# Apple Silicon M4 Pro 기반 Meta SAM 3 구축 및 에이전틱 엔지니어링 전략 보고서

## 요약 (Executive Summary)

본 보고서는 Meta의 최신 비전 파운데이션 모델인 'Segment Anything Model 3 (SAM 3)'를 Apple의 최신 하드웨어인 Mac Mini M4 Pro(24GB 통합 메모리) 환경에 구축하고 최적화하기 위한 포괄적인 기술 분석 및 실행 계획을 담고 있다. 특히, 단순한 모델 구동을 넘어 Claude Code, Google Antigravity, OpenAI Codex라는 세 가지 첨단 AI 에이전트 도구를 결합한 '에이전틱 워크플로우(Agentic Workflow)'를 통해 개발 효율성을 극대화하는 방안을 제시한다.

분석 결과, SAM 3의 8억 4,800만 파라미터 구조는 M4 Pro의 24GB 메모리 제약 내에서 추론(Inference) 단계에서는 충분히 운용 가능하나, 학습 및 미세 조정(Fine-tuning) 단계에서는 심각한 메모리 병목 현상을 초래할 수 있음이 확인되었다.1 또한, 공식 SAM 3 리포지토리가 NVIDIA GPU 전용 라이브러리인 Triton에 의존하고 있어 Apple Silicon의 Metal(MPS) 백엔드와 호환성 문제가 발생한다는 점이 주요 기술적 장벽으로 식별되었다.3

이에 대한 해결책으로 본 보고서는 Hugging Face Transformers 라이브러리를 활용한 우회 구현, EfficientSAM3를 이용한 지식 증류(Knowledge Distillation) 전략, 그리고 ExecuTorch 프레임워크를 통한 Apple Neural Engine(ANE) 가속화 방안을 제안한다. 이 모든 과정은 사용자가 직접 코드를 작성하는 대신, Antigravity를 관제탑(Mission Control)으로, Claude Code를 수석 엔지니어(Deep Engineer)로, Codex를 유틸리티 에이전트(Utility Agent)로 활용하는 '바이브 코딩(Vibecoding)' 방법론을 통해 수행된다.

## 1. 서론: 컴퓨터 비전의 패러다임 전환과 에이전틱 개발

### 1.1 프롬프트 기반 개념 분할(Promptable Concept Segmentation)의 도래

기존의 컴퓨터 비전 모델들은 특정 객체를 탐지하거나 분할하기 위해 사전에 정의된 클래스(예: 개, 고양이, 자동차)에 의존해야 했다. SAM 1과 SAM 2는 점(Point), 박스(Box)와 같은 기하학적 프롬프트를 통해 범용적인 분할(Segmentation)을 가능하게 했으나, 의미론적 이해(Semantic Understanding)는 결여되어 있었다. 2025년 11월 공개된 Meta의 SAM 3는 이러한 한계를 뛰어넘어 '프롬프트 기반 개념 분할(Promptable Concept Segmentation, PCS)'이라는 새로운 패러다임을 제시하였다.4

PCS는 텍스트 프롬프트(예: "노란 스쿨버스" 또는 "빨간 셔츠를 입은 사람")를 입력받아 영상 내의 모든 해당 객체를 식별하고, 분할하며, 비디오 프레임 전반에 걸쳐 추적(Tracking)하는 통합 기능을 의미한다.5 이는 단순한 픽셀 분할을 넘어, 모델이 시각적 정보와 언어적 개념을 동시에 이해하고 매핑할 수 있음을 의미하며, 로보틱스, 영상 편집, 자율 주행 등 다양한 산업 분야에 혁신적인 응용 가능성을 열어주었다.

