AutoCut: End-to-end advertisement video editing based on multimodal discretization and controllable generation

Supplementary Material

1. Appendix

This appendix is organized as follows:

- **1.1 Model Architecture and Hyperparameters** model components and key training settings.
- **1.2 Post-Processing and Rendering Pipeline** steps for converting predictions into final ad videos.
- **1.3 GPT-4o Evaluation Prompts** prompts used for GPT-4o-based automatic evaluation.
- **1.4 Human Evaluation Criteria** criteria and guidelines for the pairwise user study.
- **1.5 Additional Human Study with Gemini 3** criteria and guidelines for the pairwise user study.
- **1.6 Dataset Construction Details** details of alignment data, SFT filtering, and task formatting.

1.1. Model Architecture and Hyperparameters

Table 1. Hyperparameters for the video and audio RQ-VAE quantizers.

RQ-VAE Configs	Video	Audio
Input feature size	128	2048
Quantization heads	8	8
Codebook size	256	256
Codebook dim	128	256
Shared codebook	False	False
Encoder MLP size	[512, 512]	[1024, 512]
Decoder MLP size	[512, 512]	[512, 1024]
Loss type	cosine	cosine
Reconstruction weight	1.0	1.0
Commitment weight	0.0	0.0
Optimizer weight decay	$1e^{-4}$	$1e^{-4}$
Initial learning rate	$1e^{-3}$	$1e^{-3}$
Min learning rate	$2e^{-5}$	$2e^{-5}$
Batch size	8192	8192
Max epochs	20	50

To convert continuous visual and audio embeddings into discrete tokens, we train two modality-specific Residual Quantized Variational Autoencoders (RQ-VAEs). Both models use multi-head residual quantization, cosine reconstruction loss, and large-batch optimization, while differing in encoder–decoder capacity to match modality characteristics.

For **video**, 128-dimensional frame embeddings are quantized using an 8-head residual quantizer with a 256-entry codebook and lightweight two-layer MLP encoder–decoder. This configuration balances reconstruction fidelity and token compactness for downstream selection and ordering tasks.

For **audio**, the 2048-dimensional PANNs embeddings require a wider asymmetric MLP architecture, while retaining the same quantizer heads and codebook size. Training follows the same objective and optimizer settings but uses a longer schedule to accommodate higher input dimensionality.

Both quantizers are trained on large-scale advertisement datasets and monitored with Weights & Biases for stability. A consolidated comparison of all model hyperparameters is provided in Table 1.

Table 2. Hyperparameters used during the multimodal alignment and SFT stages.

Config	Alignment Stage	SFT Stage
Base model	Qwen3-8B	Aligned model
Template	qwen3	qwen3
Cutoff length	8096	4000
Packing	Enabled	Disabled
Per-device batch size	2	1
Grad. accumulation	1	1
Learning rate	5e–5	1e–5
Epochs	5	3
LR scheduler	Cosine	Cosine
Warmup ratio	0.1	0.1
Precision	bf16	bf16
Optimizer	AdamW	AdamW
Flash attention	fa3	fa3
Resize vocab	Enabled	Disabled

We adopt a two-stage training pipeline to integrate multimodal discrete tokens into the Qwen3-8B backbone and enable controllable editing. The first stage performs **multimodal alignment**, where the backbone is frozen except for the newly added multimodal embedding layers. This stage learns semantic correspondence across video, audio, and text tokens using a cosine learning-rate schedule, a long context window (cutoff 8096), and bf16 mixed precision. Training uses a per-device batch size of 2, no gradient accumulation, and runs for five epochs at a learning rate of 5e–5, with packing and DeepSpeed ZeRO-2 to improve memory and throughput.

Figure 1. Qualitative case studies of our automatic ad video editing pipeline. For each case (top: wireless earbuds, bottom: children's face cream), we show the provided product information, selected video clips, and the aligned script sentences. Each frame strip visualizes the model's clip selection and ordering, together with the corresponding sentence and clip-level timestamps, illustrating how the system produces a coherent, time-aligned ad from raw footage and product metadata.

The second stage applies **supervised fine-tuning (SFT)** on the aligned model. Unlike alignment, SFT updates all parameters and disables packing to preserve task structure. It is trained on curated datasets covering video selection, clip ordering, script generation, and BGM selection, using a shorter sequence length (cutoff 4000), a smaller learning rate (1e–5), and a per-device batch size of 1 for three epochs. The same optimizer, scheduler, and mixed-precision settings are used as in the alignment stage.

