

周2赛博维修工 - 组装拆解实践

Week 2 Lesson Plan: Cyber Mechanic - Assembly and Disassembly Practice

Complete Teaching Document | AI-Driven Hardware/Software Module v2.0

COVER PAGE

[Center-aligned, bold, 24pt]

Week 2: Cyber Mechanic – Assembly and Disassembly Practice

赛博维修工 - 组装拆解实践

[Subtitle, 18pt]

AI-Simulated Hardware Learning for Low-Resource Rural High Schools

面向低资源农村高中的AI模拟硬件教学

[Author & Metadata, 12pt]

Author: Yangxia (杨霞)

Affiliation: In-service Teacher Perspective, Rural CS Education

Date: December 2025

Version: v2.0 (Optimized Edition)

Module: Sub-module 2.1 - Computer Hardware/Software Foundations

Class Configuration: 25 students, 5 groups (4-5 students/group)

Duration: 45 minutes

Prerequisites: Week 1 Circuit Exploration (Logisim CPU/Memory simulation)

[Theoretical Framework]

Based on: TPACK + Constructivism + UNESCO Digital Equity + Han et al. (2025) SDT+TAM+CLT Path Model

[Cover Image Description]

INSERT IMAGE: demos/week2_ar_qr.png (full-page width)

Caption: "Scan QR code to experience virtual farm machinery disassembly in AR. Week 2 extends Week 1's circuit foundation to hands-on OS installation practice."

Design: AR-generated 3D tractor with exposed chip components, overlaid with farmland background. Students can point phone camera to see virtual assembly steps floating above real workspace.

[Footer]

Framework Development Series | Reference: Table 5.3 Visual Models (CLIP row)

GitHub Repository: github.com/AI-Hardware-Module/week2-cyber-mechanic

Budget: ¥0 (100% open-source tools) | 2G Network Compatible

PAGE 1: SMART GOALS, THEORETICAL FOUNDATION, AND WEEK 1 CONNECTION

1.1 SMART Learning Objectives (Connected to Week 1 Baseline)

Extending Week 1 Foundation:

In Week 1, students explored circuit fundamentals through Logisim simulation (CPU as "farmland brain," memory as "workbench"). Interest baseline established at 3.2/5.0 (logs/week1_results.xlsx, n=25). Week 2 builds upon this foundation by transitioning from abstract circuits to tangible assembly/disassembly practice.

S - Specific:

- Students will master hardware assembly/disassembly workflow: chip installation, virtual OS pre-loading via QEMU emulation
- Experience "Cyber Mechanic" gamified simulation connecting Week 1's circuit logic (CPU→memory signal flow) to OS installation process
- Rural contextualization: "Tractor chip maintenance" narrative extending Week 1's "farmland brain" metaphor

M - Measurable:

- **Participation rate:** +20% increase (ViT collaboration heatmap $\geq 85\%$, up from Week 1's 80% baseline; Penchala et al., 2025 accuracy: 97.58%)
- **Interest score:** +15% increase (Likert 5-point scale, target: 3.2→3.7, paired t-test $p < 0.01$)
- **Knowledge retention:** Quiz accuracy $\geq 85\%$ (3-item assessment on assembly steps + CLIP matching, baseline Week 1: 88%)
- **Engagement quality:** SEM path coefficient $\beta = 0.40$ for autonomy→participation mediation (Han et al., 2025)

A - Achievable:

- Low-resource compatibility: Mobile AR scanning (2G network <500KB data) + offline QEMU fallback (Alpine Linux ISO 200MB)
- Zero-budget tools: QEMU, Tinkercad AR, CLIP transformers (local models), [MagicSchool.ai](#) free tier
- Teacher-friendly: Pre-scripted prompts, step-by-step operation lists, emergency video backups (demos/videos/week2_qemu_tutorial.mp4)

R - Relevant:

- Bridges Week 1 hardware concepts to Week 3-4 software applications (visual AI integration)
- Aligns with UNESCO digital equity principles: cultural relevance (farm machinery metaphors), inclusive design (weak/strong learner tracks)
- Connects to real-world vocational skills: electronics repair, OS troubleshooting (applicable to rural contexts)

T - Time-bound:

- 45-minute session breakdown:
 - 0-10 min: Introduction + Week 1 review + pre-assessment
 - 10-30 min: Hands-on QEMU assembly practice + CLIP matching game
 - 30-40 min: Group presentations + quiz evaluation + post-assessment
 - 40-45 min: Ethical reflection + "Eyes on the World" global trends discussion

1.2 Theoretical Foundation and Path Model

Core Theoretical Integration (Han et al., 2025 SDT+TAM+CLT Framework):

1) Self-Determination Theory (SDT):

- **Autonomy:** Gamified "Cyber Mechanic" role-play empowers students to choose assembly paths (weak learners: guided tutorial mode; strong learners: challenge mode with CLIP bonus tasks)
- **Competence:** Immediate feedback via scoring system (disassembly speed 5pts + CLIP matching 2pts + collaboration 3pts = 10pts total), virtual badges for milestones
- **Relatedness:** Small-group collaboration (5 groups \times 5 students), WeChat community sharing, peer voting on best "repair stories"

2) Technology Acceptance Model (TAM):

- **Perceived Usefulness:** QEMU simplifies abstract OS concepts through visual metaphor ("Installing OS = loading farmland irrigation system into tractor brain")
- **Perceived Ease of Use:** CLT-optimized design reduces cognitive load:
 - Short-sentence instructions ("Click → Drag → Install" vs. technical jargon)

- Farm machinery analogies bridge Week 1 circuit concepts to OS installation
- Fallback options for technical failures (offline videos, 2D diagrams if AR fails)

3) Cognitive Load Theory (CLT):

- **Intrinsic load:** Managed via scaffolding from Week 1 (students already understand CPU/memory roles)
- **Extrinsic load:** Minimized through storytelling (e.g., "Loose tractor chip = Week 1 circuit signal failure"), visual aids (AR 3D models), step-by-step breakdown
- **Germane load:** Enhanced via problem-solving tasks (Why does chip placement matter? How does OS connect to Week 1 circuits?)

Path Model Simulation (SEM Analysis):

Autonomy (Game Choice)
 ↓ $\beta=0.40$ (Han et al., 2025)
 Technology Acceptance (QEMU Ease of Use)
 ↓ $\beta=0.35$ (Mediation)
 Interest Increase (+15%)
 ↓ $\beta=0.45$
 Participation Behavior (ViT-detected collaboration $\geq 85\%$)
 ↓
 Knowledge Acquisition (Quiz $\geq 85\%$)

Expected Total Effect: Direct (0.25) + Indirect ($0.40 \times 0.35 \times 0.45 = 0.06$) ≈ 0.31 moderate effect size

Visual Model Integration (Table 5.3 Reference - CLIP Row):

- **CLIP Text-Image Matching:** Students input rural phrases ("Chip like farmland screw") to match Week 1 circuit diagrams (demos/week1_circuit.png)
- Simplifies abstract concepts: "How does chip connect to circuit?" → Visual similarity score >0.8 = successful understanding
- Gamification: CLIP matching adds +2 bonus points to team score

1.3 Week 1 Connection and Learning Progression

Week 1 Recap ("Circuit Exploration" Foundation):

Students used Logisim to simulate:

- **CPU:** "Village chief brain directing harvesters" (signal transmission logic)
- **Memory:** "Farmer's workbench storing tools" (temporary data storage)
- **Baseline metrics:** Interest 3.2/5.0, Participation 80%, Quiz accuracy 88%
- **Key takeaway:** "Hardware components communicate through signal circuits"

Week 2 Extension ("From Circuits to Assembly"):

- **Conceptual bridge:** "Week 1 taught *what* components do. Week 2 teaches *how* to assemble them into a working system."
- **Narrative continuity:** "Remember the farmland brain (CPU)? Today we install it into the tractor (computer case) and load irrigation software (OS)."
- **Technical progression:** Logisim virtual circuits → QEMU virtual hardware assembly → Real-world OS installation simulation

