Hear-to-See: Automated Audio Segmentation and Visual Summarization

De Marchis Domenico, Fontanesi Francesco and Rainer Cabral Ilao

Department of Civil Engineering, Computer Science and Aeronautical Technologies
University of Roma Tre

Abstract

We propose an end-to-end system that listens to a long-form spoken narrative and returns a storyboard-like sequence of images that spotlight its key turning points. The pipeline, orchestrated in ComfyUI, (i) detects speech, (ii) separates the vocal track from music or ambient noise, when it is necessary, (iii) identifies peaks that mark semantically rich moments and (iv) each of these chunks is translated to text using OpenAI Whisper transcription model and then mapped to an illustrative frame by a commercial diffusion model. Experimental results demonstrate robust visual alignment between generated images and their corresponding spoken narrative events: part of our evaluation comprised tests on classic fairy tales and iconic film narratives, each presented as continuous audio clips of two to five minutes.

Introduction

The accelerating progress of generative models has opened new possibilities for multimodal storytelling, where text, audio and images co-evolve to convey rich narratives. Yet most current pipelines start from written text; the vast reservoirs of oral content such as podcasts, interviews, historical archives... remain largely untapped for automatic visualisation. Bridging this gap requires robust speech isolation under diverse acoustic conditions, identification of narrative pivot points without manual annotation, and tight coupling to an image generator capable of maintaining stylistic and thematic consistency.
In this paper we present a fully automated pipeline that converts long-form spoken narratives into a sequence of stylistically consistent images that highlight the most salient moments of the story. The workflow, implemented end-to-end in ComfyUI, first filters incoming audio with a two–stage acoustic classifier. We begin with a lightweight detector based on MFCC features to determine whether speech is present. When speech is detected, we evaluate three classification strategies under identical conditions. The first strategy retains the MFCC feature extraction and applies a logistic-regression classifier directly to those coefficients. The second replaces the handcrafted MFCCs with frame-level embeddings produced by the pretrained YAMNet network, again using logistic regression for final decisions. Finally, we considered a small convolutional neural network trained on log-mel spectrogram inputs; while this CNN demonstrated competitive performance, it serves mainly as an auxiliary option and will only be mentioned briefly in our oral presentation. Depending on the predicted background class, we apply either spectral subtraction combined with voice-activity detection (VAD) for noisy scenes or Demucs for vocal–music separation. The isolated vocal track is analysed with a salience function that detects abrupt pitch and energy shifts; these maxima delimit narrative meaningful "chunks". We transcribe each of these segments with the OpenAI Whisper model and then generate a thematically consistent illustration for it using a commercial diffusion model, focusing on a particular cartoonish style that we dediced, but the approach is completely replicable under different design preferences.
In summary, we make three main contributions:

1. **Background-aware voice isolation**. We use a two-stage acoustic classification pipeline. First, a lightweight MFCC-based detector filters out non-speech segments. The remaining audio is then classified as music or generic noise using one of two strategies: either a logistic regression on MFCCs or a transfer-learning classifier built on YAMNet embeddings, like we discussed earlier. Depending on that result, we dynamically apply spectral subtraction with VAD for generic noise or invoke Demucs for musical backgrounds.
2. **Saliencylience-driven segmentation**. We introduce a Saliency function approach that tracks abrupt shifts in pitch and energy on the isolated vocal track, automatically extracting semantically rich narrative "chunks" without relying on any annotations or language models; in the specific section, we propose different alternatives that can be followed

Corresponding authors:
dom.demarchis@stud.uniroma3.it
fra.fontanesi@stud.uniroma3.it
ra.ilao@stud.uniroma3.it
Published: July 10, 2025

3. **Style-consistent image storyboard**. We use a commercial diffusion model to generate a unified "cartoonish" aesthetic, ensuring the visual style remains coherent even across diverse narrative contexts

Methodologies

Input Audio Preprocessing

The signal-conditioning front end prepares each incoming waveform for the salience-based chunking stage by guaranteeing that the salient-point detector is exposed to speech that is as free as possible from interfering sources. We first apply a lightweight classifier that decides whether the recording actually contains speech; this one is a binary classifier trained on Mel-Frequency Cepstral Coefficients, as said in the previous introduction; if its decision is negative, then the pipeline terminates and the input audio is flagged as "non-speech", that's for preventing hazards in the downstream modules. When the classifier labels positive, the signal is forwarded to a second classifier that distinguishes generic background noise, such as conversational babble, traffic or musical accompaniment, making Hear - to - See robust even to songs as input audio. This is crucial because the two interference categories demand fundamentally different enhancement strategies.
Recordings labelled as "speech with generic noise" are processed by a classical Voice Activity Detection followed by Spectral Subtraction. The VAD identifies speech-dominant frames: using those labelled as non-speech we estimate an average noise spectrum and subtract it, with oversubtraction and spectral-floor safeguards, from every frame's magnitude. This combination, although algorithmically simple, suppresses broadband, non-harmonic disturbances with minimal audible artefacts, as confirmed by informal listening tests; the results we are talking about are indeed discussed with reference to the fact that we haven't clear metrics to quantify them, so we proceeded empirically. When the interference classifier, instead, reports the presence of music, the signal bypasses the VAD path and is routed to Demucs: a U-Net style convolutional network that constitutes the current state of the art for source separation; Demucs isolates the vocal stem from the musical accompaniment, enabling us to discard the latter while preserving speech fidelity.
The output of either enhancement branch is therefore a signal that can be treated as reasonably noise-free speech. Feeding this signal to the salience extractor ensures that peaks in the salience function correspond to genuine vocal events, such as intonational shifts, phrase boundaries or narrative highlights, rather than to sporadic background activity. In turn, the chunking module segments the narration at acoustically and semantically meaningful positions, meeting the core objective of the project: producing an efficient, content-aware partitioning of the input audio.

