VELOCITY EDGE

TRAINING DOMAIN-SPECIFIC AGENTS

# Learning a Way of Working from Records of Work Done Well


## ABSTRACT

This paper argues that, where a domain-specific agent is trained, it should learn its way of working from records (trajectories) of the domain's work done well. The agent should read the facts it needs as it works. The paper uses domain-driven design and UML to define the domain and check each record. That method is the paper's own reasoning. The paper sets out what a record is, where records come from, what makes a good set and how to train on them. It describes where training on records fails and how to tell whether it worked. It ends with a practical sequence. Its first step, before any training, is to measure the best no-training alternative that a firm's data-residency and control limits allow. The language-model training studies it cites use benchmarks or narrow exam sets, on models from 2023 to 2025. Where one tested a trained model against a strong general model on the same test, the trained model won some comparisons and lost others. Its wins came mostly on tasks of the kind it was trained on, against general models that were simply prompted or older than the trained model. Each comparison used the same benchmark for both models. None of those studies ran the strongest general model of its time in a purpose-built agent framework on the same test. The paper believes the cited evidence on training is thin, and mixed on whether training gives a better agent. The studies' results may not hold outside the benchmarks' environments. In the paper's reasoning, training's case may rest on cost or speed, not on a better agent. It may also rest on data-residency and control limits that exclude a stronger model. Few cited results measure that cost or speed. The paper does not review other agent-training work or later general models. Nor does it review methods that learn from past runs without training, beyond those it cites.



**Phillip Pienaar**
Velocity Edge



2026-11-21
Version 1.1.0

# Contents

Executive Summary ..... 3
1. What to Train, and What to Give It to Read ..... 3
2. What a Record of Work Done Well Is ..... 4
3. Where Records Come From ..... 4
4. What Makes a Good Set ..... 5
5. Training on the Records ..... 5
6. Where It Fails ..... 6
7. Telling Whether It Worked ..... 7
8. A Practical Sequence ..... 8
Conclusion ..... 8
Glossary ..... 9
References ..... 10
Revision History ..... 11
Licence and Disclaimer ..... 12

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0 

## Executive Summary

The paper argues that, where a domain-specific agent is trained, it should learn its way of working from records (trajectories) of the domain's work done well, and read the facts it needs. The paper uses domain-driven design and UML 2.5.1 to define the domain and check each record. That method is the paper's own reasoning, which no cited study tests.

The paper believes the cited evidence on training is thin, and mixed on whether training gives a better agent. The language-model training studies it cites use benchmarks or narrow exam sets, on models from 2023 to 2025. Where one tested a trained model against a strong general model on the same test, the trained model won some comparisons and lost others. Its wins came mostly on tasks of the kind it was trained on, against general models that were simply prompted or older than the trained model. The paper does not review other agent-training studies, later general models, or other methods that learn from past runs. That work may change these findings.

The paper argues that a firm should first measure the best no-training alternative that fits its data-residency and control limits. It should train only where that alternative falls short on quality, cost or speed.

## 1. What to Train, and What to Give It to Read

The paper argues that an agent's facts and its way of working belong in different places. Take an agent that matches supplier invoices to purchase orders. It needs facts: suppliers, approval limits, the current order. It also needs a way of working: which document to check first, when to escalate, and how a posting is made.

Facts change, so the paper argues that the agent should read them as it works, which the paper calls grounding.

In the paper's view, the way of working is harder to state fully in a prompt than facts are. On HotpotQA, a model fine-tuned on records beat the same model shown a few records in its prompt [1]. That test does not separate facts from the way of working. The fine-tuned model trailed GPT-4 (2023), prompted to reason.

One study supports grounding facts, within its setup. Retrieval beat training on raw text in nearly all of Ovadia and colleagues' comparisons [2]. The study tested three 7-billion-parameter models on multiple-choice questions, with Wikipedia as the knowledge source and no agent task. Training on raw text improved on the untrained model in most cases.

The paper takes the domain's boundary from domain-driven design: the bounded context [3]. Within it, one domain model and one language hold. The paper argues that one domain-specific agent should serve one bounded context and escalate whatever falls outside it. No cited study tests this.

