TR-15444 MobileNet codebase and onnx model

http://jira.enflame.cn/browse/TR-15444?filter=11453

参考

需求

参考下tops models里的mobilenet v2训练和推理结果，

参考torch vision里的结果 v1 or v2?

基于onnx file的推理前后处理及精度验证代码

dtu推理 使用tops infernece来dump hlir hlo

提供onnx文件 opset 13, fp32 and fp16

onnx 文件建议上传到 http/ftp server （ beijing & shanghai）

http://10.16.11.32/inference/Dorado/common/

推理patch

http://gerrit.enflame.cn/c/tops/+/35527

http://gerrit.enflame.cn/c/tops/+/36400

参考resnet torchvision

http://jira.enflame.cn/browse/TR-15359

http://git.enflame.cn/sw/tops/tree/master/models/TopsModels/research/inference/onnx/torchvision_classification/fp16

How to run topsinference to dump related HLO

工作目录

V100 /home/devdata/xiaoying.zhang/DORADO/common/mobilenet_v1

/data/srv/ftp/software/inference/Dorado/common

```
scp -r ./ app@10.16.11.32:/data/srv/ftp/software/inference/

scp -r mobilenet_v1_keras_* root@10.8.50.39:/devdata/liguo.wang/tops/models/TopsModels/research
/inference/topsexec/models/xiaoying-models/

scp -r mobilenet_v1_keras-op13-fp* root@10.8.50.39:/devdata/liguo.wang/tops/models/TopsModels
/research/inference/topsexec/models/xiaoying-models/
```

mobilenetv2

- 导出onnx文件

```
python pytorch2onnx.py --model mobilenet_v2 --opset 13 --dynamic-bs
python pytorch2onnx.py --model mobilenet_v2 --opset 13 --batchsize 1
```

- 验证FP32/FP16模型精度

```
python test_onnx.py --model mobilenet_v2

python test_onnx.py --model mobilenet_v2 --test-fp16

fp32:

acc1 = 0.71878
acc5 = 0.90286
fps = 83.691390

fp16:

acc1 = 0.7186
acc5 = 0.90314
fps = 26.292333
```

- onnx fp32 转 fp16

```
python convert_fp16.py --model mobilenet_v2
```

- dump mlir

```
export INTERNAL_FMK_SIM=DORADO

exportCOMPILE_OPTIONS_MLIR_DBG="-pass-timing-pass-statistics-print-ir-after=hlir-fusion-print-ir-after=llir-mem-static-alloc-mlir-elide-elementsattrs-if-larger=409600-log-output-path=models/xiaoying-models/irdump-fp16/"

如果模型是动态bs的dump的时候需要加上 - inputShape=2,3,224,224即可

./topsexec --device=0 --onnx=models/xiaoying-models/mobilenet-v2-op13-fp32.onnx --cluster=0 --input=input,fp32 --output=output,fp32 --iterations=30 --warmup=5 --streams=4 --threads=1
./topsexec --device=0 --onnx=models/xiaoying-models/mobilenet-v2-op13-fp32.onnx --cluster=0 --input=input,fp16 --output=output,fp16 --iterations=30 --warmup=5 --streams=4 --threads=1 --inputShape=1,3,224,224
```

mobilenetv3

参考：MobileNet V3 精度方案制定

- 导出onnx

```
官方参考：https://pytorch.org/vision/stable/models.html#classification

python pytorch2onnx.py --model mobilenet_v3_large --opset 13 --batchsize 1

python pytorch2onnx.py --model mobilenet_v3_small --opset 13 --batchsize 1
```

- 验证FP32模型精度

```
python test_onnx.py --model mobilenet_v3_large


acc1 = 0.7406
acc5 = 0.9133
fps = 38.253119

python test_onnx.py --model mobilenet_v3_small


acc1 = 0.67658
acc5 = 0.87422
fps = 75.050098
```

