Edge-Deployable AI/ML Framework for Disconnected Tactical Environments

Zach Kelling — Hanzo Team

Version 1.0 — January 29, 2026 January 2026

Abstract

We present an edge-deployable AI/ML framework designed for disconnected, degraded, and bandwidth-limited tactical environments. The system combines a Rust-native LLM inference engine supporting 15+ text and 16+ vision model architectures with aggressive quantization (2–8 bit) enabling deployment on hardware ranging from Apple Silicon laptops to embedded ARM processors. A family of 18 frontier models (0.8B–1T+ parameters) provides capability tiers from nano-scale edge inference to frontier reasoning. The ZAP zero-copy binary protocol achieves 11.5M transactions per second for AI agent coordination, while GPU-accelerated operations on Metal and CUDA provide real-time performance for cryptographic and neural network workloads. Multi-agent orchestration with 83 specialized agents enables autonomous analysis with 200–300 sequential tool calls. The framework operates fully offline with local model management, sandboxed execution, and no cloud dependencies.

Executive Summary. This is AI that runs where the network does not—on a laptop, a vehicle, or a handheld at the tactical edge, with no link back to the cloud. It is built for forces and field teams operating in disconnected, degraded, or jammed conditions who still need real-time analysis, imagery and document exploitation, and decision support. The same family of models that powers data-center systems is compressed to fit the hardware on hand, so a single device can run capability tiers from a small embedded chip up to a frontier model on a server, choosing the largest model the available memory allows. Because everything runs locally, sensitive data never leaves the device, there is no dependency on a vendor's servers, and operation continues when communications are contested or absent. For program leaders, this means frontier-grade AI delivered as a self-contained, owned capability rather than a subscription to a remote service that fails the moment the link drops.

Contents

1 Introduction

Tactical edge AI presents unique challenges absent from data center deployments: limited power, constrained memory, intermittent or absent connectivity, and the need for real-time decision support under combat conditions. Existing frameworks assume cloud connectivity, unlimited memory, and Python runtimes—assumptions that fail at the tactical edge.

Our framework addresses these constraints through three key design decisions:

1. **Rust-native inference**: No Python GIL, no garbage collection pauses, predictable latency.
2. **Aggressive quantization**: 2–8 bit weight compression enabling models larger than available RAM.
3. **Zero-copy protocols**: Sub-microsecond serialization for real-time agent coordination.

1.1 Strategic Context

The capability to run frontier models at the disconnected edge depends on owning the model itself, and that is the gap this framework fills. American frontier AI has consolidated into a closed monoculture—effectively two labs—whose models are reachable only as a metered cloud service; the only openly available frontier weights today are largely foreign. Neither option works at the tactical edge, where there is no cloud to call and foreign-origin weights are unacceptable. Hanzo, an American applied-AI lab founded in 2014 (Techstars '17) whose founder works at the frontier of both AI/ML and cryptography, is the American open-source frontier alternative: it builds the Zen model family (0.8B–1T+ parameters) with owned weights together with a production post-quantum crypto stack, so the same models can be quantized, inspected, and deployed onto hardware the operator controls—no per-token API markup, no hyperscaler dependency, no foreign weights. Owning the model and self-hosting it is also the principal lever for roughly an order-of-magnitude lower infrastructure cost at scale, since it removes both the API markup and the cloud-margin that a rented model carries on every inference.

2 Hanzo Engine: Rust-Native Inference

2.1 Architecture

Hanzo Engine is a production LLM inference server built in Rust, providing:

- **Model support**: 15+ text architectures (Transformer, MoE, hybrid), 16+ vision-language models, audio models (ASR, TTS), image generation (FLUX diffusion).
- **GPU backends**: Metal (Apple Silicon), CUDA (NVIDIA Ampere/Ada/Hopper/Blackwell), CPU (x86 AVX-512, ARM NEON/SVE), WebGPU (experimental).
- **Serving**: OpenAI-compatible REST API (`/v1/chat/completions`, `/v1/embeddings`), Ollama compatibility mode, built-in web UI.
- **Optimization**: PagedAttention with FP8 KV cache (50% memory reduction), FlashAttention V2/V3, continuous batching, prefix caching.

