Claude로 연구하기

Cowork & Code 가이드



이상오 (Sangoh Lee)

POSTECH · DSLab, As a Robotics Researcher

2026.03

2주 만에 4개 프로젝트 동시 진행

Claude Cowork + Code 도입 후 연구 생산성 변화

ICAC

Phase 1 학습
→ MSE 분석

3/4~3/9

DepthMem

구현 완료
→ 학습 시작

3/10~3/11

PhysREPA

Phase 0 준비
V-JEPA 2 셋업

3/12~3/13

MobileVAP

P-OVMM 정의
아키텍처 설계

3/13~3/14

이 모든 것이 Claude 없이는 불가능했습니다

왜 Obsidian인가?



Markdown Native

Claude가 직접 읽고 수정 가능
별도 변환 불필요



로컬 폴더 기반

Claude Cowork이 폴더 마운트로
직접 접근



Wiki-link 연결

[[노트]] 형식으로 연구 노트 간
자유로운 상호참조



Git 연동 가능

버전 관리 + Overleaf 연동
논문 작업 동기화

Notion/Google Docs: 클라우드 잠금, API 제한 → Obsidian: **로컬 파일 = Claude가 바로 접근**

PARA 방법론으로 연구 노트 정리

P **Project**
목표+데드라인 있는 진행 중 연구
VAP, ICAC, DepthMem, PhysREPA...

A **Area**
꾸준히 관리하는 영역
Daily Plan, Health, 행정...

R **Resource**
주제별 참고 자료
Papers Index, 세미나 기록...

A **Archive**
완료된 프로젝트 보관
ASM, KSC2023, EMNLP출장...

행동(Action) 기준 분류 → Claude가 맥락 파악 용이

분산 관리 → Obsidian 통합

Before

Zotero
→ 논문 PDF 관리

Mendeley
→ 논문 메모

Notion
→ 연구 노트

Google Docs
→ 미팅 노트

→

After — Obsidian

Papers Index.md
→ 논문 인덱스 (137편+)

프로젝트별 Paper Writing/
→ .md + .tex 논문 원고

Meeting Minutes/
→ 교수님 미팅 기록

Daily Plans/
→ 600+ 일일 계획

하나의 Vault에서 모든 연구 자료를 Claude가 직접 접근

Claude Cowork — 연구 매니저

방향 설정, 분석, 문서 작성을 담당하는 데스크탑 에이전트



연구 방향 탐색

Gap analysis, 논문 서베이,
아이디어 검증



논문 읽기/쓰기

Reading notes, 초안 작성,
top-tier 문체 분석



결과 분석

실험 결과 해석, 시각화,
다음 실험 제안



문서 생성

PPT, 주간 보고서,
미팅 자료 자동 생성

Obsidian Vault를 마운트 → 모든 연구 노트에 직접 접근

CLAUDE.md — 프로젝트 컨텍스트

매 세션마다 자동으로 로드되는 프로젝트 설명서

```
# CLAUDE.md

## Vault 소유자

- 이름: Sangoh / 역할: 대학원생 (Robotics)

## Project (현재 주력 연구)

- VAP: ICML 2026 제출 완료

- DepthMem: 구현 완료, 학습 중

- PhysREPA: Phase 0 준비 중

## 응답 스타일 – 연구 파트너 모드

- 꼬리를 무는 자기 질문

- 검증 먼저, 제안은 나중에

- 쉽게 꼬리 내리지 말 것
```

한 번 작성하면 모든 세션에서 맥락 유지 — "매번 설명할 필요 없음"

Claude Code — 구현 & 실험 Executor

터미널에서 직접 코딩하는 AI 에이전트

- 코드베이스를 직접 읽고 이해
- 환경 셋업부터 구현까지 자율 수행
- 실험 실행, 에러 디버깅, 로그 분석
- Git commit, branch 관리
- tmux + wandb로 백그라운드 실험

```
$ claude

> Implementation Prompt를
  읽고 DepthMem 구현해줘

✓ config.py 수정 (4ch input)
✓ depth_pipeline.py 생성
✓ MEM attention 구현
✓ train.py 수정 (LoRA)
⚠ RuntimeError 발견 → 패치
✓ 학습 파이프라인 완성
```

602줄 프롬프트 하나로 전체 구현 완성

Cowork ↔ Code 생태계



Cowork

1. 연구 방향 설정
2. 프롬프트 작성
3. 결과 분석
4. 논문 작성 코칭

프롬프트 →

← 결과



Code

1. 환경 셋업
2. 코드 구현
3. 실험 실행
4. 디버깅 & 로그



공유 Git Codebase — 양쪽 모두 같은 코드를 봄

자면서 실험하기 — tmux + wandb

23:00



Claude Code에게
실험 명령

→

23:05



tmux 세션에서
학습 시작

→

Zzz...