VAD and Spectral Subtraction

In this study we model the observed signal as

$$x(n) = s(n) + r(n)$$

where $s(n)$ is the clean speech and $r(n)$ is additive, quasi-stationary noise. The Voice Activity Detection (VAD) stage exploits the fact that, in typical acoustic scenes, speech exhibits higher average energy than background noise. By applying the Short-Time Fourier Transform (STFT) we segment the waveform into short, overlapping frames, retaining both temporal and spectral information. Frame-wise processing is essential with real world noise behaviour: statistics computed over the entire signal would blur time-varying noise characteristics; just think about a noise that lasts almost half the audio, working on the full, global frequency spectrum, not following a frame - wise approach, would result in an underestimation of the noise spectrum and in a biased Spectral Subtraction. For each frame m we compute the energy E_m and construct a boolean *speech mask* $\mathbf{M}$:

$$M_m = \begin{cases} \text{True}, & E_m > \theta \\ \text{False}, & \text{otherwise} \end{cases} \qquad \theta = 0.3 \times \text{median}\{E_m\}$$

Using the median yields robustness to the high-energy bursts typical of speech, and setting the threshold to 30% of this value provides a reliable discriminator between speech and noise, keeping in mind that, tipically, noise relies on a lower average energy than the speech component.
By inverting $\mathbf{M}$ we isolate the non-speech frames, from which we estimate the average noise spectrum

$$\widehat{R}[k] = \frac{1}{|\mathcal{N}|} \sum_{m \in \mathcal{N}} |X_m[k]|$$

with a simple mean operation on the magnitude, where $\mathcal{N}$ is the set of non-speech frames and $X_m[k]$ is the STFT of frame m.
During Spectral Subtraction we remove $\widehat{R}[k]$ from the observed magnitude spectrum, introducing an oversubtraction factor α to compensate for estimation errors and a spectral floor β to avoid negative magnitudes and so artifacts in reconstruction:

$$\widetilde{S}_m[k] = \max\big\{|X_m[k]| - \alpha\, \widehat{R}[k],\ \beta\, \widehat{R}[k]\big\}$$

The oversubtraction factor α is due to the fact that, even if we're aiming to be precise, we are still estimating. The enhanced signal is obtained via inverse STFT using the original phase, as phase errors are generally less perceptible than magnitude errors. Although based on simple assumptions, this VAD + Spectral Subtraction pipeline provides an effective solution for single-speaker noise suppression. The idea of going more into detail of this VAD + Spectral Subtraction comes from a similar technique we saw during the lessons.

MFCC Fundamentals
In this work MFCCs serve as the primary features for the first task in our pipeline: discriminating between frames that contain speech and those that do not. Because the main goal of Hear-to-See is to generate a narrative of the input audio, processing speech-free segments would be pointless. MFCCs provide a compact vector representation of the cepstrum that is particularly well-suited to classification problems. With only a dozen coefficients per frame we retain the most informative spectral cues, allowing even a simple logistic-regression classifier to reach an F1-score of **0.85**, reflecting balanced precision and recall. A speech signal can be modelled as the convolution of a glottal source $g(t)$, which is the excitation produced by the vocal-fold vibration, and the vocal-tract filter $s(t)$:

$$p(t) = g(t) * s(t)$$

Taking the Fourier transform yields

$$P(f) = G(f)\,S(f)$$

and applying the logarithm converts the product into a sum:

$$\log P(f) = \log G(f) + \log S(f)$$

In the log-spectral domain the glottal source (high quefrencies) and the vocal-tract envelope (low quefrencies) are neatly separated. Recognizing wether there's speech or not depends primarily on the slowly varying vocal-tract component, so we concentrate on the low-quefrency region. The linear time-invariant (LTI) assumption underlying this decomposition is valid only locally. We therefore compute a short-time Fourier transform (STFT) with 20–40 ms windows, within which the speech signal is approximately stationary. The real cepstrum is then obtained by taking the inverse DFT of the log-magnitude STFT coefficients. In this representation the horizontal position of a peak corresponds to the period of a repetitive event in the spectrum, so a 'spectrum periodicity' regarding the frequency content; glottal periodicity appears at high quefrencies, whereas the vocal-tract envelope occupies low quefrencies and exhibits no strong periodicity. Although we rely on librosa for the actual implementation, the standard MFCC pipeline comprises:

1. STFT of the input waveform
2. Power spectrum computation for each frame
3. Mel filterbank integration followed by a logarithm
4. Discrete cosine transform (DCT) of the log-Mel energies, yielding the coefficients c_k

The resulting c_k values are the MFCCs themselves; strictly speaking, MFCCs are not computed from the cepstrum but from the log-Mel spectrum, an operation that is mathematically analogous to the log step described above. The Mel scale warps frequency so that equal distances correspond more closely to equal perceived pitch intervals: below ≈ 1 kHz we perceive frequency changes almost linearly, whereas above 1 kHz our resolution gradually compresses. A filterbank of typically 40–128 triangular filters is defined with centres evenly spaced on the Mel axis, then mapped back to hertz; because the Mel$\to$Hz mapping is nonlinear, filters become progressively wider at higher frequencies. Applying the filterbank converts each STFT frame of length N into a vector of $H \ll N$ Mel-band energies:

$$S_j = \sum_{k=0}^{N-1} P[k]\; H_j[k]$$

where $P[k]$ is the power at the $k-th$ frequency bin and $H_j[k]$ the weight of the $j-th$ triangular Mel filter. This step both emphasises perceptually important regions and performs substantial dimensionality reduction. A DCT is finally applied to the log-Mel energies. The first 12–13 coefficients capture almost the entire spectral-envelope energy and are largely decorrelated, completing the dimensionality-reduction process while preserving the information most relevant for speech detection. Overall, the MFCC representation provides an information-dense, perceptually motivated feature set that enables accurate, computationally efficient voice-activity detection—the prerequisite for all subsequent stages of the Hear-to-See framework.