2.2 PagedAttention

PagedAttention manages KV cache as fixed-size blocks, enabling dynamic context length without pre-allocation:

- Block-based memory management with LRU eviction.
- FP8 E4M3 KV cache quantization (50% memory reduction vs. FP16).
- Block-level prefix caching for multi-turn conversations.
- Automatic fallback to standard attention when PagedAttention is unavailable.

2.3 FlashAttention Integration

Version	GPU Architecture	Compute Capability
FlashAttention V2	Ampere (A100, RTX 30)	≥ 8.0
FlashAttention V2	Ada Lovelace (RTX 40)	≥ 8.9
FlashAttention V3	Hopper (H100)	≥ 9.0
FlashAttention V2	Blackwell (RTX 50)	≥ 12.0

Table 1: FlashAttention version dispatch by GPU architecture.

3 Model Quantization for Constrained Hardware

3.1 Quantization Methods

Method	CPU	CUDA	Metal	Bits	Notes
ISQ	✓	✓	✓	2–8	Load-time, never loads full model
AFQ	–	–	✓	2–8	Metal-optimized kernels
GGUF	✓	✓	✓	2–8	Universal format
GPTQ/AWQ	–	✓	–	2–8	Marlin CUDA kernels
HQQ	✓	✓	✓	4, 8	Half-quadratic
FP8	✓	✓	✓	8	Floating-point
BNB	✓	✓	✓	4, 8	int8, fp4, nf4

Table 2: Quantization method support matrix.

3.2 In-Situ Quantization (ISQ)

ISQ is the primary edge deployment method. It quantizes weights during model loading, streaming from storage and quantizing layer-by-layer:

Algorithm 1 In-Situ Quantization Loading

for each layer ℓ in model **do**
  Stream weights W_ℓ from storage (SafeTensors/GGUF)
  Compute quantization parameters: scale s, zero-point z
  Quantize: $\hat{W}_\ell = \text{round}(W_\ell / s) + z$
  Store $\hat{W}_\ell$ in target device memory (GPU/CPU)
  Release original W_ℓ from host memory
end for
return quantized model (fits in device memory)

Key advantage: models larger than available RAM can be loaded, as only one layer's full-precision weights are in memory at any time.

3.3 AFQ: Metal-Optimized Quantization

Affine Quantization (AFQ) is designed specifically for Apple Silicon Metal:

- Group sizes: 32, 64 (default), 128.
- Optimized Metal kernel dispatch for Apple M1/M2/M3/M4.
- 2–8 bit quantization with affine transformation.
- Outperforms GGUF on Apple Silicon by leveraging unified memory architecture.

4 Zen Model Family for Defense Applications

4.1 Model Tiers

Tier	Model	Params	Active	Use Case
Edge	zen4-nano	0.8B	0.8B	On-device, embedded
	zen4-micro	2B	2B	Edge+
	zen4-mini	4B	4B	Mobile, tablet
	zen4	9B	9B	General tactical
Medium	zen4-pro	27B	27B	Balanced perf/quality
	zen4-max	35B MoE	3B	Efficient MoE
	zen4-coder	80B MoE	3B	Code analysis
Frontier	zen4-ultra	1.04T MoE	32B	Frontier reasoning
	zen4-titan	744B MoE	40B	Advanced analysis
	zen4-storm	456B MoE	45B	Large-scale

Table 3: Zen4 model family with defense deployment tiers.

4.2 Vision-Language Models

Six vision-language variants (4B–30B) provide:

- 32-language OCR for document exploitation.
- Spatial reasoning for geospatial imagery analysis.

- Video comprehension for temporal pattern recognition.
- Agent variants with native function calling for tool integration.

4.3 Mixture of Experts (MoE)

MoE models activate only 2–8 experts per token from a pool of 16–384, providing frontier-quality output with edge-scale compute:

- **zen4-coder-flash** (31B total, 3B active): 59.2% SWE-Bench.
- **zen-max** (671B total, 14B active): Extended reasoning with 96–128K thinking tokens.
- **Heavy Mode**: 8 simultaneous reasoning trajectories for complex analysis.

