Icon_primitive — Project Guide (EN/KR)

Generated: 2026-01-23

What this is (EN)

Icon_primitive is the Part-1 (ICON-Primitive) reproducible pipeline for measuring per-dimension information capacity (kappa) under a fixed measurement channel, and for deriving relative constants across primitive operations (activation, normalization, precision, skip, linear type). The project is designed to be audit-friendly: fixed indices, frozen protocol, receipt schema validation, and hashed artifacts.

이 문서의 목적 (KR)

Icon_primitive는 ICON 시리즈 1편(Primitive) 실험을 재현 가능하게 돌리기 위한 파이프라인입니다. 고정된 측정 채널(노이즈 채널)을 포함한 kappa(κ)를 측정하고, primitive 옵션(activation/norm/precision/skip/linear type)별 상수 비율을 산출합니다. 공정성/재현성을 위해 고정 인덱스, 프로토콜 고정, receipt 스키마 검증, 해시 기반 감사 가능성을 포함합니다.

Project Layout

High-level folder/file tree (EN/KR)

```
Icon_primitive/
├── assets/
│   ├── calib_indices/
│   ├── eval_indices/
│   └── frozen_stems/
├── configs/
│   ├── manifests/
│   ├── packs/
│   ├── sections/
│   └── base_protocol.yaml
├── docs/
│   ├── ICON_Primitive_ConfigPacks_v1.1_README.md
│   ├── ICON_Primitive_Experiment_Spec_v1.1.md
│   ├── ICON_Primitive_Independence_Config_Generation_Rules_v1.1.md
│   ├── ICON_Primitive_repo_review_toClaude_2026-01-18.md
│   ├── ICON_Primitive_v1.1_Gapless_Checklist_toClaude.md
│   ├── ICON_Series_Detailed_Design.md
│   └── Icon_primitive.pdf
├── icon_primitive/
│   ├── core/
│   ├── data/
│   ├── experiment/
│   ├── models/
│   ├── training/
│   └── utils/
├── schemas/
│   ├── ICON_Primitive_Receipt_Example_v1.1.json
│   └── ICON_Primitive_Receipt_Schema_v1.1.json
├── scripts/
│   ├── __init__.py
│   ├── aggregate.py
│   ├── generate_eval_indices.py
│   ├── generate_stems.py
│   ├── run_manifest.py
│   └── run_single.py
├── CHANGELOG.md
├── README.md
└── pyproject.toml
```

Notes (EN)

• The project intentionally separates code/config from data. Large datasets are downloaded/managed outside of this repo (e.g., CIFAR10/MNIST via torchvision).
• Reproducibility artifacts are stored under assets/ (fixed eval/calib indices, optional frozen stems). Run outputs and receipts are stored under outputs/.

주의사항 (KR)

• 이 프로젝트는 코드/설정과 데이터를 분리합니다. 대용량 데이터(CIFAR10/MNIST 등)는 레포 외부에서 내려받아 사용합니다.
• 재현성 아티팩트는 assets/(고정 eval/calib 인덱스, 선택적 frozen stems)에 저장됩니다. 실행 결과/receipt는 outputs/ 아래에 저장됩니다.

Key Components

Path	English (purpose)	한국어 (설명)
configs/base_protocol.yaml	Frozen protocol: training spec, seeds policy, measurement channel, MI estimators, determinism settings, receipt required fields.	프로토콜 고정: 학습 스펙, seed 정책, 측정 채널, MI 추정기, 결정론 설정, receipt 필수 필드
configs/manifests/*.csv	Run manifests: Base153 and optional Extensions. Each row defines run_id, section, probe, primitive options, seeds.	실행 매니페스트: Base153 및 확장. 각 row가 run_id/섹션/probe/옵션/seed 지정
scripts/run_single.py	Run one run_id from a manifest; produces outputs/runs//receipt.json.	매니페스트에서 run_id 1개 실행; outputs/runs//receipt.json 생성
scripts/run_manifest.py	Run many runs (optionally filtered by section). Use workers=1 for strict reproducibility during validation.	여러 run 실행(섹션 필터 가능). 검증 단계는 workers=1 권장
scripts/aggregate.py	Aggregate receipts into constants YAML; invalid receipts (schema failure) are excluded.	receipt 집계해 상수 YAML 생성; 스키마 실패는 제외
schemas/ICON_Primitive_Receipt_Schema_v1.1.json	Receipt JSONSchema gate (audit/reproducibility).	Receipt JSON 스키마 게이트(감사/재현성)
icon_primitive/experiment/runner.py	End-to-end runner: dataset -> train -> taps -> kappa -> receipt.	E2E 실행기: 데이터->학습->tap 추출->κ->receipt
icon_primitive/core/*	Kappa computation, MI estimators (InfoNCE/MINE/KSG), noise channel.	κ 계산, MI 추정기(InfoNCE/MINE/KSG), 노이즈 채널
icon_primitive/models/*	Stems, probes, primitive ops, PTQ/precision conversion.	Stem, probe, primitive 연산, PTQ/정밀도 변환

