Self-Evolving Real-Time Agents: Think While Listening, Speak While Thinking, Learn While Acting

Bojie Li

Co-Founder & Chief Scientist, Pine AI

Nov. 2025

Press Space for next page →

Pine AI: AI Agent that Make Calls to Get Things Done

🌲 What is Pine AI?

AI Agent that Takes Action

Like ChatGPT, but can make calls, send emails, and use computers to complete tasks

Your Personal Assistant

Handles tedious customer service interactions on your behalf

Reclaim What's Rightfully Yours

Skip the long hold times and unhelpful agents. Pine fights for what's rightfully yours.

💡 Key Capabilities

- 📞 **Bill Negotiation:** Average 20% savings on telecom, utilities
- ❌ **Subscription Cancellation:** Cancel unwanted services
- 📋 **Complaint Filing:** File formal complaints and get resolutions
- 💰 **Compensation & Refunds:** Recover unauthorized charges
- ✈️ **Travel Assistance:** Handle bookings and cancellations

270 min
Avg. Time Saved

93%
Success Rate

$3M+
Saved for Consumers

🔗 Learn more at 19pine.ai

Overview: Two Fundamental Challenges of Agents

Part I: Real-Time Interaction

Real-time voice agents must respond in **<1s** like humans, but traditional architectures introduce **2-10 second delays** with reasoning LLMs

VAD Challenges:

- 500-800ms unavoidable wait for silence
- "Uh-huh" mistakenly triggers interruption
- Lost acoustic info (emotions, environment)

ASR Challenges:

- No context → high errors (emails, names)
- No world knowledge → wrong transcription

LLM Challenges:

- Forced to wait, cannot think while listening
- Cannot speak while thinking (5-10s silence)
- Poor turn detection (when to speak/silence)

Part II: Learning from Experience

Models are "**intelligent**" but not "**proficient**" — like top graduates lacking real-world experience

Fixed Models Cannot Learn:

- Cannot learn from successful traces
- Cannot learn from unsuccessful traces
- Parameters frozen after deployment

Big World Hypothesis:

World is too large to pre-encode all knowledge

- Business processes are dynamic & non-public
- Verification info varies by company
- Service rules constantly change
- Pre-trained knowledge insufficient for deployment

Part I: Real-Time Agent-Environment Interaction

Interaction Targets:

- **Humans**: Dialogue and collaboration through real-time voice communication.
- **Digital World**: Operating computers, browsing web pages, using mobile devices.
- **Physical World**: Controlling robots, interacting with real environments.

A Typical Architecture of Voice Agents

🎤📹 ↓

1. Perception Layer

VAD + ASR

〰 → ▢ ▢ ▢

↓

Continuous → Discrete Events

▢ ▢ ▢ ↓

2. Thinking Layer

LLM

🧠⚡

↓

Async Processing

▢ ▢ ▢ ↓

3. Execution Layer

TTS

▢ ▢ ▢ → 〰

↓

Discrete → Continuous Actions

🔊🖱️

Layer 1: Perception Layer

Transforming Continuous Real-World Signals into Discrete Events

Input

Continuous signals: audio streams

Output

Discrete events: `speech_start`, `interrupt`, `laugh`, `speech_fragment`, etc.

Problems with Traditional VAD + ASR Architecture

VAD (Voice Activity Detection)

1. Unavoidable Latency

Must wait 500-800ms of continuous silence to confirm user finished

2. Poor Interrupt Detection

- Cannot distinguish background noise/music
- "Uh-huh" mistakenly triggers interruption

3. Low Voice Detection Accuracy

- Errors in complex acoustic environments
- Mid-sentence pauses → truncation
- "Hello" in background noise → unresponsive

ASR (Automatic Speech Recognition)

1. Low Accuracy Without Context

- VAD cuts audio into isolated segments
- Cannot use context for disambiguation
- **High errors for:** emails, names, phone numbers

2. Lack of World Knowledge

- Cannot leverage common sense
- **Low accuracy for:** addresses, brands, technical terms, amounts

