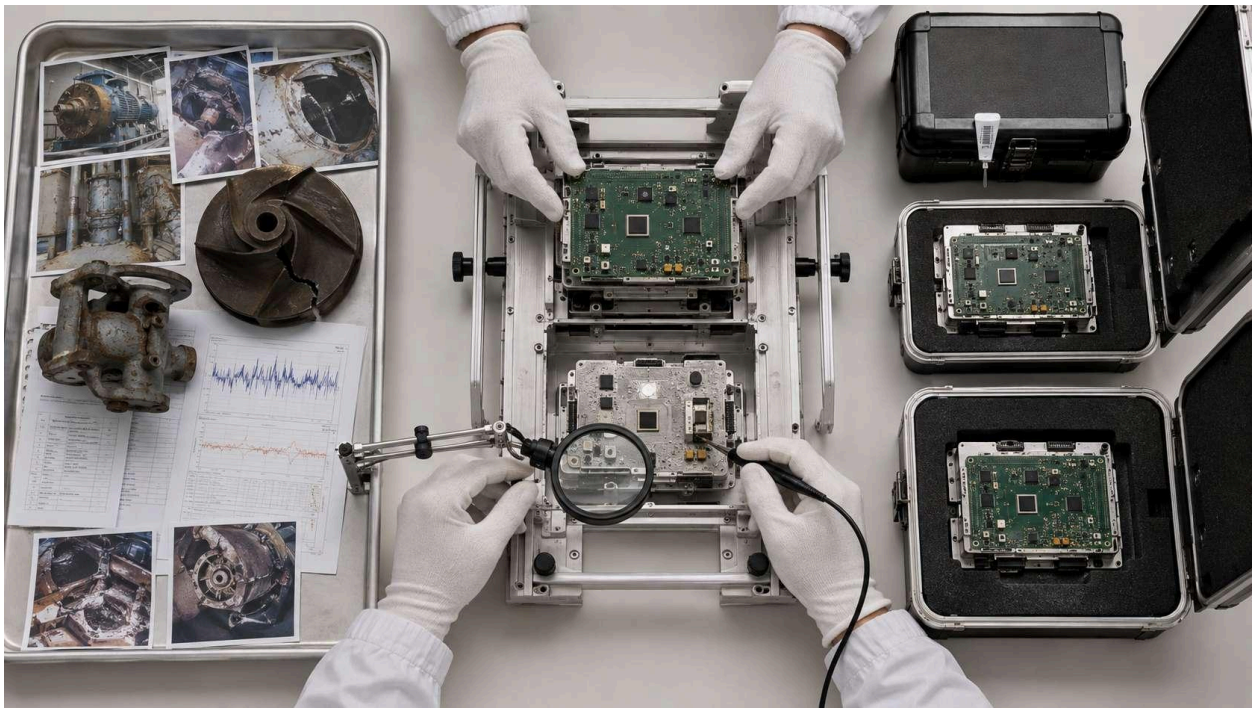
EON AI Ventures

Owning the Delta

Training our own models with LoRA — and why the trained model, not the base model, is the asset



Contents

Executive summary

A large language model is raw capability with no context. It has never seen our customers' equipment, it does not know their procedures, and — most importantly — it has no way of finding out whether what it just said was correct. Closing that gap is what turns a model into a product.

This paper describes how we close it: by training our own small, specialised models on verified data, using a technique called LoRA. The output of that process is what we call an EON Delta.

A Delta comes in two halves, and the distinction between them is what makes it contractable. The Tenant Delta is trained inside the customer's own tenant on their own records; it is their property and never leaves. The Base Delta is trained only on de-identified patterns that have been observed at two or more unaffiliated organisations; it is EON's property, it survives termination, and every participating customer receives it royalty-free in perpetuity. Each half ships with the verified dataset that produced it and a benchmark score.

The five things worth taking away

- Facts belong in a database. Judgment belongs in the weights. Putting facts into a model's weights makes them unchangeable, unexplainable, and undeletable — each of which alone would end an energy-sector procurement conversation.
- We do not retrain models. We freeze the base model and train a small adapter — roughly 0.4% of the parameters, a few hundred megabytes, a few hundred dollars, a few hours on one rented GPU.
- The base model is a commodity our competitors can download this afternoon. The Base Delta is not — it carries the de-identified judgment of an entire industry, gathered under a standard operators will actually sign. That asset appreciates as models get cheaper, and it is what our own Data Rights Standard calls load-bearing for enterprise value.
- The bottleneck is never the training run. It is whether we do the unglamorous work of assembling one thousand verified examples. That is 80% of the effort and all of the risk.
- Documented failures — incident reports, root-cause analyses, lessons learned — are the most valuable training data we can obtain, because correct procedure is already written down and the gap between procedure and reality is not.

The commercial sentence this unlocks

“Tested against 300 of your own historical incidents, using only the data available 48 hours before each event, our system flagged 214 of them in advance.”

No competitor can say that sentence, because saying it requires the customer's own history, a verification loop, and a benchmark written before the model existed. Producing it costs us no training run at all — only the discipline to build the benchmark first.

PART ONE

Plain English

Everything in this part is written to be understood without any background in machine learning. There are no formulas. If you only read one section, read section 5 — it is the step-by-step account of how we actually train a model.

---

1. The problem: capability without context

Imagine hiring the smartest twenty-two-year-old in the world. They read anything, remember everything, and think faster than you do. On Monday morning you walk them into a gas facility and say: keep this safe.

They will be useless. Possibly dangerous. Not because they are stupid, but because they have no map of the plant, no copy of the procedures, nobody checking their homework, and no way to discover whether the thing they just said was right or wrong.

