CS6886 : Assignment 1

Aghin Shah Alin K
CS17B109

February 15, 2020

Part A

Sources and example runs

Model	Dataset	Source
Resnet50-v1.	ImageNet(224x224)	torchvision modelzoo
MobileNets-v1 224	ImageNet(224x224)	mmdnn
SSD-ResNet34	COCO(1200x1200)	Link
SSD-MobileNets-v1	COCO(300x300)	Link
GNMT	WMT16	torch,onnx

Please see

- ./supplements/InferenceExamples.pdf for example outputs of both torch,onnx models.
- ./supplements/run_all.mp4 for screencast of console on running models.

Device details

Dev1	GTX 1070 Max Q
Dev2	Redmi Note 7
Dev3	Iphone XS Max
Dev4	i7-750
Dev5	RTX 2060

- Dev1
  - CPU : Intel i7-8750H
  - Cores : 6
  - Threads per core : 2
  - Clockspeed :
    - CPU MHz: 834.984
    - CPU max MHz:4100.0000
    - CPU min MHz:800.0000
  - RAM : 16 gb
  - L1d cache: 32K
  - L1i cache: 32K
  - L2 cache: 256K
  - L3 cache: 9216K

  - GPU : GTX 1070 Max Q,8gb

- Dev2
  - Chipset : Qualcomm Snapdragon 660 MSM8956
  - Graphics : Adreno 512
  - Processor : Octa core (2.2 GHz, Quad core, Kryo 260 + 1.8 GHz, Quad core, Kryo 260)
  - Architecture : 64 bit
  - Ram : 3 GB

- Dev3
  - Chipset : Apple A12 Bionic
  - Graphics : Apple GPU (four-core graphics)
  - Processor : Hexa Core (2.49 GHz, Dual core, Vortex + 1.52 GHz, Quad core, Tempest)
  - Coprocessor : M12
  - Architecture : 64 bit
  - Ram : 4 GB

- Dev4
  - CPU : Intel i7-7500
  - Cores : 6
  - Threads per core : 2
  - Clockspeed : 2.8 GHz
  - RAM : 8 gb
  - L1d cache: 32K
  - L1i cache: 32K
  - L2 cache: 256K
  - L3 cache: 9216K

- Dev5
  - CPU : Intel i7-9750H
  - Cores : 6
  - Threads per core : 2
  - Clockspeed :
    - * CPU MHz: 834.984
    - * CPU max MHz:4100.0000
    - * CPU min MHz:800.0000
  - RAM : 16 gb
  - L1d cache: 32K
  - L1i cache: 32K
  - L2 cache: 256K
  - L3 cache: 12 MB
  - GPU : RTX 2060,6 gb

Inference times

Mean,Standard Deviation is reported for 29 runs,the first run has been excluded because it's comparitively higher than the other due to caching and other warm ups. For ssd_resnet34_1200 all the runs take equal time,since the model is large its not cached completly.See ./supplements/inference_data for device details and data from each.

Model	mean	std
mobilenet_v1_224	46.8966	35.4433
ssd_mobileNet_v1_300	75.2069	1.26979
ssd_resnet34_1200	6784.34	83.3672
resnet_v1.5_224	112.379	1.69001

Table 1: dev1

Model	mean	std
mobilenet_v1_224	2972.52	23.1273
ssd_mobileNet_v1_300	6604.55	16.831
resnet_v1.5_224	2311.97	12.7075

Table 2: dev2

Model	mean	std
mobilenet_v1_224	206.69	5.724
ssd_mobileNet_v1_300	631.931	8.295
resnet_v1.5_224	253.759	4.796