### 1.2 에이전틱 개발 환경(Agentic Development Environment)의 부상

소프트웨어 개발 방식 또한 근본적인 변화를 겪고 있다. 단순히 코드를 더 빨리 작성하는 'Copilot' 단계를 넘어, 개발자가 목표를 설정하면 AI 에이전트가 계획을 수립하고, 코드를 작성하며, 디버깅까지 수행하는 '에이전틱 개발' 시대가 도래했다. Google의 Antigravity, Anthropic의 Claude Code, OpenAI의 Codex는 이러한 흐름을 주도하는 핵심 도구들이다.7

본 프로젝트는 이러한 최신 AI 기술들을 결합하여, 하드웨어의 제약을 소프트웨어적 최적화와 에이전트 기반의 자동화로 극복하는 것을 목표로 한다. 사용자의 Mac Mini M4 Pro는 단순한 로컬 컴퓨터가 아니라, 고성능 AI 에이전트들이 협업하여 SAM 3라는 거대 모델을 제어하는 'AI 워크스테이션'으로 전환될 것이다.

## 2. Meta SAM 3 아키텍처 심층 분석

### 2.1 듀얼 인코더-디코더 구조와 통합 비전 백본

SAM 3의 아키텍처는 기존 모델들과 달리 이미지 감지기(Detector)와 비디오 추적기(Tracker)가 단일 비전 인코더(Vision Encoder)를 공유하는 형태로 설계되었다.10 이 공유된 백본(Perception Encoder, PE)은 이미지의 시각적 특징을 추출하여 텍스트 및 예시 이미지(Exemplar) 임베딩과 정렬하는 역할을 한다. 총 파라미터 수는 약 8억 4,800만 개(848M)로, 이는 SAM 2의 가장 큰 모델보다 약 3~4배 더 큰 규모이다.1

| **구성 요소** | **역할** | **특징** | **파라미터 규모 (추정)** |
| --- | --- | --- | --- |
| **Perception Encoder (PE)** | 이미지/비디오 특징 추출 | 텍스트/이미지 멀티모달 정렬 | ~450M |
| **Memory Encoder** | 비디오 프레임 간 문맥 압축 | 시간적 일관성 유지 | ~150M |
| **Text Encoder** | 텍스트 프롬프트 이해 | CLIP 기반 경량화 모델 | ~100M |
| **Detector/Decoder** | 마스크 생성 및 객체 탐지 | DETR 기반 구조 | ~148M |

이러한 통합 구조는 메모리 효율성을 높이는 동시에, 정지 이미지에서 학습된 개념적 이해도가 비디오 추적 작업으로 자연스럽게 전이(Transfer)될 수 있도록 돕는다.4

### 2.2 존재 토큰(Presence Token)과 인식-위치 분리

SAM 3의 가장 혁신적인 기술 중 하나는 '존재 토큰(Presence Token)'의 도입이다. 기존의 개방형 어휘(Open-Vocabulary) 모델들은 텍스트 프롬프트와 유사한 시각적 특징이 발견되면, 실제로 해당 객체가 없더라도 억지로 마스크를 생성하는 환각(Hallucination) 문제가 빈번했다.

SAM 3는 인식(Recognition, "무엇이 있는가?")과 위치 파악(Localization, "어디에 있는가?")을 구조적으로 분리하였다.5 존재 헤드(Presence Head)는 이미지 전체를 분석하여 해당 개념이 존재하는지 여부를 먼저 확률적으로 판단한다. 이 점수가 임계값을 넘을 때만 마스크 생성 헤드가 활성화되어 구체적인 위치를 분할한다. 이는 "흰색 옷을 입은 선수"를 찾으라는 명령에 대해, 심판이나 관중을 제외하고 정확히 선수만 찾아내는 식별 능력을 획기적으로 향상시켰다.