5 GPU-Accelerated AI/ML Operations

5.1 Hanzo GPU Kernel Library

Category	Operations
Matrix	matmul, batch matmul, transposed matmul
Attention	Standard, FlashAttention, multi-head attention
Normalization	LayerNorm, RMSNorm, BatchNorm
Activation	ReLU, GELU, SiLU, sigmoid, tanh, softmax
Convolution	2D standard, depthwise
Quantization	FP quantization, blockwise FP8
Crypto	BLS, BN254, secp256k1, Ed25519, ML-DSA, NTT

Table 4: Hanzo GPU kernel categories.

5.2 Unified C API

Listing 1: GPU kernel acceleration C API (excerpt).

```
// Session management
hz_session_t* hz_create_session(hz_device_t dev);

// Tensor operations
hz_tensor_t* hz_matmul(hz_session_t* s,
    hz_tensor_t* a, hz_tensor_t* b);
hz_tensor_t* hz_attention(hz_session_t* s,
    hz_tensor_t* q, hz_tensor_t* k,
    hz_tensor_t* v, float scale);

// Crypto operations
hz_tensor_t* hz_mldsa_verify(hz_session_t* s,
    hz_tensor_t* pk, hz_tensor_t* msg,
    hz_tensor_t* sig);
```

6 On-Device Learning and Adaptation

6.1 LoRA Runtime Adaptation

Low-Rank Adaptation (LoRA) enables domain-specific fine-tuning without full model retraining:

- Dynamic adapter activation at runtime (hot-swap without reload).
- X-LoRA: mixture of adapters for multi-task adaptation.
- Quantized base model with unquantized adapter layers.
- Adapters can be pre-loaded from HuggingFace or local storage.

Tactical application: Pre-train base model in data center, deploy with mission-specific LoRA adapters. Swap adapters as mission requirements change without downloading new models.

6.2 Speculative Decoding

Draft model generates candidate tokens; target model verifies in parallel:

- 2–4× latency reduction for interactive use.
- Draft model: small (nano/eco), target: larger (pro/max).
- No quality degradation—output identical to target model alone.

7 ZAP Protocol for AI Workloads

7.1 Performance for Agent Communication

Metric	JSON-RPC	ZAP
Serialization	5,579 ns	322 ns
Memory/call	2,826 B	256 B
Allocations/op	58	2
GC pressure (50K ops)	41 pauses	3 pauses
Network (20 servers)	140K calls/s	4.2M calls/s

Table 5: ZAP vs. JSON-RPC for AI agent communication.

7.2 MCP Gateway

ZAP bridges 20+ Model Context Protocol (MCP) servers with tool prefix namespacing, health checking, and reconnection:

Listing 2: MCP gateway aggregation.

```
bridge.AddServer("filesystem", "npx",
    "@anthropic/mcp-filesystem")
bridge.AddServer("github", "npx",
    "@anthropic/mcp-github")
result := bridge.CallToolByName(ctx,
    "filesystem:read_file", args)
```

7.3 GPU Cluster Coordination

ZAP's mDNS discovery (`_hz-gpu._tcp`) enables automatic GPU node discovery for distributed inference and FHE operations. Direct zero-copy ciphertext handling enables GPU-to-GPU encrypted computation pipelines.

8 Multi-Agent Orchestration

8.1 Agent Architecture

- **83 specialized agents**: Architecture, languages, infrastructure, data/ML, QA, security, and domain-specific expertise.
- **15 multi-agent orchestrators**: Full-stack development, security hardening, ML pipelines, incident response.
- **Async-first SDK**: Compatible with OpenAI Chat Completions API.
- **State and memory**: Shared state + vector search long-term memory (LanceDB).
- **Tool integration**: 13 canonical MCP tools (fs, exec, code, git, fetch, workspace, ui, think, memory, hanzo, plan, tasks, mode).