A raw model is exactly that person. Everything that converts raw capability into a reliable worker sits outside the model: the data it can look things up in, the tools it can act with, the checks on its output, and the loop that tells it when it was wrong.

Why this matters more in industry than anywhere else

The reason AI became extraordinarily good at writing software is that software has a simulator with an automatic grader. The model writes code, runs it, and the tests instantly say pass or fail. It can try, fail, and improve thousands of times before a human looks.

Physical industry has almost none of that. You cannot let a model try things on a live pipeline. This is precisely why AI in the physical world has largely stalled at the level of a chatbot — and precisely where our digital twin becomes strategically interesting, because a twin is a safe place for a model to be tested.

2. The professor and the apprentice

The core technique in this paper has a simple shape. You cannot put the professor in a backpack, so you do not try.

Instead: the professor — a very large, very expensive, very general model — answers ten thousand questions. We check and correct the answers. Then an apprentice — a small model that fits on a device — studies only the checked answers until it can produce them itself.

The apprentice ends up as good as the professor at that one job, because it only ever had to learn one job. It is a hundred times smaller, runs offline, and costs almost nothing per query. This is called distillation.

	The professor	The apprentice
What it is	Frontier or large open-weight model	Small open-weight model plus our adapter
Where it runs	A datacentre full of GPUs	A backpack, a vehicle, a plant room
Cost per query	Material	Effectively zero
Breadth	Knows a bit about everything	Knows one job extremely well
Our use	Disposable labour that generates draft answers	The product we ship and version

The point of the analogy

The professor is rented and replaceable. The apprentice — trained on our verified data, carrying our judgment, running on our customer's hardware — is ours. Everything strategic in this paper follows from that asymmetry.

3. Where knowledge lives: three places

This is the question that causes the most confusion, so it is worth being precise. Think again about a new engineer. Knowledge they use can sit in three different places.

Place	In AI terms	What it holds	How you change it
Their brain from school	The weights	Behaviour, style, judgment, taste	Fine-tuning (this paper)
What is on their desk now	The context window	Only what you just handed them	Put it in the prompt
The filing cabinet	The database / retrieval	All facts: procedures, logs, incidents	Edit a row

3.1 Facts go in the filing cabinet, never in the brain

Nobody serious stores company facts in a model's weights. There are three reasons, and each one alone is fatal in our market.

1. **Facts change.** A plant adds a valve. In a database that is one row edited in one second. In the weights it means retraining the model.

2. **You must be able to explain yourself.** A regulator asks why the system said what it said. With a database you point at the document, page 14. With weights you cannot — the fact is smeared across billions of numbers. Unexplainable is unsellable.
3. **You must be able to delete.** A customer leaves and instructs us to remove their data. In a database that is a DELETE statement. In weights it is impossible; the information is cooked in. This alone kills contracts.

3.2 So what does go in the weights?

Judgment. Not facts — taste. The format a report should take. The order in which evidence should be weighed. The instinct to say "I am not sure" rather than guess. The ten thousand small decisions an experienced engineer makes without being able to explain them if you ask.

This is the part that cannot be written down, because nobody can articulate it. But it can be demonstrated — and a fine-tune learns from demonstration. That makes fine-tuning the mechanism for capturing the expertise our best people cannot put into a manual.

> **Why the timing favours us**
> The generation who genuinely know how these plants behave is retiring. Their intuition was never in any procedure document and cannot be interviewed out of them at scale. It can, however, be captured from graded examples of their work — which is exactly what this pipeline consumes.

4. What a LoRA actually is

LoRA stands for Low-Rank Adaptation. The plain-language version is short:

> We do not rewrite the model. We freeze it exactly as downloaded, and bolt a small extra piece onto the side. Only that small piece is trained. That piece is the EON Delta.

The consequences of that design are what make the whole approach practical:

- The trained piece is roughly 0.4% of the model's parameters — a few hundred megabytes instead of a terabyte.
- A training run takes hours on one rented GPU and costs a few hundred dollars, not weeks on a cluster.
- It is fully reversible. Unload the adapter and the base model is untouched — we cannot damage the thing we started from.
- We can hold many adapters at once: one per customer, one per job, all swapped over a single shared base model.
- Once training finishes, the adapter can be merged into the base weights, so there is no speed penalty at all when running it.

Layer	Who has it	Value to us
Base open-weight model	Anyone, free, this afternoon	None. A commodity engine.
Tenant Delta	The customer	Theirs. Trained in their tenant, never leaves it, destroyed on exit.
Base Delta	EON	High. Survives termination; licensed back to every participant.
The verified dataset	Split by tier — see section 7	High, and it is the hard part.
The benchmark score	EON	High — it is how we sell and how we gate.

5. How we actually train it: the walkthrough

This section is the one to read carefully. It describes, concretely, what happens from “we have an idea” to “a trained model is running on a device.” Eight steps.

Step 1 — Pick exactly one job

This is where projects like this succeed or fail, and it happens before any technology is involved. A small model cannot be a general assistant. It can be excellent at one narrow job.

Wrong: “answers any question about the plant.” It will be mediocre at everything, and the team will wrongly conclude that small models do not work.

Right: “Given what the technician is looking at and says, identify the equipment, surface the correct procedure step, flag unsafe conditions, and draft the inspection record.”

Write the job down in one sentence and get it agreed before anything else happens.

Step 2 — Collect the examples

Training data is pairs: an input, and the output we wish the model had produced. There are three sources, in descending order of value per unit of effort.

