EdSutra

Teaching guide

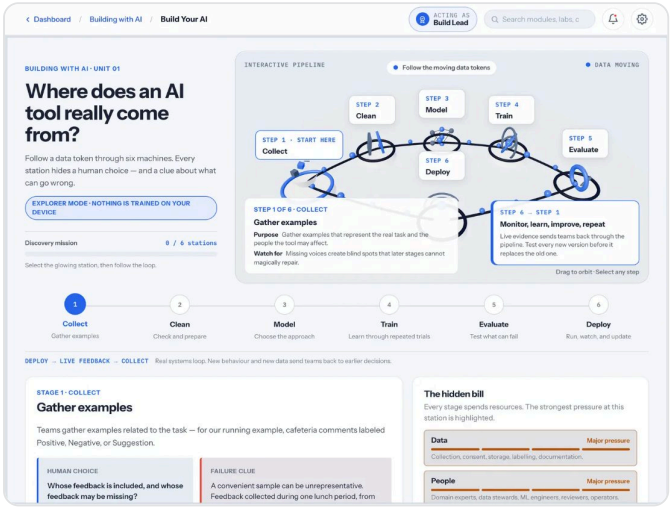
Module 06 · Building with AI · Unit 01 · Stage 3, Responsibility

Build Your AI

The Pipeline Lab

Grades 9–12 | 55 minutes | Role: **Build Lead**

Grade 9: About age 14–15 · Year 10 in the UK · Class 9 in India



The big idea

An AI tool is a chain of ordinary human decisions: collect, clean, model, train, evaluate, deploy. Each one changes quality, cost and who might be harmed. In the Pipeline Lab, training longer can make results on new comments worse, and a good overall score can hide missed Suggestions. Deciding to deploy depends on what a mistake would cost.

The learner's job

Learners are the **Build Lead** at Kestrel Works · Shop Floor. A customer is waiting. They do not care which part is difficult. Their task: take it from idea to working. Say out loud when it is not.

By the end, learners can say

- I can sequence the six pipeline stages and name a failure mode for each
- I can change a deterministic pipeline simulation and trace cause to effect
- I can diagnose why a simulated run underperforms and which stage caused it
- I can judge whether evidence is enough to deploy, revise, or run a limited trial

Before class

- One computer or tablet for each learner or pair, with EdSutra open (this Module is part of the full EdSutra studio).
- Open the studio once on each device before class, so the Unit still runs if the internet drops.
- Play the first screen and the start of the Pipeline Lab yourself, so you know where learners will click.
- Write the six station names on the board in a loop, with an arrow from Deploy back to Collect.
- Pairs work well: one learner makes the choices, the Build Lead says which station to change next, and why.
- Best after How Machines Learn and Neural Networks.
- Units open in order, and Module 1 comes first. To start a class here, switch on Open every Unit in Settings → Teachers & parents.

Words to introduce

Pipeline: The chain of steps that turns data into a working AI tool.

Epoch: One full pass of the training examples through the model.

Overfitting: Getting better on seen examples while getting worse on new ones.

Best checkpoint: The saved version that did best on new data; a candidate, not a pass.

Recall: Of all the real examples of one kind, how many the model found.

Confusion matrix: A table counting each right answer and each mix-up between kinds.

Drift: Real-world data changing after deployment, so the model fits it less well.

Running the lesson 55 minutes

Open 5 min

"When an AI tool goes wrong, people say "the computer messed up". Where else could the mistake have come from?"

Before anyone clicks, ask: if you keep training a model for longer and longer, will it keep getting better at sorting new comments? Take a hands-up vote and record it. Come back to it in Apply.