8.2 Agent Consensus

For critical decisions, agents employ distributed voting:

- Threshold-based agreement (default 67%).
- W3C DID authentication (did:key from ML-DSA public keys).
- PQ-secure agent-to-agent communication via ZAP.
- Stake-weighted voting for reputation-based trust.

9 Disconnected Edge Deployment

9.1 Hardware Deployment Matrix

Platform	Backend	Quantization	Max Model
Apple Silicon M4	Metal + AFQ	2–8 bit	zen4-pro (27B)
NVIDIA Jetson	CUDA	GGUF/ISQ	zen4 (9B)
x86 Server	AVX-512	ISQ/GGUF	zen4-ultra (1T)
ARM Embedded	CPU/NEON	GGUF 4-bit	zen4-nano (0.8B)
Browser	WebGPU	ISQ	zen4-nano (0.8B)

Table 6: Edge deployment: hardware to model mapping.

9.2 Offline Operation

- Self-contained binary with embedded model management.

- Ollama-compatible `pull`/`list`/`run` commands for model caching.
- Sandboxed execution via macOS `sandbox-exec` or Linux seccomp.
- MCP tools operate locally (filesystem, code analysis, git) with no network.
- Bearer token authentication for optional cloud relay.

10 Hanzo ML Framework

The Hanzo ML framework (Candle fork) provides Rust-native tensor operations:

- **Core**: Tensor creation, manipulation, GPU dispatch (Metal, CUDA).
- **Neural network**: Transformer layers, attention mechanisms, MLP.
- **Models**: 100+ pre-trained model implementations.
- **ONNX**: Import and run ONNX models directly.
- **PyO3**: Python bindings for integration with existing ML pipelines.
- **WASM**: Browser-based inference via WebAssembly.
- **Flash Attention**: V2 kernels for 2–3× memory bandwidth reduction.

11 Reinforcement Learning from Operational Feedback

11.1 Feedback Loop Architecture

Agent execution in tactical environments produces results paired with confidence scores. Human operators validate, correct, or override these results through a structured feedback interface:

- **Accept**: Operator confirms the agent output; recorded as a positive preference signal.
- **Reject**: Operator discards the output; the correct action (if provided) forms a negative preference pair.
- **Modify**: Operator edits the output; the original and edited versions form a ranked preference pair.

All feedback is recorded as preference pairs (y_w, y_l) where y_w is the preferred response and y_l is the dispreferred response, suitable for direct optimization.

11.2 LoRA Fine-Tuning via RLHF/DPO

On-device adaptation uses Low-Rank Adaptation to incorporate operational feedback without full model retraining:

- **RLHF pipeline**: Preference pairs train a reward model; PPO optimizes the policy against this reward while constraining divergence from the base model via KL penalty.

- **DPO (Direct Preference Optimization)**: Bypasses explicit reward modeling by directly optimizing the policy from preference pairs, reducing compute requirements for edge deployment.
- **LoRA efficiency**: Only adapter weights (rank 4–64, typically <1% of model parameters) are updated, enabling fine-tuning on tactical hardware with limited GPU memory.

11.3 Byzantine-Robust Aggregation

When multiple agents or edge nodes produce LoRA updates from independent operational feedback, aggregation must tolerate adversarial or corrupted contributions:

- **DeltaSoup**: Byzantine-robust aggregation of LoRA weight deltas from multiple agents. Outlier updates are detected via coordinate-wise median filtering before merging, ensuring that compromised or malfunctioning nodes cannot poison the shared adapter.
- **GT-QLoRA (Gate-Targeted QLoRA)**: For MoE architectures, adapter updates target expert gating layers specifically, enabling efficient adaptation of routing decisions without modifying expert weights. Combined with 4-bit quantized base weights, this enables MoE fine-tuning within edge memory budgets.

