Presentation of Progress

2022.07.18 Mon.

홍 세 현 Hong Sehyun

STRADVISION

Outline

1. Project Introduction
2. Project Progress[1] – Faster R-CNN
3. Project Progress[2] – Quantization
4. Future Work
5. Q & A

Project Introduction

PyTorch Based Quantization Aware Training of Two Stage Object Detection Model

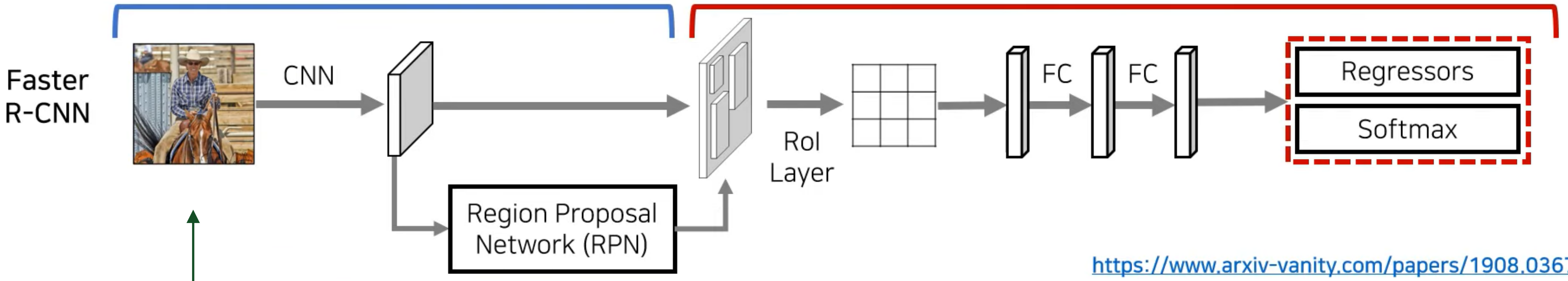
PyTorch

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with 'attention' mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (*including all steps*) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

Index Terms—Object Detection, Region Proposal, Convolutional Neural Network.

