Yolov5 pytorch mlu移植教程

官方Yolov5文档：https://github.com/ultralytics/yolov5/releases/tag/v4.0

官方模型下载地址：https://github.com/ultralytics/yolov5/releases/download/v4.0/yolov5s.pt

移植好后代码：http://gitlab.software.cambricon.com/neuware/software/ae/solution/270_sample/yolov5_demo

MLU SDK版本：1.7.0

MLU PYTORCH版本：1.3.0

【以下版本在v4版本上完成（v5和v4类似）】

高版本PyTorch降级到低版本PyTorch

为什么要做这一步？

由于官方yolov5模型是在PyTorch1.8.x版本上进行，而MLU的PyTorch还是1.3.0版本。

高版本的PyTorch具有zip压缩模型功能，而在1.3.0上不支持，如果在1.3.0的版本上直接打开高版本的pt，会出现报错。

```
Traceback (most recent call last):
  File "detect.py", line 282, in <module>
    detect()
  File "detect.py", line 84, in detect
    model = attempt_load(weights, map_location=device)  # load FP32 model
  File "/home/Cambricon-Test/zhangkun_yolov5/yolov5_/models/experimental.py", line 141, in attempt_load
    state_dict = torch.load(weights[0], map_location='cpu')
  File "/torch/venv3/pytorch/lib/python3.6/site-packages/torch/serialization.py", line 507, in load
    return _load(f, map_location, pickle_module, **pickle_load_args)
  File "/torch/venv3/pytorch/lib/python3.6/site-packages/torch/serialization.py", line 680, in _load
    raise RuntimeError("{} is a zip archive (did you mean to use torch.jit.load()?)".format(f.name))
RuntimeError: weights/yolov5s.pt is a zip archive (did you mean to use torch.jit.load()?)
```

搭建 PyTorch1.8.0环境

可以先用conda或者virtualenv创建一个python虚拟环境，本教程用的是python3.6.10，然后执行如下命令安装pytorch1.8.0 cpu版。

```
pip install torch==1.8.0+cpu torchvision==0.9.0+cpu torchaudio==0.8.0 -f https://download.pytorch.org/whl/torch_stable.html
```

降低pt模型的版本

下载官方yolov5代码：

git clonehttps://github.com/ultralytics/yolov5.git

git checkout v4.0

修改detect.py

添加代码：torch.save(model.state_dict(), "./weights/unzip.pt", _use_new_zipfile_serialization=False)

```
18 def detect(save_img=False):
19     source, weights, view_img, save_txt, imgsz = opt.source, opt.weights, opt.view_img, opt.save_txt, opt.img_size
20     webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
21         ('rtsp://', 'rtmp://', 'http://'))
22 
23     # Directories
24     save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok))  # increment run
25     (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir
26 
27     # Initialize
28     set_logging()
29     device = select_device(opt.device)
30     half = device.type != 'cpu'  # half precision only supported on CUDA
31 
32     # Load model
33     model = attempt_load(weights, map_location=device)  # load FP32 model
34     torch.save(model.state_dict(), "./weights/unzip.pt", _use_new_zipfile_serialization=False)
35     imgsz = check_img_size(imgsz, s=model.stride.max())  # check img_size
36     if half:
37         model.half()  # to FP16
38 
39     # Second-stage classifier
```

下载官方模型：

wgethttps://github.com/ultralytics/yolov5/releases/download/v4.0/yolov5s.pt

运行python detect.py --device cpu --weights yolov5s.pt

生成unzip.pt文件，这个pt可以放到MLU PyTorch使用。

到此，PT降版本工作完成。

增加算子

修改/workspace/neuware_sdk_ubuntu_prebuild/venv/lib/python3.6/site-packages/torch/nn/modules/activation.py，添加SiLU, Hardswish激活函数

```
class Hardswish(Module): # export-friendly version of nn.Hardswish()
  @staticmethod
  def forward(x):
    # return x * F.hardsigmoid(x) # for torchscript and CoreML
    return x * F.hardtanh(x + 3, 0., 6.) / 6. # for torchscript, CoreML and ONNX
class SiLU(Module): # export-friendly version of nn.SiLU()
  @staticmethod
  def forward(x):
    return x * torch.sigmoid(x)
```

修改/workspace/neuware_sdk_ubuntu_prebuild/venv/lib/python3.6/site-packages/torch/nn/modules/__init__.py，注册SiLU, Hardswish激活函数

