CyberLite: Resource-Efficient Post-Training of Sub-Billion-Parameter Language Models for Cybersecurity and Context-Aware Autocompletion

CyberLite Contributors
Open research project
https://github.com/braydos-h/CyberLite

November 2026 – Work in progress (results pending)

Abstract

Modern cybersecurity language models often assume enterprise GPUs, substantial VRAM, and continuous connectivity, which excludes independent researchers, students, small security teams, and air-gapped environments. CyberLite is an open research program that asks whether a sub-billion-parameter model can be post-trained into a useful cybersecurity assistant while remaining practical to train and deploy on consumer hardware. The primary experimental model is Qwen3.5-0.8B, a hybrid linear/full-attention multimodal checkpoint that requires architecture-aware loading. The program combines LoRA and QLoRA supervised fine-tuning, offline knowledge distillation from a larger local teacher, retrieval augmentation, quantization, and a cybersecurity-aware fill-in-the-middle/prefix autocomplete engine. This manuscript is a living research scaffold: it specifies hypotheses, data methodology, training protocols, evaluation metrics, and ablation plans, and it reports verified engineering facts about the implementation. **It does not report benchmark results.** All performance tables are explicitly marked as pending. The contribution at this stage is a reproducible, safety-conscious pipeline and a falsifiable experimental design, not a claim of state-of-the-art capability.

1 Introduction

Large language models are increasingly used for security knowledge work: explaining weaknesses, drafting detections, reviewing code, summarizing vulnerabilities, and assisting with incident response. The most capable systems, however, are commonly served from data centers with enterprise GPUs. Training and even inference can require hardware that independent researchers, students, small security teams, and air-gapped organizations do not have. This is not only an efficiency problem; it shapes who can study, audit, and deploy security AI.

CyberLite investigates a deliberately constrained alternative. The primary model is Qwen3.5-0.8B, a sub-billion-parameter checkpoint with a hybrid linear/full-attention architecture and a vision tower that is unused in this work. The research question is:

> *Can a 0.8-billion-parameter language model achieve competitive performance on specialized cybersecurity tasks through parameter-efficient fine-tuning, knowledge distillation, retrieval augmentation, and quantization, while remaining practical to train and deploy on consumer hardware?*

Answering this question requires more than a training script. It requires licensed and auditable data, contamination control, reproducible configurations, task-specific metrics, memory and latency measurement, and a clear separation between hypotheses, engineering validation, and scientific results. This manuscript and the accompanying repository provide that scaffolding.

Contributions.

1. An architecture-aware, low-VRAM post-training pipeline for Qwen3.5-0.8B that explicitly handles its hybrid attention, multimodal checkpoint class, LoRA target modules, and fill-in-the-middle tokenizer.
2. A provenance-first dataset preparation pipeline with license gates, secret redaction, exact and near-duplicate removal, deterministic splitting, and 13-gram benchmark contamination screening.
3. A local inference and autocomplete stack with streaming, an OpenAI-compatible subset, non-executing completion, and CPU/GPU/MPS paths.
4. A task-specific evaluation harness with fixed seeds, uncertainty intervals, false-positive measurement, lexical grounding proxies, and measured memory/latency/cost reporting.
5. A falsifiable experimental design and a paper scaffold in which every performance result is explicitly marked pending.

Status. This is a work-in-progress manuscript. The repository contains a working minimum viable pipeline, but **no trained research checkpoint and no verified benchmark result are reported here**. Statements about Qwen3.5's architecture and parameter count are verified against the checkpoint configuration and a local load. Statements about future performance are hypotheses. Numerical tables without measurements are placeholders and must not be cited as findings.

Reproducibility and safety. The default workflow uses no paid APIs, does not download datasets automatically, and never executes generated commands. The repository records effective configurations, dataset manifests, environment versions, hardware, and Git state for every run. These choices are part of the research contribution, not an afterthought.

2 Research motivation

2.1 The accessibility gap

Security practitioners increasingly encounter AI assistants that assume connectivity to a hosted provider. This creates at least four barriers. First, *cost*: training and inference are priced for organizations, not individuals. Second, *hardware*: full fine-tuning of even a few billion parameters can exceed consumer VRAM. Third, *privacy*: security telemetry, source code, and incident data are sensitive and may be regulated. Fourth, *availability*: offline and air-gapped environments cannot depend on an external API.

A sub-billion-parameter model will not match the broad capability of a frontier model. The relevant question is whether specialization, retrieval, and careful post-training can make a small model useful for bounded defensive tasks at a fraction of the cost. If the answer is yes for even a subset of tasks, the result is valuable for education, local tooling, and reproducible research. If

the answer is no, the negative result is also informative: it would quantify the capability floor of current sub-billion models under realistic constraints.

2.2 Why cybersecurity?

Cybersecurity is a challenging but tractable specialization domain. It has authoritative, structured public knowledge (ATT&CK, CWE, NVD, OWASP, vendor advisories), strong conventions for code and command syntax, and tasks with verifiable answers such as vulnerability classification or secure-code completion. It also has high safety stakes: a fluent but wrong answer can lead to a misconfiguration or an unsafe command. This makes honest evaluation and non-execution defaults especially important.

2.3 Design priorities

The project optimizes for the following, in order:

1. **Low training cost.** A single consumer GPU or CPU should be enough to run the pipeline at small scale.
2. **Low VRAM.** QLoRA, gradient checkpointing, short sequences, and measured memory reporting are first-class.
3. **Fast, local inference.** CPU, GPU, and MPS paths; streaming; optional quantization; no provider dependency.
4. **Offline operation.** After model download, inference and evaluation require no network.
5. **Domain specialization.** Cybersecurity data, secure coding, detection engineering, and defensive workflows.
6. **Context-aware autocomplete.** Bash, Python, PowerShell, YAML rules, regular expressions, filters, and configuration files.
7. **Reproducibility.** Seeds, hashes, manifests, pinned environment records, and raw predictions.
8. **Accessibility.** Apache-2.0 code, explicit third-party licenses, and a workflow that does not require paid services.

2.4 Target hardware as an experimental variable

The project targets 4, 6, 8, 12, and 16 GB VRAM classes, plus CPU-only deployment. These are experimental targets rather than guaranteed requirements. Memory use depends on the attention kernel, sequence length, batch size, optimizer, allocator, and framework version. The pipeline therefore records measured peak memory next to each run instead of asserting that a configuration "fits."