All training experiments for both stages are conducted on a cluster of 8 NVIDIA GPUs. All templates used during SFT follow the `qwen3` instruction format, and model checkpoints are logged using Weights & Biases. Table 2 summarizes the hyperparameters used in the alignment and SFT stages.

1.2. Post-Processing and Rendering Pipeline

In the main paper we have shown that our model achieves strong performance on all sub-tasks and outperforms comparable baselines, both in quantitative metrics and in our user study. Here, we detail the complete post-processing and rendering pipeline that converts model outputs into a final edited video. For clarity, AutoCut does not perform clip or BGM selection by directly predicting fixed database indices. Instead, the model outputs discrete token sequences that are decoded into target embeddings, and the final video or audio assets are retrieved from the material database through similarity search. This retrieval-based design allows the system to generalize to unseen clips and music tracks, rather than memorizing a closed set of asset IDs.

Editing scenarios and model outputs. We consider two typical usage scenarios.

(1) Script–driven editing. The user provides (i) product information, (ii) a pool of candidate product clips, and (iii) a target ad script. *Selection.* Given the script and clip pool, the model first performs a *select* task: it chooses clips that are relevant to the script. We constrain the number of selected clips to match the number of sentences in the script. *Sorting.* The model then performs a *sort* task on the selected clips, ordering them such that visual content aligns with the sentence-level script order, based on learned vision–language correlations. *BGM selection.* Finally, we feed the ordered clip sequence together with the script and product information into the model for *background-music (BGM) selection*. The model outputs a sequence of discrete audio tokens corresponding to a suitable BGM track for this rough cut.

(2) Footage–driven editing. In the second scenario, the user has already curated all relevant footage but has not yet written an ad script; effectively, the *select* step has been performed manually. Given the product information and the set of clips, the model first *generates an ad script* whose number of sentences matches the number of clips. We then apply the same *sort* and *BGM selection* steps as in the script–driven pipeline.

In both scenarios, the model outputs discrete video tokens (for clips), audio tokens (for BGM), and the text script.

Token decoding and retrieval. To map token sequences back to concrete media assets, we use the RQVAE models trained in the pre-processing stage. We first decode the discrete tokens into continuous embeddings, and then perform nearest-neighbor search in FAISS indices.

For **video tokens**, FAISS returns a pair (`frame_id`, `clip_id`) for each token. These identifiers are structured:

- `frame_id` is constructed from a base `photo_id` (the underlying source video) plus four digits encoding the frame index within that video.
- `clip_id` is constructed from the same `photo_id` followed by seven digits, where the first four digits denote the starting frame of the clip and the last three digits encode its duration (at 1 fps).

For **audio tokens**, FAISS returns an `audio_id` pointing to a BGM track in our library.

Rendering strategies: by frame vs. by clip. Using these identifiers, we support two strategies for assembling the final video.

By frame. We derive the starting frame from the last four digits of `frame_id` and use the stored clip length (in frames at 1 fps) to cut a contiguous segment from the original video. This provides frame-accurate control over clip boundaries.

By clip. We additionally maintain an alternative clip segmentation defined independently of script boundaries. Each source video is segmented into visually coherent clips based on changes in visual continuity: we detect boundaries where the similarity between consecutive frame embeddings drops sharply. For each such segment (computed at 20 fps), we average the frame embeddings to obtain a single clip embedding. At inference time, retrieval is performed over these "visual clips", and we stitch together the corresponding segments. This *by-clip* strategy is specifically designed to reduce visual jitter or abrupt cuts that may occur if we follow sentence boundaries alone without considering the intrinsic visual structure of the footage.

Subtitles, TTS, and BGM mixing. For each clip in the final sequence, we associate one sentence of the provided or generated script. We render the sentence as on-screen subtitles and synthesize a voice-over track using text-to-speech (TTS). In the current implementation we use a fixed "avatar" voice; extending the model to select an appropriate voice profile conditioned on the video content is left for

future work. Finally, we retrieve the BGM track using the predicted `audio_id` and mix it with the TTS audio when compositing the final video.

For all user-study videos and the demo results reported in the paper, we adopt the **script–driven** editing pipeline (given script) together with the **by-clip** rendering strategy.

1.3. GPT-4o Evaluation Prompts

We provide additional details on the automated evaluation procedures used to measure semantic alignment and script quality. Full prompts are included as separate text files; here we summarize their intended behavior.

