Duino的工具箱

浪迹天涯的程序员

个人资料

DuinoDu

关注 发私信

访问： 120369次

积分： 2717

等级： BLOG 5

排名： 第15095名

原创： 122篇 转载： 0篇

ONNX demo

标签： facebook 移动

2017-10-02 00:35 314人阅读 评论(0) 收藏 举报

分类：

pytorch（5）

目录(?) [+]

ONNX是facebook AI部门那帮人搞出来的东西，可以方便的把 caffe2，然后就可以进行部署，尤其是可以部署到移动端。想 部署到android上，是不是很激动 ~

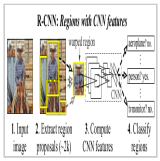
译文： 35篇 评论： 34条

博客专栏

AI看啥片

文章：1篇

阅读：384



目标检测

文章：6篇

阅读：8446

文章分类


```python
import io
import numpy as np
from torch import nn
from torch.autograd import Variable
import torch.utils.model_zoo as model_zoo
import torch.onnx
```

```python
# model definition
import torch.nn as nn
import torch.nn.init as init

class SRnet(nn.Module):
    def __init__(self, upscale_factor, inplace=False):
        super(SRnet, self).__init__()

        self.relu = nn.ReLU(inplace=True)
        self.conv1 = nn.Conv2d(1, 64, (5,5),(1,1),(2,2))
        self.conv2 = nn.Conv2d(64, 64, (3,3),(1,1),(1,1))
        self.conv3 = nn.Conv2d(64, 32, (3,3),(1,1),(1,1))
        self.conv4 = nn.Conv2d(32, upscale_factor ** 2, (3,3),(1,1),(1,
        self.pixel_shuffle = nn.PixelShuffle(upscale_factor)

        self._initialize_weights()

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.relu(self.conv3(x))
        x = self.pixel_shuffle(self.conv4(x))
        return x
```

```
24
25    def _initialize_weights(self):
26        init.orthogonal(self.conv1.weight, init.calculate_gain('relu'))
27        init.orthogonal(self.conv2.weight, init.calculate_gain('relu'))
28        init.orthogonal(self.conv3.weight, init.calculate_gain('relu'))
29        init.orthogonal(self.conv4.weight)
30
31  torch_model = SRnet(upscale_factor=3)
```

```
1  # load pretrained model
2  map_location = lambda storage, loc : storage # load to cpu
3  state_dict = torch.load('sr.pth', map_location=map_location)
4  torch_model.load_state_dict(state_dict)
5  torch_model.train(False)
```

```
SRnet (
  (relu): ReLU (inplace)
  (conv1): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(32, 9, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pixel_shuffle): PixelShuffle (upscale_factor=3)
)
```

上面这些内容，我们就完成了pytorch端的工作，接下来就要把
做“tracing”，具体实现的方式是，提供一个x，让x把整个网络
了哪些torch提供的operator。

```
1  batch_size = 1
2  x = Variable(torch.randn(batch_size, 1, 244, 244), requires_grad=T
```

```
torch_out = torch.onnx._export(torch_model,
                               x,
                               "super_resolution.onnx",
                               export_params=True)
```

torch_out输出没有什么特殊的用途，不过可以用来验证，pytorch和caffe2得到相同的结果，导出的模型存为文件，“super_resolution.onnx”。

```
import onnx
import onnx_caffe2.backend

# graph is a python protobuf object
# for different export dl platform, caffe2, cntk, mxnet, tf
# they all use protobuf object
graph = onnx.load("super_resolution.onnx")
prepared_backend = onnx_caffe2.backend.prepare(graph)

img_input = {graph.input[0]: x.data.numpy()}
c2_out = prepared_backend.run(img_input)[0]

np.testing.assert_almost_equal(torch_out.data.cpu().numpy(), c2_
```

到这里呢，我们成功地把pytorch定义的模型以及训练的参数
得很6呢？

在cpp_caffe2下运行

cpp使用官方提供的speed_benchmark.cc这个例程。我们先生成

```
c2_workspace = prepared_backend.workspace
c2_graph = prepared_backend.predict_net

from caffe2.python.predictor import mobile_exporter

init_net, predict_net = mobile_exporter.Export(c2_workspace, c2_graph, c2_graph.external_input)

with open('init_net.pd', 'wb') as f:
    f.write(init_net.SerializeToString())
with open('predict_net.pb', 'wb') as f:
    f.write(predict_net.SerializeToString())
```

