Assignment #5: TensorRT

Hanyang University

Sooyoung Kim [redacted], Gaon Choi [redacted]

Introduction

TensorRT is a high-performance neural network inference optimizer and runtime engine. It focuses on running a pre-trained network quickly and efficiently, using a GPU for the purpose of generating a result. Also, it automatically applies Network compression, network optimization, and GPU optimization techniques to deep learning models to achieve optimal inference performance on the NVIDIA platform.

Deep learning acceleration techniques include:

- Quantization & Precision Calibration
- Graph Optimization

FLOW: Model trained by PyTorch --> ONNX --> TensorRT engine

GITHUB LINK: https://github.com/Gaon-Choi/PBL4009/tree/main/lab4%20-%20opencv%20tutorial

Experiment

1. Implementation

1) Using loaded Resnet50 model, inference the image

```
input = pre_process_image("poodle.jpg").cuda()
torch_start = time.time()

# model = models.resnet50(pretrained=True)
# model = models.mobilenet_v2(pretrained=True)
model = models.densenet161(pretrained=True)

model.eval()
model.cuda()

torch_start = time.time()

############torch inference############
output = model(input)
#######################################
print("\ntorch inference time:", time.time() - torch_start, " second\n")
postprocess(output)
```

2) Export the model in onnx format by touch.onnx.export

```
if not os.path.exists(ONNX_FILE_PATH):
    #Export ONNX file with trained parameters
    # modify the blue-colored variables and fill the blank with the code below.
    torch.onnx.export(model, input, ONNX_FILE_PATH, input_names=['input'],
output_names=['output'], export_params=True)
```

3) Change the ONNX to TensorRT inference engine

3-1. Define a build_engine function

```
def build_engine(onnx_file_path):
    # initialize TensorRT engine and parse ONNX model
    builder = trt.Builder(TRT_LOGGER)

    config = builder.create_builder_config()

    # allow TensorRT to use up to 1GB of GPU memory for tactic selection
    config.max_workspace_size = 1 << 28
    builder.max_batch_size = 1


    explicit_batch = 1 << (int)(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH) # When
netowrk can't read model

    #Create Network with builder
    network = builder.create_network(explicit_batch)
    # builder.create_network(EXPLICIT_BATCH)

    #Define onnx parser
    parser = trt.OnnxParser(network, TRT_LOGGER)


    # parse ONNX
    with open(onnx_file_path, 'rb') as model:
        print('Beginning ONNX file parsing')
        parser.parse(model.read())
    print('Completed parsing of ONNX file')
```

3-2. Inference with TensorRT engine

```
    # we have only one image in batch
    builder.max_batch_size = 1
    # use FP16 mode if possible
    #if builder.platform_has_fast_fp16:
        #config.set_flag(trt.BuilderFlag.FP16)

    # generate TensorRT engine optimized for the target platform
    print('Building an engine...')

    #Build cuda engine
    engine = builder.build_engine(network, config)
    #Engine create execution context
    engine.create_execution_context()

    print("Completed creating Engine")

    return engine
```

2. Result

1) Inference Accuracy(Top-1)[1]

1-1) Resnet50, ONNX

```
$ python3 torch_to_onnx.py
class: toy poodle , confidence: 64.12075805664062 %, index: 265
class: miniature poodle , confidence: 31.424909591674805 %, index: 266
class: teddy, teddy bear , confidence: 0.999542772769928 %, index: 850
class: standard poodle , confidence: 0.7201401591300964 %, index: 267
```