1，from .activation import 中添加SiLU, Hardswish,

2，__all__ = 中添加'SiLU', 'Hardswish',

MLU移植工作

首先请确认您安装的是MLU SDK 1.7.0版本，具体的移植步骤如下：

1.环境搭建(云平台已搭好)

使用镜像：yellow.hub.cambricon.com/pytorch/pytorch:0.15.0-ubuntu16.04

```
#/bin/bash
export MY_CONTAINER="Cambricon-MLU270-v1.7.0-pytorch"
num=`docker ps -a|grep "$MY_CONTAINER"|wc -l`
echo $num
echo $MY_CONTAINER
if [ 0 -eq $num ];then
xhost +
docker run -e DISPLAY=unix$DISPLAY --device /dev/cambricon_dev0 --net=host --pid=host -v /sys/kernel/debug:/sys/kernel/debug -v /tmp/.X11-unix:/tmp/.X11-unix -it --privileged --name $MY_CONTAINER -v $PWD/Cambricon-MLU270/:/home/Cambricon-MLU270 \
-v $PWD/../tar_package/datasets:/home/Cambricon-MLU270/datasets \
-v $PWD/../tar_package/models:/home/Cambricon-MLU270/models \
yellow.hub.cambricon.com/pytorch/pytorch:0.15.0-ubuntu16.04 /bin/bash
else
docker start $MY_CONTAINER
#sudo docker attach $MY_CONTAINER
docker exec -ti $MY_CONTAINER /bin/bash
fi
```

-v部分自定义

2. 激活MLU PyTorch

```
source /torch/venv3/pytorch/bin/activate
```

3.修改部分代码

在yolov5的目录下，需要修改models/experimental.py部分代码

1）引入库

```
(pytorch) root@localhost:/home/Cambricon-MLU270/test/yolov5s-src# git diff

diff --git a/models/experimental.py b/models/experimental.py
index 2dbbf7f..c83c685 100644
--- a/models/experimental.py
+++ b/models/experimental.py
@@ -1,4 +1,6 @@
# This file contains experimental modules
+import matplotlib
+matplotlib.use('Agg')
import numpy as np
import torch
```

在 models/experimental.py，头部增加 import matplotlib 和 matplotlib.use('Agg') 两行

2). 修改读取model代码

```
def attempt_load(weights, map_location=None):
  from models.yolo import Model
 model = Model('./models/yolov5s.yaml')
 state_dict = torch.load(weights[0], map_location='cpu')
 model.float().fuse().eval()
 model.load_state_dict(state_dict, strict=False)
 model.float().fuse().eval()
 return model
```

这里默认使用yolov5s，所以固定载入 ./models/yolov5s.yaml

3）验证

```
python detect.py --device cpu --weights unzip.pt
```

```
(pytorch) root@localhost:/home/Cambricon-Test/yolov5_demo# python detect.py --device cpu --weights weights/unzip.pt
CNML: 7.10.2 0a592c0ee
CNRT: 4.10.1 a884a9a
Namespace(agnostic_nms=False, augment=False, cfg='cpu', classes=None, conf_thres=0.25, device='cpu', exist_ok=False, img_size=640, iou_thres=0.45, jit=False, name='exp', project='result', save_conf=False, save_txt=False, source='data/images', update=False, view_img=False, weights=['weights/unzip.pt'])
Using torch 1.3.0a0 CPU


                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1    156928  models.common.C3                        [128, 128, 3]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  1    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1    229245  models.yolo.Detect                      [80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
Model Summary: 283 layers, 7276605 parameters, 7276605 gradients

Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients
Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients
image 1/2 /home/Cambricon-Test/yolov5_demo/data/images/bus.jpg: run cpu
640x480 4 0s, 1 5s, Done. (0.428s)
image 2/2 /home/Cambricon-Test/yolov5_demo/data/images/zidane.jpg: run cpu
384x640 2 0s, 1 27s, Done. (0.373s)
Results saved to result/exp
Done. (0.874s)
```