4. **Records we already have.** Past inspection reports, work orders, incident write-ups. The input is the situation; the output is what the expert actually wrote. Free, real, and already validated by reality.
5. **Professor-generated, engineer-checked.** The large model drafts thousands of pairs from our documents; engineers correct them. Correcting is roughly five times faster than authoring from scratch. This is how we reach volume.
6. **Twin-generated.** Run a scenario in the digital twin and capture situation → correct action. Nobody else can produce these, which makes them the most defensible data we own.

Target for a first model: 500 to 1,000 verified pairs. Quality dominates quantity — five hundred excellent examples beat fifty thousand sloppy ones, and sloppy examples actively teach the model to be sloppy.

What one training example actually looks like

Everything above is abstract until you see a pair. Here is one, in the form the training script consumes. The tags and identifiers below are illustrative, not real Orland data.

```
# INPUT  — the situation, as the model receives it

ASSET: Compressor Station 4 — Unit B discharge line
READINGS (last 6 h):
  PT-4412  discharge pressure  61.2 -> 58.9 barg   (steady decline)
  TT-4415  discharge temp      44 -> 47 degC
```

```
  FT-4401  flow                128 -> 126 kSm3/h
LAST INSPECTION: 2026-02-14 — external corrosion noted at weld W-118;
  cathodic protection survey overdue
TECHNICIAN NOTE (voice): "there's a hissing sound near the discharge
  flange, can't see anything"
```

```
# OUTPUT — what we want the model to produce

ASSESSMENT: Possible seal or flange leak at discharge. Medium confidence.

EVIDENCE: 2.3 barg decline over 6 h against only a 1.6% flow reduction is
  not consistent with a demand change. Audible hiss localises to the
  discharge flange. Temperature rise is consistent with recirculation.

PRIOR: INC-2023-041 at CS-2 showed the same pressure/flow signature
  nine days before a gasket failure on a comparable flange.

RECOMMENDED ACTION: Portable gas detection sweep at flange F-4412-A
  before next shift. Do NOT isolate on this evidence alone.

CONFIDENCE: Medium. Missing: vibration data, CP survey result.
NOT ESTABLISHED: leak location, leak rate.
```

Study what that output contains, because it is the whole point of fine-tuning. It has a fixed structure. It separates evidence from conclusion. It cites a prior incident by number. It states its confidence. It names what is missing. And it explicitly refuses to recommend isolation on thin evidence.

None of that is a fact you could look up. It is judgment — and it is exactly what a thousand examples of this shape will teach a small model to do reliably.

Contrast: what an untuned model says to the same input

“The pressure drop may indicate a possible leak. It is recommended that maintenance investigate the issue as soon as possible. Safety should always be the top priority.”

Fluent, useless, and unfalsifiable. It cites nothing, commits to nothing, and states no confidence. The gap between that paragraph and the one above is the entire value of the training run.

Step 3 — Clean the examples: green, yellow, and the third colour

The interview method we developed with UBT applies directly here, with one addition. Incident reports in particular are full of material that will poison a model.

Colour	What it is	Examples	Action
Green	Verified fact	Timestamped sequence, measurements, physical mechanism, corrective action taken	Keep
Yellow	Narrative and drama	Blame, defensiveness, corporate hedging, speculation about motive	Discard
Third colour	Knowable only in hindsight	Anything the writer knew because the event had already happened	Label separately; keep out of the input

Why the third colour matters so much

Post-incident reports suffer from hindsight bias. They make the cause look obvious — "they should have noticed the pressure decline." At the time it was not obvious at all.

If we train on the report's after-the-fact framing, the model learns to be confidently certain about situations that were genuinely ambiguous. In the field it will then be wrong with total conviction, which is the single most dangerous failure mode a safety-adjacent system can have.

The rule: build each input from the information state as it existed before the event, and use the outcome only as the label. Feed the model only what was actually knowable at the time.

Step 4 — Write the exam before you train anything

Set aside 100 to 200 examples that the model will never see during training. This is the held-out set, and it is the only instrument that can tell us whether anything we do helps.

For the Field IQ Edge model we have an unusually good source for it: Orland's documented incidents. Each becomes a test of the form — given only what was known 48 hours before this event, does the system flag it?

> **Do this first, and do it before the training set**
>
> A few hundred incidents is small for training but excellent as a benchmark. Building the benchmark first costs no training run, produces the most credible sales artifact we could own, and gives us the gate that every future model version must clear.
>
> The held-out set is written once and never touched again. Tune against it and every number it produces becomes fiction.

Step 5 — Choose the student model

Now hardware reality enters. The size of the model is bounded by what the device can hold and power.

Parameters	Size at 4-bit	Runs on	Capable of
1–4 billion	0.7–2.5 GB	Phone, Jetson Orin Nano	Identification, extraction, procedure lookup, form filling
7–8 billion	4–5 GB	Jetson Orin NX/AGX, mini-PC	Light reasoning, drafting records, ranked warnings
12–14 billion	7–9 GB	Orin AGX 64 GB, small GPU	Stronger reasoning; battery and thermals become limiting
30 billion +	17 GB +	Rack GPU	Not a backpack — this is the base-station tier

Recommendation for Field IQ Edge: 7 to 8 billion parameters at 4-bit precision. Enough capability to reason and draft, small enough for a wearable power budget. Candidate bases include the Qwen, Llama, Gemma and Mistral small models — all open-weight. Which one matters far less than the pipeline; keep it swappable.

“4-bit” refers to quantization: the model's numbers are stored with less precision, which makes the file roughly four times smaller while degrading quality only slightly. It is what makes edge deployment possible at all.

Step 6 — Run the training

This is the step people imagine is hard. It is the easiest one.