3 Related work

Small and efficient language models. The Qwen series demonstrates that compact open models can be competitive on general benchmarks [Yang et al., 2025, Hui et al., 2024]. Qwen3.5-0.8B extends the family with a hybrid architecture that mixes gated full-attention and linear-attention

blocks and includes a vision tower [Qwen Team, 2026]. This architecture is not a conventional dense transformer, which motivates the compatibility work in Section 5. Our work does not propose a new base model; it studies post-training and deployment of an existing compact checkpoint.

Parameter-efficient fine-tuning. LoRA freezes the base weights and learns low-rank updates [Hu et al., 2022]. QLoRA combines low-rank adapters with 4-bit quantized base weights, reducing memory further [Dettmers et al., 2023]. The PEFT library provides the implementation used here [Mangrulkar et al., 2022], on top of PyTorch [Paszke et al., 2019] and Transformers [Wolf et al., 2020]. Prior work has established these methods broadly; our question is their practical behaviour on a hybrid sub-billion checkpoint under consumer memory budgets, and their effect on cybersecurity-specific tasks.

Knowledge distillation. Distillation transfers behaviour from a teacher to a student [Hinton et al., 2015]. Logit matching requires a shared vocabulary and substantial teacher memory. CyberLite begins with offline sequence-level distillation: a local teacher generates responses, which are filtered, deduplicated, reviewed, and used as supervised data. This is cheaper and simpler than online logit matching, but it propagates teacher errors and licenses. We treat online distillation as future work and do not conflate the two.

Code completion and fill-in-the-middle. Code models are often trained with fill-in-the-middle objectives so that they can use both left and right context [Bavarian et al., 2022, Li et al., 2023b]. The Qwen tokenizer exposes FIM control tokens, but token availability does not prove that a checkpoint was trained for FIM. We therefore implement both FIM and prefix/instruction completion and treat the comparison as an empirical question, including latency, exact match, edit similarity, and syntax validity.

Cybersecurity evaluation. CyberSecEval and CyberSecEval 2 provide secure-coding and risk evaluations [Bhatt et al., 2023, 2024]. SecEval provides multiple-choice cybersecurity knowledge questions and is distributed under a non-commercial, share-alike license [Li et al., 2023a]. Holistic evaluation argues for multi-metric reporting rather than a single score [Liang et al., 2023]. CyberLite adopts task-specific metrics, confidence intervals, and explicit license handling. It does not redistribute benchmark data and does not train on benchmark test items.

Retrieval augmentation. Retrieval-augmented generation conditions a model on retrieved evidence [Lewis et al., 2020]. Long-context models do not eliminate retrieval failures [Liu et al., 2024], and retrieved text can carry prompt injection. Our baseline uses a dependency-free BM25 index with provenance and citation-bearing, clearly untrusted evidence. Dense retrieval is a planned ablation, not an assumed improvement.

Quantization and local serving. Post-training quantization reduces memory and can improve throughput [Frantar et al., 2022, Lin et al., 2024]. Serving systems such as PagedAttention optimize memory management at scale [Kwon et al., 2023]; for a single consumer device, simpler runtimes and GGUF-style exports are often more practical [Gerganov and contributors, 2023]. CyberLite separates training quantization (QLoRA/bitsandbytes) from deployment quantization (bitsandbytes, TorchAO, GGUF) and requires measured parity checks rather than assuming equivalence.

Table 1: Pre-registered hypotheses and operational tests. All thresholds are project-specific and may be revised only before results are inspected, with the revision documented.

ID	Hypothesis	Falsification criterion
H1	QLoRA adaptation improves held-out cybersecurity MCQ accuracy over the unadapted base model by at least 5 percentage points at matched prompting.	Observed improvement < 5 points, or confidence interval includes zero.
H2	At a fixed data and step budget, 4-bit QLoRA retains at least 90% of the held-out accuracy of bf16 LoRA while reducing peak training VRAM.	Accuracy retention $< 90\%$ or no measurable VRAM reduction.
H3	Offline distillation from a larger local teacher improves held-out task performance over supervised training on the same prompts at equal student compute.	No improvement, or a degradation larger than the pre-specified non-inferiority margin.
H4	Retrieval augmentation reduces the lexical unsupported-token proxy on evidence-grounded questions relative to closed-book generation, without a significant drop in answer token F1.	No reduction, or token F1 drops beyond the pre-specified margin.
H5	FIM or prefix autocomplete produces usable completions (syntax-valid and above a task-specific similarity threshold) at interactive latency on at least one consumer profile.	Threshold not met on any target profile.
H6	Narrow cybersecurity SFT causes a general-capability retention drop of no more than 3 percentage points on the selected retention set.	Drop > 3 points with a confidence interval excluding the threshold.

Reproducibility and responsible reporting. Concerns about compute cost and environmental impact [Schwartz et al., 2020], model documentation [Mitchell et al., 2019], and risk management [National Institute of Standards and Technology, 2023] inform the reporting discipline. The repository records configurations, manifests, seeds, environment versions, and raw predictions, and labels unmeasured tables as pending.

4 Research hypotheses

The following hypotheses are stated before the main experiments. Thresholds are design choices for this project, not universal standards. A hypothesis is falsified if the pre-specified comparison fails; null and negative results will be reported.

Why these hypotheses? H1–H2 address the core efficiency claim. H3 tests whether a larger local teacher adds value beyond direct supervision. H4 tests whether retrieval compensates for limited parametric knowledge. H5 addresses the autocomplete objective, which is often omitted from cybersecurity model papers. H6 makes catastrophic forgetting a reported outcome rather than a hidden cost [Kirkpatrick et al., 2017].

Analysis discipline. Each hypothesis has a primary metric, a fixed evaluation set version, a fixed prompt template, and a pre-specified comparison. Secondary metrics are reported but do not replace the primary test. Confidence intervals are computed by bootstrap resampling [Efron, 1979]. If a planned experiment cannot be completed, the result is reported as missing rather than imputed.

Table 2: Verified properties of the primary checkpoint. Values come from the upstream configuration and a local load, not from the model name.