Step	What learners do	Ask the class	Listen for
1. Remember 5 min	Enter the pipeline and open the six stations in order, from Step 1 to Step 6: Collect, Clean, Model, Train, Evaluate, Deploy. Watch the data token travel around the loop.	• Which station comes right after Clean? • Where does the loop go after Deploy, and why?	Six stations in order, and Deploy sends live evidence back to Collect: the system is never really finished.
2. Understand 10 min	At each station, read the human choice and the failure clue, check the hidden bill of costs, and open the Human review queue to see which station each warning sends the team back to.	• Which human choice surprised you most? • Why might cleaning out every short comment hurt the sorter?	Every station is a human decision. Removing short comments like "More vegan?" throws away real Suggestions.
3. Apply 13 min	In the Pipeline Lab, answer the prediction, then change one choice at a time (sample, cleaning, model, epochs, pace) and press Recalculate. Drag along the training curve to find the best checkpoint.	• What happened to new-data results when you trained for many more epochs? • Which single change helped most? • Look at our vote. Were we right?	Training results kept improving while new-data results got worse. That widening gap is overfitting.
4. Analyze 12 min	Press Show the counts to open the confusion matrix, read each run's diagnosis in the Iteration history, then press Return to go back to the station that caused the problem.	• Overall accuracy looks fine. How well does it find Suggestions? • Which station did the history send you back to, and why?	Overall accuracy can hide a rare kind of comment. Weak Suggestion recall traces back to Collect or Clean.
5. Evaluate 10 min	At Deploy, decide whether staff should use the sorter with a person checking. Then judge a tool that flags cheating from quiz timing at 95% accuracy: pick a verdict and the reason that backs it.	• Which call did you make for the cafeteria sorter, and what evidence backed it? • If cheating is rare, why could many flags still be wrong?	Good enough depends on the cost of a mistake, not one accuracy number. Any choice needs a reason that fits.

Steps follow Bloom's revised taxonomy, in the order the Unit presents them.

Deepen the thinking

Critical thinking	A team trains longer and the training score rises. What evidence would you ask for before believing the model got better?
AI thinking	Deployment is not the finish line. What new kinds of comments might reach the sorter next term, and which station would they send the team back to?
Ethics and fairness	Aggressive cleaning removed short comments like "More vegan?". Whose voices get lost when data is cleaned or collected in one lunch period, and who should decide?
Connect to life	Think of an app you use that changed after an update. What might its team have seen that sent them back to an earlier stage?

Misconceptions to listen for

"AI fails because the computer went rogue."

Every station hides a human choice. The sorter's problems trace to ordinary decisions, like a narrow sample or aggressive cleaning.

"Training longer always makes a model better."

In the Pipeline Lab, training results keep improving while new-data results get worse. That widening gap is overfitting.

"High accuracy means the model is good."

Show the counts: overall accuracy can look fine while the sorter misses many Suggestions. Check every kind, and the cost of each mistake.

In real life the card learners see at the end of the Unit

- When a school or shop AI tool fails, the cause is usually an ordinary choice, like its data or testing.
- Some app updates come from monitoring: real-world feedback sends a model back to be retrained and retested.
- A tool sorting feedback can look accurate overall yet miss most of a rare kind, like suggestions.

Try it this week. Memorize the answers to five practice questions, then try five new ones on the same topic. The gap is like overfitting.

Watch out for. Training longer doesn't always help: a model can improve on practice data while getting worse on new data. That's overfitting.

Talk about it. What should a school test before using AI to spot cheating? Who could a mistake hurt?

Exit ticket

1. Name the six pipeline stations in order.

2. What is overfitting, and how does the training curve show it?

3. Why might a sorter with high overall accuracy still be failing someone?

Going further

The ML Pipeline Builder Lab trains real models on penguin or message data. A model that always guesses Normal scores high yet catches no spam, like the missed Suggestions here. For homework, use the In real life card's "Try it this week".

EdSutra

Teaching guide

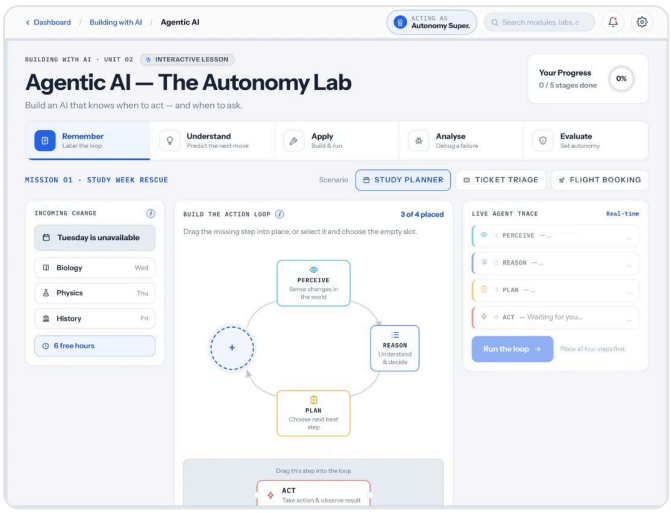
Module 06 · Building with AI · Unit 02 · Stage 3, Responsibility

Agentic AI

The Autonomy Lab

Grades 9–12 | 40 minutes | Role: **Autonomy Supervisor**

Grade 9: About age 14–15 · Year 10 in the UK · Class 9 in India



The big idea

An agent does not just answer: it runs a loop (perceive, reason, plan, act) until a rule says stop. Humans decide which tools it gets, when it must stop and when it must ask first. The right amount of autonomy depends on the stakes and how easily a mistake can be undone.