- fp32 onnx转fp16

```
python convert_fp16.py --model mobilenet_v3_large

python convert_fp16.py --model mobilenet_v3_small
```

- 验证FP16模型精度

```
python test_onnx.py --model mobilenet_v3_large --test-fp16



acc1 = 0.74066
acc5 = 0.9132
fps = 22.364632

python test_onnx.py --model mobilenet_v3_small --test-fp16



acc1 = 0.67648
acc5 = 0.87414
fps = 41.573567
```

- dump mlir

```
v100 cd /home/devdata/xiaoying.zhang/DORADO/common

scp -r mobilenet_v3_* root@10.8.50.39:/devdata/liguo.wang/tops/models/TopsModels/research
/inference/topsexec/models/xiaoying-models/



conda activate tops

cd /devdata/liguo.wang/tops/models/TopsModels/research/inference/topsexec



export INTERNAL_FMK_SIM=DORADO

export COMPILE_OPTIONS_MLIR_DBG="-pass-timing -pass-statistics -print-ir-after=hlir-fusion -
print-ir-after=llir-mem-static-alloc -mlir-elide-elementsattrs-if-larger=409600 -log-output-
path=models/xiaoying-models/mobilenet_v3_large-irdump/"

./topsexec --device=0 --onnx=models/xiaoying-models/mobilenet_v3_large-op13-fp32.onnx --
cluster=0 --input=input,fp32 --output=output,fp32 --iterations=30 --warmup=5 --streams=4 --
threads=1

export INTERNAL_FMK_SIM=DORADO

export COMPILE_OPTIONS_MLIR_DBG="-pass-timing -pass-statistics -print-ir-after=hlir-fusion -
print-ir-after=llir-mem-static-alloc -mlir-elide-elementsattrs-if-larger=409600 -log-output-
path=models/xiaoying-models/mobilenet_v3_small-irdump/"

./topsexec --device=0 --onnx=models/xiaoying-models/mobilenet_v3_small-op13-fp32.onnx --
cluster=0 --input=input,fp32 --output=output,fp32 --iterations=30 --warmup=5 --streams=4 --
threads=1
```

mobilenetv1

- 官方指标结果

Pre-trained Models

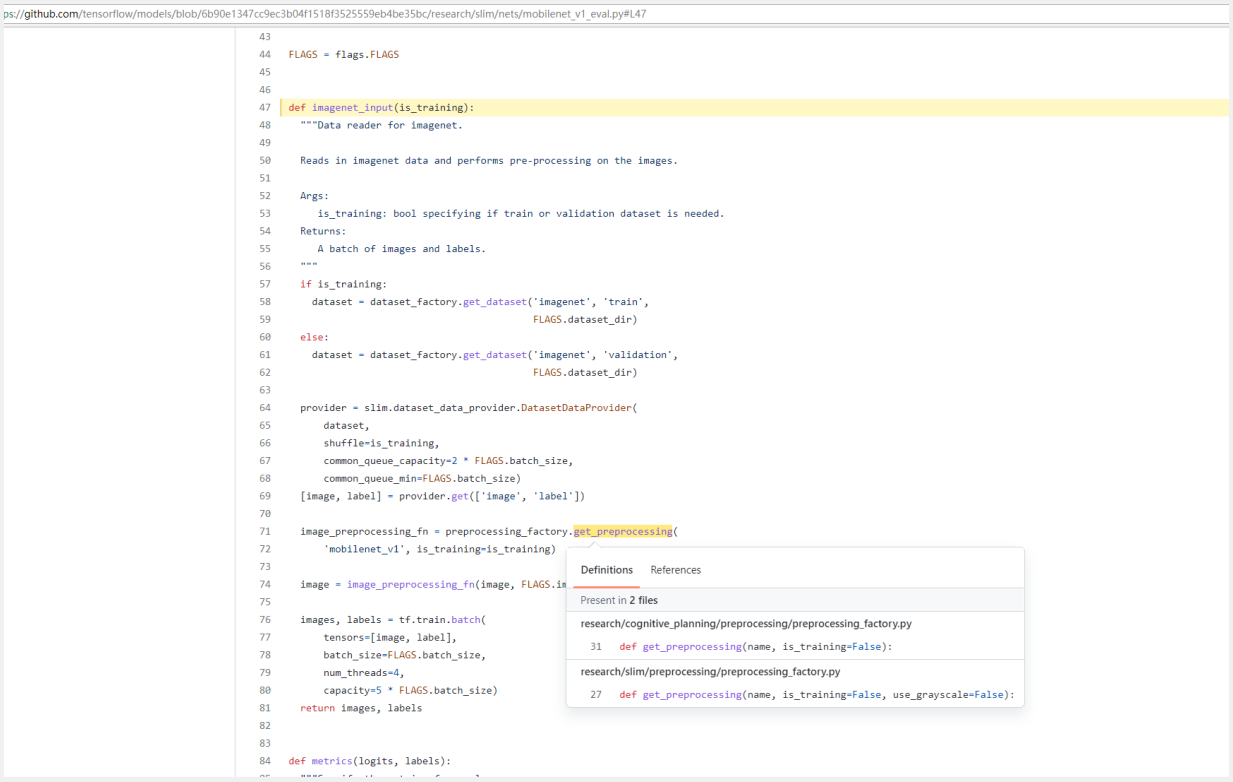
Choose the right MobileNet model to fit your latency and size budget. The size of the network in memory and on c
number of parameters. The latency and power usage of the network scales with the number of Multiply-Accumulat
the number of fused Multiplication and Addition operations. These MobileNet models have been trained on the ILS
classification dataset. Accuracies were computed by evaluating using a single image crop.