### 2.3 고밀도 메모리 뱅크와 비디오 추적

비디오 분할을 위해 SAM 3는 SAM 2에서 도입된 메모리 메커니즘을 계승하되, 이를 개념(Concept) 기반으로 확장했다. 메모리 뱅크는 과거 프레임의 객체 특징을 저장하고, 현재 프레임의 정보와 결합하여 가려짐(Occlusion)이나 형태 변화가 발생해도 동일한 객체임을 유지한다.10 그러나 이 고밀도 메모리(Dense Memory) 방식은 프레임 수가 늘어날수록 메모리 사용량이 선형적으로 증가하는 단점이 있어, 24GB RAM 환경에서는 효율적인 관리 전략(예: EfficientSAM3의 Perceiver 기반 압축)이 필수적이다.12

## 3. 하드웨어 환경 분석: Mac Mini M4 Pro (24GB RAM)

### 3.1 통합 메모리 아키텍처(UMA)의 기회와 제약

Mac Mini M4 Pro의 핵심은 CPU와 GPU가 24GB의 고대역폭 메모리를 공유하는 통합 메모리 아키텍처(UMA)이다.

* **대역폭의 이점:** M4 Pro의 메모리 대역폭은 약 273GB/s(M3 Pro 대비 향상된 수치 추정)로, 일반적인 DDR5 PC 메모리(약 50-60GB/s)보다 월등히 빠르다. 이는 대규모 모델의 가중치(Weight)를 로딩하고 추론하는 속도에서 큰 이점을 제공한다.13
* **용량의 제약:** 24GB는 일반적인 컴퓨팅에는 넉넉하지만, 848M 파라미터 모델을 운용하기에는 빠듯한 용량이다.
  + **모델 로드 (FP16):** 약 1.7GB 소요.
  + **KV Cache 및 Activation:** 비디오 처리 시 프레임 해상도와 길이에 따라 수 GB에서 십수 GB까지 급증 가능.
  + **시스템 오버헤드:** macOS 자체와 Antigravity, Claude Code 등 개발 도구가 약 4~6GB를 점유.
  + **결론:** 학습(Training)은 불가능에 가깝지만, 추론(Inference)과 경량화된 미세 조정(LoRA)은 최적화를 통해 충분히 가능하다.

### 3.2 Apple Neural Engine (ANE) 및 Metal 가속

M4 Pro 칩에는 AI 연산 가속을 위한 강력한 Neural Engine(NPU)이 탑재되어 있다. 이를 활용하기 위해서는 PyTorch의 MPS(Metal Performance Shaders) 백엔드나 Apple의 CoreML 프레임워크를 사용해야 한다.14

* **MPS (Metal):** GPU를 활용한 가속. PyTorch에서 device='mps'로 호출 가능. 대부분의 연산을 지원하지만, SAM 3가 사용하는 일부 커스텀 연산(Triton 기반 Attention 등)은 지원하지 않아 호환성 문제가 발생한다.3
* **CoreML (ANE):** NPU를 활용한 초저전력/고속 추론. 모델 변환(Export) 과정이 필요하지만, 전성비와 추론 속도 면에서 가장 유리하다. M4 Pro의 NPU는 INT8/INT4 연산에 최적화되어 있어 양자화 적용 시 성능이 극대화된다.15

## 4. 에이전틱 툴체인(Agentic Toolchain) 구성 전략

사용자가 요청한 Claude Code, Antigravity, Codex를 유기적으로 결합하여 단일 프로젝트를 수행하는 '멀티 에이전트 오케스트레이션' 전략을 수립한다.

### 4.1 도구별 역할 정의 (Role Assignment)

| **도구 (Agent)** | **역할 (Role)** | **주요 기능 및 활용 방식** |
| --- | --- | --- |
| **Google Antigravity** | **관제탑 (Mission Control)** | 전체 프로젝트의 IDE 역할. 에이전트 관리, 시각적 아티팩트(Artifact) 확인, 계획 수립(Planning), 파일 시스템 관리. |
| **Claude Code** | **수석 엔지니어 (Deep Engineer)** | 터미널 기반의 심층 코딩. 복잡한 리포지토리 분석, 의존성 충돌 해결, 아키텍처 리팩토링, 코드 생성 및 실행. |
| **OpenAI Codex** | **유틸리티 에이전트 (Rapid Utility)** | 특정 구문 생성, 단위 테스트 작성, 간단한 스크립트 생성, 깃허브(GitHub) 연동 작업. |

### 4.2 시너지 아키텍처: 바이브 코딩(Vibecoding) 환경 구축

이 세 도구를 개별적으로 사용하는 것이 아니라, 하나의 흐름으로 연결하는 것이 핵심이다.