The learner's job

Learners are the **Autonomy Supervisor** at Kestrel Works · Autonomy Bay. This machine does not wait for permission between steps. Someone decides how far it gets. Their task: decide how much this system may do alone. Then watch it do it.

By the end, learners can say

- I can name the four parts of the agent action loop — Perceive, Reason, Plan, Act
- I can predict how a reactive agent and a planning agent respond when a plan fails
- I can construct a bounded agent with a scoped tool and an explicit stopping condition
- I can trace an agent failure to the part of the loop that broke, citing transcript evidence
- I can choose an autonomy level and guardrail for a deployment and defend the trade-off

Before class

- One computer or tablet for each learner or pair, with EdSutra open (this Module is part of the full EdSutra studio).
- Open the studio once on each device before class, so the Unit still runs if the internet drops.
- Play the first two steps yourself, so you know where learners will drag and what the two agents do.
- Draw the loop on the board: Perceive, Reason, Plan, Act, with an arrow back to Perceive. Leave space to add Stop in Apply.
- Pairs work well: one learner drives, the Autonomy Supervisor decides what the agent may do alone, and says why.
- Best after Build Your AI.
- Units open in order, and Module 1 comes first. To start a class here, switch on Open every Unit in Settings → Teachers & parents.

Words to introduce

Agent: An AI that keeps choosing its next step until a job is done.

Action loop: Perceive, Reason, Plan, Act, then check the result and go again.

Tool: Something the agent is allowed to use, such as a calendar.

Stopping condition: A clear rule that tells the agent when the job is done.

Least privilege: Give the agent only the tools this one job needs.

Guardrail: A rule people set so an agent's mistakes stay small.

Autonomy level: How much the agent does before a person checks: act, report, or ask first.

Running the lesson 40 minutes

Open 4 min

"This machine does not wait for permission between steps. Who decides how far it gets, and how would they decide?"

Before anyone clicks, ask: an AI planner moves your study blocks after Tuesday becomes busy. Everything now fits. How does it know it is finished? Write the guesses down and come back to them in Apply.

Step	What learners do	Ask the class	Listen for
1. Remember 5 min	Pick a scenario (study planner, ticket triage or flight booking). Drag the missing Act step into the loop, then press Run the loop and watch the live agent trace fill in.	• What did the agent perceive first? • Where does the result go after Act?	Each part has a job. The result feeds back to Perceive, and the loop repeats until the job is done.
2. Understand 7 min	For each case, predict how a reactive agent and a planning agent handle a broken step, then watch both lanes play out. Learners finish at least two of the three cases.	• What did the reactive agent do when Wednesday was full? • In the flight case, when did each agent check the budget?	A reactive agent has one move for one trigger. A planning agent holds a goal and can reroute.
3. Apply 8 min	Run the starter agent first: it plans the week, then has no rule to stop, and pauses at the app's safety cap. Attach the stopping condition and run it again.	• Why is the Refund tool locked for a study planner? • Look at our guesses from the start. How does the agent know it is finished?	Without a stop rule, a loop is stuck, not autonomous. The agent gets only the tools this job needs.
4. Analyze 8 min	Read three failure transcripts row by row. For each, pick the broken part of the loop, the evidence that proves it, and the guardrail that would have prevented it.	• In the order case, where did "arrives Friday" come from? • Which row shows the agent going beyond its task?	Infinite loop breaks at Stop, the invented order status at Perceive, the photo clean-up at Act. Each fix is a guardrail.
5. Evaluate 8 min	For at least two of three agents (refunds, coding, study planner), choose an autonomy level, pick and rank a guardrail, set the threshold, tag the evidence and pick a reason that backs the choice.	• Why might the study planner act alone while the refund agent asks first? • What does a guardrail cost, and what does it save?	Autonomy follows the stakes: real money and hard-to-undo actions need a person sooner. More than one answer can be defended.