1-2) Resnet50, TRT

```
$ python3 onnx_to_tensorrt.py
class: doormat, welcome mat , confidence: 8.759943962097168 %, index: 539
class: paddle, boat paddle , confidence: 7.53119421005249 %, index: 693
class: folding chair , confidence: 6.754096508026123 %, index: 559
class: umbrella , confidence: 4.506896495819092 %, index: 879
class: pelican , confidence: 4.37526273727417 %, index: 144
class: black swan, Cygnus atratus , confidence: 4.20759916305542 %, index: 100
class: goose , confidence: 3.181190013885498 %, index: 99
class: American egret, great white heron, Egretta albus , confidence:
2.7166895866394043 %, index: 132
class: bib , confidence: 2.3839876651763916 %, index: 443
class: bathing cap, swimming cap , confidence: 2.296194553375244 %, index: 433
class: park bench , confidence: 2.186353921890259 %, index: 703
class: shopping basket , confidence: 2.1555368900299072 %, index: 790
class: bucket, pail , confidence: 2.100956439971924 %, index: 463
class: window shade , confidence: 2.002927303314209 %, index: 905
class: shopping cart , confidence: 1.8548719882965088 %, index: 791
class: drake , confidence: 1.731899619102478 %, index: 97
class: little blue heron, Egretta caerulea , confidence: 1.5547678470611572 %, index:
131
class: American coot, marsh hen, mud hen, water hen, Fulica americana , confidence:
1.4757874011993408 %, index: 137
class: beaver , confidence: 1.4517337083816528 %, index: 337
class: prayer rug, prayer mat , confidence: 1.309253215789795 %, index: 741
class: window screen , confidence: 1.2502630949020386 %, index: 904
class: platypus, duckbill, duckbilled platypus, duck-billed platypus, Ornithorhynchus
anatinus , confidence: 1.2200331687927246 %, index: 103
class: flamingo , confidence: 1.082710862159729 %, index: 130
class: bubble , confidence: 1.039042592048645 %, index: 971
class: seashore, coast, seacoast, sea-coast , confidence: 0.8206286430358887 %, index:
978
class: paintbrush , confidence: 0.8160631060600281 %, index: 696
class: tile roof , confidence: 0.7294796705245972 %, index: 858
class: plastic bag , confidence: 0.7241808176040649 %, index: 728
class: ashcan, trash can, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin,
trash barrel, trash bin , confidence: 0.6946664452552795 %, index: 412
class: canoe , confidence: 0.6628563404083252 %, index: 472
class: European gallinule, Porphyrio porphyrio , confidence: 0.656722366809845 %, index:
136
class: binder, ring-binder , confidence: 0.5969184041023254 %, index: 446
```

[1] The threshold of probability was set to 0.5

```
class: crane , confidence: 0.5793968439102173 %, index: 134
class: spoonbill , confidence: 0.5254786014556885 %, index: 129
class: water snake , confidence: 0.5025461912155151 %, index: 58
```

1-3) MobileNetV2, ONNX

```
$ python3 torch_to_onnx.py
class: toy poodle , confidence: 37.83669662475586 %, index: 265
class: miniature poodle , confidence: 25.23346519470215 %, index: 266
class: teddy, teddy bear , confidence: 7.828701972961426 %, index: 850
class: standard poodle , confidence: 2.6951522827148438 %, index: 267
class: Kerry blue terrier , confidence: 2.4472973346710205 %, index: 183
class: Lakeland terrier , confidence: 1.9715156555175781 %, index: 189
class: Bedlington terrier , confidence: 1.6448564529418945 %, index: 181
class: Shih-Tzu , confidence: 1.581188440322876 %, index: 155
class: Lhasa, Lhasa apso , confidence: 1.3606950044631958 %, index: 204
class: Bouvier des Flandres, Bouviers des Flandres , confidence: 1.0191229581832886 %,
index: 233
class: Maltese dog, Maltese terrier, Maltese , confidence: 0.8467124700546265 %, index:
153
class: binoculars, field glasses, opera glasses , confidence: 0.5601708889007568 %,
index: 447
class: miniature schnauzer , confidence: 0.5213196873664856 %, index: 196
class: Tibetan terrier, chrysanthemum dog , confidence: 0.5107740163803101 %, index: 200
class: cocker spaniel, English cocker spaniel, cocker , confidence:
0.5064408183097839 %, index: 219
```