First and Second Derivatives
What actually feeds our classifier are not the "raw" MFCCs alone but the MFCCs together with their first- and second-order time derivatives:

- First derivatives Δ measure the variation of each coefficient c_k across a neighbourhood of n frames, i.e. they capture the spectral velocity
- Second derivatives $\Delta\Delta$ extend this idea by taking the derivative of the first derivatives, thus yielding the spectral acceleration

The physical analogy is direct: Δ reflects how fast the spectrum is changing, while $\Delta\Delta$ indicates how quickly that rate of change itself varies. These dynamic cues allow us to keep the feature set compact yet highly informative. In the present work we fix $N_{\text{MFCC}} = 13$; applying the same dimensionality to the two derivative orders gives, for every frame, a 39-component feature vector still far smaller than the original STFT representation. By transforming a static spectral snapshot into a kinematic description, the slopes (Δ) reveal how much and in which direction the spectral envelope moves from one frame to the next, whereas the curvature ($\Delta\Delta$) tells us how rapidly those slopes themselves evolve. This dynamic information is essential for detecting the transitions that characterise speech (e.g. vowel–consonant boundaries) and therefore improves class separability whenever two sounds share similar static spectra but differ in their dynamics. For each frame we obtain 39 values (13 MFCCs + 13 Δ + 13 $\Delta\Delta$). To describe an entire clip we aggregate those frame-wise features using the mean, median, standard deviation, and min–max range for every coefficient and derivative order. This choice is motivated by the markedly different statistical profiles of our three target classes:

- Generic background noise is largely stationary; its derivatives are close to zero, and this stability is preserved after aggregation
- Music and speech share richer and similar derivative patterns, which explains mostly the classification errors
- Speech, tipically, shows higher low-frequency MFCC means for the very first c_k than music, while its standard deviations are typically larger because speech is less periodic than a sustained harmonic background; this, of course, depends on the musical genre itself. The min–max range further helps discriminate music (often wider due to stronger peaks) from speech, giving additional information to better discriminate

If we computed clip-level aggregates over an entire recording, distinctive regions (pure speech, pure music, noise) would be diluted by the large and heterogeneous number of frames, especially for long audio files. To avoid this, we build the training set from 3-s windows:

1. Each 3-s segment is treated as an independent sample
2. Segments labelled speech include pure voice as well as mixtures with music or noise, forcing the model to detect speech even under masking backgrounds
3. Purely instrumental samples are added as negative examples

At inference time, the pipeline proceeds as follows:

1. The incoming audio is split into 3-s windows
2. Aggregated MFCC + Δ + $\Delta\Delta$ statistics are computed per window
3. The trained model outputs a binary decision (speech/no-speech) for each window
4. The final label for the whole clip is assigned by simple majority voting

After training, the system was evaluated on five previously unseen clips (30 s each) containing various combinations of speech, music and noise. In every case the majority-vote decision matched the ground truth, confirming the effectiveness of the proposed feature design and segmentation strategy.

Classifiers Comparison
To examine how MFCC-based statistics behave under different acoustic conditions, we assembled three complementary datasets:

1. **Talk Dataset**
   - **Compisition**: 3 s clips in which speech is always present and mixed with either music or generic environmental noise
   - **Labels**: “music” vs. “noise” refer exclusively to the type of background, while speech presence is constant
2. **No-Talk Dataset**
   - **Compisition**: 3 clips of music-only or noise-only, with no speech component
   - **Labels**: “music” vs. “noise” again denote the background class, here under the assumption of speech absence
3. **Speak Dataset**
   - **Compisition**: 3 clips labeled according to speech presence, regardless of background; some contain speech over music or noise, others contain only music or noise
   - **Labels**: we have “speak” where speech is present, mixed with any background and “no_speak” where there’s no speech component (music and noise only)

Each dataset is balanced at approximately the clip level (50% per class) and processed with identical MFCC and derivative aggregation pipelines, enabling a direct comparison of classifier performances. In the **No - Talk dataset** we obtain a F1 - Score of 100% and that’s not due to the overfitting of the model: having the statistical properties of music and speech a similar profile, which is very different to the generic noise one, the features we’ve built are extrimely significative for this task, making the model fall in no error. In the **Speak Dataset** we obtain a F1 - Score of 85%

and, as said before, the main source of error is the similarity between speech and music in the features we've built; the no_speak class contains examples of both generic noise and music, it's for the latter that the results are not a perfect score as before.