In plain language, here is what the computer does. It shows the model an input from our file. The model predicts the output one word at a time. After each word, the script compares what the model predicted against what our expert actually wrote. Where they differ, it nudges the numbers in the adapter very slightly in the direction that would have produced the expert's word instead. Then it moves to the next example.

It does that across all thousand examples, then repeats the whole pass two or three more times. Four hours later, the adapter has absorbed the pattern.

```
# What actually gets run — conceptually

1.  provision one GPU  —  INSIDE the customer tenant for a Tenant Delta;
                          EON infrastructure only for a Base Delta
2.  load base model, frozen, quantized to 4-bit
3.  attach LoRA adapter  (rank 16, ~30 M trainable parameters)
4.  for each of 3 epochs:
        for each batch of examples:
            predict -> compare to expert output -> nudge adapter
```

```
5.  merge adapter into base weights
6.  quantize merged model to 4-bit for the device
7.  export to GGUF (llama.cpp) or TensorRT-LLM (Jetson)
```

We do not write this ourselves. Mature open tooling exists — Axolotl, Unsloth, or Hugging Face TRL — and it takes a configuration file plus our dataset. Reference hyperparameters are in section 19.

Where the training run physically happens is a contractual matter, not a convenience

A Tenant Delta is trained on Tier 1 Customer Data, which under Schedule B.2.2 does not leave the customer's tenant. The training run therefore executes inside that tenant, on their infrastructure, and the resulting adapter stays there. Renting a GPU at our end and pulling their records across would breach the standard before a single epoch completed.

A Base Delta trains only on qualified Generalized Learnings and may run on EON infrastructure. Keeping these two pipelines physically separate — different environments, different credentials, different manifests — is what makes the audit right in Schedule B.11 answerable.

Total cost of a first serious attempt: a few hundred dollars of GPU time and about a week of one engineer's attention. The barrier is never the training. It is the dataset.

Step 7 — Score it, shrink it, ship it

Run the held-out exam. Compare three numbers: how the small model scores, how the professor scores, and how a competent human scores.

- If the small model reaches 85–95% of the professor on the narrow job, ship it.
- If it does not, the fix is almost always more and better examples — not a bigger model. Resist the urge to jump to a larger base.
- Re-run the exam after quantizing for the device. Never assume the score survived compression.

Only then does it go onto hardware and into someone's hands.

Step 8 — Close the loop in the field

The deployed model is not the end of the pipeline; it is the start of the next turn of it.

- On the device, when the model is not confident, it says so and defers to the technician.
- The technician's correction is logged locally, with the full context that produced it.
- When the device next has connectivity, corrections sync home.
- The next Distillery run folds them into the dataset, retrains the adapter, and re-scores the benchmark.

This is where the compounding happens

Every day in the field makes the next version better. A competitor can download the same base model this afternoon. They cannot download our correction log, and they cannot shortcut the years of real work inside real plants that produced it.

6. Why this is good for us

The technical account above is only worth the effort if it produces commercial advantage. It does, in six distinct ways.

6.1 It resolves the data-sovereignty objection, which is otherwise fatal

An operator of critical energy infrastructure cannot send live operational data to a third-party cloud. That is frequently not a preference but a condition of their licence, their insurance, or their national regulator.

Open weights running on hardware inside the customer's perimeter mean nothing leaves. That is a physical guarantee, not a contractual promise — and it is the difference between a conversation that can proceed and one that cannot.

The single sentence that carries this: the AI never touches your pipeline; it touches the twin of your pipeline, on your own hardware.

6.2 The economics invert in our favour

Running a very large model continuously is expensive — on the order of tens of thousands of dollars a month if saturated. That only makes sense at steady high volume, or when it is a sovereign deployment on the customer's own hardware, purchased by the customer as a line item.

A distilled small model costs effectively nothing per query, runs without connectivity, and is ours to deploy as widely as we like. We should not be burning our own capital hosting a giant model; we should be selling sovereign deployments and shipping cheap adapters.

6.3 We own an asset that appreciates as models get cheaper

Base models are commoditising rapidly. In eighteen months there will very likely be a free model as capable as today's best. Anyone whose advantage was the model itself will watch it evaporate.

Our advantage is the Base Delta and the benchmark that proves it works: the de-identified judgment of an entire industry, gathered with documented consent under a published standard. Better and cheaper base models make that more valuable, not less, because they raise the ceiling on what it can be turned into. Our Data Rights Standard makes the same point in contractual terms — the survival of EON's rights in Generalized Learnings past termination is what an investor can actually verify we own.

6.4 It converts our digital twin from content into infrastructure

Today the twin is training material for humans. The argument in this paper makes it something more valuable: the grading environment where an AI's proposals can be tested before a human ever sees them.

That is the scaffolding the rest of the industry lacks. We have been building simulators of physical work for twenty-five years; almost nobody trying to deploy industrial AI has one.

6.5 It gives us a defensible number to sell on

Recall against a customer's own documented incident history is a number no competitor can produce, because producing it requires the customer's data and a benchmark built before the model existed. It also turns the sales conversation from a demonstration into an audit — a much stronger position.

6.6 One factory, two product lines

The same pipeline, the same data discipline and the same benchmark serve two very different products.

	Line 1 — Edge adapters	Line 2 — Sovereign adapters
Base model	3–8 B open-weight	Large open-weight, e.g. a Kimi-class MoE
Runs on	Backpack, vehicle, plant room	Customer's own datacentre hardware
Commercial role	Learning vehicle; cheap to iterate	High-value; customer funds the hardware
Risk	Low. Fast feedback.	Higher. De-risked by Line 1.

Line 1 teaches us the pipeline at low cost and low stakes. Line 2 monetises what Line 1 taught us. Building them in the wrong order is the main way this goes wrong.

