The City College of New York

CSC 59866-E: Senior Project I

AI Agents for Decision Making in the Real World

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

March 30, 2026 CSC 59866

Multimodal Agentic Evaluation: Post-training, Inference, Qualitative and Quantitative decision-making assessment (Trustworthy and Explainable Agents)

Logistics and Motivation

Recall Lecture 17: We covered the hardware and efficiency bottlenecks of Agentic Systems, including the KV Cache and AlphaChip.

The Final Phase: You have mathematically defined your agents, built them in PyTorch/JAX, and optimized them for deployment.

The Core Question: How do you prove your agent actually works safely?

Today's Agenda

- The Evaluation Crisis: Why standard metrics fail for autonomous agents.
- Inference Efficiency Metrics (Energy, Memory, Bandwidth).
- Explainable AI (XAI): SHAP and Vision-Language Interpretability.
- Interactive Problem: Calculating Expected Calibration Error (ECE).
- Post-Training Assessment: LLM-as-a-Judge and ELO ratings.

The Agentic Evaluation Crisis

Why Standard Metrics Fail

The Supervised Paradigm: In standard ML, we evaluate using Accuracy, F1-Score, or Cross-Entropy Loss on a static test set (e.g., ImageNet or MMLU).

The Agent Paradigm: Agents take *sequential* actions in dynamic, open-ended environments. A single mistake at step 1 completely changes the state at step 10.

Multimodal Complexity: If an agent receives a camera feed (Vision), reads a user manual (Text), and controls a robotic arm (Continuous Action), a simple "Accuracy" metric provides zero insight into *where* the reasoning failed.

Evaluation must move from "Did it guess the right word?" to "Did it accomplish the long-horizon task efficiently and safely?"

Evaluating Inference Efficiency

An agent that requires a $100,000 supercomputer to make a single decision is functionally useless. We must quantitatively assess efficiency.

Memory Utilization: Monitoring the VRAM ceiling. Agents must be evaluated on how efficiently they prune their KV Cache during long tasks.

Network Bandwidth: In decentralized swarms, evaluating the payload size (bytes per second) of agent-to-agent communication. (e.g., Does the agent compress visual data into semantic text before broadcasting?)

Energy & Power: Raw computational cost (FLOPs/Watt) required per successful episode.

Hardware & Chip-Level Evaluation

The ultimate ceiling of an agent's efficiency is determined by the physical silicon it runs on.

Recall AlphaChip (2021 & 2024): Deep RL agents are now used to generate superhuman chip floorplans (like the TPUs running your models).

Hardware Evaluation Metrics: When evaluating these specialized hardware-design agents, we measure physical proxy costs:

$$J(\pi) = \mathbb{E}_{\pi}\left[-\left(\text{Wirelength} + \lambda \cdot \text{Congestion} + \gamma \cdot \text{Density}\right)\right]$$

The Lesson: "Software" evaluation is incomplete. A truly robust AI methodology must account for the chip-level utilization and physical routing efficiency it demands.

What Makes an Agent "Trustworthy"?

Trust != Accuracy. An agent that is 99% accurate but confidently lies 1% of the time in life-or-death scenarios is untrustworthy.

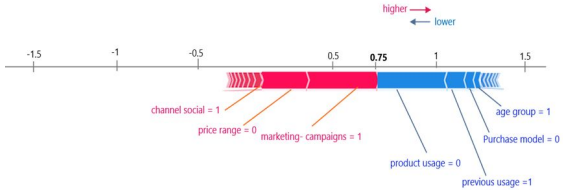
A trustworthy agent must exhibit:

- **Calibration:** Its confidence level should perfectly match its actual probability of being correct.
- **Robustness:** It maintains stable behavior under adversarial inputs (e.g., noisy sensors).
- **Explainability:** It can trace its complex decisions back to specific, understandable input features.

SHAP Values

- How do we quantitatively measure *why* an agent made a decision? We use cooperative game theory.
- **SHAP (SHapley Additive exPlanations):** Treats every input feature (a pixel, a word, a sensor reading) as a "player" in a coalition game, where the "payout" is the agent's final decision.
- The SHAP value ϕ_i for feature i is calculated by evaluating the model's output with and without feature i across all possible feature combinations:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [v(S \cup \{i\}) - v(S)]$$



- **Interpretation:** This gives us a mathematically rigorous breakdown of exactly how much each input feature contributed to the agent's action.

