unified-sdk

11.19 10:59

Unified SDK Architecture Overview

목적

> `unified-sdk`는 다양한 국산 및 상용 AI 반도체(NPU) 환경에서 AI 모델의 **컴파일(Build)** 및 **추론(Runtime)**을 일관된 인터페이스로 처리하기 위한 통합 SDK입니다.
> 본 프로젝트는 **「국산 AI 반도체 기반 마이크로 데이터센터 확산 사업」** 내 (세부 3) *이종 AI 반도체 활용을 지원하는 통합 SDK* 개발 과제로 수행되었습니다.

Directory Structure

```
unified-sdk/
├── README.md
├── requirements.txt
├── Dockerfile
├── devcontainer.json
├── examples/                    # 예제 코드
│   ├── test_tensorrt_build.py   # TensorRT 모델 빌드 예제
│   └── run_tensorrt_infer.py    # TensorRT 엔진 실행 예제
└── src/
    └── unified_sdk/
        ├── __init__.py
        ├── types.py             # 공통 데이터 구조 정의
        │
        ├── build/               # 모델 빌드(컴파일) 관련 모듈
        │   ├── __init__.py
        │   ├── api.py           # 상위 진입점 (backend-agnostic)
        │   ├── registry.py      # 백엔드 빌더 등록/조회 관리
        │   └── tensorrt_build.py # TensorRT 빌드 어댑터
        │
        └── runtime/             # 모델 실행(추론) 관련 모듈
            ├── __init__.py
            ├── api.py           # 상위 진입점 (backend-agnostic)
            ├── registry.py      # 백엔드 런타임 등록/조회 관리
            └── tensorrt_rt.py   # TensorRT 런타임 어댑터
```

Layered Design

계층	역할	주요 파일	예시
1 Common Types	빌드/런타임 공용 설정 및 결과 구조 정의	`types.py`	`BuildConfig`, `BuildResult`, `RuntimeConfig`, `RuntimeHandle`
2 Registry Layer	백엔드별 어댑터를 등록/조회	`build/registry.py`, `runtime/registry.py`	`register()`, `get_builder()`, `get_runtime()`
3 Adapter Layer	실제 SDK(TensorRT 등)에 연결되는 백엔드별 구현	`tensorrt_build.py`, `tensorrt_rt.py`	TensorRT 빌드 및 추론 수행
4 API Layer	상위에서 backend-agnostic하게 접근하는 통일된 API	`build/api.py`, `runtime/api.py`	`build_unified()`, `create_runtime()`
5 Entrypoint Layer	패키지 import 시 자동 등록 및 re-export	`build/__init__.py`, `runtime/__init__.py`	`from unified_sdk.build import build_unified`
6 Example Layer	실제 SDK 동작 확인용 예제 코드	`examples/`	TensorRT build/infer

Build System (Model Compilation)

Entry Point

```python
from unified_sdk.build import build_unified
from unified_sdk.types import BuildConfig

cfg = BuildConfig(
    backend="tensorrt",
    model_or_path="models/hrnet_coco_w32_Nx256x192.onnx",
    out_dir="build_output",
    model_name="hrnet_coco_w32_Nx256x192",
    precision="fp16"
)

result = build_unified(cfg)
print(result.compiled_model_path)
```

내부 동작

1. `build_unified(cfg)` → `registry.get_builder(cfg.backend)`
2. `tensorrt_build.py` 내 `_TensorRTBuildAdapter.build()` 호출
3. TensorRT Builder API를 통해 ONNX → Engine 변환 수행
4. 결과 `.engine` 파일과 메타데이터를 반환

Runtime System (Model Execution)

Entry Point

```
from unified_sdk.runtime import create_runtime, infer, destroy_runtime
from unified_sdk.types import RuntimeConfig
import numpy as np

cfg = RuntimeConfig(
    backend="tensorrt",
    engine_path="build_output/hrnet_coco_w32_Nx256x192_FP16.engine",
    input_name="input.1",
    output_name="2901",
    input_shape=(1, 3, 256, 192),
)

rh = create_runtime(cfg)
x = np.random.rand(*cfg.input_shape).astype(np.float32)
y = infer(rh, x)
destroy_runtime(rh)
```

내부 동작

1. `create_runtime(cfg)` → `registry.get_runtime(cfg.backend)`
2. `tensorrt_rt.py` 내 `_TensorRTRuntime.create()` 실행
3. TensorRT Engine 로드 및 메모리 바인딩
4. `infer()` 호출 시 `execute_async_v3` 또는 `execute_v2`로 추론 수행

확장 설계 (Extensible Architecture)

항목	추가 시 작성 위치	비고
새 백엔드 빌드 (예: Furiosa)	`src/unified_sdk/build/furiosa_build.py`	`class _FuriosaBuildAdapter:` + `register()`
새 백엔드 런타임 (예: Rebellions)	`src/unified_sdk/runtime/rebellions_rt.py`	`class _RebellionsRuntime:` + `register()`
공통 API 재사용	그대로 유지	`build_unified()`, `create_runtime()` 동일하게 작동

모든 신규 백엔드는 **register()로 자동 인식**되며,
상위 API는 코드 수정 없이 그대로 동작합니다.

Naming Convention

구분	규칙	
패키지	`unified_sdk.[build	runtime]`
클래스	`_TensorRTBuildAdapter`, `_TensorRTRuntime`	
공통 타입	`BuildConfig`, `RuntimeConfig`, `BuildResult`, `RuntimeHandle`	
주요 함수	`build_unified`, `create_runtime`, `infer`, `destroy_runtime`	

Example Development Flow

단계	목적	실행 파일	결과
①	모델 ONNX 변환 (PyTorch/TensorFlow → ONNX)	외부 툴	`.onnx` 파일
②	Unified SDK로 컴파일	`examples/test_tensorrt_build.py`	`.engine` 파일 생성
③	Unified SDK로 실행	`examples/run_tensorrt_infer.py`	추론 수행 및 성능 측정

향후 확장 계획

- [ ] **Rebellions / Furiosa NPU 어댑터** 추가
- [ ] **ONNXRuntime CPU fallback** 모드 지원
- [ ] **build/infer 공통 CLI** (`unified-cli build`, `unified-cli run`)
- [ ] **자동 성능 측정 & 로그 리포트 기능**

License / Acknowledgment

본 소프트웨어는 한국전자통신연구원(ETRI) 「국산 AI 반도체 기반 마이크로 데이터센터 확산 사업」의 연구 성과를 기반으로 개발되었습니다.

› **편집자** 김량수 (11.19 10:59)

☰ 댓글