1-4) MobileNetV2, TRT

```
$ python3 onnx_to_tensorrt.py
class: prayer rug, prayer mat , confidence: 8.307268142700195 %, index: 741
class: barn , confidence: 7.298793792724609 %, index: 425
class: tray , confidence: 4.825915336608887 %, index: 868
class: umbrella , confidence: 3.8814592361450195 %, index: 879
class: window shade , confidence: 3.726536512374878 %, index: 905
class: balloon , confidence: 3.6739070415496826 %, index: 417
class: doormat, welcome mat , confidence: 3.656536817550659 %, index: 539
class: seashore, coast, seacoast, sea-coast , confidence: 3.2139017581939697 %, index:
978
class: rubber eraser, rubber, pencil eraser , confidence: 2.699558734893799 %, index:
767
class: pole , confidence: 2.3170664310455322 %, index: 733
class: envelope , confidence: 2.0502676963806152 %, index: 549
class: carousel, carrousel, merry-go-round, roundabout, whirligig , confidence:
1.926021695137024 %, index: 476
class: parachute, chute , confidence: 1.9239976406097412 %, index: 701
class: matchstick , confidence: 1.8810913562774658 %, index: 644
class: theater curtain, theatre curtain , confidence: 1.8175075054168701 %, index: 854
class: paintbrush , confidence: 1.7265100479125977 %, index: 696
class: studio couch, day bed , confidence: 1.6302412748336792 %, index: 831
class: paddle, boat paddle , confidence: 1.505131483078003 %, index: 693
class: handkerchief, hankie, hanky, hankey , confidence: 1.2134952545166016 %, index:
591
class: pencil box, pencil case , confidence: 1.1891766786575317 %, index: 709
class: binder, ring-binder , confidence: 1.123469591140747 %, index: 446
class: web site, website, internet site, site , confidence: 1.0871928930282593 %, index:
916
class: bib , confidence: 1.0608808994293213 %, index: 443
class: radio, wireless , confidence: 0.8987323641777039 %, index: 754
```

```
class: throne , confidence: 0.8008816242218018 %, index: 857
class: dishrag, dishcloth , confidence: 0.7023650407791138 %, index: 533
class: beacon, lighthouse, beacon light, pharos , confidence: 0.6210627555847168 %,
index: 437
class: church, church building , confidence: 0.606907069683075 %, index: 497
class: boathouse , confidence: 0.6016800403594971 %, index: 449
class: chime, bell, gong , confidence: 0.5528774857521057 %, index: 494
class: harmonica, mouth organ, harp, mouth harp , confidence: 0.5424392819404602 %,
index: 593
class: sombrero , confidence: 0.539147138595581 %, index: 808
class: lampshade, lamp shade , confidence: 0.508216381072998 %, index: 619
class: quilt, comforter, comfort, puff , confidence: 0.5001009106636047 %, index: 750
```

2) Execution Time

2-1) ResNet50, ONNX

```
$ python3 torch_to_onnx.py

torch inference time: 38.00408482551575  second
```

2-2) ResNet50, TRT

```
$ python3 onnx_to_tensorrt.py

trt inference time:  0.1712965965270996 second
```

2-3) MobileNetV2, ONNX

```
$ python3 torch_to_onnx.py

torch inference time: 16.99754810333252  second
```

2-4) MobileNetV2, TRT

```
$ python3 onnx_to_tensorrt.py

trt inference time:  0.04459214210510254 second
```

3) Print Log and Analyze the model structure(compared to PyTorch MobileNetV2 structure)

3-1) Log

3-1-1. ResNet50

```
[TensorRT] INFO: [MemUsageChange] Init CUDA: CPU +198, GPU +0, now: CPU 270, GPU 3220 (MiB)
[TensorRT] INFO: Loaded engine size: 153 MB
[TensorRT] INFO: [MemUsageSnapshot] deserializeCudaEngine begin: CPU 424 MiB, GPU 3459 MiB
[TensorRT] VERBOSE: Using cublas a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +158, GPU +157, now: CPU 582, GPU 3616
(MiB)
[TensorRT] VERBOSE: Using cuDNN as a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuDNN: CPU +241, GPU +238, now: CPU 823, GPU 3854 (MiB)
[TensorRT] INFO: [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +0, now: CPU 823, GPU 3854
(MiB)
[TensorRT] VERBOSE: Deserialization required 3501568 microseconds.
[TensorRT] INFO: [MemUsageSnapshot] deserializeCudaEngine end: CPU 823 MiB, GPU 3854 MiB
[TensorRT] INFO: [MemUsageSnapshot] ExecutionContext creation begin: CPU 669 MiB, GPU 3701 MiB
```