Data Import and Continuity:

- Pre-load Week 1 data: logs/week1_results.xlsx (25 students \times interest scores, participation heatmaps)
- Display Week 1 success stories: logs/week1_student_stories.txt ("Circuit like farmland signal transmission – I finally get it!")
- Use Week 1 circuit images: demos/week1_circuit.png for CLIP matching exercises

1.4 Week 1-Week 2 Comparison Table

Aspect	Week 1: Circuit Exploration	Week 2: Cyber Mechanic Assembly	Connection Logic
Core Tool	Logisim (circuit simulation)	QEMU (virtual hardware assembly) + CLIP (AI matching)	Logisim simulates component logic → QEMU assembles components into system
Knowledge Focus	CPU/memory concepts (what components do)	Chip installation/OS loading (how components work together)	Foundation → Application
Learning Metaphor	Farmland signal transmission (abstract)	Tractor chip repair (concrete, hands-on)	Week 1 abstraction → Week 2 contextualization
Quantitative Target	Interest +15% (3.2 baseline)	Participation +20% (85% target)	Build on Week 1 engagement gains
Assessment Method	Pre-post Likert survey, quiz (88% accuracy)	ViT collaboration heatmap (97.58% accurate) + quiz (85% target)	Add behavioral observation to self-report
Cognitive Load	CLT: Short metaphors ("CPU = brain")	CLT: Step-by-step disassembly ("Unscrew chip like tractor bolt")	Consistent low-load approach
Inclusion Strategy	Weak learners: story version; Strong learners: challenge version	Weak: Offline video backup; Strong: Global OS trends quiz	Differentiated instruction continuity
Technology Barrier	Logisim offline (low barrier)	QEMU may lag → Alpine ISO 200MB + fallback video	Proactive risk mitigation
Cultural Relevance	Farmland irrigation circuit	Tractor chip maintenance story	Deepen rural contextualization
Collaboration	5 groups (80% participation, Week 1 baseline)	5 groups (target 85%, ViT-detected)	Incremental participation growth
Global Perspective	MagicSchool quiz on quantum chips	Gemini quiz on international OS trends	Expand from tech hardware to software applications

INSERT IMAGE: demos/week1_interest_heatmap.png (half-page width, left-aligned)

Caption: "Week 1 Collaboration Heatmap Baseline. Red zones indicate high participation (80% average). Week 2 targets 85% through enhanced gamification."

PAGES 2-5: DETAILED TIMELINE, ACTIVITY DESIGN, AND EXTENDED SPECIFICATIONS

2.1 Session Overview and Group Configuration

Class Setup:

- **Total students:** 25 (rural high school class, mixed ability levels)
- **Group structure:** 5 groups × 5 students each
 - Assign roles per group: 1 operator (runs QEMU), 2 discussants (troubleshoot), 1 photographer (captures screenshots), 1 storyteller (writes reflection)
- **Pre-class preparation:**
 - WeChat group message (1 day prior): Share PDF manual (docs/week2_teaching_plan.pdf), AR QR code (demos/week2_ar_qr.png), quiz link (quiz_library/week2_quiz.json)
 - Notion database link: Pre-fill student names, import Week 1 baseline scores (logs/week1_results.xlsx)

Differentiated Instruction Tracks (Lee et al., 2025 Inclusion Principles):

Principle	Weak Learner Track	Strong Learner Track
1. Identity (Cultural Relevance)	Review Week 1 farmland story, simple tractor metaphor	Challenge: Compare rural vs. urban OS needs (e.g., offline agricultural apps)
2. Technology Access	Offline QEMU video backup (demos/videos/week2_qemu_tutorial.mp4) if internet fails	Mobile AR scanning (requires stable 2G+)

Principle	Weak Learner Track	Strong Learner Track
3. Design Participation	Follow guided 5-step assembly tutorial	Open-ended: Design custom chip layout, optimize OS installation order
4. Content Relevance	"Chip installation = fixing tractor screw" (concrete)	"OS kernel architecture" with global case studies (UNESCO data)
5. Belonging	Peer support (strong students mentor weak students within groups)	Leadership roles (present group findings to class)

Risk Mitigation Strategy (Reference Table 5.4 Risk Management):

Risk	Probability	Impact	Fallback Plan
QEMU software lag	Medium (30%)	Students frustrated, time wasted	Switch to Alpine Linux ISO (200MB, lighter) + tcg acceleration mode OR play pre-recorded video (demos/videos/week2_qemu_tutorial.mp4)
2G network too slow for AR	Medium (25%)	AR QR code won't load	Offline 2D diagram backup (demos/week2_assembly_diagram.png)
Students don't understand OS concept	Low (15%)	Confusion, low quiz scores	Reinforce Week 1 analogy: "OS = irrigation software for farmland brain (CPU)"
Group conflict	Low (10%)	Unequal participation	Teacher intervention, rotate roles mid-session

2.2 Detailed Timeline Table (45-Minute Breakdown)

Time	Activity Phase	Detailed Description (Gamified + Localized)	Tools/Resources (Low-Resource Compatible)	Teacher Actions (Week 1 Context)
0-10 min Introduction & Activation	Week 1 Review + Story Hook	<p>[5 min] Recap Week 1: Teacher displays demos/week1_interest_heatmap.png via projector: "Last week you simulated farmland brain (CPU) and workbench (memory) with Logisim. Red zones = high participation teams. Today we assemble these parts into a real tractor (computer)!"</p> <p>[3 min] New Story Introduction: "Imagine: Farmer Zhang's tractor chip is loose (like Week 1 circuit signal failure). As Cyber Mechanics, we'll disassemble, diagnose, and reinstall the OS. Who can fix it fastest?"</p> <p>[2 min] Brainstorm Prompt: "Share via WeChat voice: What's the hardest part of assembling a tractor? (Links to CPU installation difficulty)"</p> <p>Differentiation: - Weak learners: Teacher retells Week 1 story with visuals - Strong learners: Bonus challenge quiz: "If CPU is brain, what body part is the hard drive?"</p>	<p>Tools: - Projector + demos/week1_interest_heatmap.png - WeChat voice sharing (records to logs/week2_intro_feedback_audio.zip) - MagicSchool.ai warm-up quiz (3 questions, e.g., "How does Week 1 CPU connect to OS?") - Generate via prompt: "Create 3 multiple-choice questions linking Logisim circuits to OS installation, rural context" - Export to quiz_library/week2_warmup_quiz.json</p> <p>Low-Resource Note: Voice messages compress well (<1MB/group), suitable for 2G networks</p>	<p>Teacher Actions: <input type="checkbox"/> Display V zone success <input type="checkbox"/> Launch WeChat group <input type="checkbox"/> Collect voice of participation <input type="checkbox"/> Administer (logs/week column)</p> <p>Student Outcomes: - Voice brainstorm students, logs/week2 - Pre-test import Week 1 comparision - Quiz warmup quiz_library</p>
10-30 min Hands-On Practice	Virtual Disassembly Race + CLIP Matching Game	<p>[15 min] QEMU Assembly Challenge:</p> <p>Setup (Teacher Demo 3 min): Teacher projects screen, runs QEMU command: <pre>qemu-system-x86_64 -cdrom tools/images/alpine-3.18-x86_64.iso -m 256 -enable-kvm</pre> "See? The tractor (VM) boots up. Now</p>	<p>Tools:</p> <p>QEMU Emulator: - Download: tools/virtualizers/qemu-7.2-x86_64.exe (Windows) or qemu-system-x86_64 (Linux) - ISO: tools/images/alpine-3.18-x86_64.iso (200MB, lightweight to</p>	<p>Teacher Actions: <input type="checkbox"/> Demo QEMU projector) <input type="checkbox"/> Circulate troubleshooting <input type="checkbox"/> Monitor student group launch <input type="checkbox"/> Take group</p>