In the paper's view, its approach fits rule-bound, transactional work, such as matching invoices, but not loosely structured work.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0 

## 2. What a Record of Work Done Well Is

A record is one complete run of the task: its inputs, each step, each tool call and its result, and the end state.

Evans groups objects into aggregates, each reached only through its root [3]. In the paper's reasoning, each action should act on one aggregate through its root. Posting an invoice against an order becomes two actions, one on each aggregate. A rule spanning both is an invariant of neither. The program check (below) tests that rule across the run.

Evans also recommends defining invariants for each aggregate [3]. Here, an order's invoiced amount must not exceed its approval. In UML, a protocol state machine gives the legal event sequences for an object or interface, and can carry state invariants [4]. Pairing Evans's aggregate invariants with it is the paper's own reasoning. Before each action takes effect, the program check tests its result against the invariants. It also tests the order of actions against such a machine. It fails a run that breaches an invariant, a spanning rule or the sequence. An invalid step that is refused and then corrected does not fail the run. In the paper's view, the program check has three uses: a run-time guard on each action, a filter on training records and, with a decision check, a possible reward for reinforcement learning.

The paper argues that the program check decides only whether a run stayed valid. Valid is not correct. A run that does nothing, or escalates every case, also passes. In the paper's view, work done well needs a second check, the decision check, that the outcome is correct. A named person usually makes it, on every record or a sample.

In the paper's view, records also hold facts, such as that day's prices and limits. Training could fix them in the weights, where they go stale. The paper argues that each record should show the fact being looked up. It also argues that training should learn only from the agent's own steps, as AgentTuning did [5]. The agent should then learn to look up the limit, not the limit itself. It may still learn facts that its reasoning or tool calls repeat.

## 3. Where Records Come From

Records come from people's logs, from people working cases through the agent's tools, or from model runs. The paper believes that logs, such as the domain events Evans recommends modelling [3], hold the outcome of much of the work, but not the inputs the person read or the steps they took. To become records, logs must be mapped onto the agent's tool calls, with each look-up added. A look-up added afterwards is chosen knowing the outcome. Such records show no search or failed attempts.

No agent study cited trains on people's logs of real work, so that route is the paper's own reasoning.

In FireAct and AgentTuning, models trained on GPT-4's records mostly fell short of GPT-4 [1], [5]. In three studies, a trained model beat, on some tests, the model whose output it trained on: GPT-4 on three held-in tasks [5], Qwen 2.5 with hints (Ibrahim, below) and GPT-4 (Xue, section 5).

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

Ibrahim and colleagues had GPT-4o write hints from the failed runs of Qwen 2.5 (2024) [6]. They reran Qwen 2.5 on the training tasks with retrieved hints. They then trained a student on the successful runs (on WebShop, runs scoring at least 67), with the hints removed. The student was not compared with GPT-4o. At 14 billion parameters, the student succeeded on 91% of the ALFWorld benchmark's household tasks, against 80% untrained. Fine-tuning on the untrained model's own successes reached 85%. The untrained model with retrieved hints reached 82%. At that size, on the WebShop benchmark, both alternatives came within a point of the student's success rate. These are single-run results. The study does not report whether the compared methods had equal training data.

The law may limit training on records holding personal or client data. Under UK guidance, for example, personal information may generally be reused for a new purpose only if that purpose is compatible with the original and has a lawful basis [7]. Privacy information must be updated first, unless an exemption applies. Special category data needs a further condition. This guidance does not mention training. Applying it there is the paper's reading. In the paper's view, training processes all personal data in the records, even if it learns only from the agent's own steps. In the paper's view, what it learns cannot readily be removed from the weights, so personal data should be redacted before records are kept. Firms should also check client contracts and, for model-written records, the provider's terms.

## 4. What Makes a Good Set

The paper argues for three properties.