VSC Scoring Prompt The Visual–Script Correlation (VSC) evaluator compares a single video frame with its corresponding script line. It is instructed to judge semantic consistency and output a discrete score in $\{0, 1, 2\}$, indicating mismatch, partial relevance, or strong alignment. The evaluator receives only the frame and the script line and is constrained to return a numerical score. The full prompt is provided in `vsc_prompt.txt`.

SQ Scoring Prompt The Script Quality (SQ) evaluator applies a 100-point rubric covering correctness, clarity, linguistic naturalness, selling-point communication, and timing/length alignment. It is given product metadata, a human reference script, the generated script, and a pre-computed line-level length-difference statistic. The evaluator outputs a structured JSON object with category-level scores and a brief justification. The full prompt is included in `sq_prompt.txt`.

To ensure consistent behavior across evaluation runs, both prompts explicitly define scoring boundaries, output formats, and constraints that prevent the evaluator from introducing additional commentary or unsupported interpretations.

1.4. Human Evaluation Criteria

To assess perceptual advertisement quality beyond automated metrics, we conducted a controlled user study with ten professional advertising specialists. Each participant evaluated ten video pairs, resulting in a total of 100 comparisons between videos generated by *AutoCut* and by the *GPT-4o + MGSV* baseline. For fairness, the ordering of the two videos in each pair was fully randomized, ensuring that annotators could not infer the model identity. Instead of collecting numerical scores, we adopt a pairwise comparison protocol in which annotators simply choose the better video for each criterion. This approach reduces calibration bias, avoids inconsistent use of rating scales, and yields more stable and discriminative perceptual judgments.

Annotators compared each video pair across five dimensions: visual smoothness, logical consistency, script–visual alignment, BGM–visual compatibility, and overall advertisement attractiveness. Table 3 summarizes the guidelines used by annotators to judge the winner in each dimension.

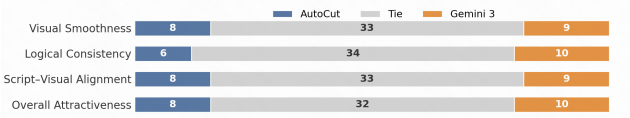

Figure 2. Additional human study comparing **AutoCut** with **Gemini 3** on 50 advertisement videos. A large proportion of cases are judged as *Tie* across all five dimensions, indicating that AutoCut achieves performance highly comparable to this stronger and more recent baseline.

1.5. Additional Human Study with Gemini 3

Since multimodal foundation models continue to improve rapidly, we further compare AutoCut with a stronger and more recent closed-source baseline, Gemini 3, which was released after our original submission. We conduct an additional pairwise human evaluation on **50** advertisement videos using the same protocol as the main user study.

For each sample, annotators compare the completed advertisement videos generated by AutoCut and Gemini 3 across four dimensions: *visual smoothness*, *logical consistency*, *script–visual alignment*, and *overall advertisement attractiveness*. As in the main study, annotators choose *Win*, *Loss*, or *Tie* for each dimension.

Figure 2 summarizes the results. We observe that **AutoCut** achieves highly comparable performance to Gemini 3, with a majority of cases resulting in *Tie* across all evaluation dimensions. This result suggests that AutoCut remains competitive even against a much stronger and more recent multimodal baseline. Importantly, AutoCut achieves this performance with a substantially smaller parameter scale and lower computational cost, highlighting its practical efficiency for advertisement video editing.

1.6. Dataset Construction Details

1.6.1. Multimodal Alignment Dataset Construction

We start from short-form advertisements with high user engagement, defined as videos whose click-through rate and like-rate both lie in the top 10% on the platform. For each video we collect product metadata (category, brand, selling points) and process the three modalities in parallel (Fig. 3).