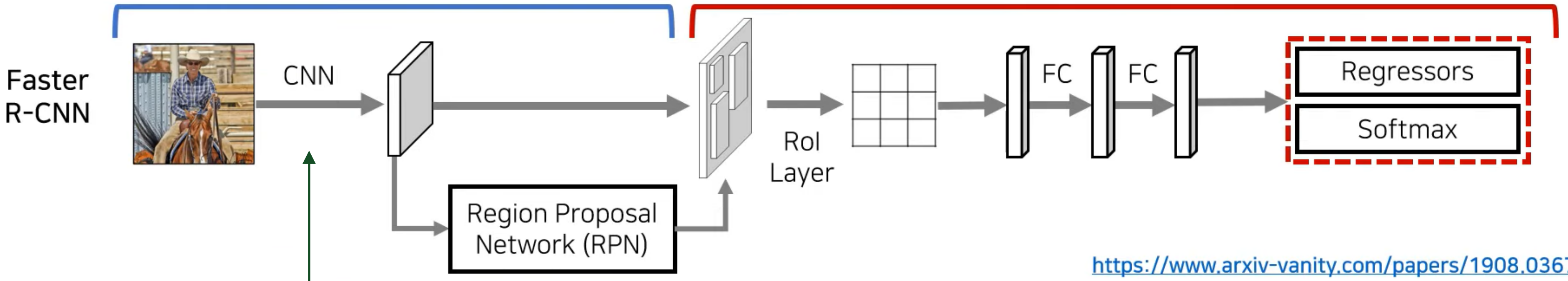


https://www.arxiv-vanity.com/papers/1908.03673/

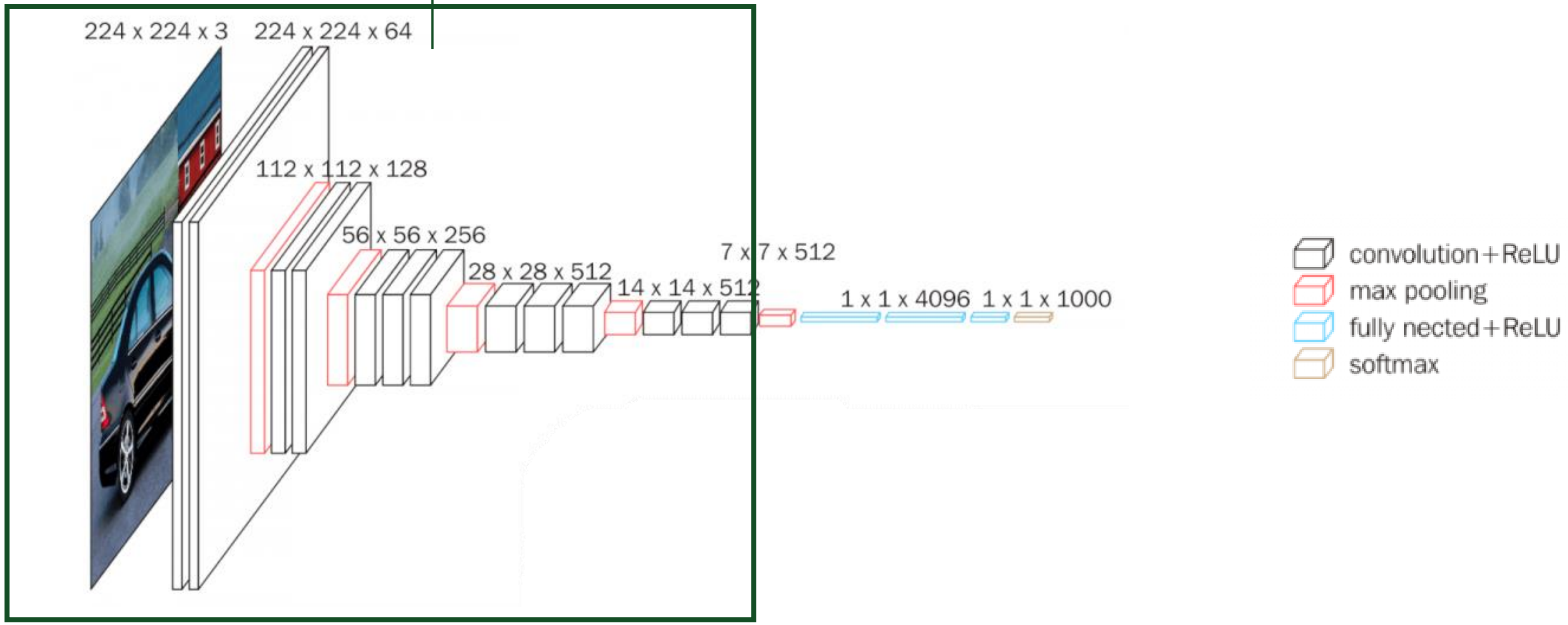
2007	20 classes: • *Person:* person • *Animal:* bird, cat, cow, dog, horse, sheep • *Vehicle:* aeroplane, bicycle, boat, bus, car, motorbike, train • *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor Train/validation/test: 9,963 images containing 24,640 annotated objects.

