Mission 16 — Model Quantization & Inference Summary Report

개요

본 과제는 기존 MNIST CNN 모델을 학습한 후, 다음의 세 가지 형태로 **모델을 변환 및 비교**하는 미션입니다.

- **FP32 (기본 PyTorch 형식)**
- **INT8 (동적 양자화 PyTorch 형식)**
- **ONNX (범언어 실행 가능 포맷)**

이를 통해 **정확도, 추론 속도, 모델 용량 변화**를 비교하여 양자화의 실효성과 ONNX 변환의 이식성을 분석합니다.

사용 데이터셋

구분	데이터셋 이름	내용	출처	비고
학습 데이터셋 (Training Set)	MNIST Train Set	60,000장의 28×28 크기 흑백 손글씨 이미지	Yann LeCun et al. (1998), `torchvision.datasets.MNIST(train=True)`	숫자(0~9) 분류 모델 학습
평가 데이터셋 (Test Set)	MNIST Test Set	10,000장의 손글씨 이미지	`torchvision.datasets.MNIST(train=False)`	모델 성능 검증 (정확도 측정용)

> **Detail**
>
> - PyTorch의 **torchvision.datasets.MNIST**를 통해 자동 다운로드 및 캐싱
> - 전처리: `transforms.ToTensor()` + `transforms.Normalize((0.1307,), (0.3081,))`
> - 평가 시 **shuffle=False**, `batch_size=256` 설정으로 고정

프로젝트 디렉토리 구조

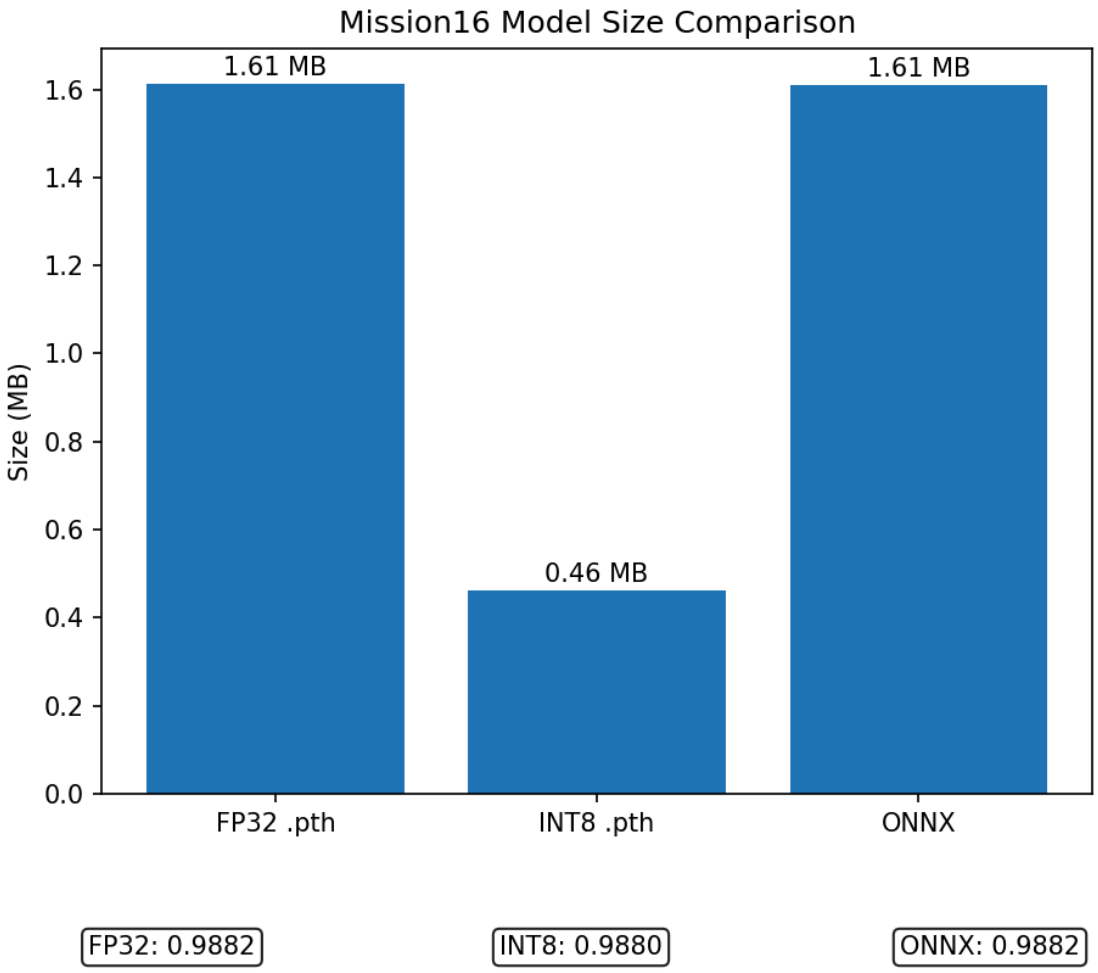
```
mission16/
├─ models/
│  ├─ mission_16_mnist_cnn.pth
│  ├─ mission_16_mnist_cnn_int8.pth
│  └─ mission_16_mnist_cnn.onnx
├─ report/
│  ├─ inference_summary.txt
│  ├─ inference_summary.csv
│  └─ sizes_and_results.png
├─ inference.py
├─ modeling.ipynb
├─ requirements.txt
└─ Summary_Report.pdf
```