Model	Million MACs	Million Parameters	Top-1 Accuracy	Top-5 Accuracy
MobileNet_v1_1.0_224	569	4.24	70.9	89.9
MobileNet_v1_1.0_192	418	4.24	70.0	89.2

官方预处理方式

tensorflow中mobilenetv1使用的预处理方式映射的是inception的预处理方式

https://github.com/tensorflow/models/blob/master/research/slim/preprocessing/inception_preprocessing.py



- 导出onnx

方法1：tf-model convert，以下方法均为使用官方slim的模型

MobileNet V1官方预训练模型的使用 https://cloud.tencent.com/developer/article/1356892

预训练模型：wget http://download.tensorflow.org/models/mobilenet_v1_2018_08_02/mobilenet_v1_1.0_224.tgz

pb2onnx-inference https://zenodo.org/record/3157894#.YT8eWfkzYuU（直接提供onnx，但是opset是8）

https://gist.github.com/guschmue/788ae7f602c1f15ce3998b8d5f56ed2e（这里将opset设置为13）

预处理：

结果

fp32	fp16
acc1 = 0.71228 acc5 = 0.89836	acc1 = 0.71228 acc5 = 0.89864

方法2：keras-model convert

```
from keras.applications.mobilenet import MobileNet

model = MobileNet(include_top=True, weights='imagenet')
```

keras2onnx：https://github.com/onnx/keras-onnx

Use the following script to convert keras application models to onnx, and then perform inference:

```
import numpy as np
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input
import keras2onnx
import onnxruntime

# image preprocessing
img_path = 'street.jpg'   # make sure the image is in img_path
img_size = 224
img = image.load_img(img_path, target_size=(img_size, img_size))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# load keras model
from keras.applications.resnet50 import ResNet50
model = ResNet50(include_top=True, weights='imagenet')

# convert to onnx model
onnx_model = keras2onnx.convert_keras(model, model.name)

# runtime prediction
content = onnx_model.SerializeToString()
sess = onnxruntime.InferenceSession(content)
x = x if isinstance(x, list) else [x]
feed = dict([(input.name, x[n]) for n, input in enumerate(sess.get_inputs())])
pred_onnx = sess.run(None, feed)
```