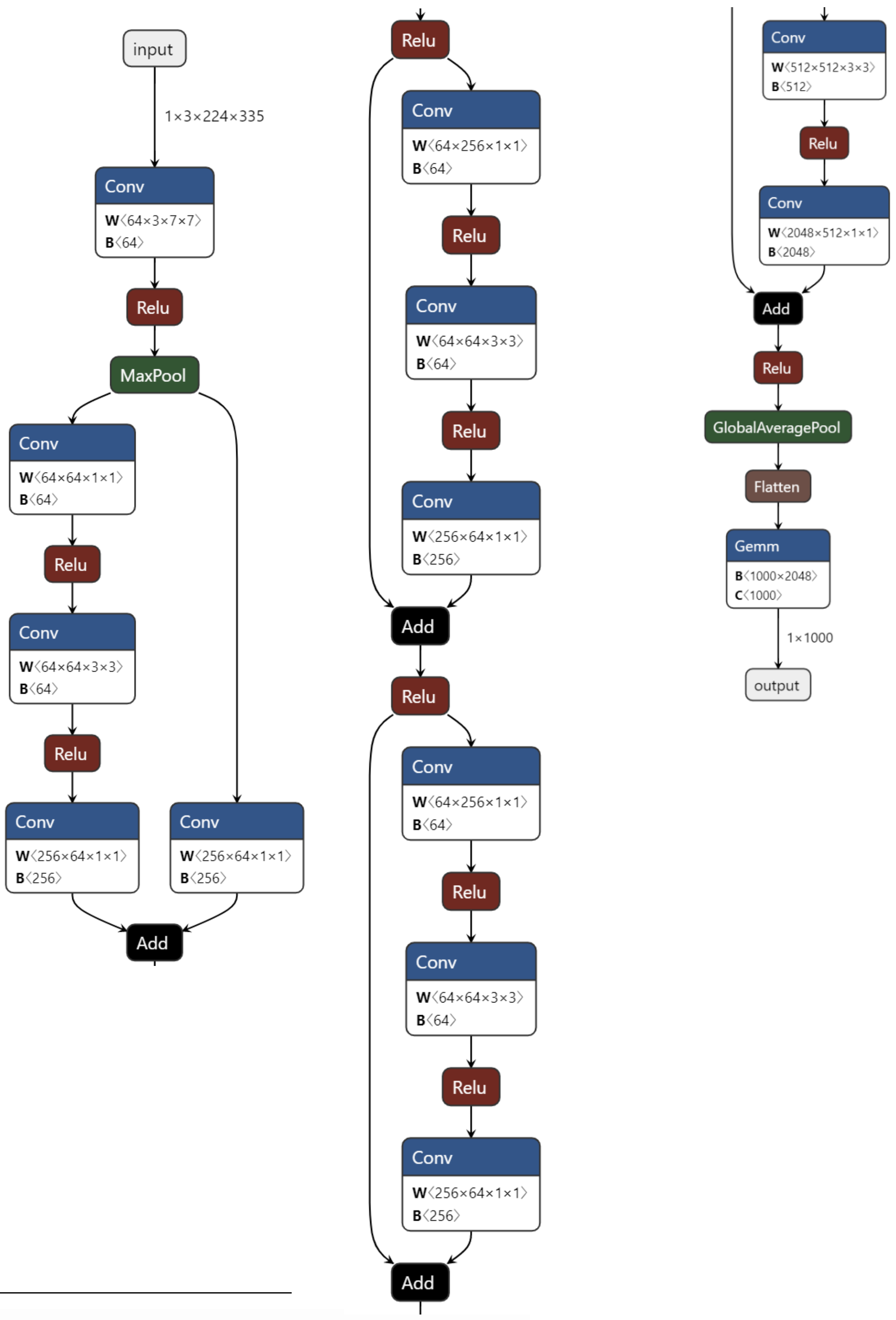
```
[TensorRT] VERBOSE: Using cublas a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +0, now: CPU 669, GPU 3701
(MiB)
[TensorRT] VERBOSE: Using cuDNN as a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuDNN: CPU +0, GPU +0, now: CPU 669, GPU 3701 (MiB)
[TensorRT] VERBOSE: Total per-runner device memory is 143390720
[TensorRT] VERBOSE: Total per-runner host memory is 129632
[TensorRT] VERBOSE: Allocated activation device memory of size 10838016
[TensorRT] INFO: [MemUsageSnapshot] ExecutionContext creation end: CPU 671 MiB, GPU 3753 MiB
```