Dataset Representativity Problem

About the **Talk dataset** we can discuss something crucial: all of the dataset we used are derived from a restricted subset of the famous GTZAN corpus, limited almost exclusively to pop-rock tracks. Speech segments were created by overlaying a single narrator reading portions from seven non-fiction books, all of which, as well as the music clips, are publicly available on Kaggle. Because of these constraints, our data do not capture the full diversity of real-world acoustic conditions: they lack variation in musical genres, vocal timbres, accents, and prosodic styles. Consequently, the performance metrics reported here should be interpreted as a proof of concept rather than as definitive evidence of real-world effectiveness. On the Talk dataset the classifier attains an F1-score of 1.00. This perfect score is partly artefactual. Although data-augmentation was applied, several vocal passages from the same songs appear, even if modified with Data Augmentation, in both the training and test partitions, introducing a degree of train–test leakage. Under these circumstances the model effectively "memorises" the common speech component and exploits the strong spectral differences between music and broadband noise in the MFCC-based feature space. When the Talk classifier is cascaded with the model trained on the Speak dataset, the overall F1-score drops to 0.85, the F1 of the Talk Classifier, a figure that we regard as a more realistic proxy for real-world performance. Despite the leakage, these experiments demonstrate that the engineered MFCC and derivative features are highly informative for disentangling speech and music from speech and noise mixtures. Future work could address the present limitations by eliminating cross-song leakage, expanding the musical and vocal diversity, potentially by using the full GTZAN collection or larger open datasets, and evaluating the system on speaker- and song-independent splits to obtain an unbiased estimate of generalisation. Something important to highlight is that, in the Talk dataset, both classes include a spoken component: therefore, the model learns that shared speech component and distinguishes the clips only by how much the background, music or generic noise, adds to the MFCC and derivative feature aggregates, backgrounds whose profiles differ sharply under the features we engineered. Consequently, as we expected, the scores are on a par with those obtained on the No-Talk dataset, because the two situations are, to a good approximation, the same problem and share the same criticalities.

Demucs and YAMNet

To explore alternative design choices, we replaced the handcrafted MFCC + derivative features used in our previous experiments with these YAMNet embeddings and trained a logistic-regression classifier under otherwise identical conditions. In earlier tests on the Talk Dataset, the MFCC-based classifier achieved an apparently perfect 100% F1 score on the music versus generic noise task, an obvious artefact of dataset bias like we discussed, yet the result suggested that the engineered features were highly informative. Our goal here was therefore to assess, via transfer learning, how well the widely adopted YAMNet embeddings perform on the same binary classification problem. Using the YAMNet feature set, the logistic-regression model achieved an F1 score of 0.68. Although this metric is subject to the same data representativeness issues as before, the comparison remains valid because both systems were evaluated on the identical dataset and subject to the same criticalities, also the same Logistic Regression model, making, even if artifacts, the two metrics directly comparable. Even discounting the MFCC model's inflated score, its true performance still exceeds that obtained with YAMNet embeddings. This finding underscores a key aspect: model capacity alone is not sufficient and feature engineering remains critical. Despite stemming from a substantially larger and more resource-intensive training procedure, the YAMNet embeddings are less effective for our specific task, even though YAMNet's 521-class taxonomy includes categories such as speech, music, rain noise, clapping, cough, animal sounds... and so forth, which closely align with our specific task of classify music, generic noise and speech. Evidently, the tailored features can outperform generic, albeit sophisticated, representations when the target problem is narrowly defined. Demucs (Deep Extractor for MUlti-Channel Sources) is an end-to-end family of music–source-separation networks introduced by Défossez et al. (2019). The first release adopted a 1-D convolutional U-Net that operates directly on the stereo waveform; encoder–decoder skip connections preserve fine temporal detail during the decomposition in stems. Later versions have provided incremental refinements, which we don't discuss further, but the model remains state of the art for decomposing a mixture of sounds into single stems like vocals, bass, drums, and accompaniment. This is therefore an attractive component for our pipeline. Although U-Nets were not covered in this course, their operation follows directly from the convolutional concepts we studied with CNN. In practice, Demucs delivers excellent perceptual results: still, we haven't formal metrics such as F1-score to not apply here, yet informal listening confirms that vocal isolation from musical accompaniment is near-perfect. Regarding the pipeline, we restrict its use to scenarios in which the background is musical and poorly handled by traditional VAD and Spectral Subtraction methods: extracting vocals from a speech-dominant background remains intrinsically difficult, but where Demucs is unnecessary we abide by the "simplest-works-best" principle and fall back on lighter classical techniques. A more aggressive design, feeding every input to Demucs, would eliminate the need for the music vs generic noise classifier and the VAD + Spectral Subtraction stages, but it offers practical benefit about computational heaviness; the empirical tests, also, suggest that Demucs was trained primarily on musical mixtures only: when the background is unstructured noise, the "noise" stem is often nearly silent and the vocal stem acquires artefacts, undermining the very purpose of this stage. For these reasons we invoke Demucs only when it demonstrably improves the overall pipeline.