可以看到，文件夹下面生成了**init_net.pb**和**predict_net.pb**，第一个文件是模型的参数文件，第二个文件是模型的定义文件。为什么这样呢？把模型的定义存为文件，这样模型的文件就是和平台无关了，pytorch和caffe2都可以使用这个文件，python和cpp代码都能使用这个文件，ubuntu和android也都能使用这个文件。

```
# Run on caffe2_python
from caffe2.proto import caffe2_pb2
from caffe2.python import core, net_drawer, net_printer, visualiz

import numpy as np
import os
import subprocess
from PIL import Image
from skimage import io, transform
```

```
img = Image.open('./cat_244x244.jpg')
img_ycbcr = img.convert('YCbCr')
```

```
img_y, img_cb, img_cr = img_ycbcr.split()

workspace.RunNetOnce(init_net)
workspace.RunNetOnce(predict_net)

print(net_printer.to_string(predict_net))
```

```
# net: torch-jit-export
11 = Conv(1, 2, kernels=[5L, 5L], strides=[1L, 1L], pads=[2L, 2L, 2L, 2L], dilations=[1L, 1L], group=1)
12 = Add(11, 3, broadcast=1, axis=1)
13 = Relu(12)
15 = Conv(13, 4, kernels=[3L, 3L], strides=[1L, 1L], pads=[1L, 1L, 1L, 1L], dilations=[1L, 1L], group=1)
16 = Add(15, 5, broadcast=1, axis=1)
17 = Relu(16)
19 = Conv(17, 6, kernels=[3L, 3L], strides=[1L, 1L], pads=[1L, 1L, 1L, 1L], dilations=[1L, 1L], group=1)
20 = Add(19, 7, broadcast=1, axis=1)
21 = Relu(20)
23 = Conv(21, 8, kernels=[3L, 3L], strides=[1L, 1L], pads=[1L, 1L, 1L, 1L], dilations=[1L, 1L], group=1)
24 = Add(23, 9, broadcast=1, axis=1)
25, _onnx_dummy1 = Reshape(24, shape=[1L, 1L, 3L, 3L, 244L, 244L])
26 = Transpose(25, axes=[0L, 1L, 4L, 2L, 5L, 3L])
27, _onnx_dummy2 = Reshape(26, shape=[1L, 1L, 732L, 732L])
```

```
# feed input
workspace.FeedBlob('1', np.array(img_y)[np.newaxis, np.newaxis
# forward net
workspace.RunNetOnce(predict_net)
# fetch output
img_out = workspace.FetchBlob('27')
```

```
# save output to image
img_out_y = Image.fromarray(np.uint8(img_out[0,0]).clip(0,255),
final_img = Image.merge(
```

```
    'YCbCr', [
        img_out_y,
        img_cb.resize(img_out_y.size, Image.BICUBIC),
        img_cr.resize(img_out_y.size, Image.BICUBIC),
    ]).convert('RGB')
final_img.save('./cat_superres.jpg')
```

```
# prepare input blob
with open('input.blobproto', 'wb') as f:
    f.write(workspace.SerializeBlob('1'))
```

编译cpp代码

我们需要编译cpp的代码，使用如下编译命令：

```
CAFFE2_ROOT=$HOME/src/caffe2
g++ speed_benchmark.cc -o demo -std=c++11 \
    -I $CAFFE2_ROOT/third_party/eigen \
    -lCaffe2_CPU \
    -lglog \
    -lgflags \
    -lprotobuf \
    -lpthread \
    -llmdb \
    -lleveldb \
    -lopencv_core \
    -lopencv_highgui \
    -lopencv_imgproc
```

能够使用这条命令的前提是，caffe2安装到了 /usr/local 下，使用了 sudo make install 进行安装。

运行cpp程序：

```
./demo --init_net init_net.pd --net predict_net.pb --input 1 --input_file input.blobproto --output_folder . --outpu
```