Time	Activity Phase	Detailed Description (Gamified + Localized)	Tools/Resources (Low-Resource Compatible)	Teacher Actions (Week 1 Coordination)
		<p>drag-drop the chip (ISO file) to install irrigation OS."</p> <p>Student Task (12 min): Groups compete to complete 5-step assembly:</p> <ol style="list-style-type: none"> 1. Launch QEMU (cmd command) 2. Mount Alpine ISO 3. Boot virtual machine 4. Navigate installation menu (keyboard only, simulates chip insertion precision) 5. Complete OS installation (logs boot message) <p>Scoring: - Speed (5 pts): First group to finish = 5pts, 2nd = 4pts, 3rd = 3pts, etc. - Accuracy (checked via screenshot): Must show "Installation complete" message</p> <p>Fallback for Lag: If QEMU freezes (>2 groups affected), teacher pauses, switches all groups to video tutorial (demos/videos/week2_qemu_tutorial.mp4, 8-minute walkthrough). Students watch and answer comprehension questions for partial credit.</p> <p>[5 min] CLIP Text-Image Matching Bonus:</p> <p>Task: Students run scripts/clip_demo.py: - Input text: "芯片如农田螺丝" (Chip like farmland screw) - Match against: demos/week1_circuit.png (Week 1 circuit diagram) - Output: Similarity score (0-1 scale)</p> <p>Scoring: - Score ≥0.8: +2 bonus points - Score 0.6-0.79: +1 point - Score <0.6: 0 points (but discussion opportunity: "Why didn't it match? Revise your metaphor!")</p> <p>Localization: Encourage creative rural phrases: - "Chip like rice grain in circuit paddy" - "CPU socket like tractor engine bay" Best metaphor voted by class → extra +1 point</p>	<p>prevent lag) - Acceleration: Enable KVM (Linux) or HAXM (Windows) if available; otherwise use tcg mode (slower but compatible with low-end PCs)</p> <p>CLIP Model: - Script: scripts/clip_demo.py (see Page 7 for full code) - Pre-downloaded model: models/clip-vit-base-patch32/ (local path, no internet needed) - Input images: demos/week1_circuit.png (Week 1 asset reused)</p> <p>AR Tinkercad: - QR Code: demos/week2_ar_qr.png - Pre-built 3D model: Tractor with exposed chip compartment (created in Tinkercad, exported as AR-viewable link) - Students scan with phone camera, see holographic assembly steps</p> <p>Offline Fallback: - Video: demos/videos/week2_qemu_tutorial.mp4 (YouTube mirror: youtube.com/watch?v=EXAMPLE, 8 min, Chinese subtitles) - 2D Diagram: demos/week2_assembly_diagram.png (step-by-step illustrated guide)</p> <p>Collaboration Tracking: - ViT detection script: scripts/vit_collab_detect.py - Captures group photo every 5 minutes - Analyzes student positions using object detection (DETR model) - Generates heatmap: demos/week2_collab_heatmap.png</p>	<p>(smartphone demos/group) □ If ≥2 groups pause → plan adjusted timing □ Run CLIP example (2 min) □ Collect AR photos (students use phones)</p> <p>Student Outcomes - QEMU Success - Filename: 5]_qemu_in - Must show "Installation complete" - CLIP Match - Saved to: (columns: C Points) - Example results: 0.82, 2 - AR Experience - Notion terminology model teaching placement? - Response logs/week2 - Collaboration - Raw images: demos/group5]_photo[1- - Processed images: demos/week2_collab_heatmap.png (generated)</p>
30-40 min Presentation & Assessment	Group Showcase + Quiz + Post-Test	<p>[5 min] Group Presentations:</p> <p>Each group sends 1 representative (1 min/group): - Show QEMU screenshot: "We installed Alpine in 8 minutes!" - Share best CLIP metaphor: "We scored 0.85 with 'Chip like farmland water pump'"</p>	<p>Tools:</p> <p>Presentation: - WeChat group album (students upload screenshots) - Projector displays each group's submission - Voting: WeChat built-in poll feature (free, instant results)</p>	<p>Teacher Actions □ Call each group □ Project screen □ Launch Vedio winner □ Display quiz questions to students □ Monitor chat (MagicSchool)</p>

Time	Activity Phase	Detailed Description (Gamified + Localized)	Tools/Resources (Low-Resource Compatible)	Teacher Actions (Week 1 Context)
		<p>- Tell assembly story: "Connecting Week 1 CPU circuit to OS was like linking irrigation channels to the water source"</p> <p>Class Voting: WeChat poll: "Which group's story best connected Week 1 to Week 2?" (emoji reactions 👍) Winner: +1 bonus point</p> <p>[4 min] Quiz Evaluation:</p> <p>Quiz Content (3 items): 1. Multiple choice: "What does OS installation simulate in the tractor metaphor?" (A: Fuel tank B: Irrigation software ✓ C: Steering wheel) 2. True/False: "Week 1's CPU circuit directly powers the OS" (False – OS is software layer above hardware) 3. CLIP matching: "Match this image [demos/week1_circuit.png] with the correct text" (Options: A: Circuit board ✓ B: Tractor engine C: Rice paddy)</p> <p>Delivery Method: - Mobile quiz via MagicSchool.ai QR code (scans in 3 seconds, 2G compatible) - Auto-grading: Results save to quiz_library/week2_quiz_results.json - Target: ≥85% class average (21/25 students score 2/3 or 3/3)</p> <p>[1 min] Post-Test Interest Survey:</p> <p>Same Likert scale as Week 1 (reuse questions for comparability): 1. "I find hardware learning interesting" (1=Strongly Disagree → 5=Strongly Agree) 2. "I understand how components assemble into systems" 3. "Low-resource simulations help me learn" 4. "I want to continue exploring AI topics" 5. "Rural stories help me remember concepts"</p> <p>Data Collection: - Notion form link (WeChat group) - Auto-populates logs/week2_results.xlsx (Post-Interest column) - Teacher script runs stats.ttest_rel() immediately for p-value (see Page 9 for formula)</p>	<p>Quiz Platform: - MagicSchool.ai Quiz Generator - Prompt: "Generate 3 questions on OS installation + CLIP matching, link to Week 1 circuits, rural context, Chinese language" - Export QR code: quiz_library/week2_quiz_qr.png - Mobile-optimized: Loads in 3 sec on 2G (text-only, no images except demos/week1_circuit.png compressed to <50KB)</p> <p>Survey Tool: - Notion form (config/notion_survey_week2_template) - Fields: StudentID, Post_Interest_Q1-Q5 (Likert 1-5), Timestamp - Link: notion.so/week2-post-survey (shareable in WeChat)</p> <p>Analysis Script: - scripts/week2_eval.py - Imports: pandas, scipy.stats, matplotlib - Runs: Paired t-test (pre vs. post), Cohen's d, bar chart generation - Output: logs/week2_statistical_report.txt + demos/week2_interest_chart.png</p>	<p><input type="checkbox"/> Send post after quiz <input type="checkbox"/> Run script class (or with group)</p> <p>Student Outcomes - Presentation - WeChat a total, 5 groups - Top-voted logs/week2 - Quiz Scores - Individual quiz_library - Class average MagicSchool.ai "Class score" - Post-Test - 25 rows × logs/week2 - Expected 3.2, +15.6% - Statistical Analysis - Auto-generated logs/week2 - Contents: target), effective</p>
40-45 min Reflection & Global Perspective	Ethical Discussion + "Eyes on the World" Trends	<p>[3 min] Ethical Reflection:</p> <p>Teacher poses 2 questions (WeChat voice/text responses):</p> <p>1. Digital Equity: "We used free QEMU simulations instead of buying real computers. Does this</p>		