到此，官方的pt模型已经可以在MLU pytorch上执行cpu推理。

量化

1. 在detect.py中，在 Run inference上增加代码

```
import torch_mlu
import torch_mlu.core.mlu_quantize as mlu_quantize
```

```
import torch_mlu.core.mlu_model as ct


...


global quantized_model
if opt.cfg == 'qua':
 qconfig = {'iteration':2,'firstconv':False}
 quantized_model = mlu_quantize.quantize_dynamic_mlu(model, qconfig, dtype='int8', gen_quant=True)


# Run inference
t0 = time.time()
img = torch.zeros((1, 3, imgsz, imgsz), device=device) # init img
....
  # Inference
 t1 = time_synchronized()
 if opt.cfg == 'cpu':
 pred = model(img, augment=opt.augment)[0]
 print('run cpu')


 elif opt.cfg == 'qua':
 pred = quantized_model(img)[0]
 torch.save(quantized_model.state_dict(), 'yolov5s_int8.pt')
 print('run qua')


if __name__ == '__main__':
...
 parser.add_argument('--cfg', default='cpu', help='qua and off')
 parser.add_argument('--jit', type=bool,default=False)
...
```

2.执行

```
python detect.py --weights weights/unzip.pt --source data/images/ --cfg qua
```

```
(pytorch) root@localhost:/home/Cambricon-Test/yolov5_demo# python detect.py --weights weights/unzip.pt --source data/images/ --cfg qua
CNML: 7.10.2 0a592c0ee
CNRT: 4.10.1 a884a9a
Namespace(agnostic_nms=False, augment=False, cfg='qua', classes=None, conf_thres=0.25, device='', exist_ok=False, img_size=640, iou_thres=0.45, jit=False, name='exp', project='result', save_conf=
False, save_txt=False, source='data/images/', update=False, view_img=False, weights=['weights/unzip.pt'])
Using torch 1.3.0a0 CPU


                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1    156928  models.common.C3                        [128, 128, 3]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  1    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1    229245  models.yolo.Detect                      [80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
Model Summary: 283 layers, 7276605 parameters, 7276605 gradients

Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients
Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients
The 'iteration' key in qconfig has been deprecated in this version. Users needn't
set this key. We will calculate quantization parameters by real forward times controlled by users.
image 1/2 /home/Cambricon-Test/yolov5_demo/data/images/bus.jpg: run qua
image 2/2 /home/Cambricon-Test/yolov5_demo/data/images/zidane.jpg: run qua
Results saved to result/exp2
Done. (1.440s)
```

torch.save(quantized_model.state_dict(), 'weights/yolov5s_int8.pt')

这句代码会在weights目录下保存一个量化好的yolov5s_int8.pt文件

逐层

由于每一层都需要跑在mlu上，通过打印可以看到其他层目前是能支持的，最后一层有mlu算子完成。

修改models/yolo.py的部分代码。

```
class Detect(nn.Module):
  stride = None # strides computed during build
  export = False # onnx export
```

```
  def __init__(self, nc=80, anchors=(), ch=()): # detection layer
    super(Detect, self).__init__()
    self.anchors_list = list(np.array(anchors).flatten())
    self.num_anchors = len(self.anchors_list)
    self.nc = nc # number of classes
    self.no = nc + 5 # number of outputs per anchor
    self.nl = len(anchors) # number of detection layers
    self.na = len(anchors[0]) // 2 # number of anchors
    self.grid = [torch.zeros(1)] * self.nl # init grid
    a = torch.tensor(anchors).float().view(self.nl, -1, 2)
    self.register_buffer('anchors', a) # shape(nl,na,2)
    self.register_buffer('anchor_grid', a.clone().view(self.nl, 1, -1, 1, 1, 2)) # shape(nl,1,na,1,1,2)
    self.m = nn.ModuleList(nn.Conv2d(x, self.no * self.na, 1) for x in ch) # output conv

    self.img_h = 640
    self.img_w = 640
    self.conf_thres = 0.25
    self.iou_thres = 0.45
    self.maxBoxNum = 1024

  def forward(self, x):
    # x = x.copy() # for profiling
    z = [] # inference output
    output = []
    self.training |= self.export

    if x[0].device.type == 'mlu':
      for i in range(self.nl):
        x[i] = self.m[i](x[i]) # conv
        y = x[i].sigmoid()
        # print('y.shape: ',y.shape)
        output.append(y)
      detect_out = torch.ops.torch_mlu.yolov5_detection_output(output[0], output[1], output[2],
                self.anchors_list,self.nc, self.num_anchors,
                self.img_h, self.img_w, self.conf_thres, self.iou_thres, self.maxBoxNum)

              # [10, 13, 16, 30, 33, 23,30, 61, 62, 45, 59, 119, 116, 90, 156, 198, 373, 326]
      return detect_out

    for i in range(self.nl):
      x[i] = self.m[i](x[i]) # conv
      bs, _, ny, nx = x[i].shape # x(bs,255,20,20) to x(bs,3,20,20,85)
```