First, every record passes the program check, and every record or a sample passes the decision check. A sample estimates how many records are wrong. It misses those not sampled. In one 7-billion-parameter test, filtering AgentTuning's records improved results on held-out benchmark tasks [5]. AgentTuning chose its data mix on those same tasks. In the paper's view, its held-out gains may therefore be optimistic.

Second, the set covers the domain's cases. UML use cases capture a system's requirements as behaviours that yield an observable result of value to its actors [4]. The paper argues for listing training cases from the use cases and the protocol state machine. Case proportions should start from real work. Add rare cases and cases the agent must escalate.

Third, the set mixes in general data from outside the domain. In AgentTuning, agent records alone did worse on held-out tasks than a mix with general data, at all three model sizes [5]. At 7 and 13 billion parameters, general data alone did about as well as the mix on held-out tasks. That mix, too, was chosen on those tasks.

## 5. Training on the Records

The paper recommends fine-tuning on the records first. Reinforcement learning can follow where runs can be sandboxed and scored. Corpus Engineering, Part 1, covers how training data shapes a model [8].

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

Reinforcement learning with verifiable rewards gives a reward only when a program verifies the answer correct [9]. Tülu 3 (2024) applied the method to mathematics and precise instruction following, with a penalty for drifting from the starting model. In some runs with a weak penalty, the model satisfied the program without doing the task.

The paper argues that the program check, with a program that checks the decision, could serve as such a reward for agents. On the invariants alone, doing nothing or escalating everything would score well. In the paper's view, a reward from the invariants alone would invite the failure the Tülu 3 study reported. No cited study uses a domain's invariants as the reward.

In WebRL, fine-tuning on WebArena-Lite's training demonstrations took Llama 3.1 8B (2024) from 4.8% to 20.6% success [10]. Simply prompted GPT-4-Turbo (April 2024) scored 17.6%, five fewer successes on the 165 test tasks. No variance is given for GPT-4-Turbo. Reinforcement learning with a learned reward model then took the fine-tuned model to 42.4%. GPT-4o generated WebRL's training tasks, in the web environments the test uses. WebRL also lists scaffolded GPT-4 and GPT-4o at 35.5% and 37.2%. Both came from other papers' runs on the full WebArena benchmark.

In one study, a software maker tested a Japanese-adapted Llama 3.1 70B (2024) on exam questions about its own product [11]. After continued pretraining and fine-tuning on GPT-4's records, the model beat GPT-4 (2023) on two of three exam sets. The two models were scored with the same prompting. Those two sets had 12 and 20 questions. The study used 42 questions and did not test for significance.

## 6. Where It Fails

In the paper's reasoning, the model copies its records' mistakes, and records that show no failed step teach little recovery. In FireAct, a fine-tuned agent coped better than a prompted one when its search tool returned nothing or random results [1]. The authors suggest its GPT-4 records already held recoveries from noisy searches.

Ross, Gordon and Bagnell find that once a model trained to copy demonstrations errs, it may meet situations unlike any it was shown, where its errors can compound [12]. Their experiments used games and handwriting. Carrying the finding to agents is the paper's reasoning.

In one study, a model fine-tuned on answers it did not know learned them slowly. As it learned them, it got more known answers wrong [13]. The study tested one model, PaLM 2-S (2023), on closed-book questions. Stopping training early largely avoided the harm.

In the paper's reasoning, a changed rule means training again, on records that do not yet exist. A prompt can change the same day. So fast-changing rules belong in what the agent reads.

Reading rules did not make an agent reliable on τ-bench. Prompted GPT-4o, reading each domain's policy, succeeded on 61% of retail and 35% of airline tasks [14]. The benchmark tests no trained agent.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

## 7. Telling Whether It Worked

The paper argues for holding out records, chosen by time, case type and entity, so the test resembles future work. In the paper's view, scoring end states needs a sandbox that simulates the tools with the facts of the time.

The paper argues for scoring each run on five measures:

- the end state, checked against the invariants and expected outcome; some benchmarks score it [14], [15];
- the decision, checked as in section 2;
- the steps taken, refused actions included, since end-to-end success alone does not show intermediate progress [15];
- cost and time per correct outcome, measures the paper adds because evaluations often overlook cost [15];
- escalation: whether the agent escalated every case that fell outside its bounded context or that its rules said to escalate, and no other case.