1. **통합 작업 공간:** Antigravity를 메인 IDE로 사용하며, 내장 터미널에서 Claude Code CLI를 실행한다.
2. **프록시(Proxy) 연결:** antigravity-claude-proxy를 사용하여 Antigravity 내부의 에이전트 기능을 Claude 3.5 Sonnet/Opus 모델로 구동하거나, Claude Code CLI가 Antigravity의 컨텍스트를 인식하게 한다.17
3. **워크플로우:**
   * **계획:** Antigravity의 'Planning Mode'를 통해 전체 구축 계획을 수립하고 PLAN.md로 저장.
   * **구현:** Claude Code CLI를 통해 PLAN.md를 읽고 실제 패키지 설치, 코드 수정, 파일 생성을 수행.
   * **보조:** 구현 중 특정 라이브러리 문법이나 간단한 스니펫이 필요할 때 Codex를 호출하여 즉시 생성.

## 5. 단계별 실행 계획 (Implementation Plan)

다음은 Mac Mini M4 Pro 환경에서 이 작업을 수행하기 위한 구체적인 단계별 가이드이다.

### 5.1 1단계: 필수 개발 도구 및 환경 설치

가장 먼저, 에이전트들이 활동할 수 있는 기초 환경을 구축해야 한다. Homebrew는 macOS의 필수 패키지 관리자이다.

1. **Homebrew 설치 (터미널 실행):**  
   Bash  
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
2. **Node.js 설치 (에이전트 도구 구동용):**  
   Claude Code와 Antigravity Proxy는 Node.js 환경에서 동작한다.  
   Bash  
   brew install node  
   node -v # 버전 확인 (v18 이상 권장)
3. **Python 환경 관리자 (Miniforge) 설치:**  
   Apple Silicon에 최적화된 패키지를 제공하는 Miniforge를 사용한다.  
   Bash  
   brew install miniforge  
   conda init zsh  
   source ~/.zshrc

### 5.2 2단계: AI 에이전트 도구 설치 및 연동

요청한 세 가지 도구를 설치하고 상호 연동한다.

**A. Claude Code 설치:** Anthropic의 공식 설치 가이드에 따라 Homebrew를 통해 설치한다.18

Bash

brew install --cask claude-code

설치 후 인증을 진행한다.

Bash

claude login

*설정 팁:* Claude Code가 터미널 명령어를 자율적으로 실행할 수 있도록 권한을 허용(Allow) 모드로 설정하면 작업 속도가 비약적으로 빨라진다.

**B. Google Antigravity 설치:** Antigravity는 독립형 IDE 애플리케이션이다.20

1. 웹사이트(antigravity.google/download) 접속.
2. **'Download for Apple Silicon'** 선택 (.dmg 파일).
3. 설치 후 실행하여 Google 계정으로 로그인하고 초기 설정(Mission Control Setup)을 완료한다.

**C. OpenAI Codex CLI 설치:** npm을 통해 Codex CLI를 전역으로 설치한다.21

Bash

npm install -g @openai/codex  
codex login # OpenAI API 키 또는 ChatGPT 계정 연동

**D. Antigravity-Claude 프록시 설정 (선택 사항이지만 강력 권장):** Antigravity IDE 내에서 Claude 모델의 추론 능력을 사용하고 싶다면 프록시를 설치한다.17

Bash

npm install -g antigravity-claude-proxy  
antigravity-claude-proxy start

이후 Antigravity 설정에서 모델 엔드포인트를 로컬 프록시(http://localhost:8080)로 지정한다.

### 5.3 3단계: SAM 3 환경 구축 및 코드 수정

공식 SAM 3 코드는 CUDA(NVIDIA GPU)에 의존적인 Triton 라이브러리를 사용하므로, M4 Pro에서는 수정이 필요하다. 이 과정을 Claude Code에게 위임한다.