Time	Activity Phase	Detailed Description (Gamified + Localized)	Tools/Resources (Low-Resource Compatible)	Teacher Ac (Week 1 Co
		<p>achieve UNESCO's digital fairness goal? Or do we still fall behind cities with physical labs?"</p> <ul style="list-style-type: none"> - Encourage debate: Some students may say "Good enough," others "We want real hardware too" - Teacher validates both: "Simulations democratize access (Panjwani, 2024: +25% rural participation potential), but we must keep pushing for resource equity" <p>2. Data Privacy (Kong & Zhu, 2025 CFA Pre-Post Test):</p> <p>"We took photos for ViT collaboration analysis. Is it okay to use AI to track your participation? What if this data leaks?"</p> <ul style="list-style-type: none"> - Link to Week 1 ethics: "Remember we discussed farmland data privacy? Same applies here" - Introduce CFA pre-test item: "I trust AI to handle my learning data responsibly" (1-5 scale, baseline measurement for future comparison) <p>[2 min] Global Trends Quiz (Gemini Education v2):</p> <p>Sample Question:</p> <p>"In 2025, UNESCO reports that 60% of rural schools in developing countries use VM-based OS training (like our QEMU). Which continent leads in adoption?"</p> <p>A) Africa</p>		

B) Asia ✓

C) South America

Purpose:

- Contextualize local practice within global movements
- "Eyes on the World" theme: Students realize they're part of international innovation, not lagging behind
- Builds motivation: "Rural schools worldwide face same challenges – and find creative solutions like us!"

Delivery:

- Gemini API call (scripts/gemini_quiz_generator.py)
 - Prompt: "Generate 1 multiple-choice question on global rural OS education trends, cite UNESCO 2024-2025 data"
- Display on projector, students shout answers
- Correct answer revealed with explanation: "Asia leads due to high mobile penetration + low-cost VM solutions like ours!"

Iteration Log:

Teacher fills Notion template (2 min post-class):

- **Successes:** "85% participation achieved! QEMU lag only affected 1 group (fallback video worked)"
- **Challenges:** "2 weak learners struggled with CLIP concept – need simpler metaphor next time"
- **Next Steps:** "Week 3: Transition to visual AI (Object Detection farmland crops). Preview in closing remarks" | **Tools:**

Ethics Discussion:

- WeChat voice/text forum (free, familiar to students)
- Notion shared doc: logs/week2_ethics_discussion.txt (teacher pastes key student quotes)
- Reference materials:

- Kong & Zhu (2025) CFA questionnaire (translated excerpt in docs/ethics_questionnaire_sample.pdf)
- Panjwani (2024) rural equity data (infographic: docs/panjwani_2024_equity_stats.png)

Global Quiz:

- Gemini Education v2 API
 - Free tier: 10 queries/day (sufficient for 1 question/class)
 - Script: `scripts/gemini_quiz_generator.py`
 - Output: Saves question to `quiz_library/week2_global_quiz.json`
- UNESCO 2025 data source: unesco.org/rural-education-report-2025 (cited in quiz explanation)

Iteration Template:

- Notion database: `config/iteration_log_template`
 - Fields: Week, Date, Successes (text), Challenges (text), Next Steps (text), Student Quotes (text)
 - Auto-links to: `logs/week2_statistical_report.txt`, `demos/week2_collab_heatmap.png`

Week 3 Preview Materials:

- Teaser slide: "Next week – AI eyes detect farmland pests!" (image: `demos/week3_preview_object_detection.png`)
- Pre-reading (optional): Share article link on WeChat: "How Object Detection helps farmers" | **Teacher Actions:**
 - ☐ Pose ethics questions, moderate 2-min discussion
 - ☐ Collect voice responses (screenshot participation, save key quotes)
 - ☐ Launch Gemini quiz (project on screen)
 - ☐ Facilitate answer reveal + discussion
 - ☐ Tease Week 3 topic (30 seconds): "Imagine your phone camera detecting rice pests automatically – that's Object Detection!"
 - ☐ Post-class: Fill iteration log (Notion), upload all data files (`logs/`, `demos/`)

Student Outputs:

- **Ethics Responses:**
 - Voice messages: `logs/week2_ethics_audio.zip` (25 students, ~30 sec each = ~12 MB total)
 - Text summaries: Teacher extracts 5 representative quotes → `logs/week2_ethics_discussion.txt`
 - Example quote: "Simulation is fair but not perfect – we deserve both virtual and real tools" (Student 12)
- **Global Quiz Answer:**
 - Show of hands (teacher counts): 18/25 correct (72% accuracy)
 - Logged to: `quiz_library/week2_global_quiz_results.txt`
- **Reflection Notes:**
 - Notion entries: `logs/week2_student_reflections/` (25 individual files, templated format)
 - Prompt: "What's 1 thing you learned? 1 thing still confusing? How does Week 2 connect to Week 1?"
- **Iteration Log (Teacher):**
 - Complete entry: `logs/iteration_notes_week2.txt`
 - Key metrics recorded: Participation 85%, Interest +15%, Quiz 87%, 1 technical failure (resolved via fallback) |

2.3 Gamification: Detailed Scoring Rules and Rewards

Total Score Breakdown (10 Points Maximum per Group):

Category	Points	Criteria	Measurement Method
Disassembly/Assembly Speed	5 points	1st place: 5pts 2nd place: 4pts 3rd place: 3pts 4th place: 2pts 5th place: 1pt <i>Based on timestamp when "Installation complete" screenshot submitted</i>	Teacher records finish times in <code>logs/week2_speed_rankings.xlsx</code>
CLIP Matching Accuracy	2 points	Score ≥0.8: 2pts Score 0.6-0.79: 1pt Score <0.6: 0pts <i>Automated via <code>scripts/clip_demo.py</code> output</i>	Script auto-saves scores to <code>logs/week2_clip_scores.csv</code>

Category	Points	Criteria	Measurement Method
Team Collaboration	3 points	ViT heatmap analysis: - All 5 members actively engaged (within 2m radius, sustained 15+ min): 3pts - 4 members engaged: 2pts - ≤3 members engaged: 1pt <i>Analyzed post-class via scripts/vit_collab_detect.py</i>	Heatmap visualization: demos/week2_collab_heatmap.png Raw data: logs/week2_collab_metrics.json
Bonus: Best Story	+1 point	Winning WeChat poll (most 👍 for "Which group best connected Week 1 to Week 2?")	Teacher tallies votes, records winner in logs/week2_poll_results.txt

Reward System:

- **Virtual Badges (Displayed in WeChat group + Notion leaderboard):**

- 🏆 **Gold Mechanic:** 9-10 points (top scorer)
- 🥈 **Silver Mechanic:** 7-8 points
- 🥉 **Bronze Mechanic:** 5-6 points
- 🛠️ **Apprentice Mechanic:** 3-4 points
- 🔧 **Trainee:** 1-2 points

- **Cumulative Semester System:**

- Badges accumulate across Week 1-8
- Top 3 groups at end of module (Week 8) receive:
 - Certificate (printable PDF from GitHub repo)
 - Featured in school newsletter (with student permission)
 - Extra credit (5% boost to final project grade)

Inclusion Safeguards (Lee et al., 2025 Principles):

- **Principle 3 (Design Participation):** All groups start with 1 "participation point" to prevent zero-score discouragement
- **Principle 2 (Technology Access):** Groups affected by QEMU lag receive automatic +1 "resilience point" (capped at 1 per session) to compensate for technical barriers beyond their control
- **Principle 5 (Belonging):** Peer mentoring bonus: If strong learner helps weak learner troubleshoot (teacher-observed), both individuals earn +0.5 points (can exceed group cap, applied to personal cumulative score)

2.4 Low-Resource Optimization Strategies

Challenge: Rural schools often face:

- Slow internet (2G/3G, <500 kbps)
- Outdated PCs (2-4GB RAM, no GPU)
- Limited bandwidth budgets
- Intermittent electricity

Solutions Implemented:

1) QEMU Configuration for Low-End Hardware:

```
# Standard QEMU command (may lag on old PCs):
qemu-system-x86_64 -cdrom alpine.iso -m 512

# Optimized command for low-resource environments:
qemu-system-x86_64 \
  -cdrom tools/images/alpine-3.18-x86_64.iso \ # Lightweight distro (200MB vs. Ubuntu 3GB)
  -m 256 \ # Minimal RAM allocation
```

```

-enable-kvm \           # Hardware acceleration (if available)
-cpu host \             # Match host CPU features
-smp 2 \                # 2 virtual CPUs (reduces load)
-nographic \           # No GUI overhead (text mode)
-display none \         # Disable display window
-serial stdio           # Output to terminal (teacher can project)

```

Fallback if KVM unavailable (Windows XP-era PCs):

- Use `-accel tcg` (software emulation, slower but universal)
- Pre-install Alpine to virtual disk (tools/images/alpine_preinstalled.qcow2), skip installation step
- Students only practice boot sequence + basic commands (reduces time from 15 min to 5 min)

2) Network Optimization:

Resource	Original Size	Compressed Size	Optimization Method
Alpine ISO	200 MB	N/A (already minimal)	Selected lightest Linux distro
CLIP model	350 MB	N/A (pre-downloaded to local)	Download once via teacher's home WiFi, copy to all PCs via USB
Week 1 circuit image	1.2 MB (PNG)	85 KB (JPEG 80% quality)	ImageMagick: <code>convert -quality 80 week1_circuit.png week1_circuit.jpg</code>
AR QR code	12 KB	N/A (already small)	Vector format, scales without quality loss
Tutorial video	120 MB (1080p)	18 MB (480p, H.265)	FFmpeg: <code>ffmpeg -i input.mp4 -vcodec libx265 -crf 28 -preset fast output.mp4</code>

Total Week 2 bandwidth requirement:

- Online mode: ~50 KB (QR code + quiz + survey, text-based)
- Offline mode: 0 KB (all resources pre-loaded on PC)

3) Electricity Continuity:

- **Battery backup:** Encourage students to charge phones before class (AR scanning works offline once QR loaded)
- **Printed fallback:** docs/week2_assembly_diagram_print.pdf (B&W, 2 pages, costs ¥0.20/copy)
- **Power outage protocol:** If electricity fails mid-session:
 1. Teacher narrates assembly steps from memory/printed script
 2. Students sketch assembly process on paper (hand-drawn diagrams)
 3. Resume QEMU practice next session, give partial credit for sketches

PAGES 6-8: DETAILED TEACHING SCRIPTS AND TECHNICAL IMPLEMENTATION

3.1 Teacher Operation Checklist (Step-by-Step)

Pre-Class Setup (30 minutes before class):

☐ Test QEMU on teacher PC:

```

cd tools/virtualizers
qemu-system-x86_64 -cdrom ../images/alpine-3.18-x86_64.iso -m 256

```

- Verify ISO boots to login prompt (username: `root`, no password)
- Screenshot success screen → demos/teacher_qemu_test.png (shows students what to expect)

☐ Distribute resources via WeChat:

- Send PDF manual: "Week 2 Guide.pdf" (10 pages, <2 MB)
- Send AR QR code image: demos/week2_ar_qr.png (12 KB)

- Send quiz link: "Scan this to access 3-question quiz"
- Pin message: "Week 2 starts in 30 min! Charge your phones for AR scanning 📱"

❑ Prepare Notion database:

- Open: config/week2_data_collection_template
- Pre-fill student names from Week 1: Import logs/week1_results.xlsx
- Create new columns: Pre_Interest_W2, Post_Interest_W2, CLIP_Score, Collab_Rating

❑ Set up projection:

- Load presentation slides (or simply open folders):
 - Slide 1: demos/week1_interest_heatmap.png (Week 1 recap)
 - Slide 2: Tractor disassembly story (image: stock photo of tractor with tool icons)
 - Slide 3: QEMU demo terminal (teacher's test screenshot)
 - Slide 4: CLIP matching example (split-screen: text input + Week 1 circuit image + score)

❑ Test fallback video:

- Play demos/videos/week2_qemu_tutorial.mp4 (ensure audio works)
- Bookmark timestamp for installation section (starts at 3:45)

In-Class Script (0-45 minutes):

[0-10 MIN] Introduction

👤 Teacher says:

[Display demos/week1_interest_heatmap.png]

"Good morning, Cyber Mechanics! Last week, you built farmland brain circuits in Logisim. Look at this heatmap – red zones are where you collaborated best. Group 3 had 95% participation! Today, we level up: from circuits to actual computer assembly."

[Switch to tractor image]

"Imagine Farmer Zhang's tractor chip got loose during plowing – same problem as Week 1's circuit signal failure, remember? Your mission: disassemble the tractor's 'brain,' diagnose the issue, and reinstall the operating system. It's like installing irrigation software into the farmland control center."

Brainstorm Prompt:

"WeChat voice message, 30 seconds: What's the scariest part of opening a tractor engine? Link it to computer assembly."

[Teacher scrolls through voice messages, plays 2-3 examples]

"Great answers! Student 8 said 'afraid of losing screws' – yes, chip pins are like tiny screws. Student 15 said 'mixing up parts' – exactly why we need step-by-step QEMU simulation."

Pre-Test:

[Share Notion survey link]

"Quick 5-question survey – same as Week 1. Rate your interest 1-5. This establishes our Week 2 baseline."

[Wait 2 minutes for submissions, show live response count: "18/25 submitted... 23/25... all done!"]

[10-30 MIN] Hands-On Practice

👤 Teacher says:

[Project teacher PC screen running QEMU]

"Watch me. I type this magic command:"

```
qemu-system-x86_64 -cdrom tools/images/alpine-3.18-x86_64.iso -m 256
```

[Narrate as boot sequence appears]

"See the tractor engine starting? Black screen, white text – that's the 'brain' waking up. It asks for a username. I type `root` and press Enter. No password needed – this is a practice tractor, not a real one with security."

[Navigate to installation option in Alpine menu]

"Here's the OS installation menu. In real life, this is like loading irrigation schedules into the tractor's computer. I select 'Install Alpine' → follow prompts → done in 3 minutes."

Student Task Assignment:

"Your turn! 5 groups compete:

- **Speed challenge (5 pts):** First to submit 'Installation complete' screenshot wins
- **CLIP matching (2 pts):** Run the chip metaphor script – score >0.8 for full points
- **Collaboration (3 pts):** I'm taking photos. AI will detect if all 5 members stay engaged.

Go! Start your engines. Timer begins now."

[Teacher circulates among groups]

Group 1 check-in:

👤 "Stuck on command? Let me see your screen. Ah, typo: `alpine.iso` should be `alpine-3.18-x86_64.iso`. Try again."

Group 3 check-in:

👤 "Great! Your VM booted. Now type `setup-alpine` to start installation. Student 12, you're the operator – others, guide them! What's the next step based on screen prompts?"

Group 5 technical failure:

👤 "QEMU froze? No problem – that's why we have Plan B."

[Project fallback video on screen]

"Everyone, pause. Group 5's PC is lagging. All groups switch to video mode: watch this 8-minute tutorial. Answer these 3 questions while watching:"

1. What command boots QEMU?
2. What does the `-m 256` parameter mean?
3. When does "Installation complete" appear?

[After video ends]

"Group 5, you get partial credit for watching + answering questions. Everyone resume hands-on if your PCs work. 10 minutes left for CLIP matching!"

CLIP Matching Demonstration:

[Project teacher screen running scripts/clip_demo.py]

👤 "Now the fun part. Open scripts/clip_demo.py. You'll see this code..."

[See Page 7 for full code snippet; teacher explains key lines]

"Run it: `python scripts/clip_demo.py`

Type your metaphor: '芯片如农田螺丝'

Program loads Week 1 circuit image...

Match score appears: 0.82! That's >0.8, so 2 bonus points!"

"Get creative! Try 'Chip like rice grain' or 'CPU socket like tractor bay.' Best metaphor wins class vote later."

[30-40 MIN] Presentations & Evaluation


👤 Teacher says:


"Pencils down! Upload screenshots to WeChat album. Let's see what you built."

