.optimizer)
```

For more details refer:
https://github.com/open-mmlab/mmcv/tree/master/mmcv/runner/optimizer

# MMDetection —— An Example (Faster R-CNN upon VOC)




## Running the Training




```
runner = EpochBasedRunner(
    model,
    optimizer=optimizer,
    work_dir=cfg.work_dir,
    logger=logger,
    meta=meta)
```

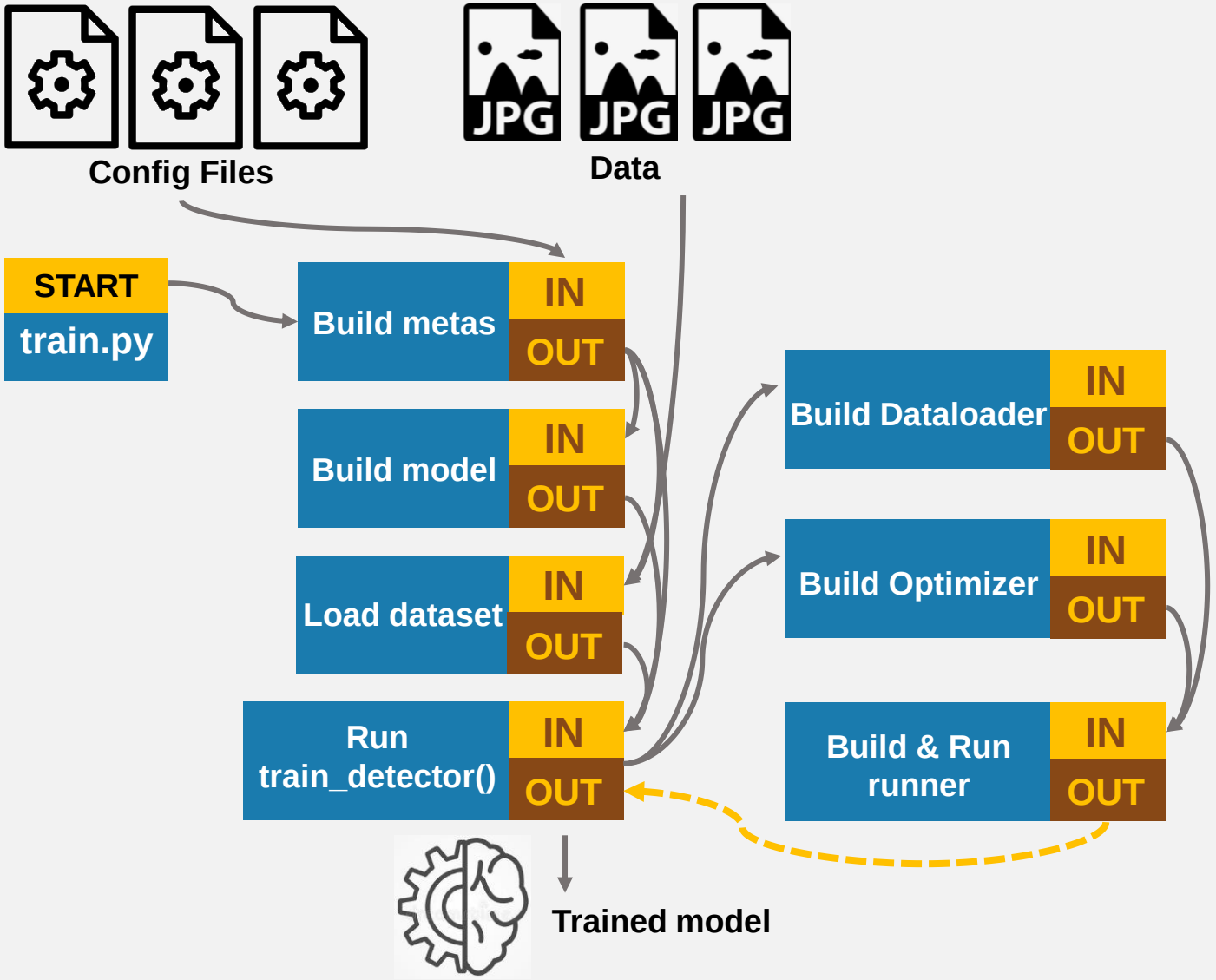
```
runner.run(data_loaders, cfg.workflow, cfg.total_epochs)
```

```
for i, data_batch in enumerate(self.data_loader):
    self._inner_iter = i
    self.call_hook('before_train_iter')
    self.run_iter(data_batch, train_mode=True)
    self.call_hook('after_train_iter')
    self._iter += 1
```

```
elif train_mode:
    outputs = self.model.train_step(data_batch, self.optimizer,
                                    **kwargs)
```

# MMDetection —— An Example (Faster R-CNN upon VOC)

## Whole Tree



# MMDetection —— Fundamental Operations

## Registry

```
In [16]: from mmcv.utils import registry

In [17]: TEST_REGISTRY = registry.Registry("test_registry")

In [18]: @TEST_REGISTRY.register_module()
    ...: class TestClass:
    ...:     def __init__(self):
    ...:         self.a = 10
    ...:     def do(self, b):
    ...:         print(self.a + b)
    ...:

In [19]: instance = TEST_REGISTRY.get("TestClass")()

In [20]: instance.do(20)
30
```

## Hook

```
self._hooks = []
```

```
for i in range(len(self._hooks) - 1, -1, -1):
    if priority >= self._hooks[i].priority:
        self._hooks.insert(i + 1, hook)
        inserted = True
        break
if not inserted:
    self._hooks.insert(0, hook)
```

```
for i, data_batch in enumerate(self.data_loader):
    self._inner_iter = i
    self.call_hook('before_train_iter')
    self.run_iter(data_batch, train_mode=True)
    self.call_hook('after_train_iter')
    self._iter += 1
```

# Q & A

## Programming at Deep Learning Field

Tao Ruan

2020.11.28