Train : Valid : Test = 1 : 1 : 2 → **TrainVal : Test = 1 : 1**

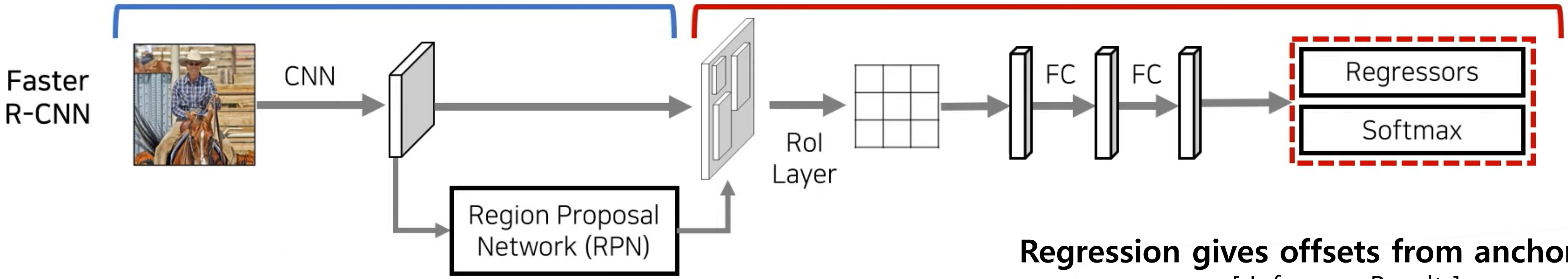
Project Progress[1] – Faster R-CNN



https://www.arxiv-vanity.com/papers/1908.03673/

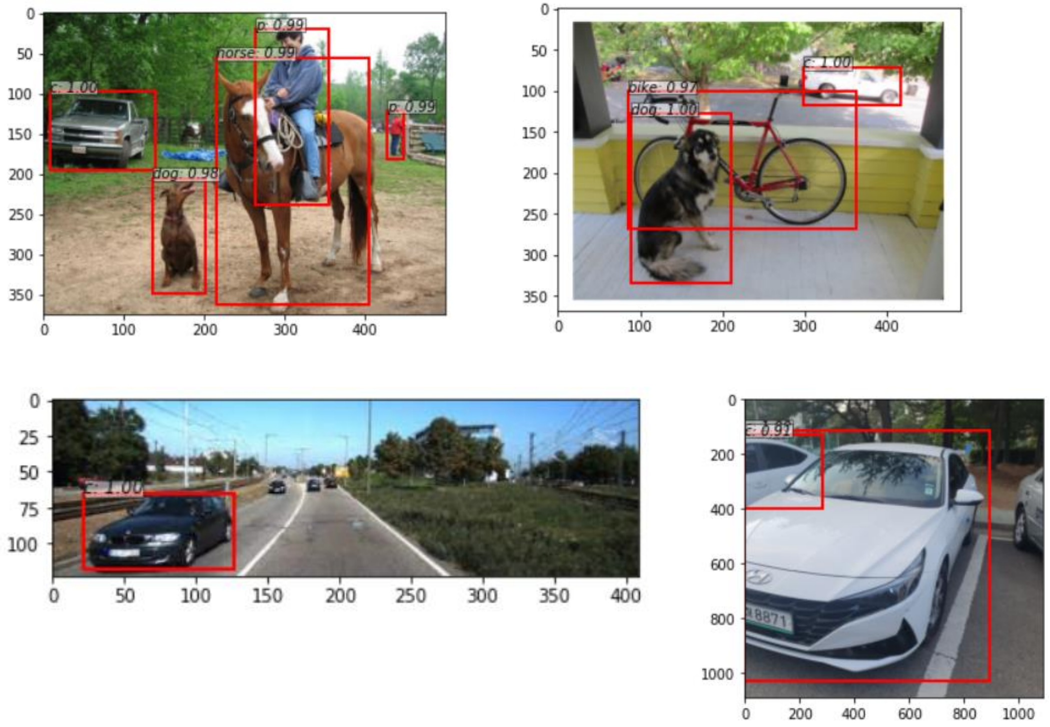


Project Progress[1] – Faster R-CNN



Regression gives offsets from anchor boxes

[Inference Result]



① **RPN Cls Loss**

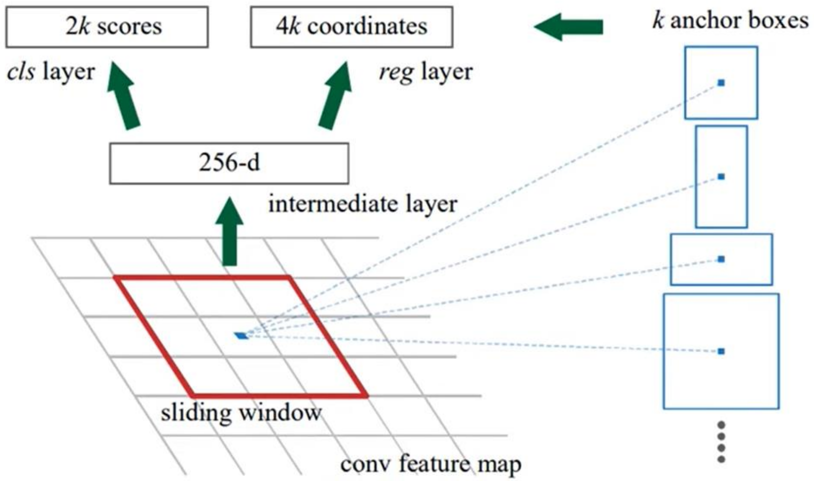
Classify

Obj. / NOT-obj.

