남도현

근본 원인을 파고들어 확장 가능한 아키텍처로 해결하는 소프트웨어 & AI 엔지니어

Email: dhnam0502@naver.com | Github: http://github.com/dhnam | Phone: 010-5219-3436 | Portfolio: https://dhnam0502.github.io

Introduction

"LLM을 단순히 호출하는 데 그치지 않고, 평가 가능한 에이전트와 운영 가능한 백엔드 시스템으로 구현합니다."

- **LLM System Design** LangGraph 기반의 멀티 에이전트·HITL 워크플로우와 RAG 파이프라인을 설계하고, 상태 관리와 툴 호출을 포함한 서비스 흐름을 구현합니다.
- **Evaluation-Driven Engineering** DeepEval 기반 모델 벤치마크와 정량 지표를 통해 응답 품질과 지연 시간을 비교하고, 프로덕션에 적합한 모델과 아키텍처를 선택합니다. (지연 시간 91.3% 단축, 채점 오차 MAE 22.5% 개선)
- **Context & Root-Cause Problem Solving** 문서와 코드베이스를 분석해 LLM에 필요한 컨텍스트를 구성하며, 데이터 누수·불균형·병목의 근본 원인을 찾아 문제 정의와 시스템을 함께 개선합니다.

Tech & Skills

AI & Computer Vision

- **Frameworks** PyTorch, TensorFlow, PyTorch Lightning (생산성 향상 및 구조화)
- **Libraries** OpenCV, Albumentations (전처리)
- **Models** YOLO(v5, Ultralytics), ResNet, EfficientNet, U-Net

Serving & Backend

- **Agent & RAG** LangGraph, LlamaIndex, LangChain
- **Web Frameworks** FastAPI, Pydantic, SSE (Streaming)
- **Database & Async** SQLModel (ORM), TaskIQ, Redis

Language & Engineering

- **Languages** Python (NumPy/Pandas)
- **DevOps & Tools** Docker, Git, Linux CLI, uv
- **Docs** Typst, Mermaid (문서화 및 시각화)

Key Projects

LLM을 이용한 Fuzzing Harness 생성 자동화 | 🔗 github

2025.03 ~ 2025.12

논문/코드 분석 / 프롬프트 엔지니어링

졸업 프로젝트 | 2명

Documentation 정보를 활용하여 Fuzzing Harness의 생성 성공률 및 커버리지 향상

- **Stack** Rust, LLM APIs, AFL++
- **[Analysis] 코드 베이스 분석** 익숙하지 않은 Rust 언어의 오픈소스 코드를 공식 문서를 통해 분석하고 로직 수정.
- **[Engineering] Context Engineering** 라이브러리 문서를 LLM Context로 주입하여 유효 하니스 생성을 최대 4.5배 향상.
  - cJSON Branch Coverage 78.37% → 87.38% (+9.01%p).

논술 AI 학습 플랫폼 | 🔗 github

2026.08.01 ~ 2026.08.25

백엔드 개발 / 백엔드 아키텍처 / 모델 벤치마크

이어드림스쿨 최종 프로젝트 | 4명

대입 논술 답안 정밀 채점·첨삭 및 회차별 비교를 지원하는 AI 학습 서비스

- **Stack** LangGraph, FastAPI, SQLModel, DeepEval, Bareun.ai, uv, React

- **[Architecture] 병렬 채점 및 중위값 집계** 3개 Replica 병렬 채점 및 중위값(Median) 집계 엔진을 구축하여 LLM 점수 편차 완화 및 응답 지연 최소화.
- **[Evaluation] DeepEval 기반 8개 모델 벤치마크** 기존 모델 대비 지연 시간 72.53s → 6.30s (91.3% 단축), 채점 오차 MAE 1.111 → 0.861 (22.5% 개선)를 기록한 `gemini-3.5-flash-lite`로 프로덕션 모델 단일화.
- **[Backend & Data] 구조화 영속 계층** `PydanticJSON` 커스텀 TypeDecorator를 개발하여 복잡한 중첩 채점 스키마와 ORM 간의 양방향 무결성 확보.
- **[Process] AI Coding Agent 협업 리드** GitHub Issue 상세 명세화 및 CLI 연계를 통해 에이전트 생성 코드의 아키텍처 정합성과 품질 유지.