The inference result is a list which aligns with keras model prediction result `model.predict()`. An alternative way to load onnx model to runtime session is to save the model first:

```
temp_model_file = 'model.onnx'
keras2onnx.save_model(onnx_model, temp_model_file)
sess = onnxruntime.InferenceSession(temp_model_file)
```

预处理：

结果：

onnx_model: mobilenet_v1_keras-op13-fp32.onnx acc1 = 0.70334 acc5 = 0.89392	onnx_model: mobilenet_v1_keras-op13-fp16.onnx acc1 = 0.70352 acc5 = 0.89194

- 验证FP32模型精度

```
mobilenet_v1_tf-op13-fp32.onnx

mobilenet_v1_keras-op13-fp32.onnx

python test_onnx.py --model mobilenet_v1_tf
```

- fp32 onnx转fp16

```
python convert_fp16.py --model mobilenet_v1_tf

python convert_fp16.py --model mobilenet_v1_keras
```

- dump hlir

```
conda activate tops

cd /devdata/liguo.wang/tops/models/TopsModels/research/inference/topsexec


export INTERNAL_FMK_SIM=DORADO

export COMPILE_OPTIONS_MLIR_DBG="-pass-timing -pass-statistics -print-ir-after=hlir-fusion -print-ir-after=llir-mem-static-alloc -mlir-elide-elementsattrs-if-larger=409600 -log-output-path=models/xiaoying-models/mobilenet_v1_tf-irdump/"

./topsexec --device=0 --onnx=models/xiaoying-models/mobilenet_v1_tf-op13-fp32.onnx --cluster=0 --input=input,fp32 --output=output,fp32 --iterations=30 --warmup=5 --streams=4 --threads=1

export INTERNAL_FMK_SIM=DORADO

export COMPILE_OPTIONS_MLIR_DBG="-pass-timing -pass-statistics -print-ir-after=hlir-fusion -print-ir-after=llir-mem-static-alloc -mlir-elide-elementsattrs-if-larger=409600 -log-output-path=models/xiaoying-models/mobilenet_v1_keras-irdump/"

./topsexec --device=0 --onnx=models/xiaoying-models/mobilenet_v1_kerassubgraph-op13-fp32.onnx --cluster=0 --input=input,fp32 --output=output,fp32 --iterations=30 --warmup=5 --streams=4 --threads=1
```

汇总

参考代码

http://git.enflame.cn/sw/tops/tree/master/models/TopsModels/research/inference/onnx/torchvision_classification/fp16

model	url	base	数据预处理	fp32	fp16
mobilenet_v1	from keras.applications.mobilenet import MobileNet	keras	keras preprocess_input  评估中不使用softmax，因为模型中存在softmax的节点	acc1 = 0.70334 acc5 = 0.89392	acc1 = 0.70352 acc5 = 0.89194

mobilenet_v1	https://cloud.tencent.com/developer/article/1356892 http://download.tensorflow.org/models/mobilenet_v1_2018_08_02/mobilenet_v1_1.0_224.tgz https://github.com/tensorflow/models/blob/master/research/slim/	tensorflow slim官方	inception  评估中不使用softmax，因为模型中存在softmax的节点 注意labelid是否与模型中的预测id对齐 tf-slim中label-id加1，和模型预测结果对齐 	fp32 acc1 = 0.71228 acc5 = 0.89836	fp16 acc1 = 0.71228 acc5 = 0.89864
mobilenet_v2		torchvision	torchvision 分类模型数据预处理	acc1 = 0.71878 acc5 = 0.90286	acc1 = 0.7186 acc5 = 0.90314
mobilenet_v3-large		torchvision	torchvision 分类模型数据预处理	acc1 = 0.7406 acc5 = 0.9132	acc1 = 0.74066 acc5 = 0.9132
mobilenet_v3-small		torchvision	torchvision 分类模型数据预处理	acc1 = 0.67658 acc5 = 0.87422	acc1 = 0.67648 acc5 = 0.87414