等一下，我们先看一眼 speed_benchmark.cc

```
#include <string>

#include "caffe2/core/init.h"
#include "caffe2/core/operator.h"
#include "caffe2/proto/caffe2.pb.h"
#include "caffe2/utils/proto_utils.h"
#include "caffe2/utils/string_utils.h"
#include "caffe2/core/logging.h"

// 定义args
CAFFE2_DEFINE_string(net, "", "The given net to benchmark.");
CAFFE2_DEFINE_string(init_net, "",
             "The given net to initialize any parameters.");
CAFFE2_DEFINE_string(input, "",
             "Input that is needed for running the network. If "
             "multiple input needed, use comma separated string
CAFFE2_DEFINE_string(input_file, "",
             "Input file that contain the serialized protobuf for "
             "the input blobs. If multiple input needed, use comn
             "separated string. Must have the same number of ite
```

```
21                "as input does.");
22  CAFFE2_DEFINE_string(input_dims, "",
23                "Alternate to input_files, if all inputs are simple "
24                "float TensorCPUs, specify the dimension using comma "
25                "separated numbers. If multiple input needed, use "
26                "semicolon to separate the dimension of different "
27                "tensors.");
28  CAFFE2_DEFINE_string(output, "",
29                "Output that should be dumped after the execution "
30                "finishes. If multiple outputs are needed, use comma "
31                "separated string. If you want to dump everything, pass "
32                "'*' as the output value.");
33  CAFFE2_DEFINE_string(output_folder, "",
34                "The folder that the output should be written to. This "
35                "folder must already exist in the file system.");
36  CAFFE2_DEFINE_int(warmup, 0, "The number of iterations to warm up.");
37  CAFFE2_DEFINE_int(iter, 10, "The number of iterations to run.");
38  CAFFE2_DEFINE_bool(run_individual, false, "Whether to benchmark individual operators.");
39
40  using std::string;
41  using std::unique_ptr;
42  using std::vector;
43
44  int main(int argc, char** argv) {
45    caffe2::GlobalInit(&argc, &argv);
46    unique_ptr<caffe2::Workspace> workspace(new caffe2::Worksp
47
48    // 读取模型参数到工作空间
49    caffe2::NetDef net_def;
50    CAFFE_ENFORCE(ReadProtoFromFile(caffe2::FLAGS_init_net, &ne
51    CAFFE_ENFORCE(workspace->RunNetOnce(net_def));
```

```
52
53  // 加载输入数据，提供两种方式，--input和--input_dims
54  if (caffe2::FLAGS_input.size()) {
55    vector<string> input_names = caffe2::split(',', caffe2::FLAGS_input);
56    if (caffe2::FLAGS_input_file.size()) {
57      vector<string> input_files = caffe2::split(',', caffe2::FLAGS_input_file);
58      CAFFE_ENFORCE_EQ(
59          input_names.size(), input_files.size(),
60          "Input name and file should have the same number.");
61      for (int i = 0; i < input_names.size(); ++i) {
62        caffe2::BlobProto blob_proto;
63        CAFFE_ENFORCE(caffe2::ReadProtoFromFile(input_files[i], &blob_proto));
64        workspace->CreateBlob(input_names[i])->Deserialize(blob_proto);
65      }
66    } else if (caffe2::FLAGS_input_dims.size()) {
67      vector<string> input_dims_list = caffe2::split(';', caffe2::FLAGS_input_dims);
68      CAFFE_ENFORCE_EQ(
69          input_names.size(), input_dims_list.size(),
70          "Input name and dims should have the same number of items.");
71      for (int i = 0; i < input_names.size(); ++i) {
72        vector<string> input_dims_str = caffe2::split(',', input_dims_li
73        vector<int> input_dims;
74        for (const string& s : input_dims_str) {
75          input_dims.push_back(caffe2::stoi(s));
76        }
77        caffe2::TensorCPU* tensor =
78            workspace->GetBlob(input_names[i])->GetMutable<caffe2
79        tensor->Resize(input_dims);
80        tensor->mutable_data<float>();
81      }
82    } else {
```

```
83      CAFFE_THROW("You requested input tensors, but neither input_file nor "
84                  "input_dims is set.");
85    }
86  }
87
88  // 加载模型定义文件，创建模型，
89  CAFFE_ENFORCE(ReadProtoFromFile(caffe2::FLAGS_net, &net_def));
90  caffe2::NetBase* net = workspace->CreateNet(net_def);
91  CHECK_NOTNULL(net);
92  net->TEST_Benchmark(
93      caffe2::FLAGS_warmup,
94      caffe2::FLAGS_iter,
95      caffe2::FLAGS_run_individual);
96
97  // 获得输出
98  string output_prefix = caffe2::FLAGS_output_folder.size()
99      ? caffe2::FLAGS_output_folder + "/"
100     : "";
101 if (caffe2::FLAGS_output.size()) {
102   vector<string> output_names = caffe2::split(',', caffe2::FLAGS_o
103   if (caffe2::FLAGS_output == "*") {
104     output_names = workspace->Blobs();
105   }
106   for (const string& name : output_names) {
107     CAFFE_ENFORCE(
108         workspace->HasBlob(name),
109         "You requested a non-existing blob: ",
110         name);
111     string serialized = workspace->GetBlob(name)->Serialize(nam
112     string output_filename = output_prefix + name;
113     caffe2::WriteStringToFile(serialized, output_filename.c_str());
```

```
114     }
115   }
116   return 0;
117 }
```

程序运行的结果是，生成了一个 27 文件，我们用python把这个文件转换为jpg。

```
1  blob_proto = caffe2_pb2.BlobProto()
2  blob_proto.ParseFromString(open('./27_mobile').read())
3  img_out = utils.Caffe2TensorToNumpyArray(blob_proto.tensor)
4  img_out_y = Image.fromarray(np.uint8((img_out[0,0]).clip(0,255)), mode='L')
5  final_img = Image.merge(
6    "YCbCr", [
7        img_out_y,
8        img_cb.resize(img_out_y.size, Image.BICUBIC),
9        img_cr.resize(img_out_y.size, Image.BICUBIC),
10   ]).convert('RGB')
11 final_img.save('./cat_superres_mobile.jpg')
```

关于放到android上运行

按照原教程的做法，可以顺利运行。这里写几点值得注意的点
- android编译的可执行程序是静态编译，以便方便地在手机设
- android程序并不是一定要用java写，这个例子便是用cpp写的
目真得很方便。
- 不过，如果仅仅在android上运行控制台程序，那也太不爽了
呢？所以，最后一定要是放到一个有界面的app中运行，这个
- 关于android环境配置，最快的方式莫过于，装一个 android s

- 上一篇 IR in deep learning
- 下一篇 [1609.04802] SRGAN中的那些loss

暂无评论

发表评论

用 户 名： weixin_35068028

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场