3. Text-Only Output Lacks Acoustic Details

- **Lost emotions:** happy, frustrated, excited
- **Lost paralinguistic:** laugh, sigh, breath
- **Lost environment:** noisy, music, quiet

Streaming Voice Perception Model: Replacing VAD + ASR

Multimodal Architecture

Model Architecture

1. Audio Encoder (from Whisper)
↓ Converts audio → audio tokens
2. Qwen LLM (autoregressive)
↓ Processes audio tokens → text + events

Key Advantages

- **Streaming:** Real-time output (not batch)
- **Context:** Full dialogue history preserved
- **In-Context Learning:** Better recognition for personal info, domain terms
- **World Knowledge:** Higher accuracy for addresses, brands, amounts

Rich Output: Text + Acoustic Events

Text Tokens

Real-time transcribed text fragments

Special Tokens (Acoustic Events)

`<speak_start>` `<speak_end>` Speech boundaries
`<interrupt>` Interruption intent
`<emotion:happy>` Emotion markers
`<laugh>` `<sigh>` Paralinguistic info
`<music>` Environmental sounds

Layer 2: Thinking Layer

Event-Driven Loop Enabling Interruptible, Asynchronous Thinking While Listening, Speaking While Thinking

Input

Discrete event stream of observations (user utterances) and tool call results

Output

Interleaved thoughts, tool calls, and output sentences (for TTS)

Interactive ReAct: Enabling Flexible Interweaving of Observation, Thinking, and Action

Traditional ReAct: Rigid OTA Loop

```
O₁: "I want to lower my Xfinity bill to $79 per month"
T₁: (thinking 5s... then interrupted, all lost)
O₂: "and I do not want to cut off any features"
T₂: (thinking 15s...)
A₁: "Got it! Here is a $79 plan with all the
features..."
```

- **Fixed Loop**: Must complete entire Observation-Thinking-Action sequence
- **Thinking Lost**: Cannot think while listening, high latency
- **Rigid**: Must wait for complete input before thinking

Interactive ReAct: Flexibly Interleaved OTA

```
O₁: "I want to lower my Xfinity bill to $79 per month"
T₁: (fast think 0.5s: user utterance incomplete, wait)
T₂: (thinking 5s... then interrupted)
O₂: "and I do not want to cut off any features"
T₃: (fast think 0.5s: user wants to lower bill to $79)
A₁: "I can help you with that! Let me check the
available plans"
T₄: (continuing thinking... 10s)
A₂: "Got it! Here is a $79 plan with all the
features..."
```

- **Think While Listening**: New observations insert anytime, thinking preserved
- **Speak While Thinking**: Fast response, then continue thinking
- **Intelligent Turn Detection**: Decide when to speak, when to stay silent

Interactive ReAct: Think While Listening

Key Insight: LLM is 20-100x Faster Than Human Speech - Use Gap Time to Think!

🧠 LLM Processing Speed

- Prefill (Input): **1000+ tokens/sec**
- Decode (Output): **100 tokens/sec**

🗣️ Human Voice Input/Output Speed

- Speaking: **5 tokens/sec (text) or 20 tokens/sec (audio tokens)**
- **LLM is 20-100x faster than humans!**

Example: Interview Agent with Async Tool Calls While Candidate Speaks

```
Candidate: My previous role involved building distributed systems...
Think: Distributed systems - need to assess depth. Let me search...
Tool Call: web_search("candidate distributed systems projects") (async!)
Candidate: ...we handled 10M requests/sec using Kafka and Redis (speaking while tool runs)
Think: Kafka+Redis is solid for high throughput. Continue listening...
Tool Result: GitHub shows 3 open-source projects, 2K+ stars total
Think: Tool result confirms experience! Integrate with what candidate said...
Assistant: That's impressive scale! (<0.5s!)
    Tell me about your toughest scaling challenge...
```

Advantage: Async tools + thinking while listening → no waiting, ultra-fast response

Interactive ReAct: Speak While Thinking

Theory: ⚡ Fast → 🐢 Slow → 🐌 Continuous Thinking Using Filler Speech

Three Phases of Thinking

1. ⚡ **Fast (0.5s, 50 tokens)**
   Quick judgment → immediate response
2. 🐢 **Slow (5s, 500 tokens)**
   Deep analysis → complete answer
3. 🐌 **Continuous (interleaved thinking and speaking)**
   Keep thinking → keep speaking