6.7 It is the only way to keep two promises we have already made

EON's published Data Rights Standard makes two commitments that are difficult to hold together. Every participating operator receives the entire body of Generalized Learnings, royalty-free and in perpetuity. And EON never sells that body as a standalone dataset — because, in the Standard's own words, the moment EON is a data broker the safety framing collapses and every operator gets nervous.

Give them everything, but never hand over the collection. A trained model is the artifact that does both. It carries the patterns without being a dataset: it answers questions about the situation in front of a technician rather than presenting a list that can be read, searched, cross-referenced and copied.

> **A data file is a pipe. A model is a teacher.**
>
> This distinction is not stylistic. A de-identified file can still be re-identified by someone with domain knowledge — an entry describing an unusual configuration in a specific service may match only a handful of facilities worldwide. Handing that file to a competitor of the contributor is precisely the hub-and-spoke exposure the Standard is designed to avoid.
>
> A model trained on the same patterns transfers the competence without transferring the record. It moves knowledge from the corpus to every participant, and never from one participant to another.

One drafting point for counsel. Schedule B.4.3 forbids making the body of Generalized Learnings available as a standalone dataset “to any third party”, while B.8.1 grants every customer access to its entirety. Whether a

participating customer is a “third party” for the purposes of B.4.3 is ambiguous on the current wording — and each participant is unambiguously a third party relative to every other. Delivering only in model form resolves the ambiguity in the direction the Standard's own strategy already points, but the wording should be tightened rather than left to interpretation.

7. What it costs and how long it takes

Weeks	Work	Output	Cost
1–2	Define the one job. Write the exam from documented incidents.	Job statement plus ~150 held-out test cases	Internal time
3–6	Assemble 1,000 verified examples. Green/yellow/third-colour pass.	Verified training set	Internal time — the real cost
7	LoRA fine-tune, merge, quantize, score against the exam.	Adapter plus benchmark score	$300–900 GPU rental
8	Deploy to hardware. Walk a site with it.	Working Field IQ Edge prototype	Device cost

Where the risk actually sits

Two months to a real answer on a few thousand dollars. Nothing in the technical path is novel or uncertain — this is well-trodden ground with mature tooling.

The risk is organisational: whether we will do the unglamorous dataset assembly in weeks three to six, or let it slip while waiting for something more interesting to work on. That is the whole ballgame.

8. Risks and honest caveats

Legal exposure on incident data

Incident reports are often legally privileged, tied to regulator filings, and occasionally to live litigation. Before any of this data enters a training pipeline we need written permission for AI training use, de-identification of individuals, and a check that the model cannot reproduce identifying details verbatim. This is a week-one task, not a week-seven task.

The over-warning trap

A model trained only on things going wrong believes everything is about to go wrong. Over-warning produces alarm fatigue; alarm fatigue means nobody reads the warnings; and a warning system nobody reads is worse than none, because it manufactures false confidence. The mitigation is to include normal days — cases where conditions looked similar and nothing happened — at a realistic ratio. Section 16 derives why this matters so much more than it sounds.

Never in the control path

Nothing produced by a language model should reach a PLC or a safety instrumented system, validator or no validator. Those loops are governed by deterministic logic under functional-safety standards with assigned integrity levels; a stochastic component upstream of them is unauditable. Our architecture is strictly one-directional: data flows out of the operational network, and advice flows to humans.

Regulatory classification

Under the EU AI Act, a system acting as a safety component in critical infrastructure management falls into the high-risk category, with conformity assessment, logging and human-oversight obligations attached. Keeping our system strictly advisory and outside the safety function is what keeps us out of that bucket. This is a design decision with legal consequences and should be confirmed with counsel rather than assumed.

Base-model churn

Every base-model upgrade invalidates the adapter, because the adapter's parameters are tied to the specific weights it was trained against. This is manageable but not free. It is also the strongest argument for treating the dataset and the benchmark as the durable assets: re-running a fine-tune takes hours, whereas rebuilding a dataset takes months.

PART TWO

The Theory

This part assumes comfort with linear algebra and probability. It gives the underlying mechanics in full, so that the choices recommended in Part 1 can be checked rather than taken on trust.

---

9. What a weight is

A transformer is a stack of layers; each layer is predominantly matrix multiplication interleaved with normalisation and a nonlinearity. A weight is a single scalar entry in one of those matrices.

```
y = W x + b                       W in R^(d_out x d_in)

Per layer, attention:             W_q , W_k , W_v , W_o
Per layer, feed-forward (MLP):    W_gate , W_up , W_down
```

For a model with hidden dimension d = 4096 and 32 layers, a single attention projection is a 4096 × 4096 matrix — 16.78 million parameters on its own. Training means adjusting those scalars so that predictions improve against an objective.

Quantity	Value
Hidden dimension d_model	4,096
Layers	32
Attention matrices per layer	4
MLP matrices per layer	3
Total parameters	~8 × 10^9
Storage at FP16 (2 bytes/param)	~16 GB

Pre-training sets all of these from scratch by optimising over internet-scale corpora: on the order of 10^25 FLOPs, months of wall-clock time, and hundreds of millions of dollars. We never do this, and neither does anyone outside a handful of laboratories.