1. **프로젝트 폴더 생성 및 Antigravity 실행:**  
   Bash  
   mkdir ~/Dev/SAM3\_M4  
   cd ~/Dev/SAM3\_M4  
   git init  
     
   Antigravity에서 해당 폴더를 연다.
2. **가상 환경 생성:**  
   Bash  
   conda create -n sam3 python=3.10  
   conda activate sam3
3. **Claude Code를 통한 설치 오케스트레이션:**  
   Antigravity 터미널에서 claude를 실행하고 다음 프롬프트를 입력한다. (Claude Code는 현재 상황을 인식하고 스스로 판단하여 명령어를 실행한다.)

**프롬프트:**  
"나는 Mac Mini M4 Pro(Apple Silicon)를 사용 중이다. Meta의 SAM 3 모델을 설치하고 싶은데, 공식 리포지토리는 Triton과 CUDA에 의존성이 있어 이 장비에서 돌아가지 않는다.

* 1. PyTorch Nightly 버전을 설치해라 (MPS 가속 지원).
  2. Hugging Face transformers 라이브러리의 최신 개발 버전을 소스에서 설치해라 (SAM 3 지원 포함).
  3. transformers를 사용하여 SAM 3 모델을 로드하고, device='mps'로 설정하여 추론하는 Python 스크립트(run\_sam3.py)를 작성해라.
  4. 만약 pin\_memory 관련 에러가 발생할 가능성이 있다면, 데이터로더 설정에서 이를 비활성화하는 코드를 포함해라."

**Claude Code의 예상 행동:**

* + pip install --pre torch torchvision --index-url... 실행.23
  + pip install git+https://github.com/huggingface/transformers.git 실행.24
  + Triton 의존성을 우회하는 Hugging Face 구현체를 활용한 Python 코드 작성 및 저장.

### 5.4 4단계: ExecuTorch를 이용한 온디바이스 최적화 (심화)

단순 실행을 넘어 성능을 최적화하기 위해, PyTorch 모델을 Apple Neural Engine(ANE)용으로 변환한다. 이는 M4 Pro의 NPU를 100% 활용하기 위한 핵심 단계이다.