3-1-2. MobileNetV2

```
[TensorRT] INFO: [MemUsageChange] Init CUDA: CPU +198, GPU +0, now: CPU 270, GPU 3206 (MiB)
[TensorRT] INFO: Loaded engine size: 14 MB
[TensorRT] INFO: [MemUsageSnapshot] deserializeCudaEngine begin: CPU 284 MiB, GPU 3233 MiB
[TensorRT] VERBOSE: Using cublas a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +158, GPU +169, now: CPU 442, GPU 3402
(MiB)
[TensorRT] VERBOSE: Using cuDNN as a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuDNN: CPU +241, GPU +239, now: CPU 683, GPU 3641 (MiB)
[TensorRT] INFO: [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +0, now: CPU 683, GPU 3641
(MiB)
[TensorRT] VERBOSE: Deserialization required 2880029 microseconds.
[TensorRT] INFO: [MemUsageSnapshot] deserializeCudaEngine end: CPU 683 MiB, GPU 3641 MiB
[TensorRT] INFO: [MemUsageSnapshot] ExecutionContext creation begin: CPU 669 MiB, GPU 3628 MiB
[TensorRT] VERBOSE: Using cublas a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +0, now: CPU 669, GPU 3628
(MiB)
[TensorRT] VERBOSE: Using cuDNN as a tactic source
[TensorRT] INFO: [MemUsageChange] Init cuDNN: CPU +0, GPU +0, now: CPU 669, GPU 3628 (MiB)
[TensorRT] VERBOSE: Total per-runner device memory is 9074176
[TensorRT] VERBOSE: Total per-runner host memory is 110608
[TensorRT] VERBOSE: Allocated activation device memory of size 11590656
[TensorRT] INFO: [MemUsageSnapshot] ExecutionContext creation end: CPU 671 MiB, GPU 3635 MiB
```

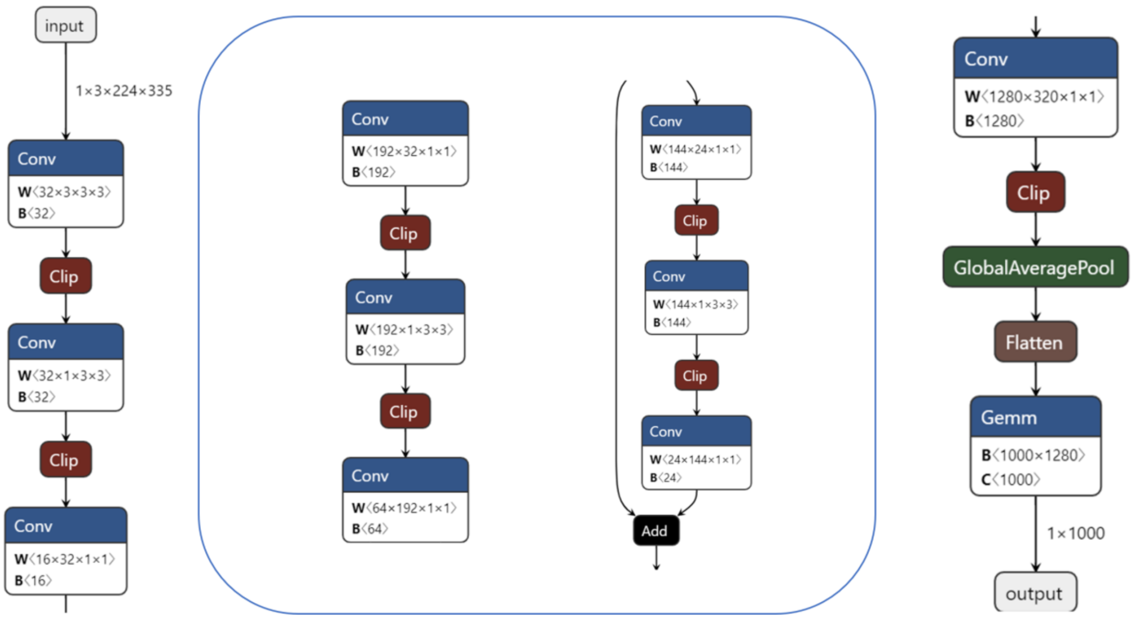
3-2) Analysis of model structure

3-2-1. ResNet50[2]



[2] Repeated parts are removed, and only key parts are marked.

3-2-2. MobileNetV2



Conclusion

We learned how to run a pre-trained network quickly and efficiently. It was useful to convert our model to ONNX format, to yield an accelerated output with TensorRT. With TRT, we can observe the elapsed time for inference decreased with higher rate.