② **RPN Reg Loss**

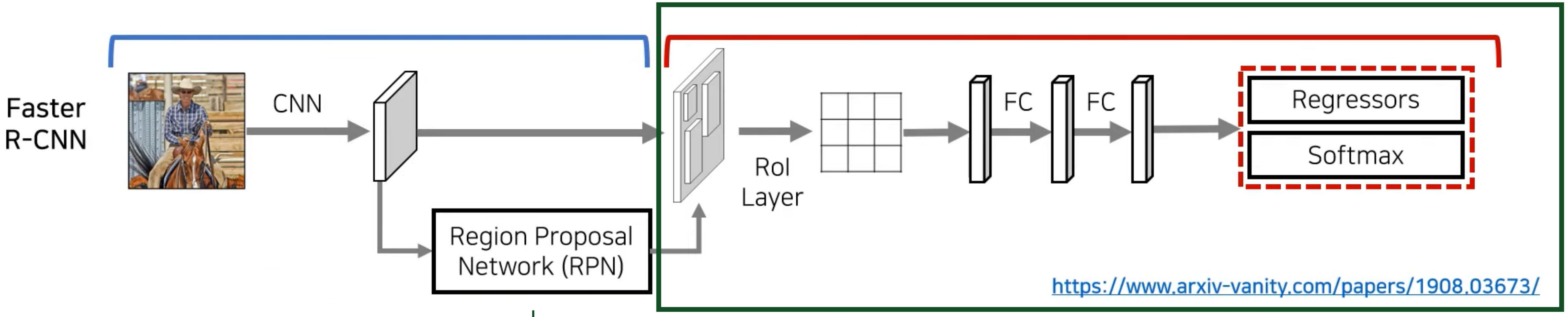
Regression

Tuning Box Locations



Using *k anchor Boxes* at each Location

Project Progress[1] – Faster R-CNN



Fast R-CNN

① RPN Cls Loss

② RPN Reg Loss

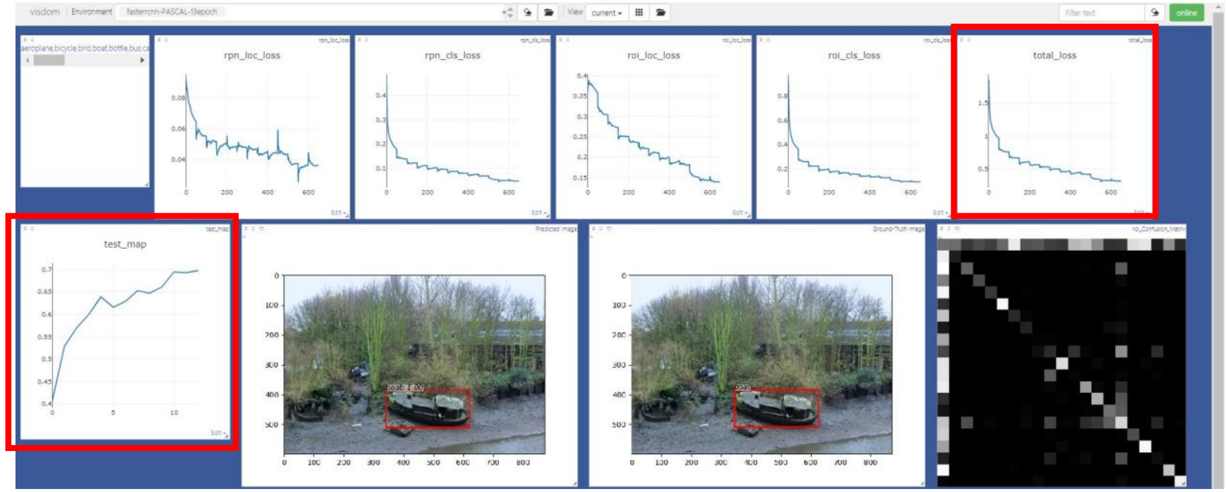
③ ROI Cls Loss

④ ROI Reg Loss

Total Loss = ① + ② + ③ + ④

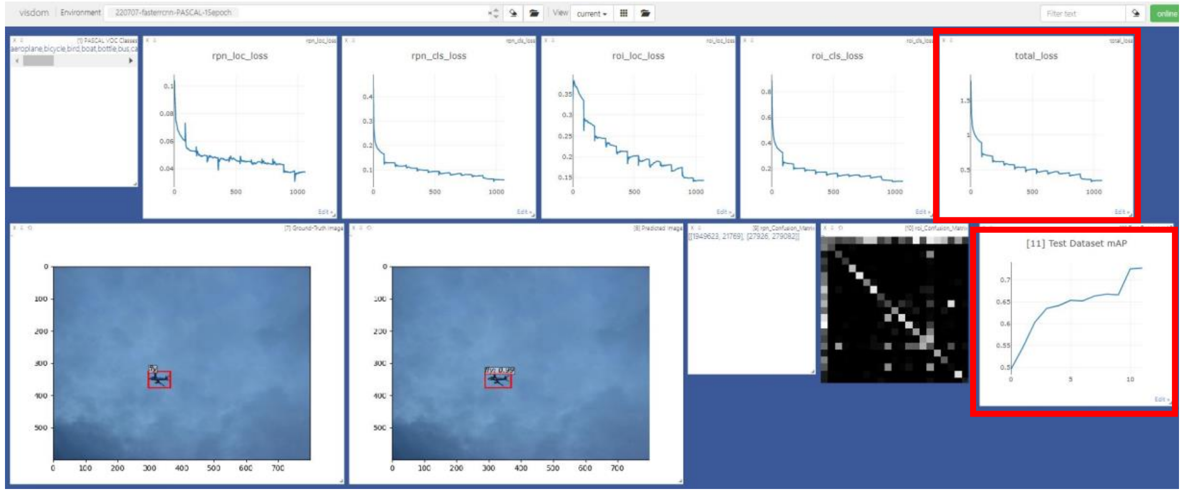
Other Parameters	
Batch Size	1
Epoch	15
Learning Rate	0.001
IoU Threshold Value	0.7

Project Progress[1] – Faster R-CNN



TrainVal : Test = 5 : 5

mAP ≅ 0.69



TrainVal : Test = 9 : 1

mAP ≅ 0.74

Project Progress[2] – Quantization

while models are becoming more efficient, high accuracy still implies high complexity!

WORKFLOWS

	Quantization	Dataset Requirements	Works Best For	Accuracy	Notes
Dynamic Quantization	weights only (both fp16 and int8)	None	LSTMs, MLPs, Transformers	good	Suitable for dynamic models (LSTMs), Close to static post training quant when performance is compute bound or memory bound due to weights.
Static Post Training Quantization	weights and activations (8 bit)	calibration	CNNs	good	Suitable for static models, provides best perf
Static Quantization-Aware Training	weights and activations (8 bit)	fine-tuning	CNNs	best	Requires fine tuning of model, currently supported only for static quantization.

Project Progress[2] – Quantization

Post Training Static Quantization WorkFlow

1. Modify Model (Layer Fusion)
2. Prepare and Calibrate
3. Convert
4. Deploy

```