Key: Use "filler speech" to maintain conversation flow during deep thinking

Example: Interview Agent Asking Complex Question

```
Candidate: I'm ready for the technical question.
```

```
Think: Complex question, need to formulate carefully
```

```
Assistant: Let me ask you a system design question. (⚡
0.5s)
```

```
Think: Need to cover scalability, consistency,
latency... (🐢 5s)
```

```
Assistant: Imagine you're building a global CDN.
```

```
Think: Continue - specify the cache invalidation
challenge...
```

```
Assistant: How would you handle cache invalidation
across
    100+ edge servers when content is updated?
```

Result: Question unfolds naturally sentence-by-sentence, no awkward silence

Future: Three Stages of AI Agent-Environment Interaction

Real-time Asynchronous Interaction with Environment is Fundamental to Agents

🗣️ Stage 1: Voice

Input: Voice

Output: Voice

Data Rate: 15-50 token/s

Latency: <500ms

Challenge: Fast-slow thinking balance

Solution: Interactive ReAct

💻 Stage 2: Computer Use

Input: Visual (screenshots)

Output: Mouse/keyboard actions

Data Rate: ~2K token/frame

Latency: <1 second

Challenge: Precise action execution

Solution: VLA models + RL

🤖 Stage 3: Physical World

Input: Vision+Voice+Tactile

Output: Voice+Joint actions

Data Rate: ~20K token/s

Latency: <100ms

Challenge: Real-time control

Solution: VLA + World Models

Key Insight: Complexity increases (data rate ↑, latency ↓), but architectural solutions transfer across stages

Part II: Agents Learning from Experience

> *"We want AI agents that can discover like we can, not which contain what we have discovered." — Richard Sutton*

Why Agents Must Learn from Experience: From "Intelligent" to "Proficient"

🎓 SOTA Models ≈ Top Graduates

✅ Knowledgeable

Master vast amounts of general knowledge

❌ Lack Experience

Underperform vs. experienced professionals on specialized tasks (e.g., accounting, tax filing)

💼 Real Challenges in Pine AI

🔑 Verification Info

1st call: learns credit card last 4 digits required
2nd call: should proactively request it

📋 Service Procedures

1st cancellation: told to fill online form instead of phone call
2nd cancellation: should directly fill online form

🎯 Service Rules

Which discounts apply? (veterans, 2-year loyalty, etc.)

💰 Price Estimation

Is $60/month for 3Gbps broadband high or low? Room to negotiate?

Core Problem: Many business processes are **dynamic and non-public**. Simply improving the base model's general capabilities **cannot solve** these "experience-based" problems.

Building Self-Evolving Agents

Making Agents Learn from Experience

Paradigm 1: Post-Training

Paradigm 2: In-Context Learning

Paradigm 3: Externalized Learning

Method 1: Post-Training - SFT Memorizes, RL Generalizes

📚 Supervised Fine-Tuning (SFT)

✅ Advantages

- Extremely sample-efficient (thousands suffice)
- Quickly solidifies formats and protocols
- Stable training, fast convergence

❌ Limitations

- Memorizes surface patterns
- Cliff-like degradation on out-of-distribution
- Hard to learn transferable strategies

🎯 Reinforcement Learning (RL)

✅ Advantages

- Learns transferable policy representations
- Robust in out-of-distribution scenarios
- Discovers new strategies beyond training data

❌ Limitations

- Low sample efficiency (100x more data and compute)
- High training cost and time
- Requires verifiable reward signals

💡 Engineering Practice: Form Before Function

- **SFT Phase:** Establish format stability, ensure parseable outputs
- **RL Phase:** Break through generalization boundaries on stable foundation
- **Key Balance:** Train SFT until "format stable, capabilities emerging"

Improving Sample Efficiency (I): On-Policy Distillation

📊 Three Training Approaches

SFT (Supervised Fine-Tuning)