Steps follow Bloom's revised taxonomy, in the order the Unit presents them.

Deepen the thinking

Critical thinking	The starter agent looked fine for several steps and still had no way to end. How would you test an agent before trusting it with a real job?
AI thinking	A reactive agent gets stuck when one step breaks; a planning agent can reroute. When might a simple reactive agent still be the better design?
Ethics and fairness	When an agent's mistake hurts someone, who is responsible: the agent, the person who set its autonomy, or the company that gave it the tool?
Connect to life	A robot vacuum senses, plans, acts and checks, again and again. What other machine or routine in your life runs a loop with a stop rule?

Misconceptions to listen for

"More autonomy is always better."
In Evaluate, the possible damage spreads further as autonomy rises. A study planner can act alone; a refund agent may need a person to confirm.

"A demo that works means the agent is ready."
In Apply, the starter agent plans the week perfectly, then has no way to end. Test what happens when things go wrong, and how it stops.

"Guardrails just slow the agent down."
In Analyze, every failure's fix is a guardrail, and in Apply the run cannot finish without a stop rule. Guardrails make autonomy safe to use.

In real life the card learners see at the end of the Unit

- Some AI assistants can now book, buy or send things, looping through steps until a stopping rule says done.
- A robot vacuum senses a room, plans a path, acts, checks the result and repeats: an agent loop.
- An email helper that suggests a reply and one that sends it for you have different autonomy.

Try it this week. Write a step-by-step loop for a chore, like packing a school bag. Add a stop rule and one step where a person should check.

Watch out for. More autonomy isn't always better. The higher the stakes, and the harder a mistake is to undo, the sooner an agent should ask a person.

Talk about it. What's one job you'd let an AI agent do alone, and one it should always ask about?

Exit ticket

1. Name the four parts of the agent loop, in order.

2. Why does an agent need a stopping condition?

3. What does least privilege mean? Give an example from the Unit.

Going further

The Open Project Canvas Lab lets learners build an agent loop from simple rules and watch it run. For homework, use the In real life card's "Try it this week": a chore loop with a stop rule and a human check.

EdSutra

Teaching guide

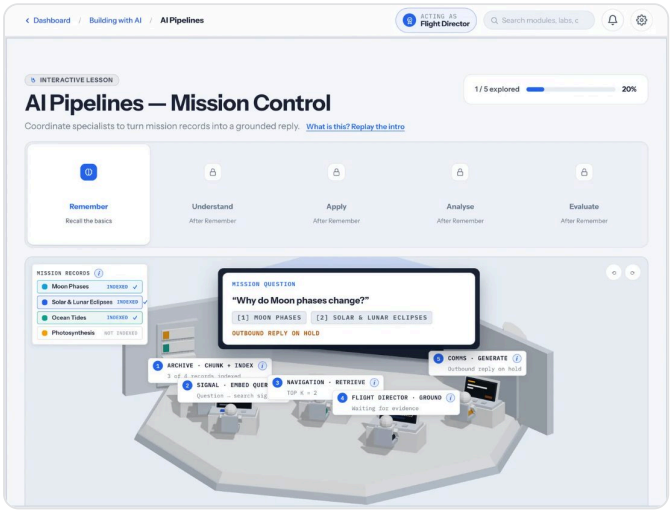
Module 06 · Building with AI · Unit 03 · Stage 3, Responsibility

AI Pipelines

Mission Control

Grades 9–12 | 50 minutes | Role: Flight Director

Grade 9: About age 14–15 · Year 10 in the UK · Class 9 in India



The big idea

An AI pipeline is a relay of specialists: index the records, turn the question into search terms, retrieve, ground, generate. Retrieval can bring in facts the model lacks, but it cannot force later stages to use them. When a reply goes wrong, trace it back to the first broken handoff and fix that stage.

The learner's job

Learners are the **Flight Director** at Kestrel Works · Mission Control. Every stage says it is fine. The thing coming out the end is not. Their task: find the stage where the reply first went wrong. Fix that one.