可以看到，这里将Detect类，修改为，我们mlu支持的算子torch.ops.torch_mlu.yolov5_detection_output

detect_out = torch.ops.torch_mlu.yolov5_detection_output

这里算子是把img宽高固定为 640 640作为输入。

detect.py需要继续增加逐层运行的方式

增加一个get_boxes的函数，用来从torch.ops.torch_mlu.yolov5_detection_output获取结果后画框，不再依赖cpu的获取框的后处理方式了。

```
import numpy as np
```

```
def get_boxes(prediction, batch_size=1, img_size=640):

  """

  Returns detections with shape:

    (x1, y1, x2, y2, object_conf, class_score, class_pred)

  """

  reshape_value = torch.reshape(prediction, (-1, 1))
  num_boxes_final = reshape_value[0].item()
  print('num_boxes_final: ',num_boxes_final)
  all_list = [[] for _ in range(batch_size)]
  for i in range(int(num_boxes_final)):
    batch_idx = int(reshape_value[64 + i * 7 + 0].item())
    if batch_idx >= 0 and batch_idx < batch_size:
      bl = reshape_value[64 + i * 7 + 3].item()
      br = reshape_value[64 + i * 7 + 4].item()
      bt = reshape_value[64 + i * 7 + 5].item()
      bb = reshape_value[64 + i * 7 + 6].item()
      if bt - bl > 0 and bb -br > 0:
        all_list[batch_idx].append(bl)
        all_list[batch_idx].append(br)
        all_list[batch_idx].append(bt)
        all_list[batch_idx].append(bb)
        all_list[batch_idx].append(reshape_value[64 + i * 7 + 2].item())
        all_list[batch_idx].append(reshape_value[64 + i * 7 + 1].item())
  output = [np.array(all_list[i]).reshape(-1, 6) for i in range(batch_size)]
  return output
```

增加 mlu 的配置方式

```
def detect(save_img=False):
  source, weights, view_img, save_txt, imgsz = opt.source, opt.weights, opt.view_img, opt.save_txt, opt.img_size
  webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
    ('rtsp://', 'rtmp://', 'http://'))



  # Directories
  save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok)) # increment run
  (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True) # make dir

  # Initialize
  set_logging()
  device = select_device(opt.device)
  half = device.type != 'cpu' # half precision only supported on CUDA

  # Load model
  model = attempt_load(weights, map_location=device) # load FP32 model
  imgsz = check_img_size(imgsz, s=model.stride.max()) # check img_size
  if half:
    model.half() # to FP16

  # Second-stage classifier
  classify = False
  if classify:
    modelc = load_classifier(name='resnet101', n=2) # initialize
    modelc.load_state_dict(torch.load('weights/resnet101.pt', map_location=device)['model']).to(device).eval()

  # Set Dataloader
  vid_path, vid_writer = None, None
  if webcam:
    view_img = True
    cudnn.benchmark = True # set True to speed up constant image size inference
    dataset = LoadStreams(source, img_size=imgsz)
  else:
    save_img = True
    dataset = LoadImages(source, img_size=imgsz)

  # Get names and colors
  names = model.module.names if hasattr(model, 'module') else model.names
  colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]

  global quantized_model
  global quantized_net
  if opt.cfg == 'qua':
    qconfig = {'iteration':2,'firstconv':False}
    quantized_model = mlu_quantize.quantize_dynamic_mlu(model, qconfig, dtype='int8', gen_quant=True)
  elif opt.cfg == 'mlu':
```

```
    from models.yolo import Model
    model = Model('./models/yolov5s.yaml').to(torch.device('cpu'))
    model.float().fuse().eval()
    quantized_net = torch_mlu.core.mlu_quantize.quantize_dynamic_mlu(model)
    #state_dict = torch.load("./yolov5s_int8.pt")
    state_dict = torch.load(weights[0])
    quantized_net.load_state_dict(state_dict, strict=False)
    quantized_net.eval()
    quantized_net.to(ct.mlu_device())

    if opt.jit:
      print("### jit")
      ct.save_as_cambricon('weights/yolov5s_int8_1_4')
      torch.set_grad_enabled(False)
      ct.set_core_number(4)
      trace_input = torch.randn(1, 3, 640, 640, dtype=torch.float)
      input_mlu_data = trace_input.type(torch.HalfTensor).to(ct.mlu_device())
      quantized_net = torch.jit.trace(quantized_net, input_mlu_data, check_trace=False)
```

