스프린트 미션 16

미션 소개

지난 미션을 수행하며 구현한 모델들을 다시 가져와서, 여러 형태의 포맷으로 모델을 변환하여 저장해보는 실습 (모델은 직접 학습하거나 사전 학습된 모델을 fine-tuning 해도 무방)

가이드라인

1. 이전 미션에서 다루었던 모델들 중 하나 이상을 자유롭게 선택하여 모델 학습을 진행합니다.
2. 아래의 3가지 타입의 모델로 변환하여 저장해봅시다.
    - `.pth` (PyTorch 기본 저장 형식)
    - `.pth` (양자화 된 버전)
    - `.onnx` (ONNX 형식)
3. ONNX 파일을 기반으로 하여 각각의 모델을 테스트할 수 있는 추론 코드를 작성합니다. 모델에 적절한 평가 데이터셋을 사용해 동작을 검증해야 합니다.

modeling.ipynb

데이터셋

- CIFAR 10
- train 데이터를 train 과 val로 split(8:2)
- train : 40000 / val : 10000
- 이미지 : 32 -> 128 resize
- RandomHorizontalFlip(p = 0.5)
- 정규화
- batch_size = 32 / 32

학습 모델

- VGG11-BN (ImageNet1K 사전학습 가중치)
- 입력 크기에 맞춰 avgpool / classifier 재구성
- Full Fine-Tuning

```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(vgg.parameters(), lr=0.005, momentum=0.9, weight_decay=5e-4)
epochs = 5
```

모델명	상세 정보
mission_16_vgg11bn_fp32.pth	- 기본 FP32
mission_16_vgg11bn_dynamic_int8.pth	- **PTQ – Dynamic** - 대상: **Linear(FC)만 INT8** (Conv/BN/ReLU=FP32) - - backend: **fbgemm**
mission_16_vgg11bn_static_int8_fx.pth	- **PTQ – Static (FX Graph)** - **Calibration 20배치**(대표 배치로 통계 수집) - 대상: **Conv + Linear → INT8** (activation per-

모델명	상세 정보
	tensor, weight per-channel) - backend: **fbgemm**
mission_16_vgg11bn_qat_int8_fp32match.pth	- **QAT (FX Graph)** - 대상: **Conv2d + Linear → INT8** - Weight: **int8 per-channel symmetric** (ch_axis=0) - Activation: **uint8 per-tensor affine**- **BatchNorm**: Conv에 **fuse**- **ReLU**: Conv/Linear와 **fuse**되어 INT8 경로에서 실행 - backend: **fbgemm**

⇒ pth 모델을 ONNX로 내보내기

inference.ipynb

데이터셋

- CIFAR 10 test 데이터
- test : 10000

Model	Size (MB)	Acc (%)	Per Img (ms)	Img/s	EP
mission16_vgg11bn_fp32.onnx	227.37	92.28	1.937	516.4	CUDA
mission16_vgg11bn_int8_dynamic.onnx	83.25	92.30	2.801	357.0	CUDA
mission16_vgg11bn_int8_static.onnx	57.03	92.24	2.291	436.4	CUDA
mission_16_vgg11bn_qat_int8_fp32match.onnx	56.99	91.37	2.166	461.8	CUDA

분석 및 회고

모델 크기

모델	크기(MB)	감소율
FP32	227.37	—
INT8 Dynamic	83.25	**63.39% ↓**
INT8 Static	57.03	**74.92% ↓**
INT8 QAT	56.99	**74.94% ↓**

- 양자화 **적용 범위가 넓을수록** 모델 크기가 더 줄어듭니다.
- FP32(4 byte) → INT8(1 byte)로 양자화하면 **이론상 ~75% 크기 감소**가 발생합니다.

정확도

- 전반적으로 **FP32 대비 ±1%p 이내** 유지
  - FP32 **92.28%** → Dynamic **92.30%**, Static **92.24%**, QAT **91.37%**
- **결론:** 적절한 양자화(특히 **Static/PTQ**, **QAT**)를 적용하면 **정확도는 사실상 동일**을 유지하면서 **모델 크기를 ~75%**까지 줄일 수 있음.

이미지 처리 속도

- **순위: FP32 > QAT ≈ INT8 Static > INT8 Dynamic**
- **이유**
  - **FP32**: CUDA의 최적화된 Conv 커널을 그대로 사용하고 **Q/DQ(양자화·역양자화) 오버헤드가 없음** → 가장 빠름.
  - **QAT / Static PTQ**: **사전 고정된 scale/zero-point**(QAT=학습, Static=캘리브레이션)를 사용해 INT8 연산 수행. 다만 **중간 Q/DQ 오버헤드**와 일부 FP32 경로로 인해 FP32보다는 **약간 느림**. 두 방식은 **유사한 속도**를 보임.
  - **Dynamic PTQ**: 실행 시마다 **activation 범위를 계산해 즉석 양자화**(주로 **Linear만 INT8**, **Conv는 FP32 유지**)하므로 **추가 연산 오버헤드**가 커져 **가장 느림**.