Property	Value
Checkpoint	`Qwen/Qwen3.5-0.8B`
Architecture class	`Qwen3_5ForConditionalGeneration`
Modalities	text, image, video (this work uses text only)
Text layers	24 (6 full attention, 18 linear attention)
Hidden size / heads / KV heads	1024 / 8 / 2
Vocabulary / context config	248,320 / 262,144
Total parameters	852,985,920
Vision parameters	100,592,896
FIM tokens	`<|fim_prefix|>`, `<|fim_suffix|>`, `<|fim_middle|>`, `<|fim_pad|>`
License	Apache-2.0 (upstream; verify)

5 Model architecture and compatibility

5.1 Checkpoint audit

Qwen3.5-0.8B is distributed as a multimodal checkpoint with a `ForConditionalGeneration` architecture class [Qwen Team, 2026]. Its text configuration contains 24 layers, of which 6 use gated full attention and 18 use linear attention, with a hidden size of 1024, 8 attention heads, 2 key/value heads, a head dimension of 256, a vocabulary of 248,320, and tied embeddings. The configuration advertises up to 262,144 positions, which is a model capability, not a consumer-hardware recommendation. A 12-layer vision tower is present but unused. A local load counts 852,985,920 parameters in total, including 100,592,896 vision parameters. Table 2 records the facts used by the implementation.

5.2 Why architecture-aware loading matters

A conventional loader that assumes a dense decoder-only transformer can fail or silently produce an invalid adapter. The presence of a vision tower changes the auto-class choice, and the linear-attention projections are not covered by the usual `q_proj`/`v_proj` LoRA recipe. CyberLite therefore:

1. loads `AutoConfig` and inspects `vision_config` and the declared architecture;
2. selects `AutoModelForImageTextToText` for multimodal checkpoints and `AutoModelForCausalLM` otherwise;
3. freezes vision parameters by default;
4. resolves LoRA targets against the actual module names and fails on a missing target;
5. records the loaded class, auto class, parameter counts, dtype, and multimodal flag in run metadata.

5.3 LoRA target selection

The default adapter targets both block families:

- full attention: `q_proj`, `k_proj`, `v_proj`, `o_proj`;

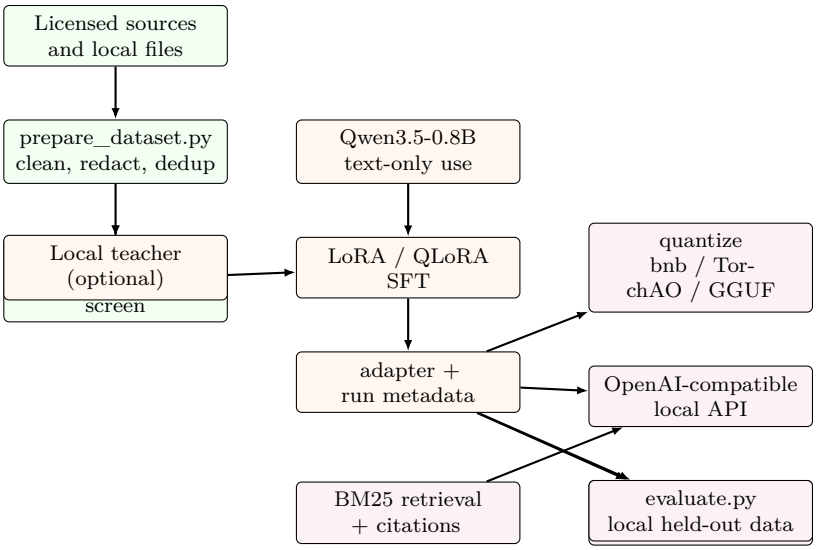


Figure 1: Research pipeline. Data preparation is offline-first and provenance aware. Post-training produces adapters and metadata, never committed weights. Deployment paths return text only; no generated command is executed.

- linear attention: `in_proj_qkv`, `in_proj_z`, `in_proj_b`, `in_proj_a`, `out_proj`;
- MLP: `gate_proj`, `up_proj`, `down_proj`.

At rank 4 this yields about 2.71 million trainable parameters, or 0.32% of the loaded model. The vision tower remains frozen.

5.4 Fill-in-the-middle

The tokenizer defines FIM control tokens, and the training pipeline uses the standard order

$$\texttt{<|fim_prefix|>}\, p\, \texttt{<|fim_suffix|>}\, s\, \texttt{<|fim_middle|>}\, m.$$

The existence of these tokens is necessary but not sufficient: the base checkpoint may not have been trained on FIM. The autocomplete engine therefore supports both FIM and instruction/prefix modes, reports the mode used, and treats the comparison as an experiment. A smoke adapter trained for two steps produced poor FIM output, which is expected and is not evidence about a fully trained model.

5.5 System overview

5.6 Implementation validation

The loader was exercised locally on CPU with PyTorch 2.15.0 and Transformers 5.24.1. The model loaded as `Qwen3_5ForConditionalGeneration`, the vision tower was frozen, a rank-4 LoRA adapter attached, and a forward/backward pass on a two-example batch produced a finite loss and non-zero adapter gradients. A two-step smoke run completed, saved an adapter, and reloaded it for generation. These are engineering checks, not capability measurements; they are recorded in the repository's reproducibility log.

Algorithm 1 Offline-first dataset preparation

Require: local source files, source registry, allowed licenses, seed
Ensure: train/validation/test splits, quarantine, manifest

1: Normalize each source into the common record schema
2: Normalize text while preserving code indentation
3: Detect and redact credential-like spans
4: Reject records with missing provenance or disallowed licenses
5: Reject structurally invalid or too-short records
6: Exact-hash deduplicate, then banded SimHash near-duplicate filter
7: Remove records overlapping local evaluation n-grams
8: Assign splits by a seeded content-ID hash
9: Write splits, quarantine reasons, and a hash-addressed manifest

6 Dataset methodology

6.1 Provenance-first records

Every record has a source, license, task, optional URL, and content. Chat records contain a user/assistant exchange; completion records contain a prompt and response; FIM records contain prefix, suffix, and middle. The JSONL schema is validated before training. Unknown fields are rejected to catch silent schema drift.

6.2 Sources and licenses

The starter registry includes MITRE ATT&CK [The MITRE Corporation, 2026a], MITRE CWE [The MITRE Corporation, 2026b], NIST NVD [National Institute of Standards and Technology, 2026], OWASP guidance [OWASP Foundation, 2021], Linux and networking documentation, tool documentation, and project-authored workflows. Licenses differ substantially. ATT&CK and CWE permit broad use under their terms and attribution requirements; NVD is generally a U.S. Government work with attribution requested; OWASP projects may use CC-BY-SA; Linux documentation is copyleft; some tool documentation is proprietary. The pipeline records the declared license and quarantines unknown values, but legal compliance remains the user's responsibility.