增加mlu的推理方式

```
  # Run inference
  t0 = time.time()
  img = torch.zeros((1, 3, imgsz, imgsz), device=device) # init img
  _ = model(img.half() if half else img) if device.type != 'cpu' else None # run once
  for path, img, im0s, vid_cap in dataset:
    img_letter = letterbox0(im0s, new_shape=imgsz)[0]
    img = torch.from_numpy(img).to(device)
    img = img.half() if half else img.float() # uint8 to fp16/32
    img /= 255.0 # 0 - 255 to 0.0 - 1.0
    if img.ndimension() == 3:
      img = img.unsqueeze(0)

    # Inference
    t1 = time_synchronized()
    #pred = model(img, augment=opt.augment)[0]
    if opt.cfg == 'cpu':
      pred = model(img, augment=opt.augment)[0]
      print('run cpu')
    elif opt.cfg == 'qua':
      pred = quantized_model(img)[0]
      torch.save(quantized_model.state_dict(), 'weights/yolov5s_int8.pt')
      print('run qua')
    elif opt.cfg == 'mlu':
      img = img.type(torch.HalfTensor).to(ct.mlu_device())
      img = img.to(ct.mlu_device())
      pred = quantized_net(img)[0]
      pred=pred.data.cpu().type(torch.FloatTensor)
      box_result = get_boxes(pred)
      det = box_result[0].tolist()
      with open("result/yolov5s_mlu_output.txt","w+") as f:
        for *xyxy, conf, cls in reversed(det):
          f.write("{}\n{}\n{}\n{}\n".format(*xyxy))
          label = f'{names[int(cls)]} {conf:.2f}'
          plot_one_box(xyxy, img_letter, label=label, color=colors[int(cls)], line_thickness=1)
        import os
        cv2.imwrite("result/mlu_out_{}.jpg".format(os.path.basename(path).split('.')[0]), img_letter)
      img = img * 255
      print('run mlu')
    elif opt.cfg == 'cpu':
      pred = model(img, augment=opt.augment)[0]
      print('run cpu')
    if opt.cfg != 'cpu':
      continue

    # Apply NMS
```

执行

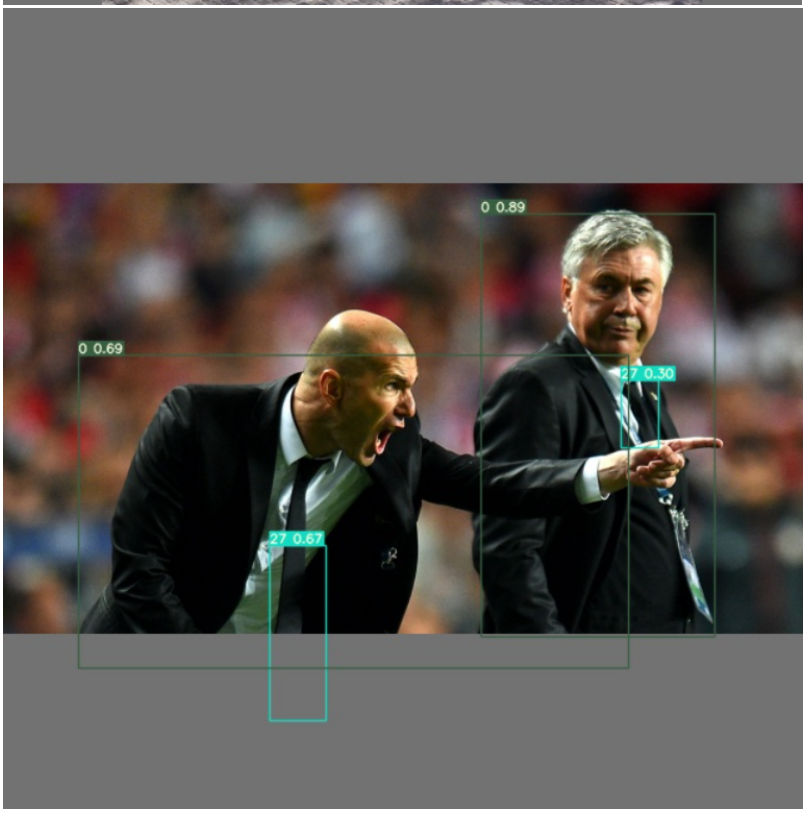
```
python detect.py --device cpu --weights weights/yolov5s_int8.pt --source data/images/ --cfg mlu
```