10. The supervised fine-tuning objective

Supervised fine-tuning is next-token prediction over our own input–output pairs. The loss measures the model's surprise at the desired output; gradient descent reduces it.

```
# Token-level cross-entropy over target sequence y given prompt x

    L(theta)  =  - sum_t  log P( y_t | y_<t , x ; theta )

# Parameter update, learning rate eta

    theta  <-  theta - eta * grad_theta L(theta)        (AdamW in practice)
```

Three practical consequences

- **Loss is masked over the prompt.** Only tokens belonging to the desired output contribute to L. The model is not trained to reproduce the question, which would waste capacity and encourage parroting.
- **Typical hyperparameters for LoRA SFT.** Learning rate 1e-4 to 2e-4, two to four epochs, effective batch size 8 to 32, cosine decay with a short warmup.
- **Overfitting is the dominant failure mode at our data scale.** With a thousand examples, more than roughly four epochs memorises rather than generalises. Monitor held-out loss; training loss will keep falling and tells you nothing.

11. LoRA, formally

The hypothesis behind LoRA is that the weight change required to specialise a pre-trained model has far lower intrinsic rank than the weight matrix itself. If that holds, there is no need to represent ΔW densely — it suffices to store two thin factors whose product is ΔW.

```
    h  =  W_0 x  +  dW x   =  W_0 x  +  (alpha / r) * B * A * x

    W_0 in R^(d x k)   frozen
    A   in R^(r x k)   trained,  init  A ~ N(0, sigma^2)
    B   in R^(d x r)   trained,  init  B = 0
    r   <<  min(d, k)

# Because B = 0 at initialisation, dW = 0 at step 0:
#   training begins exactly at the base model, never at a random perturbation.

# Trainable parameter count
    r (d + k)          versus        d * k     for a dense update

# After training, the adapter can be folded in:
    W  =  W_0 + (alpha / r) B A     =>  zero added inference latency
```

Hyperparameter	Range	Guidance
Rank r	8 – 64	Higher r gives more capacity and more overfitting risk. r = 16 is a sound default for behaviour tuning on ~1,000 examples.
Scaling α	typically 2r	Decouples effective step size from r. Holding α/r constant keeps runs comparable across ranks.
Target modules	q,k,v,o first	Add the MLP projections (gate, up, down) when the task needs more expressive capacity than attention alone provides.
Dropout	0.0 – 0.1	Mild regularisation; useful at small dataset sizes.

12. Parameter and memory arithmetic

12.1 Why 0.4% of the parameters suffices

Quantity	Expression	Value
One attention projection, dense	d × k = 4096 × 4096	`16,777,216`
Same projection, LoRA r = 16	r(d + k) = 16 × 8192	`131,072`
Share of that matrix trained	r(d+k) / dk	`0.78%`
Adapter over q,k,v,o × 32 layers	`4 × 32 × 131,072`	`~16.8 M`
Including MLP projections	7 × 32 × r(d + k_mlp)	`~30 M`
Share of the full 8 B model	30 M / 8 B	`~0.4%`

These are illustrative figures for a d = 4096, 32-layer architecture; MLP widths and therefore exact counts vary by model family.

12.2 Why the training run fits on one GPU

Training memory is dominated not by the weights but by gradients and optimizer state. Freezing the base removes almost all of it.

```
# Full fine-tune with AdamW
    params(fp16) + master copy(fp32) + 2 optimizer moments(fp32)
    ~ 16 to 20 bytes per parameter
    8 B params  ->  approx. 128 to 160 GB       (multi-GPU territory)

# LoRA over a 4-bit frozen base (QLoRA)
    base weights     ~ 0.5 bytes/param  ->  ~4.5 GB
```

```
    optimizer state  on ~30 M trainable params only
    activations + overhead
    total            ~ 6 to 10 GB              (one 24 GB GPU)
```

That single arithmetic fact is why a three-hundred-dollar run substitutes for a three-hundred-thousand-dollar one. Gradient checkpointing trades roughly 30% throughput for a further substantial memory reduction where needed.

13. Quantization

Weights are stored by default in 16-bit floating point. Quantization stores them with fewer bits: size falls roughly linearly in the bit width, while task quality degrades far less than linearly.

```
    size(bytes)  =  N_params  x  bits / 8

# Affine quantization of a weight block
    w  ~=  s * q + z ,        q in {0 ... 2^b - 1}
    s  =  (w_max - w_min) / (2^b - 1)          scale
    z  =  zero point                           offset

# Group-wise scales (e.g. one scale per 64-128 weights) preserve accuracy
# substantially better than a single scale per tensor.
```

Precision	8 B footprint	Quality impact
FP16 / BF16	~16 GB	Reference
INT8	~8 GB	Near-identical on most tasks
INT4 (e.g. Q4_K_M)	~4.5 GB	Small, usually acceptable loss
INT3 and below	~3.5 GB	Noticeable degradation; not recommended

Two distinct uses, often confused

QLoRA quantizes the frozen base during training so the run fits on one GPU. Post-training quantization shrinks the merged model so it fits the target device. They are different steps with different risks.

Always re-run the benchmark after quantizing. Never assume a score survived compression — that assumption is how a validated model becomes an unvalidated one.

14. Distillation, formally

Classical knowledge distillation minimises the divergence between the student's output distribution and the teacher's, softened by a temperature.

```
# Logit-level (Hinton et al.) — requires access to teacher logits z_T

    p_i^(T)  =  softmax( z_i / T )                 T = temperature

    L  =  alpha * L_CE(y, p_S)
          + (1 - alpha) * T^2 * KL( p_T^(T) || p_S^(T) )

# The T^2 factor rescales gradients so that soft-target magnitude does not
# shrink as T rises.
```

We will generally not do this, because teacher logits are unavailable behind a commercial API and because logit matching transfers the teacher's errors along with its competence. Instead we use sequence-level distillation on verified outputs:

```
# Sequence-level distillation — what we actually run

  1.  Teacher generates k candidate outputs  y_hat  for each input x
  2.  Filter: rule checks, twin replay, or engineer review
  3.  Correct the survivors to  y*        (this is the value-add step)
  4.  Train the student with plain cross-entropy on the pairs (x, y*)
```

Step 3 is where our proprietary advantage enters. A corrected answer carries our engineers' judgment; a teacher logit carries only the teacher's. Rejection sampling — generating k candidates and keeping only those that pass a check — is an inexpensive way to raise dataset quality before human review.