1. **ExecuTorch 설치:**  
   Bash  
   pip install executorch

./install\_requirements.sh --pybind coreml

```

2. **CoreML 변환 스크립트 생성 (Codex 활용):**

터미널에서 Codex에게 요청하여 변환 스켈레톤 코드를 생성한다.

> **Codex 명령:** "Generate a python script to export a PyTorch model to ExecuTorch with CoreML backend partitioner. Use FP16 precision."

1. **양자화(Quantization) 적용:** 24GB RAM 절약을 위해 가중치를 INT8 또는 INT4로 양자화한다. M4 칩은 INT4 가속을 하드웨어적으로 지원한다.16  
   Python  
   # Claude Code가 작성할 코드 예시  
   from executorch.backends.apple.coreml.partition import CoreMLPartitioner, CoreMLCompileSpec  
   import coremltools as ct  
     
   partitioner = CoreMLPartitioner(  
    compile\_spec=CoreMLCompileSpec(  
    compute\_units=ct.ComputeUnit.ALL, # CPU+GPU+NPU 모두 사용  
    precision=ct.Precision.FLOAT16 # 또는 필요한 경우 양자화 적용  
    )  
   )  
   # 이후 export\_to\_edge 호출...

## 6. 성능 최적화 및 트러블슈팅 가이드

### 6.1 메모리 관리 전략 (24GB RAM)

* **스와핑(Swapping) 최소화:** macOS는 메모리가 부족하면 SSD를 RAM처럼 사용하는 스와핑을 적극적으로 수행한다. 하지만 이는 모델 추론 속도를 급격히 저하시킨다. Activity Monitor를 띄워두고 'Memory Pressure(메모리 압력)'가 노란색을 넘지 않도록 배치 사이즈(Batch Size)를 1로 유지해야 한다.26
* **브라우저 탭 관리:** Antigravity와 같은 Electron 기반 앱과 Chrome 브라우저는 메모리를 많이 차지한다. SAM 3 구동 시 불필요한 탭을 닫아 약 2~4GB의 여유 공간을 확보하는 것이 중요하다.

### 6.2 Triton/CUDA 에러 해결

만약 실행 중 RuntimeError: Triton packages are not available 에러가 발생하면, 이는 코드가 여전히 NVIDIA 경로를 타고 있다는 뜻이다.

* **해결:** transformers 라이브러리 내의 SAM 3 모델 로딩 부분에서 device\_map="auto" 대신 명시적으로 device="mps"를 지정해야 한다. Claude Code에게 "Refactor the script to strictly use MPS device and disable any CUDA-specific imports"라고 지시하면 자동으로 코드를 수정해 준다.3

### 6.3 pin\_memory 충돌 해결

Apple Silicon의 MPS 백엔드에서 DataLoader의 pin\_memory=True 옵션은 간혹 치명적인 오류를 발생시킨다.27

* **해결:** 데이터 로더 생성 시 pin\_memory=False로 강제 설정한다.

## 7. 결론 및 제언

Mac Mini M4 Pro (24GB)는 SAM 3와 같은 최신 파운데이션 모델을 구동하기 위한 '마지노선'이자 동시에 '최적의 가성비'를 가진 하드웨어이다. 통합 메모리 구조 덕분에 고가의 엔터프라이즈 GPU 없이도 모델을 로드할 수 있으나, 24GB의 용량 한계와 CUDA 미지원 문제는 소프트웨어적인 최적화를 요구한다.

본 보고서에서 제시한 **Antigravity-Claude-Codex 에이전틱 워크플로우**는 이러한 기술적 난관을 개발자가 일일이 해결하는 대신, AI 에이전트들이 협업하여 해결하도록 설계되었다.

1. **Antigravity**는 전체 프로젝트의 청사진을 그리고 관리한다.
2. **Claude Code**는 M4 아키텍처에 맞게 코드를 깊이 있게 수정하고 디버깅한다.
3. **Codex**는 반복적인 스크립팅 작업을 가속화한다.
4. **ExecuTorch**와 **CoreML**은 하드웨어 잠재력을 100% 끌어내어 실시간에 가까운 추론 속도를 보장한다.

이 전략을 통해 귀하는 단순한 모델 사용자를 넘어, 최첨단 AI 에이전트와 함께 일하는 시스템 아키텍트로서 SAM 3를 성공적으로 구축하고 활용할 수 있을 것이다.

#### 참고 자료

1. Meta AI Releases Segment Anything Model 3 (SAM 3) for Promptable Concept Segmentation in Images and Videos : r/computervision - Reddit, 2월 8, 2026에 액세스, <https://www.reddit.com/r/computervision/comments/1p2flxw/meta_ai_releases_segment_anything_model_3_sam_3/>
2. Macbook m4 pro - how many params can you train? : r/deeplearning - Reddit, 2월 8, 2026에 액세스, <https://www.reddit.com/r/deeplearning/comments/1mqr0z4/macbook_m4_pro_how_many_params_can_you_train/>
3. facebook/sam3 · Cannot run on Apple Silicon (M4) due to Triton - Hugging Face, 2월 8, 2026에 액세스, <https://huggingface.co/facebook/sam3/discussions/11>
4. SAM 3: Segment Anything with Concepts - Ultralytics YOLO Docs, 2월 8, 2026에 액세스, <https://docs.ultralytics.com/models/sam-3/>
5. [2511.16719] SAM 3: Segment Anything with Concepts - arXiv, 2월 8, 2026에 액세스, <https://arxiv.org/abs/2511.16719>
6. Meta SAM 3: Text-Driven Object Segmentation and Tracking for Video - AI FILMS Studio, 2월 8, 2026에 액세스, <https://studio.aifilms.ai/blog/meta-sam3-text-segmentation-tracking>
