Reel Breakdown Skill Guide

Install the free recast skill and turn any viral reel into a reusable style prompt, all on your own machine.

Why I break a reel down before I remake it

A reel that performs is evidence of a format. The hook lands at a certain second, the picture changes at a certain rhythm, the captions sit in a certain place. All of that is copyable. The topic, the footage and the voice are not, and they are the only parts most people bother to change.

Watching a reel back twenty times and writing notes in a doc is the slow version of this. You end up with impressions like "fast cuts, big text" that stop being useful the moment you sit down to rebuild it. An impression is not a specification.

This skill gives you the specification. It pulls the reel apart into frames, cut times and a word-timed transcript, you read that evidence and write it up, and the last step turns the whole thing into a prompt document you paste into a fresh chat with a new topic. One reel in, one reusable format out. Everything runs on your laptop and nothing is uploaded.

01

How it actually works

1. Evidence. One script pulls a frame per second out of the reel, stitches those frames into contact sheets of 18 so you can scan a whole minute in one look, re-samples the first three seconds at 10 frames per second so the hook is visible frame by frame, lists the times where the picture changes sharply, and transcribes the audio with a local whisper.cpp model into word-level timestamps.
2. Reading. Claude opens every contact sheet and the hook sheet, opens single frames when a sheet is too small to judge a material or a label, reads the transcript, and writes an Analysis.md with a scene map, the visual grammar worth keeping, the pacing, and a list of every claim the reel made on screen that nobody has verified.
3. Rebuild and proof. Claude writes a short reference animation in that style, 8 to 12 seconds, built entirely in code. A checker runs it in headless Chrome, renders stills and tests that the same timestamp always gives back the same pixels. Only when that check passes does the last script assemble the prompt document.

02

The one thing almost everyone gets wrong

They copy the reel's numbers. A reel says "this made $40,000 in a week" and the remake says it too, because it was in the transcript and the transcript was sitting right there in the evidence folder. Now you

are publishing somebody else's unverified claim under your own face, and you are the one who has to defend it.

> **WATCH OUT**
>
> The Analysis.md has a section called "Claims not verified" for exactly this. Every figure, statistic, price and result the reel put on screen goes in that list, and none of it carries into your version. The same rule covers the footage, the logos, the soundtrack, the narration and the creator's name. The reel is evidence of a style. It is not material to copy.

03

What is in the skill folder

THE FOLDER LAYOUT

```
~/.claude/skills/recast/
  SKILL.md                      the instructions Claude reads
  scripts/analyze_reel.py       frames, cuts, audio, transcript
  scripts/check_style.py        runs a style file in headless Chrome
  scripts/assemble_prompt.py    writes the TXT and the DOCX
  assets/prompt_template.txt    the document shape
  assets/lib/stage.js           the STAGE kit (K)
  assets/lib/sound.js           the SOUND kit
  assets/lib/three.core.min.js  three.js 0.187.0, core only
  assets/lib/three.module.min.js
  assets/lib/SpaceGrotesk.ttf   the bundled fonts, with OFL licenses
  assets/lib/IBMPlexMono-Medium.ttf
  assets/lib/IBMPlexMono-SemiBold.ttf
  references/exploded_view_style.js
  references/exploded_view_style_card.txt
```

The three scripts are the whole machine. **analyze_reel.py** needs ffmpeg and ffprobe on your PATH and writes a folder called `<reel name>-analysis` next to the reel, holding meta.json, a frames folder, the contact sheets, hook.jpg, cuts.txt, audio.wav at 16 kHz mono, transcript.txt, transcript.srt and words.json. **check_style.py** needs Python Playwright and writes check.json plus a stills folder. **assemble_prompt.py** writes the TXT, converts it to DOCX with the macOS textutil command, and prints both paths.

Two files are the templates everything else gets measured against. Claude writes an Analysis.md in this shape:

ANALYSIS.MD

```
# Analysis

## Source
File name, duration, size, fps. This report describes the file. It does not
claim which app or which creator made it.

## What the style is
One or two paragraphs. 3D, 2D or mixed. What carries the look.

## Scene map
| Time | What is visible | What the motion is for |
|------|-----------------|------------------------|
| 0.0 to 2.4 | ... | ... |

## Visual grammar to keep
Background, palette, materials, light, camera habits, scale changes, how text
and labels are drawn, how things move.

## Audio and pacing
Narration or not, words per minute, where the hook lands, how often the picture
changes.

## What to reuse and what to replace
The look stays. The topic and its objects get replaced.

## Claims not verified
Every number or fact the reel put on screen, listed and marked unverified.
```

And a style card in this shape. All ten fields have to carry text or assemble_prompt.py refuses to run:

STYLE_CARD.TXT

```
Name: Paper Cut Ledger
Family: 2D collage
Use: explainers where one number has to land
Look: ...
Motion: ...
Light: ...
Materials: ...
Camera: ...
Finish: ...
Good for: ...
```

Set it up in Claude Code or Codex

1. Put the skill folder at `~/.claude/skills/recast/` for Claude Code. The folder name has to match the `name:` line in the front matter of SKILL.md, which is `recast`. A folder called recast-skill or Recast will not be found.
2. For Codex, drop the same folder in your project or at `.codex/skills/recast/`, and add one line to AGENTS.md pointing at it so the agent knows it is there.
3. Install ffmpeg. On a Mac that is `brew install ffmpeg`. The analyze script checks for both ffmpeg and ffprobe before it touches your file and prints this exact fix if either one is missing.
4. Install Playwright for Python with `pip3 install playwright`. The style checker drives headless Chrome through it. You only need it for the rebuild step, so a breakdown-only run works without it.
5. Point the skill at a whisper.cpp model. It looks for one at a few known paths by itself, and you can override that with a `WHISPER_MODEL` environment variable or the `--model` flag. Ask Claude to add the export line to your shell profile, then open a new terminal window so it takes effect.
6. Run the first test on a short reel you already have on disk. In Claude Code, say "recast this" and give the full path to the MP4. There is no download step built in, so if your reel is a link, save the file first and hand over the path.
7. A successful first run prints the analysis folder, then a check report whose last lines read `"deterministic": true` and `"ok": true`, then two file paths ending in _prompt.txt and _prompt.docx. If you get the first two and not the third, the style file failed its check and Claude will keep fixing it until it passes.

> 💡 **QUICK TIPS**
>
> - Keep the analysis folder. When a style stops working months later, the contact sheets are the only record of what you were aiming at.
> - Name the style after the look, never after the creator or the brand whose reel you studied.
> - A portrait reference keeps text 150 px from the top, 170 px from the bottom and 60 px from the sides, with headline words no smaller than 56 px. Captions that crop on a phone are the most common thing to fix on a second pass.

04

The commands I actually run

THE WHOLE PASS, ONE LINE

```
Recast this reel: [FULL PATH TO THE MP4]. Name the style yourself and tell me what you picked.
```

BREAKDOWN ONLY, NO REBUILD

```
Run the recast evidence step on [FULL PATH TO THE MP4], then write Analysis.md and stop. I want the scene map and the Claims not verified list before you write any code.
```

PULL THE EVIDENCE BY HAND

```
python3 ~/.claude/skills/recast/scripts/analyze_reel.py "[FULL PATH TO THE MP4]"
```

STUDY JUST THE HOOK

```
Open hook.jpg in [PATH TO THE ANALYSIS FOLDER] and tell me, frame by frame, what is on screen for the first three seconds and which frame the first spoken word lands on.
```

REUSE A STYLE ON A NEW TOPIC

```
Here is the prompt document for [STYLE NAME]. Build a [LENGTH] second video in that exact style about [YOUR TOPIC], using the voiceover at [PATH TO THE AUDIO]. Keep the look, replace every piece of content.
```

When it breaks

ERROR	FIX
ffmpeg is not installed. Fix: brew install ffmpeg	Run brew install ffmpeg. The script checks for ffmpeg and ffprobe, so install the full package rather than a single binary.
No such file: /path/to/reel.mp4	The path is wrong, or the reel is still a link. Save the MP4 to disk and pass the full path, quoted if it has spaces in it.
This file has no video stream.	ffprobe found audio only. You have handed over an audio file, or a container whose video track did not copy. Re-export the reel as an MP4.
Python Playwright is missing. Fix: pip3 install playwright	Run pip3 install playwright. If Chrome itself is missing afterwards, run playwright install chromium.
The style card is missing: Light, Camera	All ten style card fields need text after the colon: Name, Family, Use, Look, Motion, Light, Materials, Camera, Finish and Good for.
The style file uses Math.random() which breaks frame-by-frame rendering.	Every frame has to be reproducible from its timestamp alone. Replace Math.random, Date.now and new Date with K.rng(seed) and arithmetic on t.
check.json says "deterministic": false	Something in the style builds up between calls, usually a counter or a position that gets added to instead of set. Make update(t) set the whole scene from t every time it runs.
Name_prompt.txt already exists. Pick another name, or pass --force to replace it.	The script will not overwrite a prompt you already made. Give the style a different name, or add --force when you do mean to replace it.
DOCX: not written	The DOCX conversion runs the macOS textutil command. The TXT is already written and holds the same text, so paste from that instead.

Ready to go deeper?

Want the videos made for you? Paste any link or idea into Content Engine and get a published video in under 3 minutes, with DM automation armed to capture leads.

Meet Content Engine →

https://builds.digicuratoragency.com/content-engine/