Reel Breakdown Playbook

Turn any viral reel into frames, cut times and a scene map you can rebuild on, all on your own machine.

Why the copy flops

I used to remake reels by watching them four or five times and guessing. The guess was always about the wrong thing. I matched the footage and the look, published, and watched my version do a third of the original's numbers. I was copying the surface. What made the original work sat underneath it: when the hook ended, how often the picture changed, where the caption sat on the screen.

The fix is a breakdown pass before I write anything. One command turns the reel into evidence I can read: frames, cut times, and a transcript with word level timestamps. From that I write a scene map, rebuild on the skeleton with my own topic, and check my version against the original before it goes out. I have made close to a hundred reels this way.

All of it runs on your own machine. The reel is never uploaded anywhere, which matters, because the file is usually somebody else's video sitting in your downloads folder.

01

How it actually works

1. Pull the evidence. One Python command runs ffmpeg over the reel and writes an analysis folder next to it: one frame per second, contact sheets of 18 frames each, the first three seconds at 10 frames per second, a list of cut times, and a transcript with word level timestamps from a local whisper.cpp model.
2. Read it and write the scene map. You scan the contact sheets and the hook sheet, then fill in a short table: time range, what is on screen, what the voice is doing, and what that beat is for. The last column is the only one that carries over to your topic.
3. Rebuild, then check. You write your version against the scene map, keeping the beat order and the hook length, then put your cut next to the original and compare before you publish. A hook that grew by one sentence is obvious side by side and invisible on its own.

02

The one thing almost everyone gets wrong

They judge the hook from the one frame per second sheet. At that sampling rate a three second hook is three pictures, and three pictures cannot show you a cut at 0.4 seconds, a caption that flashes for half a second, or a zoom that lands on the word that matters. That is why the breakdown grabs the first three seconds separately at 10 frames per second. Thirty frames of the opening is where the format is actually decided, and it is the sheet I spend the most time on.

> **WATCH OUT**
>
> Never carry the original's numbers across. Every statistic a reel shows on screen is unverified until you check it yourself, and "I saw it in a reel" is not a source. Keep a list of those claims in your analysis file so you know exactly what you are leaving behind.

03

The files

Three files, and the folder of evidence the script writes. Paste the first two in as they are. The third is the template I fill in by hand.

~/.CLAUDE/SKILLS/REEL-BREAKDOWN/SKILL.MD

````
---
name: reel-breakdown
description: >
  Break a reel down into frames, cut times and a timestamped transcript, then write the
scene
  map. Use when the user drops an MP4 and asks what made it work, what the structure
is, or
  asks to rebuild it on a new topic.
---

# Reel Breakdown

A reel goes in. A folder of readable evidence comes out, then a scene map you can build
on.
Everything runs on this machine. Nothing is uploaded.

## 1. Pull the evidence

```bash
python3 ~/.claude/skills/reel-breakdown/scripts/analyze_reel.py "<reel.mp4>"
```

## 2. Read it

Open every `seconds-N.jpg` sheet and `hook.jpg` with the Read tool. Open single frames
from
`frames/` when a sheet is too small to judge a caption or a transition. Read
`transcript.txt`
and `cuts.txt`.

## 3. Write the scene map

Write `Analysis.md` in the analysis folder using the template below. Only describe what
is in
the frames and the transcript. Use the times the evidence gives. Never invent
narration.

## 4. Rebuild

Write the new script against the scene map, one beat per row, same beat purposes, new
topic.
Keep the hook inside the original's hook length. List every claim from the original
that you
have not verified, and do not carry any of them over.
````