Saliency-driven Segmentation

In the audio chunking step, the primary objective is to segment long-form audio recordings—typically consisting of a single narrator recounting a story—into meaningful chunks that closely correspond to the distinct narrative components or events within the story. This task is inherently challenging when working with raw audio, as it requires approximating semantic or story-based boundaries using only the acoustic properties of the signal, without access to transcripts or explicit textual cues. However, an important advantage in this particular application is that all incoming audio files are already pre-processed to be essentially noise-free, or at least have such a low level of background noise that it can be safely ignored in the analysis. This high-quality audio greatly increases the reliability and interpretability of signal-based chunking methods, since common pitfalls caused by noise—such as spurious silence detection or artificial energy fluctuations—are virtually eliminated. To approach the chunking task, several complementary algorithms have been implemented, each leveraging different aspects of the speech signal. The first, and perhaps most intuitive, is chunking on silence. This technique scans the audio for extended periods of low energy, under the assumption that a skilled narrator will naturally insert pauses at logical breakpoints in the story—such as the end of sentences, paragraphs, or dramatic moments. In a noise-free environment, the detection of these silences becomes highly robust, since there is little risk of background noise being mistaken for speech or vice versa. The algorithm can thus use relatively low energy thresholds and short minimum silence durations, resulting in accurate detection of genuine pauses and, consequently, segment boundaries that often align well with the story's structure. A second, more naive method is fixed interval chunking, which simply divides the audio into equal-length segments regardless of content. While this approach ignores the narrative flow entirely and is likely to split sentences or story events in unnatural places, it serves as a useful baseline and can be helpful when uniformity of segment length is required for downstream processing tasks. The saliency-based chunking method builds upon the silence approach by analyzing the dynamic energy profile of the audio. Instead of only looking for complete silence, it detects significant drops in energy, which may correspond to softer speech, brief hesitations, or natural breaks in prosody. In high-quality audio with minimal noise, this method can be tuned to be very sensitive, capturing even subtle cues in the narrator's delivery that might signal a shift in the narrative or a change in emotional tone. This results in segmentations that are both fine-grained and contextually relevant, especially in expressive storytelling. The prosody-based chunking technique leverages the rhythmic and intonational properties of speech—specifically, abrupt changes in pitch and loudness—to identify boundaries between narrative units. Prosodic cues are often used by speakers to indicate emphasis, transitions, or the conclusion of an idea, even in the absence of silence. In low-noise recordings, these acoustic features are preserved and can be detected with greater confidence, allowing the algorithm to find meaningful segment boundaries even when the narrator does not pause audibly. By combining pitch and energy analysis, and filtering out minor fluctuations with statistical smoothing, this method is particularly adept at mirroring the "flow" of storytelling and marking points where the narrative shifts or new events begin. Collectively, these methods offer a robust toolkit for automatic story segmentation in high-quality, noise-free single-speaker audio. The absence of noise not only improves the accuracy of silence and energy-based chunking but also enhances the sensitivity and reliability of more advanced, prosody-aware algorithms. While none of these techniques can perfectly replicate a human's understanding of story structure without explicit semantic information, their performance is significantly strengthened by the clean input data and the careful tuning of parameters made possible by it. As a result, you can expect chunk boundaries that are much closer to natural narrative divisions, facilitating downstream tasks such as transcription, annotation, or further audio analysis.

To assess the robustness of the implemented chunking methods, tests were conducted using two types of input: recordings with natural human speech and recordings with artificially generated speech produced by TTS (text-to-speech) systems. The results clearly show that natural speech yields superior outcomes, as pauses, energy variations, and prosodic cues are more distinct and better aligned with the narrative flow, allowing the algorithms to identify more meaningful and coherent segment boundaries. In contrast, synthetic speech—especially from less advanced TTS models—tends to lack expressive pauses, exhibits flatter intonation, and shows limited prosodic variation. As a result, the segmentation often appears more arbitrary, with chunks falling at unnatural points that do not correspond to actual narrative transitions. These findings highlight the importance of speech naturalness and expressiveness not only for human comprehension but also for the effectiveness of signal-based automatic segmentation algorithms.

Hear-to-See Pipeline Theory

The preprocessing of the previous pipeline generated a noise/music-suppressed, speaker-agnostic audio file that has been automatically segmented into coherent acoustic units (using the different and proper methods described previously). The present section details the multi-stage pipeline that transcribe each chunks using a SOTA OpenAi model (Whisper) achieving an audio2text output that is merged in a single text file where each chunk is separated by specific charates. After doing so the structured file is used in its integrity as the userInput of a engenieered prompt exectued via Groq API to achieve what can be called a text2prompt output. In this way the model has a well-structured vision of the story and its protagonist and can generate from each chunk a prompt for an image generation model. The pipeline thus implement a way to generate images from sections of the story creating a visual rappresentation of it.

From audio chunks to transcriptions The preceding stage of the pipeline delivers a noise-suppressed waveform split into temporally coherent chunks (typically 15–30s each).This segmentation is essential: it maintains linguistic context while ensuring that every segment fits comfortably into GPU memory and keeps inference latency low. Each chunk is exported as an individual WAV file whose sampling rate (usually 16 kHz) and cleaned from potential noise. The remainder of this chapter describes how these WAV chunks are converted into text using the Whisper automatic-speech-recognition (ASR) system.

What Is Whisper?
Whisper (OpenAI, 2022) is an encoder–decoder Transformer trained end-to-end on 680kh of multilingual speech and corresponding transcripts. It performs speech recognition, language identification, and (optionally) speech translation within a single model.Unlike classical ASR (which strings together separate acoustic, pronunciation, and language models), Whisper learns a single mapping from audio to text tokens.

How Whisper Works
Using the python library we can access the Whisper API and pass to it an audio file. To achieve the transcriptions 3 steps are followed:

Acoustic Front-End– From Waveform to Log-Mel Spectrogram
At the outset, the incoming audio is standardized to 16 kHz mono, guaranteeing that every sample shares the same time resolution and channel configuration. Once resampled, the continuous waveform is sliced into brief, overlapping segments using 25 ms analysis windows stepped every 10 ms. Each window is transformed into the frequency domain via a Short-Time Fourier Transform, producing a dense spectrogram of magnitudes. To emphasize perceptually relevant bands, this spectrum is then passed through an 80-channel Mel filter bank, whose outputs are compressed using a logarithmic scaling. The result is an $80 \times T$ matrix—effectively a "time-frequency image" that serves as the foundational input for Whisper's neural encoder.