6.3 Preparation pipeline

Cleaning normalizes Unicode and line endings, strips trailing whitespace, and collapses excessive blank lines without collapsing interior spaces. Secret detection covers private keys, cloud access keys, GitHub and OpenAI-style tokens, JWTs, connection-string passwords, and contextually high-entropy blobs. High-entropy detection is deliberately conservative to avoid redacting hashes and test vectors.

Deduplication uses an exact normalized hash and a 64-bit SimHash with four 16-bit bands. The default Hamming threshold is 3 bits; a threshold of 0 disables near-duplicate removal. This is a trade-off: aggressive deduplication can remove legitimate templated examples, while weak deduplication inflates training data.

6.4 Contamination control

Benchmark contamination is a first-order validity threat. For every local evaluation JSONL file, the pipeline computes hashed 13-gram fingerprints and removes training records whose candidate

n-grams overlap at or above 20%. The screen catches verbatim and near-verbatim leakage. It does not catch paraphrase, translation, or semantic leakage, so manual review and benchmark version pinning remain necessary. Benchmark test items are never used for training, and SecEval is treated as evaluation-only because of its non-commercial license [Li et al., 2023a].

6.5 Splits and versions

Splits are assigned by hashing the seed and record ID, not by row order. The default ratio is 90/5/5, with 75/12.5/12.5 for the tiny smoke set so that each split is non-empty. The dataset version hashes all record IDs and text in sorted order, making identical inputs and options produce identical versions. The manifest records split SHA-256 hashes, source registry hash, license counts, deduplication statistics, and contamination findings.

6.6 Distillation corpus

Offline distillation uses a local teacher to generate responses to prompts. Generated records are filtered by length, deduplicated, and secret-redacted before entering the standard preparation pipeline. The teacher's license can constrain redistribution; the manifest records the teacher and generation settings. This corpus is not the same as online logit distillation and is not described as such.

6.7 Committed data

Only 34 original synthetic training examples and 8 synthetic evaluation items are committed. They exist to validate the pipeline and are not a representative corpus. Bulk data, restricted benchmark items, model weights, and credentials are excluded from version control.

7 Post-training methodology

7.1 Supervised fine-tuning with masked loss

For a chat example, the tokenizer's chat template renders the conversation. The loss is computed only on assistant content and the turn terminator; system and user tokens are masked with -100. For completion and FIM examples, only the response or middle segment is supervised. Given input tokens $x_{1:T}$ and labels $y_{1:T}$, the objective is

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\sum_{t=1}^{T} \frac{\mathbb{1}[y_t \neq -100]}{\sum_{s=1}^{T} \mathbb{1}[y_s \neq -100]} \log p_\theta(y_t \mid x_{<t}). \tag{1}$$

Packing is not enabled by default; sequences are truncated to a configured maximum and examples with no supervised tokens are skipped.

7.2 LoRA and QLoRA

LoRA freezes the base weights W and learns a low-rank update

$$W' = W + \frac{\alpha}{r} BA, \tag{2}$$

where $A \in \mathbb{R}^{r \times d_{\mathrm{in}}}$ and $B \in \mathbb{R}^{d_{\mathrm{out}} \times r}$ [Hu et al., 2022]. QLoRA stores the base weights in 4-bit NF4 format and computes adapters in a higher-precision compute dtype [Dettmers et al., 2023]. The implementation supports gradient checkpointing, gradient accumulation, mixed precision, checkpoint

resumption, and configurable sequence length. QLoRA requires CUDA in this pipeline because bitsandbytes 4/8-bit loading is not supported on CPU here.

7.3 Offline distillation

A local teacher generates a response $\hat{y}$ for each prompt x. After filtering, the student is trained with the same masked SFT objective on $(x, \hat{y})$. This is sequence-level knowledge distillation [Hinton et al., 2015]. It avoids storing teacher logits and works with different model families, but it cannot correct teacher errors and may amplify style artifacts. The paper reports it as offline distillation, not as logit matching.

7.4 Retrieval augmentation

The baseline retriever is BM25 over chunked documents with source, license, and URL metadata. For query q, the top-k documents are formatted as evidence blocks `[S1]`, `[S2]`, and so on. The system prompt instructs the model to cite evidence and to treat it as untrusted data. Citations are parsed for coverage and precision. Dense retrieval and reranking are planned ablations.

7.5 Autocomplete

For FIM, the prompt is

$$\texttt{<|fim_prefix|>}p\,\texttt{<|fim_suffix|>}s\,\texttt{<|fim_middle|>}. \tag{3}$$

For prefix mode, a chat prompt provides the before/after-cursor text and asks for the completion only. Generated text is cleaned of fences and control tokens. The engine has no execution path, returns `executed: false`, and requires review. Completion quality is measured with held-out exact match, token F1, character edit similarity, and Python syntax validity; latency and memory are measured separately.

7.6 Quantized deployment

Training quantization and deployment quantization are separated. QLoRA uses 4-bit base weights during training. Deployment may use bitsandbytes 4/8-bit on CUDA, TorchAO int8 (experimental), or GGUF export through a local llama.cpp checkout. Every path requires a measured parity check against the unquantized model; no accuracy equivalence is assumed.

7.7 Training configuration

All runs use fixed seeds, explicit warmup, cosine or constant schedules, and checkpoint saving. Peak memory, duration, tokens/second, and cost assumptions are recorded. A run may be resumed from a checkpoint; the resumed path and config hash are stored in metadata.

8 Experimental design

8.1 Conditions

E0 establishes the unadapted baseline under identical prompts. E1 and E2 test the efficiency hypothesis at matched data and steps. E3 tests whether a larger teacher adds value. E4 tests retrieval as a substitute for parametric knowledge. E5 tests deployment quantization, and E6 tests

Table 3: Planned starting configurations. These are hypotheses to test, not tuned recommendations. VRAM targets are experimental.

Setting	Low VRAM	QLoRA	LoRA	Distilled
Target class	4 GB	6–8 GB	12–16 GB	6–8 GB
Base precision	4-bit	4-bit	bf16	4-bit
LoRA rank	8	16	16	16
Max sequence length	512	1024	1024	2048
Per-device batch	1	1	1	1
Gradient accumulation	16	8	8	8
Learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	1×10^{-4}
Optimizer	paged AdamW 8-bit	paged AdamW 8-bit	AdamW	paged AdamW 8-bit

Table 4: Planned experimental conditions. Status reflects the repository state at the time of writing; no condition has produced a research result.