wandb 로깅
자동 기록 중

→

08:00



Cowork에게
결과 분석 요청

연구자가 자는 동안에도 실험은 계속됩니다

Overleaf + Git 연동

Claude Code가 .tex를 직접 편집 → git push → Overleaf 반영

Paper Writing/ 구조

PhysREPA/Paper Writing/

```
├── main.tex
├── Sections/
│   ├── abstract.tex
│   ├── introduction.tex
│   ├── method.tex
│   ├── experiments.tex
│   └── analysis.tex
└── math_commands.tex
```

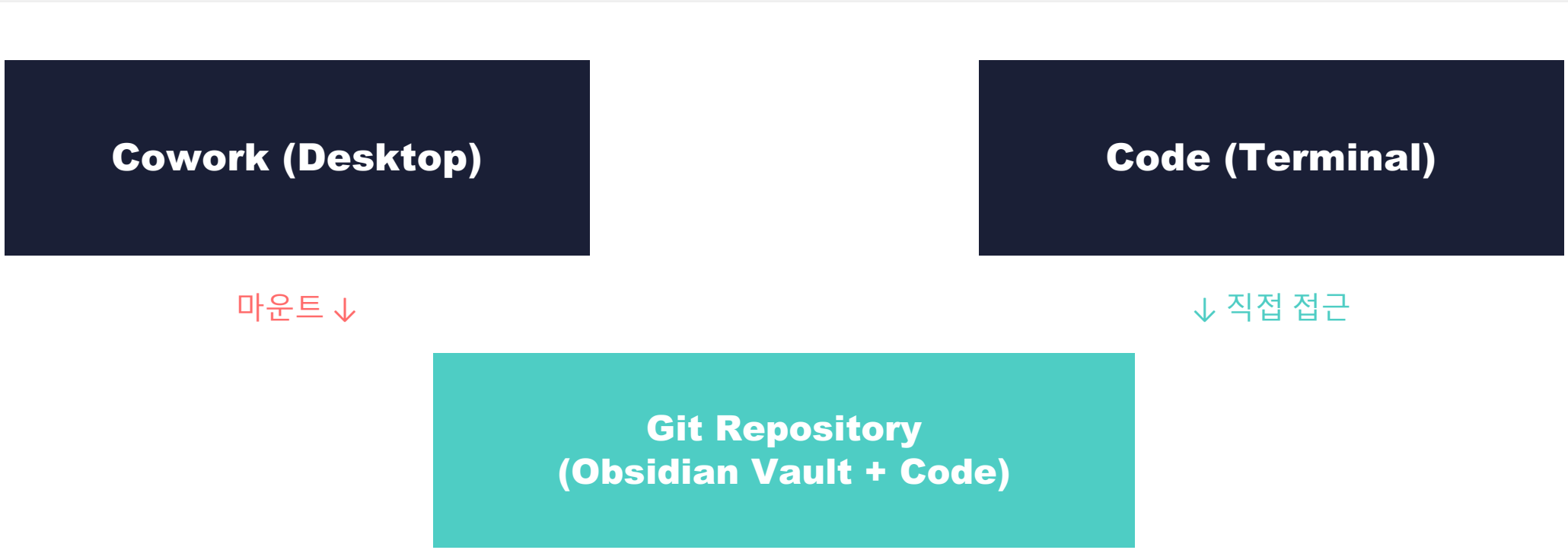
워크플로우

1. Cowork이 초안 작성 (.md)
2. Code가 .tex로 변환/편집
3. git push → Overleaf 반영
4. Overleaf에서 공동 편집 가능

현재 3개 논문 (ICAC, PhysREPA, MobileVAP) 동시 관리 중

공유 Git Codebase

Cowork과 Code가 동일한 코드베이스를 보는 구조



Cowork이 설계한 프롬프트를 Git에 저장 → Code가 읽고 바로 실행

커넥터 (MCP) — 외부 서비스 연동

Model Context Protocol로 다양한 서비스를 Claude에 직접 연결

Slack

채널 메시지 읽기
논문 링크 자동 수집

Google Drive

문서 검색/읽기
자료 직접 접근

Google Docs

문서 읽기/수정
미팅 노트 실시간 편집

Gmail

메일 검색/읽기
초안 작성

Calendar

일정 확인/생성
미팅 스케줄 관리

Notion

페이지 읽기/생성
데이터베이스 접근

설정에서 커넥터 추가 → Claude가 해당 서비스의 데이터를 직접 활용

Google Docs 연동 실전 예시

교수님 미팅 노트

1. Google Docs 미팅 문서 공유
2. Claude가 문서 직접 읽기
3. 핵심 피드백 추출 + Vault 반영
4. 다음 TODO 자동 생성

공동 연구 문서 편집

1. 공유 문서 내용 읽기
2. 수정사항 직접 반영
3. 댓글로 설명 추가
4. 동기들과 실시간 협업

자료 취합 & 정리

1. Drive에서 관련 문서 검색
2. 여러 문서 내용 통합
3. 요약 보고서 생성
4. Obsidian에 정리노트 저장

별도의 복붙 없이 Claude가 Google Docs를 직접 읽고 수정

Skills — 반복 워크플로우 자동화

키워드 트리거 → 전문 지식이 담긴 프롬프트 자동 로드

`paper-reading-note`	*"논문 정리해줘"*	arXiv → 구조화된 노트 + Index 업데이트
`paper-writing-coach`	*"논문 작성"*	top-tier 문체 분석 → 작성 코칭
`related-work-surveyor`	*"related work"*	9개 학회 최신 논문 종합 서베이
`weekly-research-digest`	*"주간 정리"*	Daily notes → 주간 연구 요약
`research-seminar-ppt`	*"세미나 PPT"*	Vault 노트 → 발표 자료 자동 생성
`slack-paper-collector`	*"슬랙 논문 정리"*	Slack 링크 → Papers Index 자동 추가
