EDGE RESEARCH · TECHNICAL REPORT 05

Instructions as programs for live video

How a typed instruction, an answer shape and a validator become an app that runs over a live stream, and what still needs engineering.

Edge Research, Edge Video AI Version 1.0 · 15 Nov 2026

Status: Edge Research technical report. Not peer reviewed. It describes Edge's own systems and internal measurements; every number carries its scope. Third-party work is cited with links.

ABSTRACT

A language model takes an instruction and returns an answer. Edge Video AI applies the same pattern to live video: an instruction, video evidence and an enforced answer shape, followed by application code that checks, remembers and routes the answer. We describe how this works in three places in our system: the built-in enrichment task, where the instruction and answer shape are fixed by engineers; channel guidance, where a channel owner's plain-language preferences are turned into instructions stored with the channel and applied without a code change; and a builder interface, where a person types what they want, the vision language model (VLM) watches the stream and the language model underneath builds the app. We set out which changes need only a new instruction, which need a new answer shape and validator, and which need new engineering, and we describe real instructions typed into that builder interface during a recorded session. We also list what we have not yet shown.

1. Introduction

The most useful thing about language models is that you can change what they do by changing what you say. We wanted the same property for live video: to be able to say "show me the brands on screen" or "build a panel with the ad categories for this moment" and have it happen over a live stream, without a new release.

The honest version of that promise has conditions. An instruction can change what the model looks for and how it answers. It cannot make the model see what is not there, and it cannot by itself create a new place for the answer to go. This report is about where those lines fall.

2. Related work

Instruction following in language models was shaped by work such as InstructGPT [1]. Using language models to compose tools and programs is explored by ReAct [2], Toolformer [3], and, for vision, VisProg [4] and ViperGPT [5], where a language model produces a program that calls visual modules. Visual instruction tuning [6] trains vision language models to follow instructions about images directly. Our system combines a vision-language perception layer with a language layer that builds apps, as described in Technical report 03.

3. The pattern

Every programmable behaviour in our system has four parts:

Part	What it is	Example
Instruction	What to look for and how to choose	"Return grounded keywords and allowlisted categories; do not name individuals."
Evidence	The video, what is said, and any prior context	The current segment, its transcript, candidate keywords
Answer shape	A strict schema the answer must match	Keywords, keyword–context pairs with one category id each
Checks and routing	Code that validates, remembers and sends the answer somewhere	Category ids checked against the allowlist, then held for the ad system

Change the instruction and the behaviour changes within the same answer shape. Change the answer shape and you also need new checks and somewhere for the new fields to go.

4. Three places instructions live

4.1 Built-in tasks

The enrichment task described in Technical report 02 has an instruction and answer shape defined by our engineers. Brand recognition has its own, stricter instruction and its own answer shape. These are the foundations other behaviour builds on.

4.2 Channel guidance

A channel owner can describe, in plain language, what they want: topics to favour or avoid, brands they work with, the language of the audience. A language model turns that guidance into short, specific instructions for each decision the shopping path makes: analysing the scene, choosing a product, choosing a deal, choosing between candidates. These instructions are stored with the channel's settings and applied to each relevant task, with priority over the general instructions, every time the channel's settings are refreshed. The owner changes behaviour without anyone editing code. This is the clearest case of an instruction acting as a program in our system today.

4.3 The builder interface

In the builder interface, a person points the system at a live stream and types what they want. The VLM watches the stream; the language model underneath builds the app, a panel beside the picture or an overlay on it; and the app can be saved and switched on over the live stream. In a recorded builder session on a live sports stream, instructions of this kind were typed and produced working apps. These two are as shown in our explainer film:

> *Build an ad enrichment panel: show the IAB categories, scene keywords, brands on screen and a brand safety level for this moment.*

> *Now change it to brand recognition: list every brand visible on screen and where it appears.*

The second instruction changed the app on the same stream without new code. Both apps read the same underlying signals; the language model built different views over them.

5. What changes with words, and what needs engineering

Change	What it takes
Emphasise or avoid topics on a channel	Instruction only (channel guidance)
Prefer a list of brands on a channel	Instruction and channel settings
A new view over existing signals	Typed instruction
A different taxonomy version in the output	Supported setting, from the versions we already map
A new kind of output field	New answer shape, new checks, and somewhere for it to go
A new destination for answers	Engineering on the receiving side
Recognising exact content on a TV	Registration and fingerprints, not an instruction (see Technical report 01)

6. Keeping instructions safe

Two risks come with programmable behaviour. The first is that words in the stream act as instructions. Our perception instructions treat transcripts and on-screen text as evidence, never as commands. The second is that a typed instruction asks for something the model cannot ground, such as an exact product match from a blurry shot. Our answer shapes allow "nothing found", and our apps are built to show nothing rather than guess.

7. Limitations

- **Not every use is one instruction away.** Enrichment, brand recognition and shopping have different instructions, answer shapes and surrounding code. A genuinely new use usually needs all four parts of the pattern.
- **Shopping is shown as demos.** Shopping apps built by typing detect products and scenes; connecting every app to live product catalogues is ongoing work, and we make no claims

about exact-item matching.

- **Recorded sessions are evidence, not benchmarks.** The builder examples above show that typed instructions produced working apps on a live stream. They are not a measure of how often that succeeds.

References

1. L. Ouyang et al. Training Language Models to Follow Instructions with Human Feedback. NeurIPS 2022. arXiv:2203.02155
2. S. Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. ICLR 2023. arXiv:2210.03629
3. T. Schick et al. Toolformer: Language Models Can Teach Themselves to Use Tools. NeurIPS 2023. arXiv:2302.04761
4. T. Gupta and A. Kembhavi. Visual Programming: Compositional Visual Reasoning Without Training. CVPR 2023. arXiv:2211.11559
5. D. Surís, S. Menon and C. Vondrick. ViperGPT: Visual Inference via Python Execution for Reasoning. ICCV 2023. arXiv:2303.08128
6. H. Liu, C. Li, Q. Wu and Y. J. Lee. Visual Instruction Tuning. NeurIPS 2023. arXiv:2304.08485