- Sampling: Off-policy (teacher's trajectories)
- Reward: Dense (token-by-token)
- Problem: Compounding errors in student's states

RL (Reinforcement Learning)

- Sampling: On-policy (student's rollouts)
- Reward: Sparse (only final outcome)
- Problem: One signal per episode, inefficient

✨ On-Policy Distillation

- Sampling: **On-policy** (student's trajectories)
- Reward: **Dense** (teacher grades each token)
- **Best of both worlds!**

🔧 How It Works

```
# Sample from student
trajectory = student.generate(prompt)

# Teacher grades EVERY token
for token in trajectory:
    teacher_logprobs = teacher(token | ctx)
    student_logprobs = student(token | ctx)

    # Minimize reverse KL
    loss = KL(student || teacher)
```

🎯 Key Benefits

- **10x more efficient** than RL
- Student learns to **recover from its own mistakes**
- Can **reuse training data** (multi-epoch)
- Enables **continual learning**

Improving Sample Efficiency (II): Feedback-Guided Sampling

❌ Traditional GRPO/DAPO

Process:

- Generate N independent rollouts
- Later attempts repeat same errors

```
Rollout 1: Requires SSN → ❌ Failure
Rollout 2: Requires SSN → ❌ Failure
Rollout 3: Requires SSN → ❌ Failure
...(wasting environment feedback)
```

✅ Feedback-Guided Sampling

Sequential Process:

- 1st rollout: From original prompt
- 2nd rollout: Prompt + 1st feedback in context
- Nth rollout: Accumulate feedback from N-1 rollouts

```
Rollout 1: Requires SSN → ❌ Failure
Rollout 2: [Knows SSN] Prepared → ✅ Success
Rollout 3: [Knows SSN] Prepared → ✅ Success
...(rapid adaptation within batch!)
```

📈 **Result: More high-quality samples per batch**

This is essentially an **online learning process**:

- **Externalized Learning:** Feedback accumulated in knowledge base after each rollout
- **Online RL:** Agent adapts its policy based on accumulated feedback within the batch

Method 2: In-Context Learning

⚠️ Common Misconception

"With long context, just put all history in and let the model automatically reason"
This is a serious misconception about context capabilities!

🔍 What Context Really Does

Nature: Retrieval, NOT reasoning engine

Mechanism: Key-value similarity matching (like RAG)

✅ **Good at:** Finding relevant information

❌ **Poor at:** Statistical aggregation & counting

⚠️ Real Case: Three-Call Limit

Rule: Max 3 calls to same merchant

Context: Trajectory has multiple Xfinity calls

Problem:

- Must scan entire trajectory to count
- Easily miscounts → makes 4th call
- Even if correct, wastes reasoning tokens

Cost: O(trajectory length) per decision

System Hint: Making Implicit State Explicit

Solution: Pre-aggregate information → Reduce O(n) to O(1) context lookups

💡 How System Hint Works

```
<system_hint>
Tool call summary:
- 'phone_call' called 3 times
  - Xfinity: 3 times (limit reached)

Constraint check:
- Cannot call Xfinity again
</system_hint>
```

✅ **Benefit:**

- Complexity: O(n) → O(1)
- Model uses aggregated info directly
- No scanning or counting needed

📋 Four Types of System Hints

1. Task Planning

```
TODO: [✅] Call customer service
      [ ] Call retention dept
```

2. Side-Channel Info

```
[2025-06-25 11:00:20] User message
```

3. Environment State

```
Current dir: /home/ubuntu
OS: Ubuntu 24.04
```

4. LLM-Generated Summary

```
Conversation summary:
User wants $79 Xfinity plan with all current features
```

Method 3: Externalized Learning (Knowledge Base)

🚫 NEVER Store Raw Cases Directly in Knowledge Base

Storing raw dialogues/cases without distillation leads to incomplete retrieval and wrong conclusions

🐱 Case 1: Cat Counting Problem

Scenario:
100 cases: 90 black cats, 10 white cats (all separate)
Question: "What's the ratio?"