`research-direction-explorer`	*"연구 방향"*	Gap analysis + Tier 랭킹

SKILL.md에 best practice를 한 번 작성 → 이후 매번 자동 적용



데모 1: 논문 읽기 → 구조화된 노트

paper-reading-note 스킬 실연

데모 2: Related Work 조사

PhysREPA Related_Work_Index.md — 46편 논문, 8개 카테고리

```
## 1. REPA & Representation Alignment

| Paper | Venue | Tier |

| REPA (2502.09725) | ICLR'25 Oral | Must-cite |

| VideoREPA | CVPR'25 | Must-cite |


## 4. Physics in Neural Networks

| Interpreting Physics (2504.12345) | Nature MI | Must-cite |

| PhysDreamer (2404.13026) | ECCV'24 | Should-cite |


## Positioning: 우리 연구의 위치

→ REPA를 image DiT에서 action DiT로 확장하는 첫 연구
```

comprehensive한 서베이가 한 번의 요청으로 완성



데모 3: 주간 연구 정리

weekly-research-digest 스킬 실연



데모 4: 세미나 PPT 자동 생성

research-seminar-ppt 스킬 실연 (이 PPT도 Claude가 만듦!)

실전: Claude Code 버그 발견 & 해결

DepthMem Implementation Log에서 발췌

발견된 버그

⚠ RuntimeError: size mismatch

config.json의 num_channels=3

→ 4ch input과 충돌

⚠ Trainable params 불일치

예상: 4M → 실제: 19.8M

LoRA 18.9M + patch embed 904K

자동 해결

✓ Dynamic patching 적용

runtime에서 config 동적 수정

pretrained weight 유지

✓ Param 분석 & 문서화

LoRA rank=16 → 18.9M

정확한 breakdown 기록

사람이 놓치기 쉬운 config 불일치를 Claude가 즉시 감지 & 수정

Tips & 주의사항

추천 Tips

★ Max 모드 강력 추천

연구용이면 반드시 Max plan 사용
(일반 모드 대비 압도적 사용량 차이)

CLAUDE.md 꼼꼼히 작성

프로젝트 구조, 연구 맥락 상세히

프롬프트는 구체적으로

"잘 해줘" X → 구체적 스펙 O

Skills 적극 활용

반복 작업은 스킬로 템플릿화

주의사항

Hallucination 주의

논문 인용, 수치는 반드시 검증
존재하지 않는 논문 지어낼 수 있음

코드 결과 직접 확인

"됐습니다" 를 맹신하지 말 것
wandb/로그로 항상 검증

민감 정보 주의

API 키, 비밀번호 노출 금지

과도한 의존 경계

직접 이해하지 못하면 연구 아님

Claude는 도구이지 연구자가 아닙니다 — 주도권은 항상 본인에게



Q & A

질문 있으시면 편하게 해주세요

이상오 · leesangoh@postech.ac.kr