~/.CLAUDE/SKILLS/REEL-BREAKDOWN/SCRIPTS/ANALYZE_REEL.PY

```
#!/usr/bin/env python3
"""Reel breakdown: pull the evidence out of a reel so it can be studied. Everything
stays on this machine.

  python3 analyze_reel.py /path/to/reel.mp4 [--out DIR] [--model /path/to/ggml-
model.bin]

Writes into DIR (default: <folder of the reel>/<reel name>-analysis):
  meta.json              duration, size, fps, audio info
  frames/second-NNN.jpg  one frame per second
  seconds-N.jpg          contact sheets of those frames, 18 per sheet
  hook.jpg               the first 3 seconds at 10 frames per second
  cuts.txt               times where the picture changes sharply
  audio.wav              16 kHz mono audio
  transcript.txt/.srt    speech transcript (local whisper.cpp), if a model is found
  words.json             word-level timestamps
"""
import argparse, json, os, re, shutil, subprocess, sys
from pathlib import Path

MODELS = [os.environ.get('WHISPER_MODEL', ''),
          str(Path.home() / 'models/ggml-small.en.bin')]

def run(cmd, **kw):
    return subprocess.run(cmd, capture_output=True, text=True, **kw)

def main():
    ap = argparse.ArgumentParser(); ap.add_argument('video', type=Path);
ap.add_argument('--out', type=Path); ap.add_argument('--model')
    a = ap.parse_args()
    if not a.video.is_file(): sys.exit(f'No such file: {a.video}')
    for tool in ('ffmpeg', 'ffprobe'):
        if not shutil.which(tool): sys.exit(f'{tool} is not installed. Fix: brew
install ffmpeg')
    out = a.out or a.video.parent / f'{a.video.stem}-analysis'
    (out / 'frames').mkdir(parents=True, exist_ok=True)

    probe = json.loads(run(['ffprobe', '-v', 'error', '-show_format', '-show_streams',
'-of', 'json', str(a.video)]).stdout)
    v = next((s for s in probe['streams'] if s['codec_type'] == 'video'), None)
    au = next((s for s in probe['streams'] if s['codec_type'] == 'audio'), None)
    if not v: sys.exit('This file has no video stream.')
    num, den = (v.get('avg_frame_rate') or '30/1').split('/')
    meta = {'file': a.video.name, 'duration': float(probe['format']['duration']),
'width': v['width'], 'height': v['height'],
            'fps': round(float(num) / float(den or 1), 3), 'has_audio': bool(au),
            'aspect': 'portrait' if v['height'] > v['width'] else 'landscape' if
v['width'] > v['height'] else 'square'}

    V = str(a.video)
    for f in (out / 'frames').glob('second-*.jpg'): f.unlink()
    run(['ffmpeg', '-y', '-v', 'error', '-i', V, '-vf', 'fps=1,scale=540:-2', '-q:v',
'3', str(out / 'frames/second-%03d.jpg')])
    n = len(list((out / 'frames').glob('second-*.jpg'))); meta['frames_sampled'] = n
    for f in out.glob('seconds-*.jpg'): f.unlink()
```

```
    run(['ffmpeg', '-y', '-v', 'error', '-i', V, '-vf', 'fps=1,scale=300:-2,tile=6x3',
'-q:v', '3', str(out / 'seconds-%d.jpg')])
    run(['ffmpeg', '-y', '-v', 'error', '-t', '3', '-i', V, '-vf',
'fps=10,scale=240:-2,tile=10x3', '-frames:v', '1', '-q:v', '3', str(out / 'hook.jpg')])
    sc = run(['ffmpeg', '-v', 'info', '-i', V, '-vf',
"select='gt(scene,0.3)',showinfo", '-an', '-f', 'null', '-'])
    cuts = [round(float(x), 2) for x in re.findall(r'pts_time:([0-9.]+)', sc.stderr)]
    (out / 'cuts.txt').write_text('\n'.join(f'{c:.2f}' for c in cuts) + '\n');
meta['likely_cuts'] = len(cuts)

    meta['transcript'] = 'none'
    if au:
        wav = out / 'audio.wav'
        run(['ffmpeg', '-y', '-v', 'error', '-i', V, '-ar', '16000', '-ac', '1',
str(wav)])
        model = next((p for p in [a.model or ''] + MODELS if p and Path(p).is_file()),
None)
        if not shutil.which('whisper-cli') or not model:
            meta['transcript'] = 'skipped: whisper-cli or a ggml model was not found
(pass --model or set WHISPER_MODEL)'
        else:
            run(['whisper-cli', '-m', model, '-f', str(wav), '-otxt', '-osrt', '-of',
str(out / 'transcript')])
            run(['whisper-cli', '-m', model, '-f', str(wav), '-ml', '1', '-sow', '-oj',
'-of', str(out / 'words_raw')])
            raw = out / 'words_raw.json'
            if raw.is_file():
                words = [{'w': s['text'].strip(), 't': s['offsets']['from'] / 1000,
'e': s['offsets']['to'] / 1000}
                         for s in json.loads(raw.read_text())['transcription'] if
s['text'].strip()]
                (out / 'words.json').write_text(json.dumps(words)); raw.unlink()
                meta['spoken_words'] = len(words); meta['transcript'] = f'automatic,
{Path(model).name}'
                if len(words) < 3: meta['transcript'] += '. Almost no speech found: the
reel may be music or ambience only.'
    (out / 'meta.json').write_text(json.dumps(meta, indent=1))
    print(json.dumps(meta, indent=1)); print('Analysis folder:', out)

if __name__ == '__main__':
    main()
```

ANALYSIS.MD (THE TEMPLATE I FILL IN)

```
# Analysis: <reel file name>

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

**What the format is.** One or two paragraphs. Talking head, screen capture, text on
footage,
or a mix. What carries the video.

## Scene map

| Time | On screen | Voiceover | What the beat is for |
|---|---|---|---|
| 0.0 to 2.3 | Close up, big caption | "Most creators do X wrong" | Name the mistake |
| 2.3 to 6.0 | Screen capture, cut every 1.8s | The correction | Show the proof |
| 6.0 to 11.4 | Three quick shots | The method | Make it look doable |
| 11.4 to end | Back to close up | The ask | CTA |

## Visual grammar to keep
Background, palette, caption placement and treatment, camera habits, how often the
picture changes.

## Audio and pacing
Words per minute. The second the hook ends. Average gap between cuts. Where the longest
shot sits.

## What to reuse and what to replace
Reuse: timing, cut rhythm, caption layout, beat order.
Replace: topic, footage, voiceover, music, on-screen claims, any name or brand.

## Claims not verified
Every number or fact the reel shows on screen, listed, none of them carried into my
version.
```