By the end, learners can say

- I can name what each pipeline specialist contributes: index, embed, retrieve, ground, generate
- I can predict which failures retrieval can fix — and which it cannot guarantee
- I can operate a RAG system: tune TOP K, inspect scores, and build a cited evidence brief
- I can trace a bad reply to the first broken stage, citing the artifact that proves it
- I can choose and defend a sequential, parallel, or conditional architecture under constraints

Before class

- One computer or tablet for each learner or pair, with EdSutra open (this Module is part of the full EdSutra studio).
- Open the studio once on each device before class, so the Unit still runs if the internet drops.
- Play the first two missions yourself, so you know which consoles learners click and what each prediction case shows.
- Write the five specialists on the board in order: Archive, Signal, Navigation, Flight Director, Comms. Learners add each one's job in Remember.
- Pairs work well: one learner operates, the Flight Director names the stage they suspect before anyone checks.
- Best after Build Your AI and Agentic AI.
- Units open in order, and Module 1 comes first. To start a class here, switch on Open every Unit in Settings → Teachers & parents.

Words to introduce

Pipeline: A chain of specialists, each doing one job and handing work on.

Retrieval: Searching stored records for the ones that best match a question.

TOP K: How many of the best-scoring records the search returns (1 to 3 here).

Grounding: Attaching the retrieved records to the brief, so every claim cites a source.

RAG: Retrieval-augmented generation: search first, then write the reply from what was found.

Fallback: A safe reply, like "I don't have enough evidence", when nothing relevant is found.

Architecture: How stages are arranged: one after another, side by side, or only when needed.

Running the lesson 50 minutes

Open 5 min

"Every stage says it is fine. The reply coming out the end is wrong. How would you find where it went wrong?"

Before anyone clicks, ask: if a chatbot can search the right documents before answering, will its answers always be right? Take a hands-up vote and record it. Come back to it in Understand, after the ignored bulletin.

Step	What learners do	Ask the class	Listen for
1. Remember 6 min	Click each console in Mission Control to meet the five specialists: Archive, Signal, Navigation, Flight Director and Comms. Then answer: what must the Flight Director give Comms?	• Which specialist finds the records, and which one writes the reply? • Why does Comms need the records, not just the question?	Comms writes from the question plus the retrieved records. That handoff is grounding, so every claim can point to a source.
2. Understand 8 min	Two cases. For each, predict whether retrieval will rescue the reply, then watch the replay: an outdated rock-sample answer, and a safety bulletin that is retrieved but never used.	• Why did retrieval fix the rock-sample answer? • The bulletin was found. Why was the reply still wrong? • Look at our vote. Were we right?	Retrieval helps when the fact is in the records. It cannot make later stages use what it found.
3. Apply 11 min	Pick a brief, choose which records are indexed, set TOP K, and run the search. Read every keyword score, attach only useful records, and generate a grounded reply.	• What changed when you set TOP K to 1, then 3? • What happened when you attached only the Eclipses record?	Scores are counted keyword matches. More records can help or distract, and with no relevant evidence the reply says so.
4. Analyze 10 min	Inspect three bad replies. For each, follow the record from Archive to Comms and pick the first broken stage, the evidence that proves it, a repair and the reason.	• Was the Auroras record ever in the Archive? • In the crater case, which stages worked, and which one broke?	The first break, not the last: wrong record at Navigation, dropped bulletin at the Flight Director, reversed claim at Comms.
5. Evaluate 10 min	For all three scenarios (routine check-ins, research reports, emergency support), pick sequential, parallel or conditional, simulate the messages, tag at least two trade-offs, keep the no-evidence fallback, and choose a matching defense.	• Why does running two models on routine check-ins waste compute? • In emergency support, which messages go to the expert, and why?	The best design spends extra compute only where it buys reliability. More branches are not automatically better.

Steps follow Bloom's revised taxonomy, in the order the Unit presents them.

Deepen the thinking

Critical thinking	In Analyze, the stage nearest the bad reply was not always the broken one. Where else might blaming the last step hide the real cause?
AI thinking	This search counts matching keywords. The question about glowing skies near the poles never says "aurora". How could a search find records by meaning instead?
Ethics and fairness	A school help bot answers from an outdated handbook and a learner misses a deadline. Who is responsible: the bot's builders, the handbook's owners, or the learner?
Connect to life	Think of a group project where each person did one part. How did a mistake early on travel through to the final result?

Misconceptions to listen for

"Retrieval stops AI from making things up."
In Understand, the bulletin is retrieved but never reaches the brief, and in Analyze, Comms reverses a claim it was given.

"More evidence is always better."
At TOP K 3, the search also returns Ocean Tides for a Moon phases question. Extra records can distract, so attach only what answers it.