 # original model
# all tensors and computations are in floating point
previous_layer_fp32 -- linear_fp32 -- activation_fp32 -- next_layer_fp32
                     /
    linear_weight_fp32

# statically quantized model
# weights and activations are in int8
previous_layer_int8 -- linear_with_activation_int8 -- next_layer_int8
                     /
  linear_weight_int8
```

```
[14]
print_model_size(model_vgg)
print_model_size(model_vgg_int8)
```

```
58.87 MB
14.83 MB
```

Project Progress[2] – Quantization

Quantization Aware Training WorkFlow

1. Model Load
2. Layer Fusion
3. QuantStub / DeQuantStub
4. Quantization Configuration
5. CUDA, QAT Training
6. Change Float to Int

```
 # original model
# all tensors and computations are in floating point
previous_layer_fp32 -- linear_fp32 -- activation_fp32 -- next_layer_fp32
                     /
    linear_weight_fp32

# model with fake_quants for modeling quantization numerics during training
previous_layer_fp32 -- fq -- linear_fp32 -- activation_fp32 -- fq -- next_layer_fp32
                          /
   linear_weight_fp32 -- fq

# quantized model
# weights and activations are in int8
previous_layer_int8 -- linear_with_activation_int8 -- next_layer_int8
                     /
   linear_weight_int8
```

Future Work

To Do LIST

- Train New Faster R-CNN Framework
- Apply Static Quantization to Faster R-CNN Model
- Apply Quantization Aware Training to Faster R-CNN Model

Q & A

Thank you ☺