Past work may hold few cases outside the context; add some.

The paper argues for running each case more than once. On τ-bench's retail tasks, GPT-4o averaged over 60% success per run. The chance that it succeeded on a given task in all eight runs was under 25% [14].

The paper argues for comparing three agents: the untrained model, the trained one and the best no-training alternative.

GEPA has a model reflect on past runs and rewrite prompts [16]. With Qwen3 8B (2025), it beat GRPO, a reinforcement learning method, on five of six benchmark tasks, losing one in mathematics. None was an agent choosing its own tool calls. The authors ran GRPO themselves, with 24,000 runs per task. Agent Workflow Memory adds procedures to what an agent reads, drawn from its past runs that a model judged successful [17]. With GPT-4 (2023) on the WebArena benchmark, the agent with the procedures scored 35.5%. The framework's published score without them was 23.5%, 12 points lower. The authors' own run without them scored 15.0%, 20.5 points lower. The agent drew its procedures from its runs on the test tasks, without labels. Neither study reported variance. Wrongly judged runs can yield wrong procedures that lower success [17].

Where a cited study tested a trained model against a strong general model on the same test, the trained model won some comparisons and lost others [1], [5], [10], [11]. Its wins came mostly on tasks of the kind it was trained on, against general models that were simply prompted or older than the trained model. Each comparison used the same benchmark for both models. None of them ran the strongest general model of its time in a purpose-built agent framework on the same test.

On HotpotQA, fine-tuning GPT-3.5 (2023) cut its time per trial to 2.7 seconds, from 9.0 when prompted with examples [1]. Fine-tuned tokens cost eight times as much, but the model used fewer per trial, so each trial cost about 15% less at 2023 prices. At 14 billion parameters, Ibrahim's student also used fewer tokens per episode than the untrained model, with or without hints [6].

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

The paper believes the cited evidence on training is thin, and mixed on whether training gives a better agent. The studies' results may not hold outside the benchmarks' environments. In the paper's reasoning, training's case may rest on cost or speed, not on a better agent. It may also rest on data-residency and control limits that exclude a stronger model. Managing Context in Long-Running Agents, since replaced by the Context Engineering series, covers what a mechanism outside the model can hold [18].

## 8. A Practical Sequence

This order is the paper's own, untested by any cited study.

1. Draw the bounded context, its aggregates, their invariants and their legal event sequences.
2. Write the program and decision checks (section 2), and build the sandbox (section 7).
3. Guard every action with the program check before it takes effect. The guard stops invalid single actions; valid but wrong ones pass.
4. Map past work into records, lawfully (section 3). Hold out two sets (section 7): one to develop every agent on, one for steps 5 and 7. Never adjust data, prompts or settings on results from the second, and refresh it with new work before each repeat. Set floors for the decision and escalation measures, ceilings for cost and time per correct outcome, and name cost or time as the deciding measure.
5. On the second held-out set, measure the best no-training alternative that fits the firm's data-residency and control limits. Give it a written procedure, the same facts and checks, examples from records not held out, prompts and procedures learned from past runs, and as much development effort as the trained agent gets.
6. Train only if the alternative misses a floor or a ceiling. Gather records lawfully (section 3). Include escalations, look-ups and recoveries (section 6), collected or generated on purpose. In the paper's view, training should learn the correction, not the failed step. Keep records passing the checks (section 4). Fine-tune, then consider reinforcement learning.
7. Score all three agents on the second set, on the five measures, each case more than once. Keep the trained agent only if its decisions and escalations reach their floors and its cost and time stay under their ceilings. Of agents meeting every floor and ceiling, it must lead on the deciding measure. Its lead must exceed the variation across runs and cases. Count the costs of wrong outcomes, of escalations and of building, checking and retraining.
8. Repeat steps 5 to 7 when a rule, the untrained model or the alternative changes.