Set it up in Claude Code or Codex

1. Install ffmpeg first. The script exits with "ffmpeg is not installed" without it. On macOS that is `brew install ffmpeg`. On Windows, ask Claude to install ffmpeg and put it on your PATH, then open a new terminal so the change takes.
2. Get whisper.cpp working if you want the transcript. You need the `whisper-cli` binary on your PATH and a ggml model file on disk. Ask Claude Code: "build whisper.cpp, download ggml-small.en.bin, and tell me the full path to both". Then either pass `--model /path/to/ggml-small.en.bin` on every run, or ask Claude to add `WHISPER_MODEL` to your shell profile and open a new terminal. The frames, the contact sheets and the cut list all work without this, so do not let it block you on day one.
3. Make the skill folder. Claude Code reads skills from `~/.claude/skills/`, one folder per skill. Create `~/.claude/skills/reel-breakdown/`, put SKILL.md at the top of it and analyze_reel.py in a `scripts/` folder inside it.
4. Check the folder name matches the skill name. The folder is `reel-breakdown` and the `name:` line in SKILL.md says `reel-breakdown`. If those two differ the skill will not load, and nothing will tell you why.
5. On Codex, put the same folder in the project instead, at `.codex/skills/reel-breakdown/`, and add one line to AGENTS.md pointing at it so the agent knows it is there.
6. Run the first test on a reel you already have on disk: `python3 ~/.claude/skills/reel-breakdown/scripts/analyze_reel.py ~/Downloads/reel.mp4`
7. A good first run prints a JSON block with duration, width, height, fps, frames_sampled, likely_cuts and spoken_words, then the path to the analysis folder. Open that folder, look at seconds-1.jpg and hook.jpg, and you are done setting up.

> 💡 **QUICK TIPS**
>
> - Drag the video file into the terminal instead of typing the path. It quotes spaces for you and kills most "No such file" errors.
> - Keep each analysis folder next to the reel it came from. Six months later the folder name is the only thing that tells you which video it was.
> - Start with a reel in your own niche that you already know performed. You will read the sheets faster when you recognise the content.

04

The commands I actually run

> **BREAK THE REEL DOWN**
>
> `Run the reel breakdown skill on [PATH TO REEL.MP4]. When it finishes, read every seconds-N.jpg sheet and hook.jpg, then tell me three numbers: the second the hook ends, how many cuts there are, and the words per minute.`

WRITE THE SCENE MAP

Using the analysis folder for [REEL NAME], write Analysis.md with the scene map table, the visual grammar to keep, the audio and pacing numbers, and a list of every claim the reel makes on screen. Only use what is in the frames and the transcript. Do not guess at anything you cannot see.

TEAR APART THE HOOK

Look only at hook.jpg for [REEL NAME]. Describe what happens in each of the first thirty frames: what is on screen, when the first cut lands, when the caption appears, and what the first spoken words are. Then tell me in one sentence what the hook promises.

REBUILD ON MY TOPIC

Write my version of this reel about [MY TOPIC]. Follow the scene map beat for beat: same number of beats, same purpose per beat, hook no longer than [N] seconds, cuts at roughly the same times. Use my own examples. Carry none of the original's claims, footage, names or wording across.

CHECK IT BEFORE I PUBLISH

Put my cut of [MY TOPIC] next to the original's contact sheets and compare the structure. Tell me where my hook is longer, where my pacing drifts, and which beat lost its purpose. List the fixes in order of how much they cost me.

When it breaks

ERROR	FIX
No such file: /path/to/reel.mp4	The path is wrong, or it has spaces in it. Drag the file into the terminal window instead of typing the path.
ffmpeg is not installed. Fix: brew install ffmpeg	Install ffmpeg, then open a new terminal window. A terminal that was already open will not see the new PATH.
This file has no video stream.	You passed an audio file, or the download failed halfway. Re-download the reel as an mp4 and check it plays before running the script again.
transcript: skipped: whisper-cli or a ggml model was not found	whisper-cli is not on your PATH, or there is no ggml model on disk. Pass --model /path/to/ggml-small.en.bin, or set WHISPER_MODEL and open a new terminal. Everything except the transcript still ran.
Almost no speech found: the reel may be music or ambience only.	Nothing is broken. That reel has no narration, so build the scene map from the frames and the cut times, and do not invent a voiceover that was never there.
cuts.txt is empty or looks too short	Scene detection fires on hard cuts. Slow crossfades and push transitions do not trip it, so read the list as a floor and confirm the real count against the contact sheets.
The contact sheets are too small to read a caption	Open the single frames in frames/ instead. They are saved at 540 pixels wide, the sheets at 300.
Python Playwright is missing. Fix: pip3 install playwright	Only the check step needs it. Run pip3 install playwright, and make sure Google Chrome is installed, because the check drives your real Chrome.

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/