Caffe2笔记

个人 API 接口

一个观察：在Caffe2里面，一个网络net其实是一系列operator按照添加顺序组成的集合（可用`core.Net._net.op`获取所有op）。在执行网络时（`workspace.RunNet(some_net)`），按照添加的顺序逐个完成操作。这是最简单的一种模式，可在`core.Net._net.type`中进行设置，默认为`simple`（猜测为DAG模式）。反向传播可通过`core.Net.AddGradientOperator`添加进op集合，附在前向op的后面。

To Do

- [ ] 整理API

Caffe2基本使用流程

先介绍Caffe2的基本用法：

- **创造输入op和blob**：可使用`TensorProtoDBInput`等函数

```
# Add input to the net
def add_input(model, batch_size, db, db_type):
    ### load the data from db - Method 1 using brew
    #data_uint8, label = brew.db_input(
    #    model,
    #    blobs_out=["data_uint8", "label"],
    #    batch_size=batch_size,
    #    db=db,
    #    db_type=db_type,
    #)
    ### load the data from db - Method 2 using TensorProtosDB
    data_uint8, label = model.TensorProtosDBInput(
        [], ["data_uint8", "label"], batch_size=batch_size,
        db=db, db_type=db_type)
    # cast the data to float
    data = model.Cast(data_uint8, "data", to=core.DataType.FLOAT)
    # scale data from [0,255] down to [0,1]
    data = model.Scale(data, data, scale=float(1./256))
    # don't need the gradient for the backward pass
    data = model.StopGradient(data, data)
    return data, label
```

- **创造网络op和输出blob**：可使用`model.SOME_OP`的形式添加操作，或者使用`brew`接口定义网络各层。具体API待整理

```
# Define a mlp net
def mlp(model, data):
    dim_in = 28*28
    num_classes = 10
    fc1 = brew.fc(model, data, 'fc1', dim_in, 1024)
    fc1 = brew.relu(model, fc1, fc1)
    fc2 = brew.fc(model, fc1, 'fc2', 1024, 2048)
    fc2 = brew.relu(model, fc2, fc2)
    fc3 = brew.fc(model, fc2, 'fc3', 2048, num_classes)
    pred = brew.softmax(model, fc3, 'softmax')
    return pred
```

- **设定loss，指标与优化器**
- **训练**
- **保存，部署**

```
# Toy example: MNIST
import numpy as np
import os
import shutil
import operator
import caffe2.python.predictor.predictor_exporter as pe
import caffe2.proto as caffe2_pb2
from caffe2.python import (
    brew,
    core,
    model_helper,
    optimizer,
    visualize,
    workspace,
)

# Initialize caffe2 core
core.GlobalInit(['caffe2', '--caffe2_log_level=0'])
print("Necessities imported!")

# Get the mnist lmdb
db_url = "http://download.caffe2.ai/databases/mnist-lmdb.zip"
data_dir = 'path/to/the/dataset/dir/mnist'
output_dir = 'path/to/the/output/dir'
def download_mnist():
    '''Downloads resources from s3 by url and unzips them to the provided
 path'''
    import requests, zipfile, StringIO
    print("Downloading... {} to {}".format(db_url, data_dir))
    r = requests.get(db_url, stream=True)
    z = zipfile.ZipFile(StringIO.StringIO(r.content))
```

```
    z.extractall(data_dir)
    print("Completed download and extraction.")

# Get input
download_mnist()
arg_scope = {'order':'NCHW'}
gpu_id = 0
device_opt = caffe2_pb2.DeviceOption(caffe2_pb2.CUDA, gpu_id)
with core.DeviceScope(device_opt):
    train_model = model_helper.ModelHelper(name='train_model')
    train_data, train_label = add_input(train_model,
                        batch_size=64,
                        db=os.path.join(data_dir, 'mnist-train-nchw-
lmdb'),
                        db_type='lmdb')
    pred = mlp(train_model, train_data)
    xent = train_model.LabelCrossEntropy([pred, train_label], 'xent')
    loss = train_model.AveragedLoss(xent, 'loss')
    train_acc = brew.accuracy(train_model, [pred, train_label], 'train_ac
c')
    train_model.AddGradientOperator([loss])

    # Define optimizer
    optimizer.build_sgd(train_model,
                        base_learning_rate=1e-2,
                        policy='step',
                        stepsize=1,
                        gamma=0.999)

    # Initialize
    workspace.RunNetOnce(train_model.param_init_net)
    workspace.CreateNet(train_model.net, overwrite=True)

    # Train process
    max_epoch = 1000
    for epoch in range(max_epoch):
        workspace.RunNet(train_model)
        t_loss = workspace.blobs['loss']
        t_acc = workspace.blobs['train_acc']
        print 'Epoch {}, Loss {}, Acc {}'.format(epoch, t_loss, t_acc)

    # For test
    test_model = model_helper.ModelHelper(name='test_model')
    test_data, test_label = add_input(test_model,
                        batch_size=64,
                        db=os.path.join(data_dir, 'mnist-test-nchw-
lmdb'),
                        db_type='lmdb')
    pred = mlp(test_model, test_data)
    test_acc = brew.accuracy(test_model, [pred, test_label], 'test_acc')

    # Initialization
```

```
    workspace.RunNetOnce(test_model.param_init_net)
    workspace.CreateNet(test_model.net, overwrite=True)
    for epoch in range(100):
        workspace.RunNet(test_model)
        # te_acc = workspace.FetchBlob('test_acc')
        te_acc = workspace.blobs['test_acc']
        print 'Epoch {}, Acc {}'.format(epoch, te_acc)

    # Define the deploy net
    deploy_model = model_helper.ModelHelper(name='deploy_model')
    pred = mlp(deploy_model, 'data')

    # Save the model
    pe_meta = pe.PredictorExportMeta(
                predict_net=deploy_model.net.Proto(),
                parameters=[str(b) for b in deploy_model.params],
                inputs=["data"],
                outputs=["softmax"],
            )

    # Save the model to a file. Use minidb as the file format
    pe.save_to_db("minidb", os.path.join(root_folder, "mnist_model.minid
b"), pe_meta)
    t_data = workspace.blobs['data']

    # Reset the workspace, and load
    workspace.ResetWorkspace()
    # Load the model
    predict_net = pe.prepare_prediction_net(os.path.join(output_dir, "mni
st_model.minidb"), "minidb")
    workspace.FeedBlob('data', t_data)
    workspace.RunNetOnce(predict_net)
    t_pred = workspace.FetchBlob['softmax']
```

Caffe2安装

参考官方

Caffe2基础模块

core