Transformer Encoder
After the raw audio has been converted into a grid of log-Mel features and split into two-dimensional patches, these patch embeddings are fed into Whisper's Transformer encoder, which consists of a deep stack of identical self-attention blocks. Depending on the chosen model size, from the compact "tiny" variant to the expansive "large" network, the encoder may comprise anywhere between 12 and 32 layers. Within each layer, multiple attention heads (typically 8 in smaller models and up to 16 in the largest) jointly attend to every patch embedding, allowing the network to dynamically reweight and integrate information across the entire time–frequency plane. Each attention operation is followed by a feed-forward network whose hidden dimension is four times the embedding width, giving the model ample capacity to recombine and transform the attended features. To ensure stable gradients during training, every sub-layer applies layer normalization before its core operation and wraps the result in a residual connection, so that the model learns only the incremental refinement needed at each depth. Abso-

lute two-dimensional positional embeddings, added to the patch projections at the input, teach the encoder to respect both temporal order and frequency locality over windows of up to 30 seconds. When processing longer recordings, Whisper seamlessly applies a sliding-window strategy with overlapping context frames, stitching together the outputs of each segment to preserve coherence and avoid boundary artifacts. The final output of the encoder is a sequence of high-level acoustic embeddings that concisely capture phonetic detail, intonation patterns, and prosodic contours, providing the decoder with a rich, contextualized representation of the spoken content.

Tokenisation & Language-Aware Decoding
Once the encoder has distilled the acoustic signal into a sequence of dense embedding vectors, the decoder takes over by first ingesting a special start-of-transcript token that signals the beginning of the output sequence. From that point onward, Whisper generates its transcription one token at a time: at each step, the model examines both the previously produced tokens and the full set of encoder embeddings, combining information about the evolving text hypothesis with the underlying audio features. Rather than predicting individual characters or words directly, Whisper leverages a shared byte-pair encoding (BPE) vocabulary of roughly fifty thousand subword units, which succinctly captures common words, inflections, punctuation marks and even language identifiers. As each new BPE token is proposed, the decoder computes not only the raw probability of that token given the preceding context but also incorporates learned language-model log-probabilities, which were co-trained alongside the acoustic parameters. This joint scoring mechanism allows the beam-search algorithm to maintain multiple competing hypotheses, weighing acoustical consistency against linguistic plausibility to select the most coherent sequence. Because punctuation, capitalization and sentence breaks were included in the training transcripts, Whisper naturally learns where to place commas, quotation marks and paragraph divisions—delivering a fluid, human-readable output without the need for any additional cleanup or rule-based post-processing.

Transformer-Based LLM Architecture for Text–to-Prompt Conversion
At the heart of the text-to-prompt stage lies a large language model built on the Transformer paradigm, repurposed to map a sequence of transcribed words into a concise, descriptive prompt. The process begins by tokenising the transcript into a sequence of discrete units—typically subword tokens drawn from a shared byte-pair vocabulary. Each token is converted into a dense embedding vector and augmented with a positional encoding, allowing the model to distinguish the order of words and capture long-range dependencies across the entire utterance. These embedded tokens then flow through a deep stack of identical Transformer blocks. Each block applies masked multi-head self-attention: several parallel "attention heads" compute scaled dot-product affinities between every pair of positions in the partial output sequence, but causally mask future positions to enforce an autoregressive generation. The attention outputs from all heads are concatenated and linearly projected back to the model dimension, enabling the network to integrate information from different subspaces of the hidden representation. A residual connection adds the block's input back to this attention result, and a layer-normalization step stabilises the ensuing gradient flow. Following self-attention, each block incorporates a position-wise feed-forward network with a two-layer architecture: an initial linear expansion (often four times the model dimension), a non-linear activation (such as GELU), and a contraction back to the original embedding size. Again, residual addition and layer-normalization ensure that the model can learn both fine-grained token interactions and higher-order abstractions without vanishing or exploding gradients. By stacking L such blocks, where L can range from a few dozen to several hundred depending on model scale, the LLM cultivates richly contextualised representations at every position. Early layers capture local collocations and syntactic patterns, while deeper layers absorb broader semantic and discourse-level cues. At inference time, the decoder begins with the special start-of-sequence token and, at each step, attends to all previously generated tokens to predict the next subword. The raw output logits are transformed into probabilities via a softmax over the vocabulary, and selection strategies such as top-k or nucleus sampling can be employed to balance fidelity and creativity in the resulting prompt. Finally, the chosen tokens are detokenised into a human-readable string that succinctly encapsulates the imagery implied by the transcript. Throughout this theoretical framework, the interplay of masked self-attention, deep non-linear transformations, and careful normalization enables the LLM to perform the sophisticated mapping from spoken description to evocative prompt—forming the crucial bridge between audio understanding and image generation.

Achieving Image Generation through a Diffusion Model
The final step of the pipeline employs a Stable Diffusion XL (SDXL 1.5) model, which interprets the text

prompt as a conditioning signal and synthesises an image via a learned denoising process. At a high level, diffusion models treat image generation as the inversion of a gradual noising process: starting from pure Gaussian noise, the model iteratively "denoises" through a trained reverse Markov chain until a coherent image emerges.

Forward and Reverse Diffusion Processes
During training, clean images x_0 are progressively corrupted by adding Gaussian noise over T discrete timesteps according to a fixed noise schedule $\{\beta_t\}_{t=1}^{T}$. At each step t, the noisy sample

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

blends signal and noise in a way that admits a closed-form posterior for x_t given x_0. The reverse process, which the network must learn, predicts x_{t-1} from x_t by estimating the noise component ϵ. The denoising objective minimises the expected squared error between the predicted and true noise terms, yielding a score-matching formulation that drives the model to approximate the true reverse transition kernels.