Local Approximations

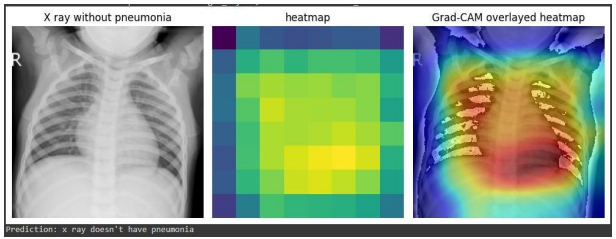
- **LIME (Local Interpretable Model-agnostic Explanations):** LIME takes a faster, localized approach than SHAP's exhaustive Game-Theoretic approach
- **The Intuition:** A complex agent's neural network is highly non-linear globally, but if we zoom in close enough to a single decision, the logic looks linear and interpretable.
- **The Math:** LIME learns a simple, interpretable model g (like linear regression) that locally approximates the complex agent f around a specific state x:

$$\xi(x) = \text{argmin}_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

- $\mathcal{L}(f, g, \pi_x)$: We want the simple model g to match the complex model f when we randomly perturb the inputs, weighting the results by their proximity π_x to the original state x.
- $\Omega(g)$: A penalty for complexity. We want a simple explanation!

Evaluating Multimodal Vision

Grad-CAM (Gradient-weighted Class Activation Mapping): Uses the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map.



The Math: We calculate neuron importance weights α_k^c by global-average-pooling the gradients, then perform a ReLU-activated weighted combination of forward activation maps.

Agentic Use Case: If an autonomous car brakes suddenly, Grad-CAM allows us to evaluate if the VLM was looking at the pedestrian, or erroneously looking at a shadow on the wall.

Expected Calibration Error (ECE)

Expected Calibration Error (ECE)

The Scenario: Your team has trained an LLM-based agent to diagnose mechanical failures. It outputs a decision and a **Confidence Score (0.0 to 1.0)**.

To evaluate its trustworthiness, we calculate the Expected Calibration Error (ECE). We group its predictions into M equally sized bins based on confidence, and compare the average confidence to the actual accuracy in that bin.

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|$$

Expected Calibration Error (ECE)

Your Task: Calculate the ECE for an agent evaluated over n=100 total test scenarios, divided into 2 bins:

- **Bin 1 (50 scenarios):** The agent was highly confident. Average Confidence = 0.90. Actual Accuracy = 0.80.
- **Bin 2 (50 scenarios):** The agent was unsure. Average Confidence = 0.60. Actual Accuracy = 0.40.

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|$$

Solution

Step 1: Calculate the error in Bin 1.

- Weight: 50 / 100 = 0.5 ; Absolute Error: |0.80 - 0.90| = 0.10; Contribution: 0.5 \times 0.10 = 0.05

Step 2: Calculate the error in Bin 2.

- Weight: 50 / 100 = 0.5 ; Absolute Error: |0.40 - 0.60| = 0.20 ; Contribution: 0.5 x .20 = 0.10

Step 3: Sum for total ECE. ECE = 0.05 + 0.10 = **0.15**

Analysis: The ECE is 15%. This agent is dangerously *overconfident*. It thinks it is right 90% and 60% of the time, but is actually only right 80% and 40% of the time.

Post-Training Assessment

LLM as a Judge

- When evaluating open-ended agent tasks (e.g., "Negotiate a better price with the other AI agent"), there is no objective ground-truth label.
- **LLM-as-a-Judge:** We use a powerful, separate foundation model (like GPT-4 or Gemini 1.5 Pro) to evaluate the transcript of our agent's behavior.
- **The Methodology:** You must write a strict, quantitative rubric into the Judge's prompt.
    - "Score the agent's negotiation from 1-10. -2 points for repeating arguments. -5 points for accepting a price above $100."
- **Correlation:** Recent studies show LLM-as-a-Judge correlates highly (above 85%) with human expert evaluations, at a fraction of the cost.

Red Teaming and ELO

- **Red Teaming:** You cannot just test your agent on normal tasks. You must deploy an "Adversarial Agent" specifically trained to confuse, attack, or find safety vulnerabilities in your main agent.
- **Agentic ELO:** Since environments are infinite, we evaluate agents by pitting them against each other in arenas.
- Their performance is updated using the Elo rating system (borrowed from Chess):

$$R_{\text{new}} = R_{\text{old}} + K \cdot (S_{\text{actual}} - S_{\text{expected}})$$

- This provides a dynamic, constantly evolving ranking of which agent architecture is truly the most robust.

Relevance for Projects

- **Efficiency:** You must evaluate the inference cost (Memory, Bandwidth, Latency) alongside the logic.
- **Explainability:** Utilize XAI tools (SHAP, Grad-CAM) to prove *why* your multimodal agents are making decisions.
- **Calibration:** Calculate metrics like ECE to prove your agent can be trusted in high-stakes environments.

Questions?