15. Dataset construction and the temporal cutoff

The hindsight-bias problem described in Part 1 has a precise formulation. Let I_t denote the information set available at time t, and let the outcome of interest occur at t + Δ.

```
    Correct construction:      x = f( I_t )         y = outcome( t + delta )

    Leakage:                   x = f( I_{t + delta} )

# Any field in x whose value could only have been recorded after the event
# gives the model a shortcut to the label. Test accuracy rises; field
# accuracy does not. This is the most common silent defect in incident-
# derived datasets.
```

Enforcement is mechanical rather than a matter of judgment: every source field carries a timestamp, and the dataset builder rejects any field whose timestamp exceeds the cutoff. Where a report does not permit reconstruction of I_t, the example is discarded rather than approximated.

Composition targets

Class	Share	Rationale
Documented incidents (positive)	10 – 20%	The rare, high-value cases we must detect
Near-misses that resolved	10 – 20%	Teaches discrimination near the boundary
Normal operating days (negative)	50 – 70%	Calibrates the base rate; prevents over-warning
General instruction-following	5 – 10%	Mitigates catastrophic forgetting

16. The base-rate problem, derived

This section justifies a claim that sounds counterintuitive: a detector that is 99% accurate can be operationally worthless. It follows directly from Bayes' rule when the event is rare.

```
# Setup
    pi           = P(incident)      = 1e-4  per inspection-day
    sensitivity  = P(+ | I)         = 0.99
    specificity  = P(- | not I)     = 0.99

# Over 10,000 inspection-days
    true positives   TP = 10000 * pi * 0.99             ~= 0.99
    false positives  FP = 10000 * (1 - pi) * 0.01       ~= 99.99

# Precision (positive predictive value)
    P(I | +) = TP / (TP + FP) ~= 0.0098      ->   about 1%
```

Roughly one hundred false alarms for every genuine catch. The detector is, by the usual reporting convention, “99% accurate”, and it will be switched off within a month.

The structural point is that when π is very small, precision is governed by the false-positive rate rather than by accuracy. No amount of additional model quality escapes this — only a change of objective does.

What to do instead

7. **Rank rather than alarm.** Return the k highest-risk assets per shift rather than a binary flag per asset. Human review cost becomes bounded and predictable, and the metric becomes precision@k.
8. **Permit abstention.** “Insufficient data” must be an available and rewarded output. A model with a zero abstention rate is guessing on its hardest cases.
9. **Report precision@k and recall, never accuracy.** Accuracy on an imbalanced problem is a number that flatters the system and misinforms the buyer.
10. **Calibrate.** A stated 30% risk should correspond to events occurring about 30% of the time. Calibration is what makes a probability actionable rather than decorative.

17. Evaluation

Metric	Definition	Why it governs the ship decision
Teacher-relative score	student score ÷ professor score on the held-out set	Ship at 85–95%. Below that, add examples rather than parameters.
Precision@k	fraction of the top-k flagged items that are genuine	Matches how a crew consumes output: a bounded worklist per shift.

Metric	Definition	Why it governs the ship decision
Recall on the incident set	share of documented incidents flagged from pre-event data only	The customer-facing headline number and the contractual gate.
Brier score	$BS = (1/N) \Sigma (p_i - o_i)^2$	Measures calibration; penalises confident wrongness, the dangerous failure.
Abstention rate	share of cases returning “insufficient data”	Should be non-zero and stable. Zero means the model is guessing.
Regurgitation probe	attempts to elicit verbatim training text	Legal requirement where training data is privileged.

The one procedural rule that makes all of the above meaningful

The held-out set is authored before the first training run and never used to make a tuning decision. If it influences a hyperparameter choice even once, it stops being an estimate of field performance and becomes a description of our own fitting process.

Where iteration against a scored set is genuinely needed, use a third split — a validation set — and keep the held-out exam sealed.

18. Failure modes and mitigations

Failure mode	Mechanism	Mitigation
Catastrophic forgetting	Narrow tuning degrades unrelated capability, including instruction-following and refusal behaviour	Low rank, low learning rate, few epochs; retain 5–10% general examples; adapters isolate damage — unload and the base is intact
Hindsight leakage	Post-event fields let the model read the label off the input	Enforce a per-example temporal cutoff; discard examples where I_t cannot be reconstructed
Over-warning	Positive-only training distorts the implied base rate	Composition targets in section 15; report precision@k
Base-model drift	Adapter factors are tied to specific W_0; upgrades invalidate them	Treat dataset and benchmark as durable assets; re-running a fine-tune is hours
Memorisation of sensitive text	Verbatim recall of names or privileged report content	De-identify before training, cap epochs, include a regurgitation probe in the benchmark

Failure mode	Mechanism	Mitigation
Quantization regression	Compression shifts behaviour after validation	Re-score the benchmark post-quantization, on the target runtime