On the **video** side, we sample frames at 1 fps with `ffmpeg` and encode them using an internal CNNv2 to obtain dense visual embeddings. On the **audio** side, we extract the audio track, separate background music from speech, and embed the BGM with a PANNs-based service into 2048-D acoustic features. On the **text** side, ASR is applied to the speech channel to obtain time-stamped transcripts; a lightweight Qwen3-0.6B filter removes videos whose tran-

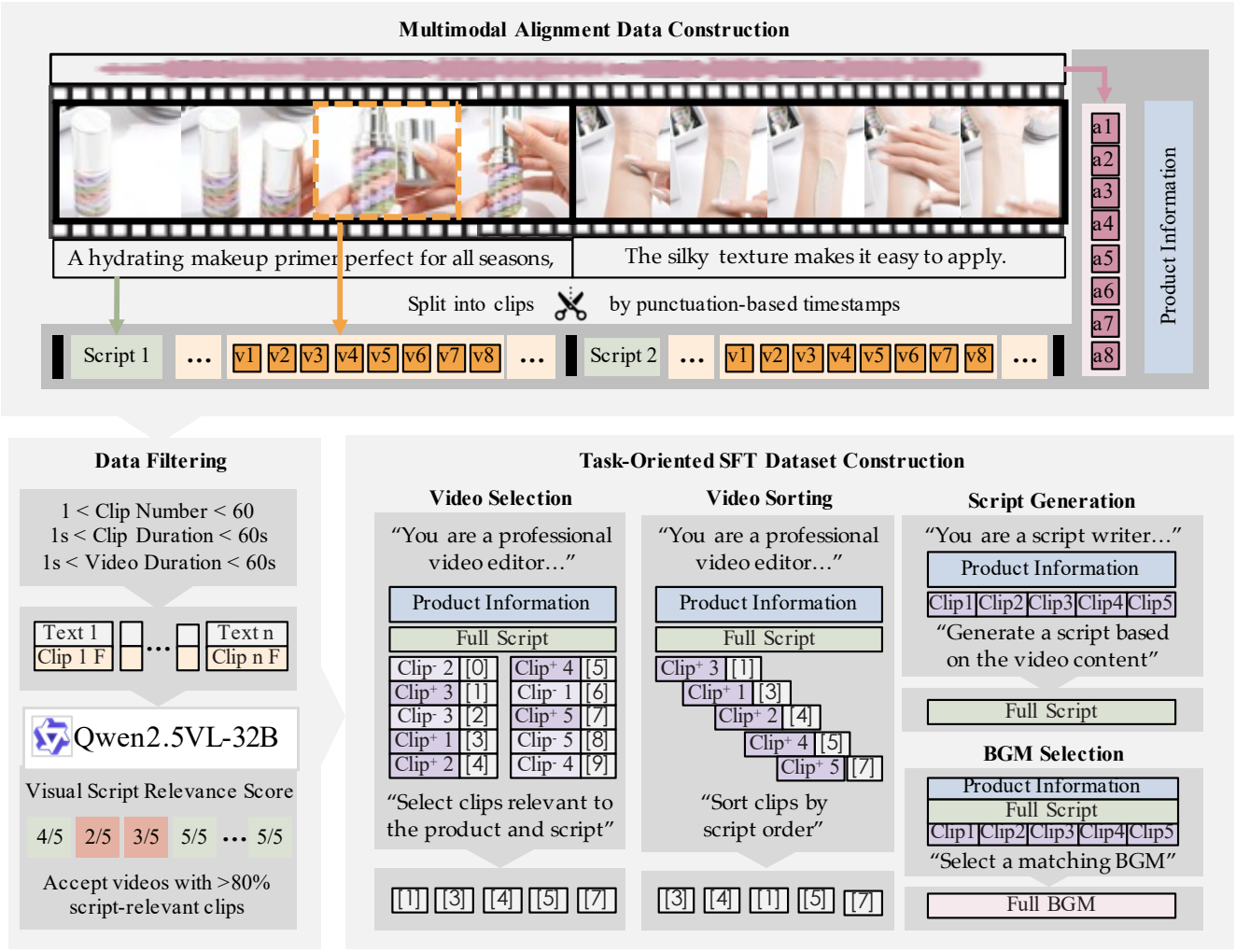

Figure 3. Overview of our dataset construction pipeline. *Top*: multimodal alignment data construction. Raw ad videos are paired with sentence-level scripts and product information, then segmented into short clips using punctuation-aligned timestamps, producing aligned video tokens and audio tokens. *Bottom-left*: data filtering. We keep videos whose clip number and duration fall within preset ranges and whose visual–script relevance score (estimated by Qwen2.5-VL-32B) exceeds an 80% threshold. *Bottom-right*: task-oriented SFT dataset construction. From the filtered aligned data we instantiate four supervision signals: video selection, video sorting, script generation, and BGM selection, each conditioned on product information and/or the full script.

scripts mainly contain song lyrics or lack coherent product descriptions.