❌ Raw Storage Problem:

- Top-k=20 retrieves partial cases only
- Incomplete sample → Wrong inference

✅ Distilled Approach:

```
"Total 100 cats:
90 black (90%), 10 white (10%)"
```

→ Single retrieval, accurate!

💼 Case 2: Discount Rule Error

Scenario:
3 cases: Veteran John ✅, Doctor Sarah ✅, Teacher Mike ❌
Question: "I'm a nurse, discount?"

❌ Raw Storage Problem:

- "Nurse" ≈ "Doctor" → Retrieves Sarah only
- Cases A, C missed → Wrong inference

✅ Distilled Approach:

```
"Xfinity discount: ONLY
veterans & doctors qualify"
```

→ Complete rule, correct answer!

Active Knowledge Distillation: Compression is Understanding

Core Principle: Invest extra compute now (LLM summarization) → Save reasoning tokens later

💡 Why Distillation?

❌ Raw trajectory (3 calls):

```
10:00 Call Xfinity (billing)
10:30 Call Xfinity (transfer)
11:00 Call Xfinity (negotiate)
```

Model must scan O(n) to count

✅ After distillation:

"Called Xfinity 3 times (limit)"

O(1) lookup, instant recognition

📊 Three Levels of Knowledge Distillation

1. Statistical Aggregation

100 cases → "90% black, 10% white"

Reduce density, improve retrieval

2. Rule Distillation

3 cases → "Only veterans & doctors"

Leap from cases to abstract rules

3. Structured Knowledge Extraction

- RAPTOR: Tree summaries
- GraphRAG: Entity networks

Summary: 3 Paradigms of Agent Continual Learning

Paradigm 1: Post-Training

Core Finding: SFT memorizes, RL generalizes

- **SFT**: Solidifies formats and protocols, high sample efficiency
- **RL**: Learns transferable strategies, out-of-distribution robust

Paradigm 2: In-Context Learning

Core Insight: Context ≠ Memory

- **Nature**: attention is similar to RAG
- **Methods**: system hints, explicit summarization

Paradigm 3: Externalized Learning

3.1 Knowledge Base

Advantages: leverages extra compute for knowledge extraction
Methods: contextual retrieval, RAPTOR hierarchical summaries

3.2 Tool Generation

Advantages: Codifies processes, efficient, reliable, composable
Philosophy: Minimal predefinition + Maximum self-evolution (Alita)

Summary: Self-Evolving Real-Time Agents

Part I: Real-Time Interaction

Think While Listening, Speak While Thinking

❌ **Problem**

Serial architecture: VAD waits → ASR transcribes → LLM thinks → TTS speaks

✅ **Solution**

- **Perception:** Streaming model produces context-aware transcription and acoustic events
- **Thinking:** Event-driven, can think while listening and speaking

💡 **Example: Telecom Plan Query - No Awkward Silence**

O "Should I order this plan?"
T_1 (fast 0.5s) Need more time
A_1 "Let me check the details..."
T_2 (slow 5s) Analyze plan...
A_2 "Yes, saves $30/month!"

Part II: Learning from Experience

Learn While Acting

❌ **Problem**

Fixed models cannot learn from experience after deployment
Big world: business processes are dynamic & non-public

✅ **Solution**

- **Post-Training:** Learn from interactions via RL
- **In-Context:** Aggregate info via system hints
- **Externalized:** Distill knowledge, generate tools

💡 **Example: Credit Card Verification**

1st call: ❌ Doesn't have last 4 digits of credit card
Learn: Store "Xfinity needs last 4..."
2nd call: ✅ Proactively requests it
→ Experience-based improvement with high sample efficiency

> *"We want AI agents that can discover like we can, not which contain what we have discovered." — Richard Sutton*

Thank You!

Self-Evolving Real-Time Agents

💭 **Think While Listening** | 🗣️ **Speak While Thinking** | 🎯 **Learn While Acting**

Bojie Li

🌲 Pine AI

AI Agent that makes calls and uses computers to get things done

Your personal assistant to contact customer service on your behalf

💰 **Lower bills** ❌ **Cancel subscriptions** 📋 **File complaints** 💵 **Get refunds**

🔗 **19pine.ai**

Powered by Slidev