## Conclusion

The argument rests on the paper's own reasoning and on cited evidence it believes is thin, and mixed on whether training gives a better agent. Its advice: measure the best no-training alternative that fits the firm's data-residency and control limits, and train only where it falls short.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

Velocity Edge, the publisher, sells services on this subject. The paper reports no results of the publisher's own. Every empirical result it cites comes from a published study or preprint. On privacy, it argues that the applicable law governs, with UK guidance as the example. It was drafted and checked with Claude models (Opus 5.5 and Fable 5.2, November 2026).

## Glossary

**Agentic prompting** Prompting a model to work as an agent, in steps and with tools. With no agent framework built around it, the model still counts as simply prompted.

**Aggregate** A cluster of domain objects within one boundary, reached through one root, as Evans describes it; in the paper's approach, what one of the agent's actions acts on.

**Bounded context** The boundary within which one domain model and its language hold, as Evans defines it; in the paper's approach, the agent's scope.

**Continued pretraining** More training on raw text.

**Data-residency and control limits** Limits on where data may go and who may hold or run the model.

**Decision check** A check, by a named person or another program, that a run's decision was correct, on every record or a sample.

**Domain-specific agent** An agent built and checked for one domain's work, whatever its model.

**Done well** Of a run: ending in an outcome checked as correct. The program check decides only that a run stayed valid; the decision check is also needed.

**Escalate** Hand a case to a person.

**Fine-tuning** Training on chosen examples of the output wanted.

**Floor** The lowest score the firm will accept on a measure.

**Grounding** Giving the model facts as it works, without changing the model.

**Held-out** Of records or tasks: kept out of training; used to develop agents or to test them.

**Invariant** A rule an aggregate must satisfy after every change.

**Learned reward model** A model trained to score runs.

**Program check** A program that tests a run three ways: each action against the aggregates' invariants; the order of actions against their protocol state machines; and the whole run against rules spanning aggregates.

**Record** One complete run of the task: its inputs, each step, each tool call and its result, and the end state; also called a trajectory.

**Recovery** A refused or failed step, then corrected, within one record.

**Reinforcement learning** Training in which the model acts, its results are scored, and the scores guide the change.

**Retrieval** Fetching documents for the model to read as it works; one way of grounding.

**Scaffolded** Of a model: run inside an agent framework built around it.

**Simply prompted** Of a model: given the task and a basic prompt, in one turn or step by step, with no agent framework built around it.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

**Student model** A model trained on a teacher's runs; the teacher may be another model or the same model given extra help.

**Training** Changing the model's weights, by any method.

**UML** The Unified Modeling Language, a standard notation for describing systems.

**Verifiable reward** A score a program computes from the result, with no model judging it.

**Way of working** The domain's procedure, the form of its outputs, and how the agent uses what it reads.

**Weights** The numbers inside a model that training changes.

## References

[1] B. Chen, C. Shu, E. Shareghi, N. Collier, K. Narasimhan, and S. Yao, "FireAct: Toward language agent fine-tuning." arXiv:2310.05915, 2023. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2310.05915

[2] O. Ovadia, M. Brief, M. Mishaeli, and O. Elisha, "Fine-tuning or retrieval? Comparing knowledge injection in LLMs." arXiv:2312.05934; EMNLP 2024, 2024. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2312.05934

[3] E. Evans, "Domain-driven design reference: Definitions and pattern summaries." Domain Language, Inc., CC BY 4.0, 2015. Accessed: Nov. 16, 2026. [Online]. Available: https://domainlanguage.com/wp-content/uploads/2016/05/DDD_Reference_2015-03.pdf

[4] Object Management Group, "OMG unified modeling language (OMG UML), version 2.5.1." formal/2017-12-05, Dec. 2017. Accessed: Nov. 16, 2026. [Online]. Available: https://www.omg.org/spec/UML/2.5.1

[5] A. Zeng *et al.*, "AgentTuning: Enabling generalized agent abilities for LLMs." arXiv:2310.12823; Findings of ACL 2024, 2024. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2310.12823

