A Reflective CLIP Agent that Learns From Its Own Fails

Kelian Schulz

April 23, 2025

Abstract

Can an image-search agent figure out when it totally messes up — and do better next time? I built one that gives it a shot. It starts with a goal (like "a fire truck in the rain"), uses CLIP to search for matching images, and if it flops, GPT steps in to rewrite the goal. I thought giving GPT strict rules would help. It didn't. Letting it rephrase more freely actually worked better. This paper is about that shift — and what I learned about trusting the process.

1 Intro

Multimodal agents — like ones powered by CLIP — are super flexible. They can take a text prompt and find matching images.But when they don't find anything useful, they usually just... stop. No reflection. No second try. I wanted to change that. So I built a basic loop: Start with a goal, search with CLIP, check if anything hits — and if not, ask GPT to rephrase the goal. Then try again. That's it. Simple, but kind of powerful.

2 What's Already Out There

CLIP and Vision Models. CLIP is the go-to for zero-shot image-text stuff. It's super handy when you don't want to lock into predefined labels.

Changing Goals. There's been some work on agents that adapt their plans, mostly with reinforcement learning. But letting a language model rethink the actual goal? Thats new.

Prompt Tweaks. Prompt engineering is its own art now. People get cool results by carefully crafting instructions for GPT. But too much control can also kill creativity — especially in small, simple datasets like CIFAR-10. **Knowing When You Fail.** Most systems rely on outside rules to figure out when they're stuck. I let the agent decide: If its CLIP scores are too low, it counts as a fail.

3 How It Works

The agent runs through this loop:

1. **Image Search with CLIP.** I use CLIP (ViT-B/32) to turn both the goal phrase and all CIFAR-10 images into vectors. Then I check cosine similarity.
2. **Failure Check.** If fewer than 5 images score above 0.28, it's a flop. Time to rethink.
3. **Goal Rethink with GPT.** I send the goal to GPT-3.5 and ask it to come up with a few short alternatives.

Example Prompt

To generate new goals, the agent sends GPT a short instruction based on the original phrase and a list of previously tested suggestions. The prompt looks like this:

```
Original: a fire truck in the rain
Tried: "rainy␣fire␣truck", "wet␣fire␣truck", "wet␣emergency␣vehicle"
Suggest 1 short visual phrase (max 4 words) close in meaning.
Use CIFAR-10 terms. Avoid tried. No quotes. Just the phrase.
```

4 A Quick Run

Here's what happened when I tested the agent on the goal "a fire truck in the rain":

```
[
    {"goal": "a␣fire␣truck␣in␣the␣rain", "n_hits": 0, "avg_score": 0.0},
    {"goal": "rainy␣fire␣truck", "n_hits": 2, "avg_score": 0.29},
    {"goal": "wet␣fire␣truck", "n_hits": 1, "avg_score": 0.287},
    {"goal": "wet␣emergency␣vehicle", "n_hits": 0, "avg_score": 0.0},
    {"goal": "rainy␣fire␣engine", "n_hits": 11, "avg_score": 0.286}
]
```

Eventually, it found a solid match with "rainy fire engine". With 11 hits.

Figure 1: Images returned for the final goal: "rainy fire engine"

5 What I Learned

At first, I wanted GPT to act like a surgical tool. **I broke the goal into "object", "scene", "action"**". I told GPT: "Only change the scene." I gave it examples. Avoid-lists. Whitelists. Semantic filters. **But the more I tightened the rules, the more GPT froze**. It stopped being helpful. The suggestions became generic, safe, or completely off-track — like **"underwater dive expedition" or "carriage"**. **I realized I was trying to make GPT act like a rule-based system. But GPT isn't a tool for control. It's a tool for variation — for language.** When I let it rephrase the whole goal freely — no constraints, no scene-labels — it suddenly worked again. "Rainy fire engine." Boom. Eleven matches. That's more than any engineered prompt had delivered. So yeah: I learned that in this setup, **trusting GPT a bit more** gave better results than micromanaging it.

6 Wrapping Up

In the end, I built a loop: fail, reflect, retry. It's simple. But it works — not because of some clever constraint logic, but because I gave the system space to reflect and adapt in its own language. That's not just a lesson for vision agents. That's a lesson for me, too. Sometimes, the real progress happens not when you control more — but when you learn to let go.