Visual and acoustic features are then discretized by the RQ-VAE tokenizers (Sec. 3.3) to produce sequences of video and audio code indices. We define clips by ASR punctuation boundaries and *soft-align* visual tokens whose timestamps fall inside each sentence span. For every ad, clip-level text and video tokens are concatenated into a single sequence, and the audio-token summary of the full BGM track is appended. This yields our multimodal alignment corpus for the first training stage, containing roughly **700K** advertisements.

1.6.2. SFT Data Filtering Pipeline

From the alignment set, we derive an SFT-ready subset of about **100K** high-quality ads using the following filters:

- **Duration:** keep videos shorter than 120 s and clips within 2–60 s.
- **Caption quality:** discard ads with empty ASR, incoherent lines, or heavy repetition.
- **Visual–text relevance:** for each clip, pair the first frame with its ASR line and query Qwen2.5-VL-32B for a 0–5 relevance score; a video is retained only if at least **80%** of its clips score ≥ 4.
- **Deduplication:** remove near-duplicates by hashing $\{brand, product, script\}$.

This pipeline enforces strong multimodal consistency and clean semantic grounding before constructing the task-oriented SFT datasets.

1.6.3. Task-Oriented SFT Dataset Construction

The four supervised tasks introduced in Sec. 3 are constructed from the filtered pool. For training convenience, some SFT tasks are expressed in an index-based output format. These indices serve only as task-level supervision over the local candidate pool. In the full AutoCut pipeline, however, final video and audio asset grounding is performed by embedding-based retrieval from the material database, as described in Sec. 1.2. Since the main text focuses on task definitions, here we summarize only the input–output formatting used to build the SFT dataset. All tasks are encoded in a **ShareGPT-style two-turn** format with a `system` role and a single `human–assistant` exchange.

- **Video Selection. Input:** product metadata, the full ad script, and a pool of candidate multimodal clips (each clip represented by its text snippet and video tokens). The clips are presented in random order, and we make sure that positive and negative clip ratios are 1:1. **Output:** the subset of candidate clips that should appear in the final ad, represented as indices within the local candidate pool.
- **Video Sorting. Input:** product metadata, the ordered reference script, and a set of selected clips whose order has been randomly shuffled. **Output:** an ordering of the selected candidate clips, represented as a permutation over local candidate indices.
- **Script Generation. Input:** product metadata and an ordered sequence of clips. **Output:** a multi-line ad script whose number of sentences matches the number of clips, providing a sentence-level description aligned with the visual sequence.
- **BGM Selection. Input:** product metadata, the ordered clips, and the full script. **Output:** a discrete audio-token sequence representing the target background-music style, which is later grounded to a concrete track through embedding-based retrieval.

For each task we construct approximately **25K–30K** instances, and adopt a **95/5** train–validation split. The clip-level multimodal structure and the end-to-end SFT data construction process are summarized in Fig. 3.

Table 3. Human evaluation criteria and decision guidelines used in the pairwise comparison study.

Criterion	Positive indicators	Negative indicators
Visual Smoothness	• Transitions feel natural and continuous. • No obvious stutter, jitter, or dropped frames. • Motion and camera movement appear stable.	• Noticeable lag, stutter, or frame skipping. • Abrupt cuts that break visual flow. • Shaky or inconsistent motion that distracts the viewer.
Logical Consistency	• Shot order follows a clear story or explanation. • The same product and context are maintained throughout. • Scene changes support a coherent narrative or demo.	• Scenes feel random or out of order. • Products or contexts change without explanation. • Transitions break the story or confuse the viewer.
Script–Visual Alignment	• The script accurately describes what appears on screen. • Key selling points are shown at the right visual moments. • No hallucinated objects, brands, or functions.	• The script mentions things not visible in the video. • Important visual content is not reflected in the script. • Timing between narration and visuals is clearly off.
BGM–Visual Compatibility	• Music tempo matches motion and editing rhythm. • Overall mood fits the product and scenario. • Volume and energy support, rather than overshadow, the content.	• Music is too fast/slow compared with visual rhythm. • Style or emotion of the music feels inappropriate. • Sudden changes in volume or style that distract the viewer.
Overall Attractiveness	• The ad is engaging and easy to follow. • Product value and advantages are clearly conveyed. • You would be more willing to consider buying the product.	• The ad feels dull, confusing, or unconvincing. • Product value is unclear or poorly communicated. • You would not be motivated to learn more or purchase.