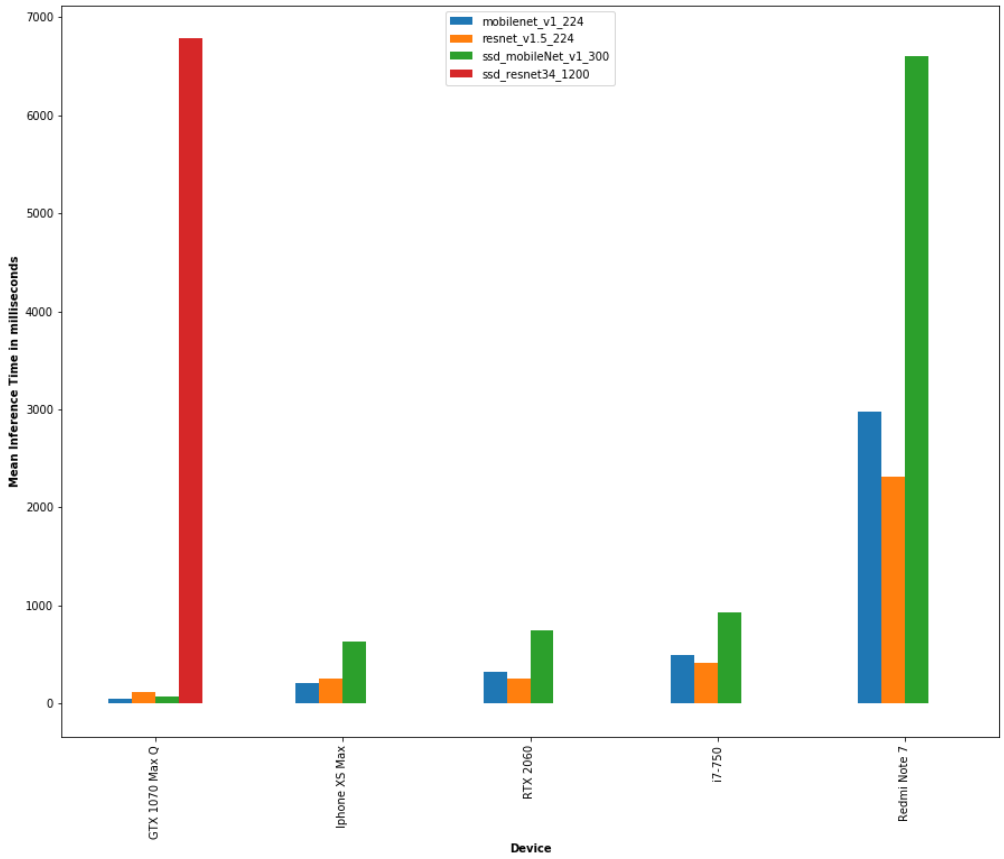
Table 3: dev3

Model	mean	std
mobilenet_v1_224	490.759	29.09
ssd_mobileNet_v1_300	933.69	110.825
resnet_v1.5_224	411.724	66.65

Table 4: dev4

Model	mean	std
mobilenet_v1_224	323.9	61.70
ssd_mobileNet_v1_300	749.2	139.198
resnet_v1.5_224	255.7	53.47

Table 5: dev5



(a) Mean Inference Time vs Device

Correlations

	dev1	dev3	dev5	dev4	dev2
dev1	1	-0.692056	-0.713916	-0.801255	-0.7254
dev3	-0.692056	1	0.986368	0.974104	0.98417
dev5	-0.713916	0.986368	1	0.99097	0.999807
dev4	-0.801255	0.974104	0.99097	1	0.993068
dev2	-0.7254	0.98417	0.999807	0.993068	1

(a) Pearson's correlation coefficient between devices

Top 5 most correlated(absolute) devices

dev5	dev2	0.999807
dev4	dev2	0.993068
dev5	dev4	0.990970
dev3	dev5	0.986368
dev3	dev2	0.984170

	mobilenet_v1_224	resnet_v1.5_224	ssd_mobileNet_v1_300	ssd_resnet34_1200
mobilenet_v1_224	1	0.999344	0.999049	-0.348606
resnet_v1.5_224	0.999344	1	0.999155	-0.336652
ssd_mobileNet_v1_300	0.999049	0.999155	1	-0.35617
ssd_resnet34_1200	-0.348606	-0.336652	-0.35617	1

(a) Pearson's correlation coefficient between models

Most correlated(absolute) models

mobilenet_v1_224	resnet_v1.5_224	0.999344
resnet_v1.5_224	ssd_mobileNet_v1_300	0.999155
mobilenet_v1_224	ssd_mobileNet_v1_300	0.999049
ssd_mobileNet_v1_300	ssd_resnet34_1200	0.356170
mobilenet_v1_224	ssd_resnet34_1200	0.348606
resnet_v1.5_224	ssd_resnet34_1200	0.336652

Part B

Changes are made to *./lib/execution-plan.ts* in root folder of onnx repo (see fork).See ./supplements/with_log_layer.mp4 for example run with log of time taken per layer.

```
92  93          const inputTensors = inputList as Tensor[];
93      -       Logger.verbose(
94      -           'ExecPlan',
95      -           `Runing op:${thisOp.node.name} (${
96      -               inputTensors.map((t, i) => `'${thisOp.node.inputs[i]}': ${t.type}[${t.dims.join(',')}]`).join(', ')})`
    94  +       const operationInfo = `Runing op:${thisOp.node.name} (${
    95  +           inputTensors.map((t, i) => `'${thisOp.node.inputs[i]}': ${t.type}[${t.dims.join(',')}]`).join(', ')})`;
    96  +       Logger.verbose('ExecPlan', operationInfo);
97  97
98  98          const outputList = await this.profiler.event('node', thisOp.node.name, async () => {
99  99            const op = thisOp.op;
    100 +          console.log('<ExecPlan> Running OP :', operationInfo);
100 101           if (!op.checkInputs(inputTensors)) {
101 102             throw new Error(`invalid inputs detected; op: ${thisOp.node.name}`);
102 103           }
103     -
    104 +         const start = new Date();
104 105           const result = op.run(inferenceHandler, inputTensors);
105     -
    106 +         const end = new Date();
    107 +         console.log('<ExecPlan> Time Taken:', end.getTime() - start.getTime());
106 108           return result;
107 109         });
```

End