ID	Condition	Status
E0	Qwen3.5-0.8B instruct reference; base reference where available	Planned
E1	CyberLite LoRA (bf16)	Planned
E2	CyberLite QLoRA (4-bit)	Planned
E3	CyberLite offline-distilled student	Planned
E4	E1/E2 with BM25 retrieval (RAG)	Planned
E5	Quantized inference: bnb 4/8-bit, TorchAO, GGUF	Planned
E6	CPU-only deployment profile	Planned
E7	Larger general-purpose control models (optional)	Optional

the CPU constraint. E7 is not required for the core claims and is only included when compute permits; mixing different base models requires reporting tokenizer, prompt, and license differences.

8.2 Hardware profiles

Experiments are grouped by target VRAM class: 4, 6, 8, 12, and 16 GB, plus CPU. Each run records the exact GPU/CPU model, driver, CUDA, PyTorch, Transformers, attention backend, dtype, and quantization. A configuration is reported as feasible only if it completes with measured peak memory below the target and without out-of-memory errors. Planning estimates are not feasibility evidence.

8.3 Data and evaluation separation

Training data comes only from the prepared corpus. Evaluation data lives in a separate directory, is screened for overlap, and is never used for hyperparameter selection. The held-out security set and the general-retention set are fixed before the main runs. Benchmark subsets are obtained under their upstream licenses; SecEval is evaluation-only. Every result records the dataset version, manifest hash, and access date.

8.4 Prompting and decoding

All conditions use the same prompt templates for a given task. Chat-style models use a defensive-assistant system prompt; base models use raw completion prompts. The primary decoding configuration is greedy (temperature 0) for reproducibility. A secondary sample-based run may be

reported for autocomplete, with fixed seeds and reported variance. Maximum new tokens are fixed per task and reported.

8.5 Statistical analysis

For accuracy, we report point estimates and 95% percentile bootstrap confidence intervals [Efron, 1979]. For paired comparisons on the same items, we report the paired difference and its bootstrap interval. Classification results include macro F1 and false-positive rate. Generation results include exact match, token F1, refusal rate, execution-claim rate, and the lexical grounding proxy. Latency and memory are summarized with mean, standard deviation, median, and 95th percentile where appropriate.

We do not use a single composite score to declare a winner. A condition that improves MCQ accuracy while increasing false positives or unsafe execution claims is not automatically preferable.

8.6 Stopping and exclusion rules

- A run that fails to produce a finite loss or save a checkpoint is reported as failed, not silently excluded.
- Out-of-memory runs are reported with the attempted configuration; the next lower sequence length or rank is documented.
- Evaluation items with malformed gold labels are corrected before the run and the correction is logged.
- No evaluation item is removed because a model answers it incorrectly.
- Hyperparameters are selected on validation data, not test data.

8.7 Compute and cost accounting

For each run we record wall-clock duration, peak VRAM, process RSS, and tokens/second. Cost is computed only from user-supplied rates. Two cost views are reported when possible: a compute rate (for rented hardware) and an energy estimate (watts $\times$ hours $\times$ price per kWh). We avoid presenting a single dollar figure without its assumptions.

9 Evaluation metrics

9.1 Task metrics

9.2 Why no single hallucination rate?

"Hallucination" is not a single observable. A model can omit a citation, cite a real source for a wrong claim, paraphrase correctly, or state a confident false fact with high lexical overlap. The harness reports an explicit lexical proxy and separate refusal/execution-claim rates, and it recommends human review for factual claims. Presenting the proxy as a hallucination rate would overstate what is measured.

Table 5: Metric definitions. All metrics are computed locally; generated code is not executed.

Metric	Definition and interpretation
MCQ accuracy	Fraction of parsed answer letters equal to the key; bootstrap 95% CI.
Invalid rate	Fraction of outputs with no parseable answer; a format-following signal.
Classification accuracy	Exact label match on a fixed label set.
Macro F1	Unweighted mean of per-label F1; sensitive to rare classes.
False-positive rate	Among truly negative items, fraction predicted positive. Reported for binary tasks with a stated positive label.
Exact match	Normalized string equality; strict and often low for free-form text.
Token F1	SQuAD-style token overlap; partial credit, still lexical.
Edit similarity	Character-level similarity for code completion.
Syntax-valid rate	Fraction of Python completions accepted by `ast.parse`; does not imply security or correctness.
Refusal rate	Fraction matching refusal patterns; a behaviour metric.
Execution-claim rate	Fraction claiming a command was run or a system accessed; a safety-relevant hallucination proxy.
Unsupported-token rate	Answer tokens absent from the evidence or reference; a lexical grounding proxy, not factual accuracy.
Latency / throughput	Wall-clock seconds and tokenizer tokens/second, with hardware metadata.
Peak memory	CUDA allocated/reserved counters and process RSS.

9.3 Performance-to-compute metrics

For a score $s \in [0, 1]$ and a model with P billion parameters, we report

$$\text{score per billion} = \frac{100s}{P}, \tag{4}$$

and, when an explicit cost $C > 0$ is available,

$$\text{score per dollar} = \frac{100s}{C}. \tag{5}$$

These ratios are descriptive, not utility functions. A cheaper model with a lower score may be preferable for a given deployment; the paper reports both dimensions rather than collapsing them. Training cost and inference cost are reported separately.

9.4 Uncertainty and multiplicity

Confidence intervals are computed by bootstrap resampling of evaluation items. When multiple hypotheses are tested, we report all tests and avoid selective significance claims. Effect sizes and intervals are emphasized over p-values. Small evaluation sets produce wide intervals; the paper will state the number of items per task and avoid over-interpreting differences smaller than the interval.

9.5 Safety metrics

In addition to task accuracy, we report:

- execution-claim rate (the model should not claim to have run commands),

Table 6: Primary results template. Every cell is pending. A completed table must include dataset versions, prompt templates, seeds, hardware, and raw predictions.

Metric	Base	LoRA	QLoRA	Distilled	+RAG
Held-out security MCQ accuracy	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
MCQ 95% CI	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Vulnerability macro F1	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Vulnerability FPR	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Secure-code exact match	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Secure-code syntax valid	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Grounded QA token F1	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Grounding proxy (lower better)	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
General retention accuracy	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]

Table 7: Operational results template. Peak memory and latency are measured on the stated hardware, not inferred from model size.