"The stage nearest the bad reply is the broken one."
In the auroras case, Comms wrote the wrong reply, but the break came earlier: Navigation returned the wrong record. Fix the first break.

In real life the card learners see at the end of the Unit

- Some chatbots search documents first and show the sources they used: retrieval plus generation.
- A school or company help bot answering from its handbook gives outdated answers if the handbook is outdated.
- At a library help desk, finding the right book comes before the answer; if that step fails, everything after does.

Try it this week. Answer a question using only two textbook pages, noting which page backs each sentence. Did any sentence come from neither?

Watch out for. Showing sources doesn't guarantee a right answer. Retrieval can fetch the wrong record, and the earliest broken step is the one to fix.

Talk about it. Would you trust an answer more if it showed its sources? What would you still check?

Exit ticket

1. What does the Flight Director add to the brief before Comms writes?

2. Retrieval found the right record. Name one way the reply can still be wrong.

3. When is a conditional pipeline a better choice than a parallel one?

Going further

The ML Pipeline Builder Lab lets learners fill four stages, run them for a real score, and swap one block at a time to see which stage matters. For homework, use the In real life card's "Try it this week".

EdSutra

Teaching guide

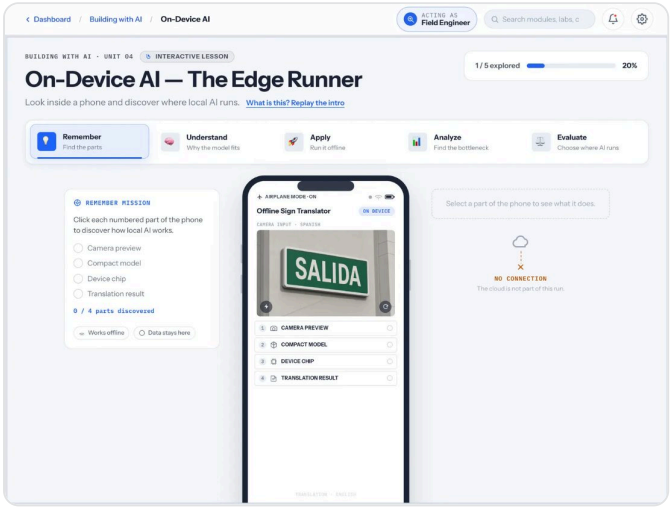
Module 06 · Building with AI · Unit 04 · Stage 3, Responsibility

On-Device AI

The Edge Runner

Grades 9–12 | 45 minutes | Role: **Field Engineer**

Grade 9: About age 14–15 · Year 10 in the UK · Class 9 in India



The big idea

Some AI runs on the device itself: the model is downloaded once, stored locally and run by the device's own chip, so nothing needs to be sent away. To fit limited memory, models are made smaller, which can cost quality. Where AI should run depends on the task, connection, privacy, memory and safety.

The learner's job

Learners are the **Field Engineer** at Kestrel Works · Field Unit. This has to run with the network off and the battery draining. Their task: make it work on the device. Say what you gave up to fit.

By the end, learners can say

- I can name the four parts of a local AI pipeline: camera, compact model, device chip, result
- I can explain why lower-precision models use less memory — and that this can affect quality
- I can run a simulated offline translation and identify what made it possible without a connection
- I can diagnose a failed local run as a memory, compute, or task-fit problem, and choose a valid repair
- I can choose and defend where an AI task should run — on-device, cloud, or hybrid — using named constraints

Before class

- One computer or tablet for each learner or pair, with EdSutra open (this Module is part of the full EdSutra studio).
- Open the studio once on each device before class, so the Unit still runs if the internet drops.
- Play the first two steps yourself, so you know where learners will click and how the precision control works.
- Draw a phone on the board with four boxes: camera, compact model, chip, result. Learners fill in each job during Remember.
- Pairs work well: one learner drives, the Field Engineer says what the smaller build gave up to fit.
- Best after Build Your AI and AI Pipelines.
- Units open in order, and Module 1 comes first. To start a class here, switch on Open every Unit in Settings → Teachers & parents.

Words to introduce

On-device (edge) AI: AI that runs on your phone or laptop, not on a distant server.

Inference: Using a trained model to produce an answer, like a translation.

AI memory budget: How much of the device's memory is free for running AI.

Precision (bits): How many bits store each number in the model; fewer bits, smaller file.