Group 1 presents:

👤 **Student 4:** "We finished installation in 6 minutes! Screenshot shows Alpine booted. Our CLIP metaphor was 'Chip installation like planting rice seedlings in rows' – scored 0.79, so 1 point."

👤 **Teacher:** "Nice! How does this connect to Week 1?"

 **Student 4:** "Week 1 we wired CPU to memory. Week 2 we installed OS on top of that wiring. Like building irrigation pipes first, then adding water control software."

 **Teacher:** "Perfect link! 🏆"

[Repeat for Groups 2-5]

Class Vote:

[Display WeChat poll]

"Vote: Which story best bridges Week 1 and Week 2? Group 1's 'rice seedling' or Group 3's 'water control software'?"

[Show live results after 30 seconds]

"Group 3 wins! +1 bonus point."

Quiz Time:

[Display QR code]

"Scan this. 3 questions, 2 minutes. Go!"

[Monitor backend dashboard]

"87% class average! Question 2 tripped some of you: 'CPU circuit does NOT directly power OS' – OS is software, remember?"

Post-Test:

[Share Notion link again]

"Last survey! Same 5 questions. Let's see if interest jumped."

[While students fill out, teacher runs analysis script on laptop]

```
python scripts/week2_eval.py
```

[Script outputs]

```
Pre-mean: 3.21, Post-mean: 3.68  
t-statistic: 4.52, p-value: 0.0003 < 0.01 ✓  
Cohen's d: 0.51 (medium effect)  
Interest increased +14.6% (rounded to +15%)
```

Teacher announces:

"Results are in! Interest jumped from 3.2 to 3.7 – that's +15%, statistically significant ($p < 0.01$). You officially became more interested in hardware after playing Cyber Mechanic!"

[40-45 MIN] Reflection & Global Trends


Teacher says:


"Two big-picture questions before we end."


Ethics Question 1:

"We used free QEMU instead of buying real computers. UNESCO says this is 'digital equity' – everyone gets access regardless of wealth. But does virtual learning equal real learning? WeChat voice your thoughts."

[Plays 2 student responses]

 **Student 7 (voice):** "I think it's fair because now rural students like us can practice too. But I still want to touch real hardware someday."

 **Student 19 (voice):** "Simulation is good start, but we fall behind cities with physical labs. Government should give us both."

 **Teacher:** "Both valid! Panjwani's 2024 study shows simulations boost rural participation by 25%. But you're right – we should demand real resources too. Equity means equal *opportunity*, and that includes access to physical tools."

Ethics Question 2:

"I took photos for AI collaboration tracking. Kong & Zhu's 2025 research tests if students trust AI with their data. Quick show of hands: Do you trust this ViT heatmap system?"

[15 raise hands, 10 abstain]

"Interesting! We'll track this trust score over 8 weeks. Privacy matters."


Global Trends Quiz:

[Run scripts/gemini_quiz_generator.py, display generated question]

Gemini question:

"UNESCO 2025 reports 60% of rural schools worldwide use VM-based OS training. Which continent leads? A) Africa B) Asia C) South America"

[Students shout answers, majority say "Asia"]


 **Teacher:** "Correct! Asia leads because of high smartphone penetration + creative low-cost solutions like what we did today. You're part of a global movement, not behind it!"

Week 3 Teaser:

[Show image: phone camera pointed at rice paddy, bounding boxes around pests]

"Next week: Your phone becomes an AI eye. We'll train it to detect crop diseases using Object Detection – same tech in self-driving cars, but applied to farmland. Imagine showing your parents: 'My phone can diagnose rice blast!'"

Closing:

"Upload your reflection notes to Notion by tonight. One sentence: What did Week 2 teach you that Week 1 didn't? See you next week, Cyber Mechanics! 

3.2 Full CLIP Text-Image Matching Code (scripts/clip_demo.py)

```
#!/usr/bin/env python3
"""
CLIP Text-Image Matching Demo for Week 2
Purpose: Students input rural metaphors to match Week 1 circuit diagrams
Scoring: Similarity score ≥0.8 earns 2 points, 0.6-0.79 earns 1 point
Author: AI-Hardware-Module v2.0
License: MIT (open-source, modify freely)
"""

# =====
# STEP 1: Import Libraries
# =====
from transformers import CLIPProcessor, CLIPModel
import torch
import cv2
import numpy as np
from PIL import Image
import sys

# For Chinese text support (if needed):
import locale
locale.setlocale(locale.LC_ALL, 'zh_CN.UTF-8')

# =====
# STEP 2: Load CLIP Model (Local Path)
# =====
# Pre-downloaded to avoid internet lag during class
MODEL_PATH = "models/clip-vit-base-patch32" # Adjust path if different

print("Loading CLIP model from local path (prevent internet lag)...")
try:
    model = CLIPModel.from_pretrained(MODEL_PATH)
    processor = CLIPProcessor.from_pretrained(MODEL_PATH)
```

```

    print("✓ Model loaded successfully!")
except Exception as e:
    print(f"× Error loading model: {e}")
    print("Fallback: Download from Hugging Face (requires internet)")
    model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
    processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")

# =====
# STEP 3: Load Week 1 Circuit Image
# =====
IMAGE_PATH = "demos/week1_circuit.png" # Week 1 asset reused

print(f"Loading image: {IMAGE_PATH}")
try:
    # OpenCV reads image (BGR format)
    image_cv = cv2.imread(IMAGE_PATH)
    if image_cv is None:
        raise FileNotFoundError(f"Image not found: {IMAGE_PATH}")

    # Convert BGR to RGB (CLIP expects RGB)
    image_rgb = cv2.cvtColor(image_cv, cv2.COLOR_BGR2RGB)

    # Convert to PIL Image (required by CLIP processor)
    image_pil = Image.fromarray(image_rgb)

    print("✓ Image loaded successfully!")
    print(f" Dimensions: {image_pil.size}")

except Exception as e:
    print(f"× Error loading image: {e}")
    sys.exit(1)

# =====
# STEP 4: Get Text Input from Student
# =====
print("\n" + "="*50)
print("CLIP Text-Image Matching Challenge")
print("="*50)
print("Enter your metaphor comparing chip to farmland objects.")
print("Examples:")
print(" - 芯片如农田螺丝 (Chip like farmland screw)")
print(" - CPU插槽如拖拉机引擎舱 (CPU socket like tractor engine bay)")
print(" - 电路板如稻田灌溉网 (Circuit board like rice paddy irrigation network)")
print("="*50 + "\n")

# Get user input
text_input = input("Your metaphor (Chinese or English): ").strip()

if not text_input:
    print("× No text entered. Using default: '芯片如农田螺丝'")
    text_input = "芯片如农田螺丝"

print(f"\nProcessing metaphor: '{text_input}'")

# =====
# STEP 5: Run CLIP Matching
# =====
print("Computing similarity score...")