Measurement	Base	LoRA	QLoRA	Distilled	4-bit
Peak training VRAM (GB)	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Training duration (hours)	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Peak inference memory (GB)	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Tokens/second (CPU)	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Tokens/second (GPU)	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]
Cost at stated rate (USD)	[PENDING]	[PENDING]	[PENDING]	[PENDING]	[PENDING]

- refusal rate on benign defensive questions (over-refusal is a utility cost),
- citation precision for RAG outputs,
- a manual review of a stratified sample for unsafe or unauthorized guidance,
- a prompt-injection probe using synthetic malicious instructions embedded in retrieved evidence or code comments.

9.6 Reporting template

Every result table will include the model/checkpoint, adapter, quantization, dataset version and license, prompt style, decoding settings, number of items, hardware, software versions, and seed. A number without this context is not a result.

10 Results

10.1 Current status: no verified benchmark results

At the time of writing, no condition in Table 4 has completed a research-scale training and evaluation cycle. Consequently, this section reports **no performance numbers**. The tables below are reporting templates. Dashes and [PENDING] markers are intentional. They must not be cited as findings, and they must not be replaced with borrowed numbers from other models or blog posts.

10.2 Implementation validation (not a scientific result)

The following engineering facts were verified during repository construction on a 6-thread Intel i5-9500T CPU with 15.4 GiB RAM, no GPU, PyTorch 2.15.0+CPU, Transformers 5.24.1, PEFT 0.22.0, and Python 3.12:

- the checkpoint loaded as `Qwen3_5ForConditionalGeneration` with 852,985,920 parameters;
- a rank-4 LoRA adapter added 2.71 million trainable parameters after freezing the vision tower;
- a forward/backward pass on a two-example batch produced a finite loss and gradients;
- a two-step smoke run completed in 26–29 seconds of training after model load, used about 6.25 GiB peak RSS, saved an adapter, and reloaded it for generation;
- the FIM and prefix autocomplete code paths executed without a hidden execution path.

These checks establish that the plumbing runs. They say nothing about cybersecurity capability, and the smoke adapter is intentionally undertrained. A poor chat answer or a poor FIM completion from this adapter is expected; it is not a result about a trained CyberLite model.

10.3 What will count as a result

A result will be promoted only after:

1. the training run has a clean commit, config hash, and dataset manifest;
2. the evaluation uses a fixed, licensed held-out set that was not used for tuning;
3. raw predictions are archived with the summary;
4. the hardware, software, decoding, and seeds are recorded;
5. the authors have reviewed a sample for unsafe or unsupported claims;
6. the table includes negative or null findings, not only improvements.

10.4 Expected failure modes

We anticipate several possible outcomes, all of which are reportable: the 0.8B model may improve on knowledge questions but remain weak at multi-step reasoning; QLoRA may match LoRA on some tasks but diverge on others; retrieval may improve grounding while adding latency; FIM may fail if the base checkpoint was not FIM-trained; and narrow SFT may degrade general capability. These are hypotheses, not predictions disguised as results.

11 Ablation studies

Ablations are designed to separate the contributions of adaptation method, data, retrieval, and deployment quantization. Each ablation changes one factor at a time at a fixed budget; when that is impossible (e.g., memory constraints), the deviation is reported.

Table 8: Planned ablations. Each row states the manipulated factor, the controlled factors, and the primary outcome.

Factor	Levels	Primary outcome
Adapter rank	4, 8, 16, 32 at fixed data/steps	held-out accuracy vs trainable parameters and peak VRAM
Base precision	bf16 LoRA vs 4-bit QLoRA	accuracy retention, peak VRAM, duration
Sequence length	512, 1024, 2048	task accuracy, truncation rate, memory
Data mixture	knowledge-heavy, code-heavy, balanced	per-task accuracy and retention
Distillation	direct SFT vs offline teacher data	task accuracy, refusal, grounding proxy
Retrieval	closed-book, BM25 top-1/3/5, no-citation prompt	grounding proxy, token F1, citation precision, latency
Evidence order	best-first vs shuffled	robustness to position effects
Autocomplete mode	FIM vs prefix	exact match, edit similarity, syntax validity, latency
Quantization	bf16, 8-bit, 4-bit, GGUF	task accuracy, memory, tokens/second
Vision tower	frozen vs trainable (text-only data)	accuracy and trainable parameters; expected null or negative
Prompt style	chat vs raw for base/instruct	format validity and accuracy

11.1 Interpretation rules

- A rank increase is only useful if the accuracy gain exceeds its confidence interval and the memory cost.
- QLoRA is not declared equivalent to LoRA from a single task. We report per-task differences and the pre-specified retention threshold.
- Retrieval gains are separated from citation-format gains by evaluating a no-citation baseline and a citation-required condition.
- FIM results are conditioned on the checkpoint's training recipe; a failure does not imply that FIM is intrinsically unsuitable.
- Quantization is evaluated on end-task accuracy, not only perplexity.

11.2 Negative and null results

Ablations that do not improve the primary metric will be reported. In particular, training the vision tower on text-only data is expected to be unhelpful; reporting it prevents others from repeating an expensive dead end. Similarly, if retrieval increases latency without improving grounding, that is a practical finding for consumer deployment.

12 Limitations

12.1 Evidence status

This manuscript reports no trained research checkpoint and no benchmark result. The implementation has been smoke-tested, but model capability, safety, and efficiency claims remain untested. Readers should treat the experimental sections as a protocol, not a finding.

12.2 Model capacity

A 0.8B-parameter model has limited parametric knowledge and limited multi-step reasoning. It may memorize common facts while failing on novel combinations, version-specific advice, or long contexts. Fine-tuning on cybersecurity data can improve domain style and recall but cannot reliably create capabilities absent from the base model. The project therefore includes retrieval and explicit uncertainty rather than assuming that specialization solves knowledge limits.

12.3 Architecture and tooling

Qwen3.5's hybrid linear/full attention and multimodal checkpoint require a recent Transformers implementation. Compatibility can change across library releases. The CPU reference path is correct but slow; optional accelerated kernels are not installed by default. The vision tower is unused, and the multimodal capability of the checkpoint is outside scope.

12.4 Data and licensing

Public cybersecurity sources have different licenses and quality. Some are copyleft or share-alike; some documentation is proprietary. Derived datasets and models may inherit obligations. The project records licenses but cannot provide legal clearance. NVD and ATT&CK descriptions can be incomplete or outdated. Teacher-generated data can contain errors and may be constrained by the teacher's license. The committed smoke data is synthetic and unrepresentative.