Quantization: Storing a model's numbers with less precision so it uses less memory.

Distillation: A small student model learns to imitate a larger teacher model.

Hybrid: Part of a task runs on the device and part in the cloud.

Running the lesson 45 minutes

Open 5 min

"No signal, no help, just the device. Can a phone still translate a sign? Where would the AI have to live?"

Before anyone clicks, ask: if we store a model's numbers with less detail, will the file get smaller? Will its answers change? Record a hands-up vote for each and come back to it in Understand.

Step	What learners do	Ask the class	Listen for
1. Remember 5 min	Click each numbered part of the phone (camera preview, compact model, device chip, translation result) to see how the sign SALIDA becomes EXIT. Then answer where the image was processed.	• Which part does the actual AI calculation? • How much of the image was sent to a server?	The model is already stored on the phone and its chip runs it. The run sends 0 KB of the image away.
2. Understand 9 min	Answer two predictions, then move the precision control from 32-bit down to 8-bit and 4-bit, and watch the model's size change against the phone's 2 GB AI memory budget.	• At which precision did the model first fit the budget? • Why might fewer bits change the answers? • Look at our vote. Were we right?	Fewer bits per number means a smaller file that can fit, but quality can change, depending on the model and task.
3. Apply 8 min	Choose a phone or tablet and a model (compact 0.8 GB or general 4.2 GB), keep airplane mode on, and run. Then pick the three things that made the offline run possible.	• What happened when you ran the general model? • Did airplane mode make the translation work? What did?	The 4.2 GB model cannot load. The compact one works: stored locally, fits the budget, run by the chip.
4. Analyze 9 min	Read the evidence table for each failed run (model doesn't fit, too slow, wrong task). Pick what stopped it and a repair that really fixes it. Learners need two of the three right.	• The medical answer came back fast. Why is it still a failure? • Why won't turning Wi-Fi on fix the memory case?	Memory, compute and task fit are different problems with different fixes. A fast, confident answer is not a safe one.
5. Evaluate 9 min	For at least two scenarios (hiking, broad research, private journal, medical symptoms), choose on-device, cloud or hybrid, watch the consequences change, tag two constraints and pick a defense that matches.	• Why is the cloud ruled out on the hiking trail? • In the medical case, does it matter where the AI runs?	Choose the least resource-heavy place that can do the task responsibly. For medical symptoms, a person is needed wherever it runs.

Steps follow Bloom's revised taxonomy, in the order the Unit presents them.

Deepen the thinking

Critical thinking	An app says its AI is "private because it runs on your phone". What would you check before believing that?
AI thinking	A compact model fits by using fewer bits. Which tasks would you trust a smaller model with, and which would you test carefully first?
Ethics and fairness	Many people own older or cheaper phones with less memory. Who might be left out if useful AI only runs on new devices, or only in the cloud?
Connect to life	When have you needed something to work with no signal, like on a trip? What would you want your phone to do offline?

Misconceptions to listen for

"On-device AI is just a smaller cloud model."
In Remember and Apply, the model file is already on the phone and its chip does the work. 0 KB of the image is sent anywhere.

"A smaller model is just as good."
In Understand, fewer bits make the file smaller, but output quality can change. Narrow tasks may stay strong; hard or sensitive ones need testing.

"Airplane mode is what makes offline AI work."
Airplane mode only proves the cloud is unreachable. The run works because the model is stored on the phone, fits its memory and runs on its chip.

In real life the card learners see at the end of the Unit

- Some phones can do offline translation, voice typing or photo search on the device, with no data sent away.
- Cars with lane warnings run their AI on board, because they can't wait for an internet signal.
- Downloading a language pack for offline translation is "download once, run offline" in action.

Try it this week. With a parent or teacher, find one phone feature that still works in airplane mode. What does that tell you about where it runs?

Watch out for. Smaller models aren't free: shrinking one can lower its quality on hard or sensitive tasks. Some jobs still need the cloud.

Talk about it. What would you rather keep on your own device instead of sending online, and why?

Exit ticket

1. Name the four parts of the local translation, in order.

2. Why does storing numbers with fewer bits help a model fit on a phone?

3. Why did the 4.2 GB model fail on the phone, and what fixed it?

Going further

For homework, use the "Try it this week" activity from the In real life card: with a parent or teacher, find one phone feature that still works in airplane mode, and say what that shows about where it runs.