12 In-Memory Graph Analytics

12.1 GraphVM

The GraphVM provides in-memory graph analytics optimized for intelligence workloads where relationship discovery across heterogeneous data sources is critical:

- **Entity resolution**: Probabilistic matching across intelligence sources (HUMINT, SIGINT, OSINT) to unify identities despite name variations, transliterations, and deliberate obfuscation.
- **Network topology analysis**: Communication pattern extraction from intercept data—call graphs, message chains, and temporal co-occurrence networks.
- **Organizational hierarchy mapping**: Social network analysis to infer command structures, identify key nodes, and detect cells within adversary organizations.
- **Temporal behavior tracking**: Time-series graph snapshots enabling detection of behavioral changes, new associations, and pattern-of-life deviations.

12.2 Integrity and Verification

- **Merkle-verified graph state**: Every graph mutation is committed to a Merkle tree, providing tamper-evident audit trails. Analysts can verify that graph state has not been modified since ingestion.
- **Provenance tracking**: Each edge and node carries source metadata (collection platform, time, confidence), enabling analysts to assess intelligence reliability at the relationship level.

12.3 Defense Applications

- **Intelligence link analysis**: Multi-hop traversal to discover indirect connections between persons of interest across compartmented datasets.
- **Threat network mapping**: Real-time graph updates as new intelligence arrives, with automatic re-scoring of centrality metrics to identify emerging leaders or facilitators.
- **Force protection**: Pattern-of-life deviation detection for insider threat and force protection scenarios, comparing current behavior graphs against historical baselines.

13 Explainable and Adaptable AI

13.1 Design Principles

The framework implements AI that is *visible*, *accessible*, *explainable*, and *adaptable* in accordance with defense acquisition requirements for trusted AI systems.

13.2 Inherent Explainability

- **Chain-of-thought reasoning**: Zen models produce 96–128K thinking tokens as extended reasoning traces. These traces expose the model's intermediate logic, hypotheses, and evidence evaluation—providing inherent explainability without post-hoc interpretation methods.
- **Heavy Mode**: 8 parallel reasoning trajectories with explicit comparison. Operators can inspect each trajectory's reasoning chain, observe where paths diverge, and understand why the selected answer was preferred.
- **Tool call logging**: Every external tool invocation (filesystem reads, code execution, database queries, API calls) is recorded with input arguments and output results, producing a complete audit trail of agent actions.
- **Memory retrieval traces**: When agents access long-term memory (LanceDB vector search), the retrieved context and similarity scores are logged, making it transparent which prior knowledge influenced the current decision.

13.3 Adaptability Mechanisms

- **LoRA adapter swapping**: Mission-specific adapters can be loaded, unloaded, and swapped at runtime without model restart. This enables rapid reconfiguration—e.g., switching from ISR analysis to SIGINT processing by swapping a single adapter file.
- **X-LoRA mixture**: Multiple adapters can be activated simultaneously with learned mixing weights, enabling multi-mission configurations from a library of single-purpose adapters.

13.4 Verifiable Decision Records

- **W3C Decentralized Identifiers (DIDs)**: Each agent holds a `did:key` identity derived from its ML-DSA public key. All decision outputs are signed with this identity, providing non-repudiation.

- **Verifiable Credentials**: Agent outputs are packaged as W3C Verifiable Credentials containing the decision, reasoning trace, confidence score, and input data hash. Third parties can verify authenticity and integrity without access to the issuing agent.
- **Tamper-evident chains**: Sequential decision records are hash-chained, enabling detection of any retroactive modification to the decision log.

14 AR/VR Integration for Warfighter Performance

14.1 Vision-Language Models as AR Engines

Zen vision-language models (4B–30B parameters) serve as real-time data overlay engines for augmented reality systems:

- **Real-time spatial reasoning**: Object detection, scene understanding, and spatial relationship inference from head-mounted camera feeds.
- **32-language OCR**: On-the-fly text recognition and translation from signage, documents, and captured materials displayed as AR overlays.
- **Contextual annotation**: Automatic identification and labeling of vehicles, equipment, structures, and personnel with threat assessment scores projected into the operator's field of view.
- **Edge deployment**: Vision-language inference runs on tactical hardware (Metal on Apple Silicon, CUDA on NVIDIA Jetson, ARM NEON on embedded processors) with quantized models (4–8 bit) for real-time frame rates.