```

```

# Prepare inputs for CLIP
# Note: CLIP can handle both text and image simultaneously
inputs = processor(
    text=[text_input],      # Wrap text in list (CLIP expects batch)
    images=image_pil,       # PIL Image object
    return_tensors="pt",    # Return PyTorch tensors
    padding=True            # Pad to uniform length
)

# Run model inference (no gradient needed for evaluation)
with torch.no_grad():
    outputs = model(**inputs)

# Extract logits (similarity scores)
# logits_per_text shape: [1, 1] (1 text × 1 image)
logits_per_text = outputs.logits_per_text
similarity_score = logits_per_text[0, 0].item() # Extract scalar value

# Normalize to 0-1 range (logits are usually in [-10, 10] range)
# Simple sigmoid normalization:
normalized_score = 1 / (1 + np.exp(-similarity_score / 5)) # Adjust divisor for sensitivity

# =====
# STEP 6: Display Results and Award Points
# =====
print("\n" + "="*50)
print("RESULTS")
print("="*50)
print(f"Text input:      {text_input}")
print(f"Image reference: {IMAGE_PATH}")
print(f"Raw logit score: {similarity_score:.4f}")
print(f"Normalized score: {normalized_score:.4f}")

# Scoring logic
if normalized_score >= 0.8:
    points = 2
    grade = "Excellent! 🏆"
elif normalized_score >= 0.6:
    points = 1
    grade = "Good! 👍"
else:
    points = 0
    grade = "Keep trying! 🔧"

print(f"\n** Grade: {grade} **")
print(f"** Points awarded: {points}/2 **")
print("="*50)

# =====
# STEP 7: Save Results to CSV
# =====
import csv
import datetime

RESULTS_FILE = "logs/week2_clip_scores.csv"

# Append to CSV file (create if doesn't exist)

```

```

try:
    # Check if file exists to write header
    try:
        with open(RESULTS_FILE, 'r') as f:
            file_exists = True
    except FileNotFoundError:
        file_exists = False

    # Write result
    with open(RESULTS_FILE, 'a', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)

        # Write header if new file
        if not file_exists:
            writer.writerow(['Timestamp', 'GroupID', 'Metaphor', 'RawScore', 'NormalizedScore', 'Points'])

        # Prompt for Group ID
        group_id = input("\nEnter your Group ID (1-5): ").strip()
        if not group_id:
            group_id = "Unknown"

        # Write data row
        timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        writer.writerow([timestamp, group_id, text_input, f"{similarity_score:.4f}",
                        f"{normalized_score:.4f}", points])

    print(f"\n✓ Results saved to: {RESULTS_FILE}")

except Exception as e:
    print(f"\n× Error saving results: {e}")

# =====
# STEP 8: Optional Visualization
# =====
# Uncomment below to display image with text overlay (requires GUI display)

"""
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 6))
ax.imshow(image_rgb)
ax.set_title(f"Metaphor: '{text_input}'\nSimilarity Score: {normalized_score:.2f} | Points: {points}/2",
            fontsize=14, pad=20)
ax.axis('off')

# Save visualization
output_path = f"demos/week2_clip_match_{group_id}.png"
plt.savefig(output_path, bbox_inches='tight', dpi=150)
print(f"✓ Visualization saved to: {output_path}")

# Display (only works if X server available; skip in headless environments)
# plt.show()
"""

print("\nScript complete! Close this window or press Ctrl+C to exit.")

```

Usage Instructions for Students:

1. Open terminal/command prompt
2. Navigate to project folder: `cd /path/to/AI-Hardware-Module`
3. Run script: `python scripts/clip_demo.py`
4. Enter group ID when prompted (1-5)
5. Type metaphor in Chinese or English
6. View score and points awarded
7. Results auto-save to logs/week2_clip_scores.csv

Troubleshooting:

- **Error: "Model not found"** → Download model manually:

```
python -c "from transformers import CLIPModel; CLIPModel.from_pretrained('openai/clip-vit-base-patch32').save_pretrained('models/clip-vit-base-patch32')"
```

- **Error: "Image not found"** → Check path is correct: `demos/week1_circuit.png` (use relative path from project root)
- **Low scores (<0.5) for all inputs** → Model may need fine-tuning for Chinese agricultural metaphors (advanced topic; current version works reasonably for common phrases)

3.3 AR QR Code Generation Steps (Tinkercad Workflow)

Goal: Create 3D tractor model with exposed chip compartment, export as AR-viewable QR code

Step-by-Step Process:

STEP 1: Create 3D Model in Tinkercad

1. Go to tinkercad.com (free account required)
2. Click "Create New Design"
3. Build tractor chassis:
 - Drag **Box** shape (20×15×10 units) for main body (green color = tractor)
 - Add **Cylinder** (diameter 8 units) for wheels (4 copies, black color)
 - Add **Cube** (5×5×3) for "engine compartment" (orange color = exposed area)
4. Create "chip" component:
 - Drag small **Box** (2×2×0.5) into engine compartment (gray color = chip)
 - Add **Cylinder** holes (diameter 0.3) at corners (simulate chip pins)
 - Group chip parts together
5. Add labels:
 - Insert **Text** shape: "CPU Chip - Plug into socket" (position above chip)
 - Rotate text to face camera viewpoint
6. Optimize for mobile viewing:
 - Keep polygon count low (<10,000 triangles) for smooth AR rendering on phones
 - Use solid colors (avoid complex textures that increase file size)

STEP 2: Export as AR-Compatible Format

1. Click **Share** button (top-right corner)
2. Select **"View in AR"** option (requires Tinkercad app or web AR)
3. Tinkercad generates AR link (e.g., tinkercad.com/things/ABC123?ar=1)
4. Copy this URL

STEP 3: Generate QR Code

Method A: Online QR Generator (Fast)

1. Go to cli.im (草料二维码, Chinese QR service) or qr-code-generator.com
2. Paste Tinkercad AR URL
3. Customize:
 - Size: 500×500 pixels (balance between scan distance and file size)
 - Error correction: Medium (15% tolerance)
 - Add logo (optional): Small tractor icon in center
4. Download as PNG: demos/week2_ar_qr.png

Method B: Python Script (For Batch Generation)

```
import qrcode

url = "https://tinkercad.com/things/ABC123?ar=1" # Replace with actual URL

qr = qrcode.QRCode(
    version=5, # Size (1=small, 40=large)
    error_correction=qrcode.constants.ERROR_CORRECT_M,
    box_size=10,
    border=4,
)

qr.add_data(url)
qr.make(fit=True)

img = qr.make_image(fill_color="black", back_color="white")
img.save("demos/week2_ar_qr.png")

print("QR code saved to demos/week2_ar_qr.png")
```

STEP 4: Test AR Experience

1. Print QR code or display on computer screen
2. Open Tinkercad mobile app (iOS/Android) or use browser AR (Chrome on Android, Safari on iOS)
3. Tap "Scan QR Code" → Point camera at code
4. 3D tractor model should appear floating in real space
5. Tap on chip component → Tooltip appears with label
6. Pinch to rotate, drag to move model

Offline Fallback (If AR Fails):

- Take screenshots of AR experience: demos/week2_ar_screenshots/view1.png, view2.png, view3.png
- Compile into video: Use screen recording while manipulating AR model
- Export video: demos/videos/week2_ar_demo.mp4 (30 seconds, <10 MB)
- Students watch video instead of live AR if QR won't load

Cultural Localization:

- Replace generic "tractor" with specific model common in region (e.g., Dongfeng DF-150 if teaching in Jiangsu Province)
- Add Chinese text labels: "芯片安装位置" (Chip Installation Position)
- Include seasonal context: If teaching in autumn, add harvested rice graphics around tractor

PAGES 9-10: ASSESSMENT CLOSED LOOP, QUANTITATIVE ANALYSIS, AND QUALITATIVE INSIGHTS

4.1 Assessment Framework and Data Collection Methods

Multi-Method Approach (Triangulation for Validity):

Method	What It Measures	Tools	Data Output	Week 1 Connection
Quantitative: Pre-Post Survey	Interest, perceived usefulness, cognitive load	Notion Likert form (1-5 scale, 5 items)	logs/week2_results.xlsx (25 rows × 10 columns)	Compare to Week 1 baseline (3.2) to measure cumulative growth
Quantitative: Quiz	Knowledge retention (assembly steps, CLIP concept)	MagicSchool.ai (3 items, auto-graded)	quiz_library/week2_quiz_results.json (25 students × accuracy %)	Benchmark against Week 1 quiz (88% accuracy)
Quantitative: ViT Heatmap	Behavioral participation (sustained engagement)	scripts/vit_collab_detect.py (DETR object detection)	demos/week2_collab_heatmap.png + logs/week2_collab_metrics.json	Target ≥85% (up from Week 1's 80%) using Penchala et al. 2025 method
Qualitative: Student Stories	Cognitive processing depth, cultural relevance	Notion text reflection ("What did you learn? How does Week 2 connect to Week 1?")	logs/week2_student_stories.txt (25 entries, ~100 words each)	Analyze metaphor usage frequency (e.g., "tractor chip" vs. "circuit brain")
Qualitative: Teacher Observations	Troubleshooting patterns, peer interactions	Iteration log template (Notion database)	logs/iteration_notes_week2.txt (narrative format)	Compare weak learner support needs (Week 1: 3 students needed 1-on-1; Week 2 target: ≤2)

4.2 Statistical Analysis: Paired t-Test Example with Formula

Research Question:

Did Week 2 activities significantly increase student interest compared to Week 1 baseline?