19. Reference recipe

A concrete starting configuration for the first Field IQ Edge model. These are defaults to be beaten, not settled conclusions.

```
base model          7-8 B open-weight, instruction-tuned variant
quantization (train) 4-bit NF4, double quantization  (QLoRA)
adapter rank r       16
alpha                32
dropout              0.05
target modules       q_proj, k_proj, v_proj, o_proj,
                     gate_proj, up_proj, down_proj
learning rate        1.5e-4, cosine decay, 3% warmup
epochs               3       (monitor held-out loss; stop early if it rises)
effective batch      16      (per-device 2 x grad-accum 8)
max sequence length  4096
optimizer            paged AdamW, 8-bit
precision            bf16 compute
gradient checkpoint  on

dataset              ~1000 verified pairs, composition per section 15
held-out exam        150 cases, sealed

export               merge adapter -> quantize Q4_K_M -> GGUF
target runtime       llama.cpp, or TensorRT-LLM on Jetson
hardware (train)     1 x A100 40GB or H100, ~4 h
```

20. Data rights compliance at the model layer

Every control in the Data Rights Standard was drafted for a corpus — a body of patterns held in a database. A trained model is a lossy, queryable compression of its training data, and models memorise. Applying the Standard's controls only to the dataset and not to the artifact trained on it would leave the substantive obligation unmet.

The table below maps each control onto its model-layer equivalent. None of these are optional refinements; each is the technical form the contractual obligation has to take.

Clause	What it requires	Model-layer implementation
B.5.1 Multi-source	Observed at two or more unaffiliated organisations	Governs training-set composition, not merely the pattern list. An adapter where the large majority of examples originate with one contributor encodes that contributor regardless of field-level de-identification. Enforce a maximum share per contributor and record it in the manifest.
B.5.2 Non-attribution	Not identifiable directly or by inference	Extraction and membership-inference testing against the trained artifact. The regurgitation probe becomes a release gate rather than a quality check, and its result is a compliance record.
B.5.4 Temporal	Minimum three-month lag; nothing current or prospective	A hard cutoff enforced by the dataset builder, applied to source timestamps rather than to the reviewer's judgement.
B.7 Review window	Thirty-day item-level customer veto	A vetoed example cannot be surgically removed from a trained adapter — the adapter must be retrained. Batch review windows to close before a training run begins. LoRA's low retraining cost is here a compliance property, not only an economic one.
B.11 Audit	Customer may audit the separation controls	Immutable training manifests: which examples, drawn from which tier, at what qualification status, entered which adapter version, trained in which environment.
B.13.4 Post-termination	No new Generalized Learning derived from Customer Data	Pipeline-level revocation of the contributor's examples for future runs. Adapters trained before termination retain their learning under B.4.4, which is intended and should be stated plainly to the customer rather than discovered later.

The failure mode to design against

If a Base Delta can be prompted into revealing something attributable to a single contributor, EON has become the channel between competitors — with the additional difficulty that the transfer happened inside a model, where it is harder to detect and harder to prove was prevented.

This is why the probe belongs in the release gate. It is the only control that tests the artifact actually shipped rather than the process that produced it.

Appendix A — Glossary

Term	Meaning
Weights / parameters	The scalars inside a model's matrices. Collectively, everything the model has learned.
Fine-tuning	Continuing to train an already-trained model on a smaller, task-specific dataset.
LoRA	Low-Rank Adaptation. Freezing the base model and training two small factor matrices whose product is the weight update.
Adapter	The trained LoRA factors, stored as a separate file that can be attached to or merged into a base model.
EON Delta	Our unit of delivery: an adapter, the verified dataset that produced it, and its benchmark score. Comprises a Tenant Delta and a Base Delta.
Tenant Delta	The half trained inside the customer tenant on their own data. Customer property; never leaves; destroyed or returned on exit.
Base Delta	The half trained only on qualified Generalized Learnings. EON property; survives termination; licensed back to every participant royalty-free and in perpetuity.
Customer Data (Tier 1)	Raw operational capture and the customer's own procedures and reports as written. Customer property; not licensed to EON for training any model given to another party.
Derived Insights (Tier 2)	Analysis or models specific to the customer's sites, assets, crews or people. Customer property.
Generalized Learnings (Tier 3)	De-identified, non-attributable patterns meeting the four qualification conditions. EON property.
Distillery	The internal pipeline that produces EON Deltas. A factory, not a deployment target.
Distillation	Training a small model on the checked outputs of a larger one.
Quantization	Storing weights at lower numerical precision to shrink the model.
QLoRA	LoRA training performed over a quantized frozen base, so the run fits on a single GPU.
Epoch	One complete pass over the training dataset.
Held-out set	Examples deliberately withheld from training, used once to estimate real performance.
Base rate	The underlying frequency of the event being predicted.
Precision@k	Of the top k items the system flags, the fraction that are genuine.
Calibration	Agreement between stated confidence and observed frequency.
GGUF	A model file format used by llama.cpp for efficient CPU and edge inference.

Term	Meaning
Context window	The amount of text a model can consider at once. Temporary, not stored.

Appendix B — Immediate next steps

Three decisions gate the eight-week plan in section 7. None requires new technology and all three can be taken this week.

11. **Name the one job.** A single sentence describing exactly what the Field IQ Edge model does. Narrow succeeds; broad fails. Nothing else can start until this is agreed.
12. **Write the exam before the model.** 150 held-out cases drawn from documented incidents, constructed from pre-event data only. This is simultaneously the engineering gate and the strongest sales artifact we can own.
13. **Clear the legal path on incident data.** Written permission for AI training use, de-identification, and a regurgitation check in the benchmark. Week one, not week seven.
14. **Confirm the two-layer split with counsel.** That the Tenant Delta maps onto Derived Insights under Schedule B.3.1 and B.5.6, and the Base Delta onto Generalized Learnings under B.4.1 — and tighten the B.4.3 “third party” wording discussed at section 6.7. No external document should describe ownership until this is settled.

The sentence to remember

The base model is rented. The EON Delta is owned.