14.2 3D Mission Rehearsal

- **Babylon.js rendering**: Interactive 3D environments for mission planning and rehearsal, supporting terrain visualization, building interiors, and route planning with physics simulation.
- **Digital twin integration**: The Netrunner network simulation framework feeds real-time system state into VR environments, enabling operators to visualize network topology, communication flows, and system health in an immersive 3D space.
- **Collaborative planning**: Multiple operators can share a VR mission space with synchronized state, annotating terrain, marking objectives, and rehearsing maneuvers in a shared virtual environment.

14.3 Training and System Design

- **Scenario generation**: LLM-driven procedural generation of training scenarios based on historical after-action reports and doctrinal templates.
- **Performance analytics**: Operator actions during VR training sessions are recorded and analyzed by multi-agent systems to identify decision patterns, reaction times, and areas for improvement.
- **System design visualization**: Digital twins of proposed system architectures rendered in VR, enabling stakeholders to walk through data flows, identify bottlenecks, and validate designs before deployment.

15 Defense Application Scenarios

15.1 ISR Analysis

Zen vision-language models (4B–30B) process full-motion video, satellite imagery, and document scans:

- 32-language OCR for captured document exploitation.
- Spatial reasoning for geospatial analysis.
- Temporal video comprehension for activity detection.
- LoRA adapters for theater-specific terrain recognition.

15.2 Tactical Decision Support

- Extended reasoning (96–128K thinking tokens) for complex COA analysis.
- Heavy Mode: 8 parallel reasoning trajectories for wargaming.
- Multi-agent orchestration for multi-domain analysis.
- Runs entirely on tactical hardware, no reach-back required.

15.3 SIGINT Natural Language Processing

- 100+ programming language understanding for CYBER analysis.
- 32+ natural language support for COMINT processing.
- Intent detection and sentiment analysis via Insights pipeline.
- Named entity recognition and relationship extraction.

16 Conclusion

We have presented a comprehensive edge AI/ML framework that operates fully disconnected from cloud infrastructure while providing frontier-quality model capabilities through aggressive quantization and hardware-optimized inference. The combination of Rust-native performance, zero-copy agent coordination, GPU-accelerated operations, and multi-agent orchestration provides a complete tactical AI stack deployable on hardware from embedded ARM processors to multi-GPU servers.

References

[1] T. Dao *et al.*, "FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning," ICLR 2024.

[2] W. Kwon *et al.*, "Efficient Memory Management for Large Language Model Serving with PagedAttention," SOSP 2023.

[3] E. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," ICLR 2022.

[4] E. Frantar *et al.*, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," ICLR 2023.

[5] J. Lin *et al.*, "AWQ: Activation-aware Weight Quantization for LLM Compression," MLSys 2024.

[6] W. Fedus *et al.*, "Switch Transformers: Scaling to Trillion Parameter Models," JMLR 2022.

[7] Hugging Face, "Candle: Minimalist ML Framework for Rust," 2023.

[8] Anthropic, "Model Context Protocol Specification," 2024.

[9] Hanzo Team, "ZAP: Zero-Copy Application Protocol for AI Agent Communication," Hanzo Industries, 2026.

[10] G. Gerganov, "GGUF: GGML Universal File Format," llama.cpp, 2023.

[11] Y. Leviathan *et al.*, "Fast Inference from Transformers via Speculative Decoding," ICML 2023.

[12] Y. LeCun, "A Path Towards Autonomous Machine Intelligence," 2022.

[13] Zen LM, "Zen Model Family Technical Report," 2026.

[14] LanceDB, "LanceDB: Serverless Vector Database," 2024.

[15] OpenTelemetry, "Observability Framework for Cloud-Native Software," CNCF, 2024.

Reproducibility

Source code. `github.com/hanzoai/engine, hanzoai/jin` (commit `TBD`).
Build. cargo build –release; LanceDB embedded by default.
Hardware tested. Apple M-series (Metal), NVIDIA Jetson, x86_64 Linux + CUDA.
Standards referenced. OpenTelemetry, ONNX Runtime, OpenAI API.
Contact. research@hanzo.ai.