[6] H. Ibrahim, N. Rozanov, and M. Rei, "Fine-tuning with RAG for improving LLM learning of new skills." arXiv:2510.01375, 2025. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2510.01375

[7] Information Commissioner's Office, "What does data protection law say about the compatible reuse of personal information?" Compatibility and the reuse of personal information, UK GDPR guidance and resources; first read in the Internet Archive snapshot of 27 March 2026, the live page re-read on 20 November 2026, 2026. Accessed: Nov. 20, 2026. [Online]. Available: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/compatibility-and-the-reuse-of-personal-information/what-does-data-protection-law-say-about-the-compatible-reuse-of-personal-information/

[8] P. Pienaar, "Corpus engineering, part 1: What an AI is trained on, and why it matters." Velocity Edge, 2026. Accessed: Nov. 16, 2026. [Online]. Available: https://velocityedge.io/papers/corpus-engineering

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0 

[9] N. Lambert *et al.*, "Tülu 3: Pushing frontiers in open language model post-training." arXiv:2411.15124, 2024. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2411.15124

[10] Z. Qi *et al.*, "WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning." arXiv:2411.02337; ICLR 2025, 2025. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2411.02337

[11] Y. Xue, M. Tsunokake, Y. Koreeda, E. M. Amin, T. Sumiyoshi, and Y. Sogawa, "Agent fine-tuning through distillation for domain-specific LLMs in microdomains." arXiv:2510.00482, 2025. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2510.00482

[12] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning." arXiv:1011.0686; AISTATS 2011, 2011. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/1011.0686

[13] Z. Gekhman *et al.*, "Does fine-tuning LLMs on new knowledge encourage hallucinations?" arXiv:2405.05904; EMNLP 2024, 2024. Accessed: Nov. 18, 2026. [Online]. Available: https://arxiv.org/abs/2405.05904

[14] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan, "τ-bench: A benchmark for tool-agent-user interaction in real-world domains." arXiv:2406.12045, 2024. Accessed: Nov. 18, 2026. [Online]. Available: https://arxiv.org/abs/2406.12045

[15] A. Yehudai *et al.*, "Survey on evaluation of LLM-based agents." arXiv:2503.16416, 2025. Accessed: Nov. 16, 2026. [Online]. Available: https://arxiv.org/abs/2503.16416

[16] L. A. Agrawal *et al.*, "GEPA: Reflective prompt evolution can outperform reinforcement learning." arXiv:2507.19457v2; ICLR 2026, 2026. Accessed: Nov. 20, 2026. [Online]. Available: https://arxiv.org/abs/2507.19457

[17] Z. Z. Wang, J. Mao, D. Fried, and G. Neubig, "Agent workflow memory." arXiv:2409.07429v1, 2024. Accessed: Nov. 20, 2026. [Online]. Available: https://arxiv.org/abs/2409.07429

[18] P. Pienaar, "Managing context in long-running agents: A configuration model, a criterion for what can be held, and a reference design." Velocity Edge, 2026. Accessed: Nov. 16, 2026. [Online]. Available: https://velocityedge.io/papers/managing-context

## Revision History

| Version | Date | Change |
|---|---|---|
| 1.0.0 | 2026-11-19 | First version |
| 1.1.0 | 2026-11-21 | Adds where the method fits, the three uses of the checks, recovery records, redacting personal data, and no-training alternatives that learn from past runs; reads the evidence on quality as mixed, with training's case on cost, speed or control as the paper's reasoning; drops the property on documents that do not help and the continued-pretraining result, among other cuts, to stay within 3,000 words |

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

## Licence and Disclaimer

© 2026 Darkwave Trading Limited, trading as Velocity Edge. This paper is licensed under the Creative Commons Attribution 4.0 International licence (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/): you may share and adapt it for any purpose, provided you credit Phillip Pienaar and Velocity Edge, keep this copyright notice, link to the licence, link to the original where reasonably practicable, and indicate any changes. The Darkwave and Velocity Edge names and logos are not covered by this licence.

This paper is for education and information. It is not investment, legal or regulatory advice.

The publisher has not yet run the sequence in Section 8 end to end.