Latent Space Representation and Variational Autoencoder
Instead of applying diffusion directly to high-dimensional pixel arrays, SDXL 1.5 performs its noising and denoising operations within a compact latent space. A pretrained variational autoencoder (VAE) first encodes each image x_0 into a lower-dimensional tensor $z_0 \sim q_\phi(z \mid x_0)$, capturing the image's core semantic and structural information while discarding fine-grained noise. The diffusion model then perturbs and refines these latent vectors z_t over T timesteps, dramatically reducing both memory and compute requirements. Once the reverse diffusion chain converges on a clean latent $\hat{z}_0$, the VAE decoder $p_\phi(x \mid z)$ reconstructs the full-resolution image, mapping the learned latent representation back into pixel space. This two-stage VAE–diffusion framework allows the model to focus its capacity on generating meaningful image content rather than expending resources on modeling every high-frequency detail.

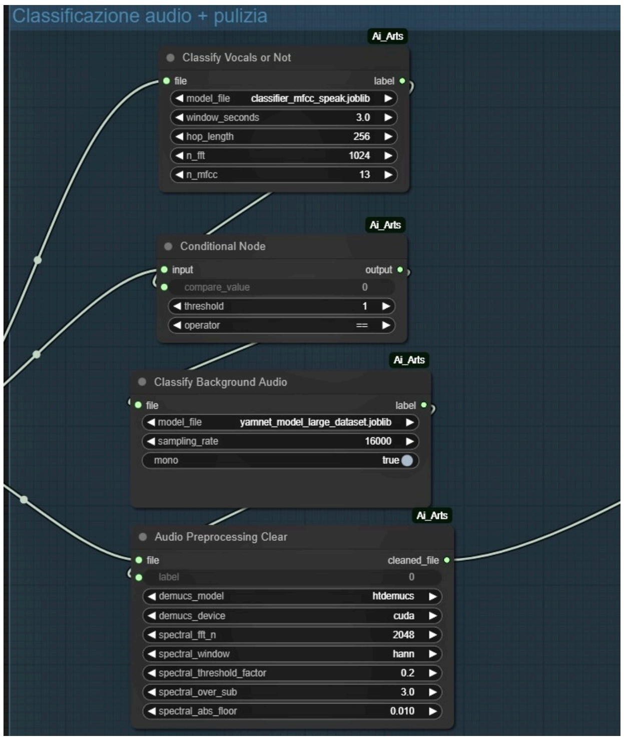
U-Net Backbone with Cross-Attention
The SDXL 1.5 architecture is based on a hierarchical U-Net: an encoder–decoder with symmetric downsampling and upsampling paths. At each resolution level, the network employs residual convolutional blocks interleaved with spatial self-attention, enabling the model to capture both local textures and long-range structure. Critically, at each upsampling stage the U-Net injects text information via cross-attention modules: the encoded text prompt is projected into key and value tensors, which attend to the diffusion features' query projections, allowing fine-grained alignment between words in the prompt and pixels in the image. Positional embeddings for both spatial location and diffusion timestep t are added to the feature maps, ensuring the model knows "when" in the denoising chain it is operating.

Noise Schedulers and Samplers
To generate an image from a text prompt, one selects a noise scheduler (e.g., linear, cosine, or the improved Karras schedule) that defines the β_t sequence, and a sampler algorithm (such as DDIM or PLMS) that trades off speed and sample quality. The scheduler dictates how aggressively noise is removed at each iteration, while the sampler interprets the model's noise estimates to step backwards through time. Reducing the total number of steps can accelerate inference, but may slightly degrade fidelity; conversely, a larger step count yields crisper images at the expense of runtime.

Parameter Configuration and Trade-Offs
Key parameters—number of diffusion steps, sampler type, noise schedule, and guidance scale—control the balance between generation speed, image sharpness, and adherence to the prompt. For instance, increasing the guidance scale improves alignment to the text but can introduce artifacts or reduce variability. Similarly, choice of sampler influences how closely the discrete inference path approximates the learned continuous diffusion process. By tuning these hyperparameters, SDXL 1.5 can flexibly span applications from rapid prototyping to production, quality art generation, all within the same theoretical framework of score, based generative modelling.

The ComfyUI Tool

To coordinate the sequence from audio classification and preprocessing through LLM prompt construction to SDXL image synthesis we employ ComfyUI, an open-source, node-based workflow engine designed for visual AI pipelines. ComfyUI provides a graphical canvas on which each computational block is represented as a node, complete with configurable inputs and outputs. Data flows from one node to the next according to directed connections, enabling us to assemble complex, multi-stage processes without writing bespoke orchestration code. The usefulness of ComfyUI lives in how the nodes expose parameters such as sample rate for audio nodes, model selection and decoding strategy for LLM nodes, and scheduler settings for diffusion nodes. By grouping related parameters into collapsible panels, the interface keeps the canvas uncluttered while allowing rapid experimentation. Nodes can also be nested within "meta-nodes," encapsulating common subnetworks (for example, a reusable subgraph that packages the Whisper front-end and encoder together), improving readability and reusability. By stringing together both standard and custom nodes in ComfyUI, we achieve a fully automated workflow: raw audio files are dropped into the "Load audio" node, transcripts emerge a few steps later from the "Whisper Transcribe" node, prompts are refined and queued through our LLM nodes, and finally SDXL images are generated and collected through the "KSampler" node. This visual programming paradigm not only accelerates development and debugging but also enables non-programmers on the team to inspect, tweak, and extend the pipeline with minimal training, a critical advantage for interdisciplinary collaboration.