게임 서비스센터 Agentic AI | 🔗 github

2026.07.16 ~ 2026.07.21

에이전트 아키텍처 설계 / 백엔드 비동기 스트리밍 / RAG 파이프라인 구축

이어드림스쿨 프로젝트 | 1명

LangGraph 기반 HITL 멀티 에이전트 및 비동기 SSE 스트리밍 게임 서비스센터 시스템

- **Stack** LangGraph 1.x, LlamaIndex, FastAPI, SSE, Docker, uv
- **[Architecture] HITL 워크플로우** LangGraph Checkpoint & Interrupt를 활용해 유저/버그 신고 시 턴제 대화 상호작용 및 세션 상태 제어.
- **[Backend] 비동기 SSE 스트리밍** FastAPI 및 `astream_events` (v2) 기반으로 LLM 토큰 및 툴 실행 이벤트를 클라이언트에 실시간 노출.
- **[RAG & Tools] 검색 및 검증 최적화** LlamaIndex Vector Store 인덱스 영속화(Persist) 및 신고 대상 유저 검증 커스텀 툴(`resolve_target_user_tool`) 개발.

딥페이크 탐지 어플리케이션 | 🔗 github

2026.1.28 ~ 2026.2.27

모델 구현 / 백엔드 비동기 태스크 구현 / 코드 품질 향상

멋쟁이사자처럼 부트캠프 프로젝트 | 5명

딥페이크 여부를 알려주는 어플리케이션 및 서비스

- **Stack** ViT, FastAPI, TaskIQ, SQLModel, React Native
- **[Implementation] Paper-to-Code** 공식 구현체가 없는 최신 논문의 아키텍처를 분석하여 모델을 직접 구현. 누락된 파라미터와 세부 설정은 유사 논문을 교차 참조하여 보완 및 구현.
- **[Evaluation] 데이터 누수 해결** 프레임 단위 분할에서 발생한 Validation 98% 이상 / Test 약 70%의 불일치를 Video-level Split으로 수정하여 Validation 지표를 Test와 유사한 약 82% 수준으로 정렬.
- **[Architecture] 비동기 추론 파이프라인** 고부하 AI 모델 추론 시 서버 병목을 방지하기 위해 TaskIQ 기반의 비동기 작업 큐 시스템을 설계 및 구축.
- **[Engineering] Modern Python Tooling** uv를 활용한 의존성 관리와 Type Hinting, SQLModel을 적용하여 확장성과 가독성이 높은 백엔드 코드 베이스 구축

차선 위반 차량 탐지 시스템 | 🔗 github

2026.01.16 ~ 2026.01.23

아키텍처 설계 / 데이터 불균형 해결 / 이미지 분류 모델 학습

멋쟁이사자처럼 부트캠프 프로젝트 | 4명

차선 세그멘테이션 정보를 활용한 위반 차량 식별 프로젝트

- **Stack** YOLOv26-seg, ResNet, Streamlit
- **[Problem 1] 극심한 클래스 불균형** 정상:주정차 위반:차선 위반 비율이 35:1:2로, 단일 모델 학습 불가.
- **[Problem 2] 라벨 부족** 다수의 차량 중 일부만 라벨링되어 있어, 직접 학습 시 미라벨링 된 차량을 배경으로 오인 학습할 위험 존재.
- **[Solution] 문제 정의 변경** 객체 인식 문제를 '위치 탐지'(차량/차선)와 '위반 분류'로 분리.
  - **차량 위치 탐지** YOLOv26-seg를 이용해 미라벨 차량의 마스크 추출.
  - **위반 판단** 추출된 객체에 대한 별도의 Classification 적용.
- **[Data] Undersampling** 정상 차량 데이터의 34/35를 무작위 제거하여 클래스 비율을 35:1:2에서 약 1:1:2로 조정.

딸기 질병 객체 인식 및 진단 | 🔗 github

2026.01.09 ~ 2026.01.16

멋쟁이사자처럼 부트캠프
프로젝트 | 4명

딸기의 병든 잎과 줄기를 인식하고 질병을 분류하는 2-Stage 솔루션

- **Stack** YOLOv11, EfficientNet, Albumentations
- **[Problem] 라벨 부족 및 불균형** Positive-Unlabeled(PU) 상황과 유사한 라벨 결핍 발생.
- **[Solution] Pseudo-Labeling 적용** 신뢰도 높은 예측값을 다시 학습 데이터로 활용.
  - Recall 성능 약 9% 향상 (약 0.55 → 0.60)
- **[Architecture] 2-Stage Pipeline** '객체 탐지' 후 '질병 분류' 구조로 설계하여 정확도 개선.
- **[Optimization] 학습 병목 해결** Albumentations 기반의 리사이징 및 전처리 최적화.
  - Epoch당 학습 시간 50% 단축 (10m → 5m).

Education

한양대학교 | 컴퓨터소프트웨어학과 (2020.03 ~ 2026.08)

멋쟁이사자처럼 | AI CV 부트캠프 3기 (2025.12 ~ 2026.02)

Elice | 이어드림스쿨 심화 (2026. 06 ~)

Awards

한양대학교 | 우등상 (2026.08)