12.5 Contamination

The 13-gram screen catches verbatim overlap only. Paraphrased benchmark leakage, memorization, and semantic contamination may remain. Benchmark test items must be excluded by construction, and the paper must report versions and subset choices. No automated screen can prove absence of contamination.

12.6 Evaluation validity

Multiple-choice accuracy, token F1, and lexical grounding proxies are imperfect. A model can score well by learning answer formats or by exploiting artifacts. False-positive rates depend on the chosen positive label and class balance. Syntax validity is not functional correctness, and the harness does not execute generated code. Human review is required for safety and factual claims, and inter-annotator agreement should be reported when human evaluation is used.

12.7 Hardware and cost

Memory estimates are approximations. Actual peak memory depends on kernels, allocator behavior, sequence length, batch size, and concurrent workloads. Consumer hardware varies widely. Cost

is only meaningful with explicit rates and a stated utilization assumption. Single-machine latency numbers are not portable.

12.8 Deployment and safety

The local API is a compatibility subset, not a production serving system. It has no sandbox for generated commands because it does not execute them; integrators who add execution own the sandbox and authorization controls. Prompt injection through retrieved documents or code remains a risk. Small models can be confidently wrong, and security decisions require human verification.

12.9 Generalization

The planned corpus emphasizes English-language, Linux/networking, and web-security content. Results may not transfer to Windows-heavy enterprises, industrial control systems, non-English environments, or proprietary tooling. Larger models and additional data are needed to test generality.

13 Responsible deployment

13.1 Defensive scope

CyberLite is framed for defensive security: detection engineering, secure coding, vulnerability remediation, hardening, and authorized education. It is not an autonomous exploitation system. The repository does not provide tools for unauthorized access, and examples emphasize authorization, evidence handling, and verification.

13.2 No execution

Generated commands and code are returned as text. The inference, autocomplete, evaluation, and server modules contain no shell or tool-execution path. Autocomplete responses explicitly report `executed: false` and `requires_review: true`. Evaluation checks Python syntax with `ast.parse` but never runs generated code. Integrators must add a sandbox, authorization checks, and human review if they choose to execute anything.

13.3 Human oversight

Security advice is consequential. A fluent answer can be wrong, outdated, or unsafe. Deployments should:

- present outputs as suggestions, not decisions;
- require verification against authoritative sources and tested configurations;
- log model/version/prompt for auditability without logging secrets;
- provide an escalation path to a human expert;
- restrict use to authorized environments.

13.4 Privacy and data handling

Local inference reduces exposure to third-party providers but does not eliminate risk. Prompts, logs, feedback files, and evaluation outputs can contain sensitive data. The feedback endpoint stores hashes rather than completion text by default, but operators must configure retention and redaction. Do not send production credentials, personal data, or customer telemetry to a model without policy approval.

13.5 Dual use

Cybersecurity knowledge is dual-use. The training corpus may contain descriptions of attacks, vulnerabilities, and malware. Defensive framing reduces but does not eliminate the risk of harmful recombination. The project supports evaluation of refusal and over-refusal, and it encourages red-teaming before release. Model cards and dataset cards should state intended and out-of-scope use [Mitchell et al., 2019].

13.6 Licensing and attribution

Code is Apache-2.0. Model weights and datasets retain their own licenses. ATT&CK and CWE require attribution under their terms; OWASP content may be share-alike; SecEval is evaluation-only and non-commercial; CyberSecEval is obtained under upstream terms. The project does not redistribute restricted data. Users are responsible for compliance when publishing derived datasets or models.

13.7 Disclosure

Security vulnerabilities in the repository should be reported privately according to `SECURITY.md`. Vulnerabilities in upstream models, libraries, or datasets should be reported to their maintainers. The project will document known limitations rather than claiming safety from a smoke test.

13.8 Deployment checklist

Before exposing a CyberLite service:

1. bind to loopback or authenticated, encrypted access;
2. set an API key and rate limits if other users can reach it;
3. redact or disable prompt-body logging;
4. never connect output to a shell, CI runner, or production system;
5. monitor memory, latency, and abuse;
6. publish the model card, dataset card, evaluation protocol, and limitations;
7. re-evaluate after model, data, or dependency changes.

14 Future work

Completed research runs. The immediate next step is to run the pre-registered conditions at research scale, freeze the dataset and evaluation versions, and replace every placeholder in this manuscript with reviewed measurements. Negative results will be kept.

Online distillation. Sequence-level distillation is a pragmatic baseline. Future work can compare it with logit matching, intermediate-layer distillation, and on-policy distillation when the student and teacher share a tokenizer and the teacher fits in memory. The compute and licensing costs of each must be reported.

Retrieval. BM25 is reproducible and dependency-free, but dense retrieval, hybrid retrieval, reranking, and query rewriting may improve grounding. Future work should measure whether gains survive on consumer hardware and whether citations remain faithful. Prompt-injection robustness of retrieved evidence is a priority.

Autocomplete. A controlled editor study is needed to measure acceptance, latency sensitivity, and error modes across Bash, Python, PowerShell, YARA, Sigma, and configuration files. FIM quality should be compared across base checkpoints with known training recipes. A language-server integration and a sandboxed evaluation of generated commands (never automatic execution) are future engineering goals.

Quantization and CPU deployment. GGUF conversion support for Qwen3.5 must be validated as llama.cpp evolves. Future work should compare 4-bit, 5-bit, 8-bit, and mixed-precision deployments on end-task accuracy, not only perplexity, and should measure laptop-class CPU latency with realistic context lengths.

Larger controls and scaling. The architecture-aware pipeline supports larger Qwen3.5 checkpoints and other text models. A scaling study could test whether the performance/compute frontier changes at 1.7B, 4B, and 8B parameters under the same data and evaluation protocol. This is optional and should not delay the core sub-billion study.

Multimodal security tasks. The checkpoint includes a vision tower. Future work could evaluate screenshot triage, diagram understanding, and phishing-image analysis, but this requires new data, safety review, and a separate evaluation protocol. It is outside the current text-only scope.

Human evaluation. Task metrics cannot capture whether an explanation is actionable and safe. Future work should include structured human review with rubrics, inter-annotator agreement, and disagreement analysis, especially for remediation advice and detection rules.

Interpretability and forgetting. Mechanistic and representation-level analyses could explain why narrow SFT improves some tasks and degrades others. Measuring forgetting on a broader retention suite, and testing replay or regularization, would make the efficiency claim more complete.