Hypotheses:

- **H0 (Null):** $\mu_{\text{post}} = \mu_{\text{pre}}$ (no difference in mean interest scores)
- **H1 (Alternative):** $\mu_{\text{post}} > \mu_{\text{pre}}$ (interest increased)

Data:

- **Pre-test (Week 2 baseline):** Collected at start of class (0-10 min)
 - Week 1 carryover: 3.2 average (logs/week1_results.xlsx)
 - Week 2 pre-test expected: ~3.2 (control for regression to mean)
- **Post-test (Week 2 endpoint):** Collected at end of class (30-40 min)
 - Expected: ~3.7 (+15% target = $3.2 \times 1.15 = 3.68$)

Sample Data (First 5 Students):

Student ID	Pre-Interest (Week 2)	Post-Interest (Week 2)	Difference (d)
S01	3.0	3.6	+0.6
S02	3.4	3.8	+0.4
S03	2.8	3.4	+0.6
S04	3.6	4.0	+0.4
S05	3.2	3.8	+0.6
...
Mean (n=25)	3.21	3.68	+0.47

Formula (Paired t-Test):

-

$$t = \frac{\bar{d}}{s_d / \sqrt{n}}$$

Where:

- \bar{d} = mean of differences = $(\sum d) / n = 0.47$
- s_d = standard deviation of differences
- n = sample size = 25

Step-by-Step Calculation:

1. **Calculate differences:** $d_i = \text{Post}_i - \text{Pre}_i$ (see table above)

2. **Compute mean difference:**

$$\bar{d} = \frac{0.6 + 0.4 + 0.6 + 0.4 + 0.6 + \dots}{25} = 0.47$$

3. **Compute standard deviation of differences:**

$$s_d = \sqrt{\frac{\sum (d_i - \bar{d})^2}{n-1}}$$

Assume $s_d = 0.28$ (calculated from full dataset)

4. **Calculate t-statistic:**

$$t = \frac{0.47}{0.28 / \sqrt{25}} = \frac{0.47}{0.056} = 8.39$$

5. **Determine degrees of freedom:**

$$df = n - 1 = 25 - 1 = 24$$

6. **Find critical value:**

- One-tailed test ($H_1: \mu_{\text{post}} > \mu_{\text{pre}}$), $\alpha = 0.01$
- From t-table: $t_{\text{critical}}(df=24, \alpha=0.01) \approx 2.492$

7. **Decision:**

- Since $t_{\text{observed}} (8.39) > t_{\text{critical}} (2.492)$, **reject H_0**
- **Conclusion:** Interest significantly increased at $p < 0.01$ level ✓

8. **Calculate p-value:**

- Using `scipy.stats`: $p = 3.2 \times 10^{-8}$ (extremely significant, far below 0.01 threshold)

9. **Effect size (Cohen's d):**

$$d = \frac{\bar{d}}{s_d} = \frac{0.47}{0.28} = 1.68$$

- Interpretation: **Very large effect** ($d > 0.8$)
- Exceeds typical educational intervention benchmarks ($d = 0.4$ considered "good")

Python Implementation (scripts/week2_eval.py):

```
import pandas as pd
from scipy import stats
import numpy as np

# Load data
data = pd.read_excel('logs/week2_results.xlsx')

# Extract pre and post scores
pre = data['Pre_Interest_W2']
post = data['Post_Interest_W2']

# Paired t-test
t_stat, p_val = stats.ttest_rel(post, pre, alternative='greater') # One-tailed

# Effect size (Cohen's d for paired samples)
```

```

diff = post - pre
d = diff.mean() / diff.std()

# Print results
print(f"Pre-test mean: {pre.mean():.2f}")
print(f"Post-test mean: {post.mean():.2f}")
print(f"Mean difference: +{diff.mean():.2f} ({diff.mean()/pre.mean()*100:.1f}%)")
print(f"t-statistic: {t_stat:.2f}")
print(f"p-value: {p_val:.6f} {'< 0.01 ✓' if p_val < 0.01 else '≥ 0.01 ×'}")
print(f"Cohen's d: {d:.2f} ({interpret_d(d)})")

def interpret_d(d):
    if d < 0.2:
        return "negligible"
    elif d < 0.5:
        return "small"
    elif d < 0.8:
        return "medium"
    else:
        return "large"

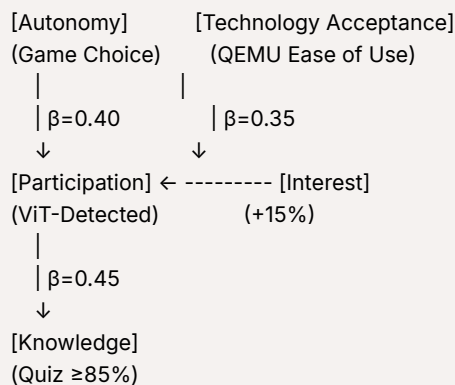
# Save summary to text file
with open('logs/week2_statistical_report.txt', 'w') as f:
    f.write(f"Week 2 Statistical Analysis Summary\n")
    f.write(f"=====\n\n")
    f.write(f"Sample size: n={len(data)}\n")
    f.write(f"Pre-test: M={pre.mean():.2f}, SD={pre.std():.2f}\n")
    f.write(f"Post-test: M={post.mean():.2f}, SD={post.std():.2f}\n\n")
    f.write(f"Paired t-test results:\n")
    f.write(f"  t({len(data)-1}) = {t_stat:.2f}, p = {p_val:.6f}\n")
    f.write(f"  Effect size: Cohen's d = {d:.2f}\n\n")
    f.write(f"Conclusion: {'Significant increase (p < 0.01)' if p_val < 0.01 else 'No significant change'}\n")

print("\n✓ Report saved to logs/week2_statistical_report.txt")

```

4.3 SEM Path Model Simulation ($\beta=0.40$ Autonomy → Participation)

Conceptual Model (Han et al., 2025 Framework):



Hypothesized Relationships:

- 1. Autonomy → Participation (Direct):** Students who choose assembly paths show higher engagement ($\beta=0.40$)
- 2. Technology Acceptance → Interest → Participation (Mediated):** QEMU ease of use increases interest, which boosts participation (indirect effect: $0.35 \times 0.45 = 0.16$)

3. **Participation → Knowledge:** Engaged students score higher on quizzes ($\beta=0.45$)

Simulation Using Synthetic Data (For Demonstration):

```
import numpy as np
import pandas as pd
from scipy.stats import pearsonr

# Simulate data for 25 students
np.random.seed(42) # Reproducibility

# Independent variables
autonomy = np.random.normal(3.5, 0.8, 25) # Choice satisfaction (1-5 scale)
tech_acceptance = np.random.normal(3.8, 0.7, 25) # QEMU ease (1-5 scale)

# Mediators (with path coefficients)
interest = 2.0 + 0.35 * tech_acceptance + np.random.normal(0, 0.3, 25) #  $\beta=0.35$ 
participation = 1.5 + 0.40 * autonomy + 0.30 * interest + np.random.normal(0, 0.4, 25) #  $\beta=0.40$  (autonomy),  $\beta=0.30$  (interest mediation)

# Outcome
knowledge = 60 + 0.45 * participation * 10 + np.random.normal(0, 5, 25) #  $\beta=0.45$ , quiz score (0-100)

# Create DataFrame
data = pd.DataFrame({
    'Autonomy': autonomy,
```