Workflow

A. Install & Environment (EN/KR)

EN: Recommended Python 3.10+ (project requires Python >= 3.10). For AWS g5/p4d, conda env (python=3.10) is usually most stable.
KR: 권장 Python 3.10+ (프로젝트 요구사항: Python >= 3.10). AWS g5/p4d에서는 conda 환경(python=3.10)이 가장 안정적입니다.

```
conda create -y -n icon310 python=3.10
conda activate icon310

cd Icon_primitive
pip install -U pip
pip install -e .
python -c "import icon_primitive; print('import ok')"
```

B. Reproducibility settings (EN/KR)

EN: For strict reproducibility during validation, fix CPU threads and keep workers=1. KR: 검증 단계에서는 CPU thread 고정 + workers=1을 권장합니다.

```
export ICON_PRIMITIVE_NUM_THREADS=1
export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
```

C. Smoke validation on g5 (EN/KR)

EN: Run a small set (e.g., 1A baseline, 1B linear_type, 1D precision, 1F independence) and then aggregate.
KR: 1A/1B/1D/1F에서 대표 run을 몇 개만 실행한 뒤 aggregate로 상수표를 생성합니다. receipt가 생성되고 스키마 검증이 통과하면 p4d로 확장 실행합니다.

```
MANIFEST=configs/manifests/ICON_Primitive_Run_Manifest_Base153_v1.1.csv

python scripts/run_single.py \
  --manifest $MANIFEST \
  --run_id 1A_A00_s0 \
  --output_root outputs --data_root data

python scripts/run_single.py --manifest $MANIFEST --run_id 1B_B00_s0 --output_root outputs --data_root data
python scripts/run_single.py --manifest $MANIFEST --run_id 1D_D00_s0 --output_root outputs --data_root data
python scripts/run_single.py --manifest $MANIFEST --run_id 1F_F01_s0 --output_root outputs --data_root data

mkdir -p outputs/constants
python scripts/aggregate.py \
  --receipts outputs/runs \
  --template configs/packs/ICON_Primitive_Constants_Template_v1.1.yaml \
  --output outputs/constants/Icon_primitive_Constants.yaml
```

Scaling up on p4d.24xlarge

EN/KR

EN: Copy assets/ from g5 to p4d (fixed indices, optional frozen stems) and keep the same code revision. First re-run the same smoke run_ids, then run the full Base153 manifest with parallel workers.
KR: 공정성/재현성을 유지하려면 g5에서 생성된 assets/(고정 인덱스/선택적 frozen stems)를 p4d로 그대로 복사하고, 동일한 코드 리비전으로 실행합니다. 먼저 스모크 run_id 4~6개를 재실행한 뒤, Base153 전체를 병렬로 실행합니다.

```
# (on g5) package assets for transfer
tar -czf icon_assets.tgz assets

# (on p4d) unpack
tar -xzf icon_assets.tgz

MANIFEST=configs/manifests/ICON_Primitive_Run_Manifest_Base153_v1.1.csv
python scripts/run_single.py --manifest $MANIFEST --run_id 1A_A00_s0 --output_root outputs --data_root data

# full run (start with workers=8 and scale up)
python scripts/run_manifest.py \
  --manifest $MANIFEST \
  --output_root outputs --data_root data --workers 8

# aggregate
python scripts/aggregate.py \
  --receipts outputs/runs \
  --template configs/packs/ICON_Primitive_Constants_Template_v1.1.yaml \
  --output outputs/constants/Icon_primitive_Constants.yaml
```

Reproducibility checklist (EN/KR)

```
- EN: Do not modify configs/base_protocol.yaml during a run campaign.
- EN: Keep assets/eval_indices and assets/calib_indices fixed once generated.
- EN: Ensure each outputs/runs/<run_id>/receipt.json validates against schemas/ICON_Primitive_Receipt_Schema_v
- EN: Prefer workers=1 when validating; scale workers only after smoke validation.
- KR: 실행 중 configs/base_protocol.yaml 변경 금지.
- KR: assets/eval_indices, assets/calib_indices는 한 번 생성 후 그대로 유지.
- KR: 모든 run의 receipt.json이 스키마 검증을 통과해야 함.
- KR: 검증 단계에서는 workers=1 권장, 이후 확장 단계에서만 병렬화.
```

Troubleshooting

Common issues (EN)

- Python version mismatch: the project requires Python >= 3.10. Use conda/venv with python=3.10+.
- Torchvision import errors: install matching torch/torchvision wheels; use the same CUDA version family.
- Slow runs: start with workers=1 and ICON_PRIMITIVE_NUM_THREADS=1 during validation; then scale on p4d.
- Missing receipts in aggregation: check schema validation failures and confirm run output paths.

자주 발생하는 문제 (KR)

- Python 버전 불일치: Python >= 3.10 필요. conda/venv로 python=3.10+ 환경 사용.
- torchvision import 오류: torch/torchvision 호환 버전 설치(CUDA 패밀리 일치).
- 실행이 느릴 때: 검증 단계는 workers=1, ICON_PRIMITIVE_NUM_THREADS=1로 시작 후 p4d에서 확장.
- aggregate에 receipt가 누락될 때: 스키마 실패 여부 확인, 출력 경로 확인.