모델별 결과 요약

Model (파일명)	Accuracy	Avg Time (ms)	Size (MB)
FP32 .pth	**0.9882**	31.42	1.61
mission_16_mnist_cnn.pth			
INT8 .pth	**0.9880**	29.10	0.46
mission_16_mnist_cnn_int8.pth			
ONNX	**0.9882**	22.25	1.61
mission_16_mnist_cnn.onnx			

> **[결과 해석]**
>
> - INT8 모델은 **약 71% 용량 감소**에도 정확도 손실이 거의 없음
> - ONNX 모델은 **CPU 추론 시 가장 빠른 속도**를 보임
> - 세 포맷 모두 정확도 차이가 0.0002 이내로 안정적임

Model Size & Accuracy Visualization



> **Detail**
>
> - 각 모델의 용량(MB)을 한눈에 비교 가능
> - 하단에 정확도(Accuracy) 수치를 주석 형태로 표시
> - FP32 대비 INT8의 용량 절감율: **약 71.4%**

디버깅 및 오류 해결 과정

Case	이슈	원인	해결 포인트	해결 과정
1	가상환경 활성화 실패	`.venv/Script` 경로 오타	쉘별 정확한 경로 사용	Git Bash: `source .venv/Scripts/activate`, PowerShell: `.\.venv\Scripts\Activate`
2	경로/타이핑 오타 및 이스케이프 경고	`\m`, `os.apth` 등 오타	슬래시(`/`) 사용, 정확한 함수명	`"models/..."`, `os.path.exists()` 로 수정
3	`forward()` 누락	클래스에 `forward()` 미구현	모델 구조 동일성 확보	`features→classifier→return` 구현
4	INT8 state_dict 불일치	FP32 구조에 INT8 가중치 로드	동일한 양자화 구조 생성	`quantize_dynamic` 적용 후 로드 (CPU 평가)
5	report 폴더 없음	저장 시 폴더 미생성	사전 폴더 생성	`os.makedirs('report', exist_ok=True)` 추가
6	`file_mb` 미정의	유틸 함수 스코프 누락	전역 유틸 함수로 이동	`def file_mb(path): ...` 추가
7	시각화 라벨 겹침	하단 여백 부족	여백 및 텍스트 오프셋 조정	`plt.subplots_adjust(bottom=0.20)` 적용
8	MNIST 404 다운로드 실패	원본 URL 404	자동 미러 확인	Torchvision이 S3 미러에서 자동 복구
9	PyTorch 경고 출력	TypedStorage deprecated	경고 필터링	`warnings.filterwarnings('ignore')` 추가

분석 및 결론

- **INT8 양자화 모델**은 Linear 계층만 양자화했음에도 **약 71% 용량 절감**, 정확도 유지.
- **ONNX 모델**은 추론 속도가 가장 빠르며, **범언어 호환성**이 높아 실제 서비스 이식에 유리.
- 디버깅 과정을 통해 모든 오류를 해결하고 **재현 가능한 end-to-end 코드**를 완성.
- **inference.py** 실행 시 자동으로 정확도·속도·용량 결과 및 보고용 그래프 생성.