结果：

```
Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients
Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1    156928  models.common.C3                        [128, 128, 3]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  1    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1    229245  models.yolo.Detect                      [80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
Model Summary: 283 layers, 7276605 parameters, 7276605 gradients

Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients
batchNum: 1
image 1/2 /home/Cambricon-Test/yolov5_demo/data/images/bus.jpg: num_boxes_final:  5.0
run mlu
batchNum: 1
image 2/2 /home/Cambricon-Test/yolov5_demo/data/images/zidane.jpg: num_boxes_final:  4.0
run mlu
Results saved to result/exp3
Done. (11.260s)
```



融合

detect.py中增加部分代码，如下的opt.jit部分

```
  elif opt.cfg == 'mlu':
    from models.yolo import Model
    model = Model('./models/yolov5s.yaml').to(torch.device('cpu'))
    model.float().fuse().eval()
    quantized_net = torch_mlu.core.mlu_quantize.quantize_dynamic_mlu(model)
    #state_dict = torch.load("./yolov5s_int8.pt")
    state_dict = torch.load(weights[0])
    quantized_net.load_state_dict(state_dict, strict=False)
    quantized_net.eval()
    quantized_net.to(ct.mlu_device())



    if opt.jit:
      print("### jit")
      ct.save_as_cambricon('weights/yolov5s_int8_1_4')
      torch.set_grad_enabled(False)
      ct.set_core_number(4)
      trace_input = torch.randn(1, 3, 640, 640, dtype=torch.float)
      input_mlu_data = trace_input.type(torch.HalfTensor).to(ct.mlu_device())
      quantized_net = torch.jit.trace(quantized_net, input_mlu_data, check_trace=False)
```

执行

```
python detect.py --weights weights/yolov5s_int8.pt --cfg mlu --jit True
```

最后得到 yolov5s_int8_1_4.cambricon 离线模型

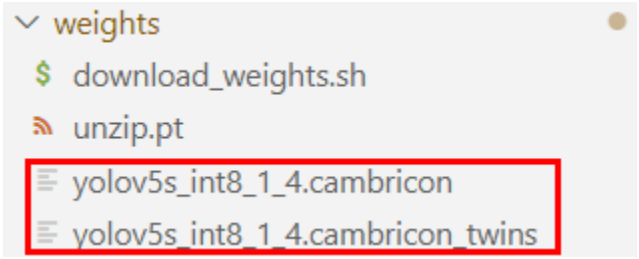
结果：

```
Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 229245 gradients
### jit
batchNum: 1
batchNum: 1
image 1/2 /home/Cambricon-Test/yolov5_demo/data/images/bus.jpg: num_boxes_final:  5.0
run mlu
[ERROR][/pytorch/catch/torch_mlu/csrc/jit/fuser/fused_kernel.cpp][line:44][CheckMFUSInputShape][thread:139855880419136][process:30076]:

The 0th input shape in fusion op is not equal to real input!
Please check the python code where the 'mfus forward' is called.

For example:
traced_model = torch.jit.trace(net, example_input, ...)
output1 = traced_model(real_input1)
output2 = traced_model(real_input2)
output3 = traced_model(real_input3)
...
The input shape of fusion op has been detetermined after traced_model.forward has been called at the first time.
Please note that real_input2.shape, real_input3.shape, ... should be equal to real_input1.shape!
If the real_input2, real_input3, ... are tuples of tensors, shape of each tensor in these tuples should be equal to the shape of corresponding tensor in real_input1.

image 2/2 /home/Cambricon-Test/yolov5_demo/data/images/zidane.jpg: num_boxes_final:  0.0
run mlu
Results saved to result/exp4
Done. (16.032s)
```



完整代码参考：http://gitlab.software.cambricon.com/neuware/software/ae/solution/270_sample/yolov5_demo