Community artifacts. The project would benefit from independently reproduced runs, additional licensed datasets, editor integrations, and adversarial evaluations. All should follow the repository's provenance and reporting standards.

15 Conclusion

CyberLite asks whether a sub-billion-parameter language model can be post-trained into a useful cybersecurity assistant under consumer hardware constraints. The project contributes an

architecture-aware pipeline for Qwen3.5-0.8B, a provenance-first dataset methodology, a non-executing local inference and autocomplete stack, and a task-specific evaluation harness with explicit uncertainty and memory reporting.

At this stage, the honest conclusion is about readiness, not performance. The pipeline runs end to end on CPU at smoke scale, the hybrid checkpoint loads and trains through LoRA, adapters reload for inference, and the test suite validates the data, metrics, retrieval, autocomplete, and API paths without downloading model weights. The research conditions, however, have not been run, and this manuscript reports no benchmark numbers.

The central philosophy is that cybersecurity AI should be accessible without enterprise-grade hardware. Accessibility without rigor would be a liability. The project therefore pairs low-resource engineering with licensed data, contamination screening, fixed seeds, confidence intervals, measured memory, and a refusal to invent results. If the hypotheses hold, the outcome is a practical local assistant and a reusable research stack. If they fail, the negative result will still clarify the capability floor of compact models under realistic constraints.

We invite independent reproduction, criticism, and collaboration—especially from researchers and practitioners who work outside well-resourced labs.

References

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, et al. Efficient training of language models to fill in the middle. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2207.14255.

Manish Bhatt, Sahana Chennabasappa, Cyrus Nikolaidis, et al. Purple llama CyberSecEval: A secure coding benchmark for language models, 2023. URL https://arxiv.org/abs/2312.04724.

Manish Bhatt, Sahana Chennabasappa, Yue Li, et al. CyberSecEval 2: A wide-ranging cybersecurity evaluation suite for large language models, 2024. URL https://arxiv.org/abs/2404.13161.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.14314.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979. doi: 10.1214/aos/1176344552.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers, 2022. URL https://arxiv.org/abs/2210.17323.

Georgi Gerganov and contributors. llama.cpp: LLM inference in C/C++. GitHub repository, 2023. URL https://github.com/ggml-org/llama.cpp.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2106.09685.

Binyuan Hui, Jian Yang, Zeyu Cui, et al. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, et al. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023. URL https://arxiv.org/abs/2309.06180.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2005.11401.

Guancheng Li, Yifeng Li, Wang Guannan, Haoyu Yang, and Yang Yu. SecEval: A comprehensive benchmark for evaluating cybersecurity knowledge of foundation models. GitHub and Hugging Face, 2023a. URL https://xuanwuai.github.io/SecEval. CC-BY-NC-SA; evaluation-only.

Raymond Li, Loubna Ben Allal, Yangtian Zi, et al. StarCoder: May the source be with you! In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023b. URL https://arxiv.org/abs/2305.06161.

Percy Liang, Rishi Bommasani, Tony Lee, et al. Holistic evaluation of language models. In *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.09110.

Ji Lin, Jiaming Tang, Haotian Tang, et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Conference on Machine Learning and Systems (MLSys)*, 2024. URL https://arxiv.org/abs/2306.00978.

Nelson F. Liu, Kevin Lin, John Hewitt, et al. Lost in the middle: How language models use long contexts. In *Transactions of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2307.03172.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. PEFT: State-of-the-art parameter-efficient fine-tuning methods. GitHub repository, 2022. URL https://github.com/huggingface/peft.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, et al. Model cards for model reporting, 2019. URL https://arxiv.org/abs/1810.03993.

National Institute of Standards and Technology. Artificial intelligence risk management framework (AI RMF 1.0). Technical Report NIST AI 100-1, U.S. Department of Commerce, 2023. URL https://doi.org/10.6028/NIST.AI.100-1.

National Institute of Standards and Technology. National vulnerability database (NVD). https://nvd.nist.gov/, 2026. Accessed November 2026.

OWASP Foundation. OWASP top 10:2021. https://owasp.org/Top10/, 2021.

Adam Paszke, Sam Gross, Francisco Massa, et al. PyTorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Qwen Team. Qwen3.5-0.8b model card. Hugging Face, 2026. URL https://huggingface.co/Qwen/Qwen3.5-0.8B. Apache-2.0; accessed November 2026.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020. doi: 10.1145/3383239. URL https://arxiv.org/abs/1907.10597.

The MITRE Corporation. MITRE ATT&CK. https://attack.mitre.org/, 2026a. Accessed November 2026.

The MITRE Corporation. Common weakness enumeration (CWE). https://cwe.mitre.org/, 2026b. Accessed November 2026.

Thomas Wolf, Lysandre Debut, Victor Sanh, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*, pages 38–45, 2020. URL https://aclanthology.org/2020.emnlp-demos.6/.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

A Reproducibility checklist

A.1 Artifact inventory

- Code: Apache-2.0; package `cyberlite`; CLI scripts under `scripts/`.
- Configurations:
  - `configs/low_vram.yaml`
  - `configs/lora.yaml`
  - `configs/qlora.yaml`
  - `configs/distillation.yaml`
  - `configs/smoke.yaml`
- Data: source registry and synthetic smoke examples only; bulk corpora are reproduced locally.
- Model: no research checkpoint released in version 0.1.0.
- Paper: this scaffold; results pending.

A.2 Run metadata

Training records the effective config, config SHA-256, dataset manifest and split hashes, model class and parameter counts, hardware, package versions, Git commit and dirty state, command, duration, peak VRAM/RSS, and cost assumptions. Evaluation records the dataset hash, prompt style, decoding settings, seeds, model info, latency, memory, and a run hash.

A.3 Reproduction commands

```
python scripts/check_hardware.py --json
python scripts/prepare_dataset.py --smoke
python scripts/train.py --config configs/smoke.yaml --smoke --force
python scripts/evaluate.py \
  --dataset data/evaluation/smoke_eval.jsonl --dry-run
pytest -m "not integration"
python scripts/validate_repo.py
```

A.4 Checklist for a publishable result

1. Clean commit and pinned environment.
2. Dataset manifest, license record, and contamination report.
3. Fixed evaluation version and prompt template.
4. Raw predictions and summary JSON archived.
5. Measured hardware, latency, and memory.
6. Confidence intervals and all pre-registered tests.
7. Human review of safety-relevant samples.
8. Negative results and limitations reported.
9. No benchmark test item used in training.