7. Introducing Claude Opus 4.6, 2월 8, 2026에 액세스, <https://www.anthropic.com/news/claude-opus-4-6>
8. Introducing Google Antigravity, a New Era in AI-Assisted Software Development, 2월 8, 2026에 액세스, <https://antigravity.google/blog/introducing-google-antigravity>
9. Codex | AI Coding Partner from OpenAI, 2월 8, 2026에 액세스, <https://openai.com/codex/>
10. facebookresearch/sam3: The repository provides code for running inference and finetuning with the Meta Segment Anything Model 3 (SAM 3), links for downloading the trained model checkpoints, and example notebooks that show how to use the model. - GitHub, 2월 8, 2026에 액세스, <https://github.com/facebookresearch/sam3>
11. SAM 3: Concept-Based Visual Understanding and Segmentation - PyImageSearch, 2월 8, 2026에 액세스, <https://pyimagesearch.com/2026/01/26/sam-3-concept-based-visual-understanding-and-segmentation/>
12. EfficientSAM3: Progressive Hierarchical Distillation for Video Concept Segmentation from SAM1, 2, and 3 - arXiv, 2월 8, 2026에 액세스, <https://arxiv.org/html/2511.15833v1>
13. On Device Llama 3.1 with Core ML - Apple Machine Learning Research, 2월 8, 2026에 액세스, <https://machinelearning.apple.com/research/core-ml-on-device-llama>
14. Deploy YOLO26 on Mobile & Edge with ExecuTorch - Ultralytics YOLO Docs, 2월 8, 2026에 액세스, <https://docs.ultralytics.com/integrations/executorch/>
15. Core ML Backend — ExecuTorch 1.0 documentation, 2월 8, 2026에 액세스, <https://docs.pytorch.org/executorch/1.0/ios-coreml.html>
16. Overview — Guide to Core ML Tools - Apple, 2월 8, 2026에 액세스, <https://apple.github.io/coremltools/docs-guides/source/opt-overview.html>
17. How to Use Free Antigravity AI Models in Claude Code - Syntackle, 2월 8, 2026에 액세스, <https://syntackle.com/blog/claude-code-free-using-antigravity-proxy/>
18. Claude Code's native installer is now generally available. : r/ClaudeAI - Reddit, 2월 8, 2026에 액세스, <https://www.reddit.com/r/ClaudeAI/comments/1ol0af7/claude_codes_native_installer_is_now_generally/>
19. claude-code - Homebrew Formulae, 2월 8, 2026에 액세스, <https://formulae.brew.sh/cask/claude-code>
20. Google Antigravity Download, 2월 8, 2026에 액세스, <https://antigravity.google/download>
21. openai/codex: Lightweight coding agent that runs in your terminal - GitHub, 2월 8, 2026에 액세스, <https://github.com/openai/codex>
22. antigravity-claude-proxy/README.md at main - GitHub, 2월 8, 2026에 액세스, <https://github.com/badrisnarayanan/antigravity-claude-proxy/blob/main/README.md>
23. Is MPS/Apple silicon deprecated now? Why? : r/pytorch - Reddit, 2월 8, 2026에 액세스, <https://www.reddit.com/r/pytorch/comments/1lc09q5/is_mpsapple_silicon_deprecated_now_why/>
24. SAM3 Video: Remove call to pin\_memory if the target device is not CUDA #43061 - GitHub, 2월 8, 2026에 액세스, <https://github.com/huggingface/transformers/pull/43061>
25. LLM Quantization: BF16 vs FP8 vs INT4 in 2026 - AIMultiple research, 2월 8, 2026에 액세스, <https://research.aimultiple.com/llm-quantization/>
26. Mac Mini M4 Pro 24GB – Smooth Performance but High Swap Usage. Should I Upgrade?, 2월 8, 2026에 액세스, <https://www.reddit.com/r/macmini/comments/1ii3hu6/mac_mini_m4_pro_24gb_smooth_performance_but_high/>
27. SAM3 inference fails on MPS (Mac M2) due to pin\_memory() usage · Issue #22954 - GitHub, 2월 8, 2026에 액세스, <https://github.com/ultralytics/ultralytics/issues/22954>