Classify Vocals or Not
- model_file: classifier_mfcc_speak.joblib
- window_seconds: 3.0
- hop_length: 256
- n_fft: 1024
- n_mfcc: 13

Conditional Node
- compare_value: 0
- threshold: 1
- operator: ==

Classify Background Audio
- model_file: yamnet_model_large_dataset.joblib
- sampling_rate: 16000
- mono: true

Audio Preprocessing Clear
- label: 0
- demucs_model: htdemucs
- demucs_device: cuda
- spectral_fft_n: 2048
- spectral_window: hann
- spectral_threshold_factor: 0.2
- spectral_over_sub: 3.0
- spectral_abs_floor: 0.010

Figure 1: *Audio classification and preprocessing group node*

Audio classification and preprocessing

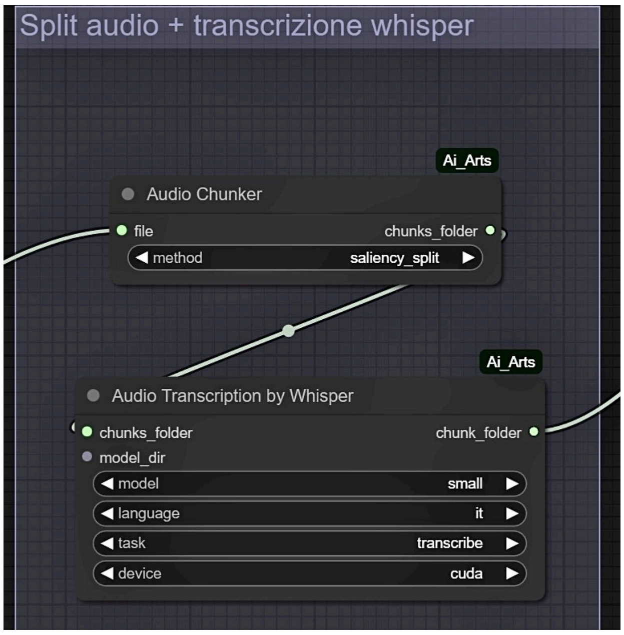
The pipeline begins by routing each incoming waveform into a *Classify Vocals or Not* node, which applies a pretrained MFCC-based classifier to decide whether speech is present. Its binary output (0=no speech, 1=speech) is sent to a downstream *Conditional* node that gates further processing: only segments flagged as containing vocals are allowed to proceed. In parallel, the same audio file is passed through a *Classify Background Audio* node—powered by a YAMNet model—that distinguishes musical content from ambient noise, again producing a 0/1 label. Finally, these two labels drive the behavior of the *Audio Preprocessing Clear* node: if music is detected, a Demucs source-separation routine is invoked to isolate and remove musical stems; if noise is detected instead, a spectral-subtraction algorithm is applied to suppress background hiss. The result is a cleaned audio stream, automatically selected according to the classification outputs, ready for transcription in the next stage of the workflow.

The Whisper Transcription Block

Once the audio has been cleansed of unwanted noise or music, it is handed off to an *Audio Chunker* node, which uses the previously stated methods to generate segments. These segments are written out to a designated "chunks" directory, preserving their original temporal order. Immediately downstream, each chunk is consumed by the *Audio Transcription by Whisper* node: here, the pretrained "small" Whisper model (configured for Italian transcription) is loaded onto the CUDA device and applied to every segment in turn. As each snippet is processed, Whisper produces a time-aligned transcript, and the collection of resulting text files is merged in a single text file where each chunk is separated by a specific sequence of character, setting the stage for precise prompt generation in the subsequent LLM block. The reason behind the choice of using a single file is to give the LLM the right context to maintain coherence.

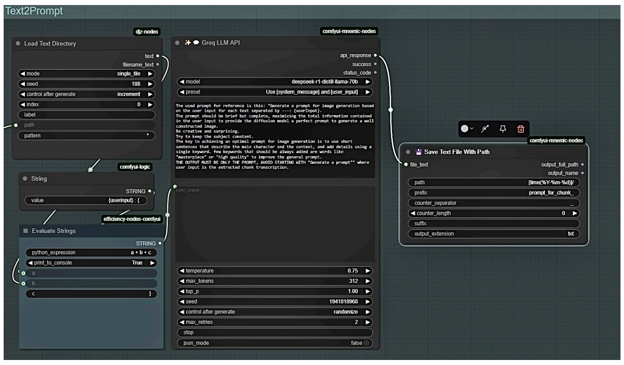
Text-to-Prompt Composition Block

At this stage, the per-chunk transcripts are consolidated and transformed into a single text prompt file. First, a *Load Text Directory* node (configured in "single_file" mode) ingests the folder of chunk transcripts and emits a unified string containing all segments in temporal order. That string is bound to a *String* node labeled {`userInput`}, which represents the raw user message for the LLM. Downstream, an *Evaluate Strings* node concatenates three components—(a) a fixed system message or template stored in the "preset" port, (b) the {`userInput`} transcript, and (c) any suffix keywords—to assemble the final prompt text. This assembled prompt is fed into the *Groq LLM API* node: here, model selection (e.g. `llama3-8b-8192`), temperature, maximum token count, and other inference parameters are specified. The node issues a synchronous API call and outputs a JSON response containing the LLM's "best" generated text. A second *Evaluate Strings* node then extracts or formats the relevant field from that JSON (for example, trimming extraneous metadata and appending style descriptors), yielding the precise image-generation prompt. Finally, a *Save Text File With Path* node writes the complete set of prompts into one file on disk. Using a timestamp-based prefix and a counter suffix, the node ensures unique and chronologically ordered filenames. The result is a single, ready-to-use text file containing all LLM-crafted prompts, which will serve



```
method     saliency_split
model      small
language   it
task       transcribe
device     cuda
```

Figure 2: *Chunking and transcroption group node*



```
Load Text Directory
mode                   single_file
seed                   188
control after generate increment
index                  0
label
path
pattern                *

String
value                  {userinput} : {

Evaluate Strings
python_expression      a + b + c
print_to_console       True
c                      }

Groq LLM API
model                  deepseek-r1-distill-llama-70b
preset                 Use [system_message] and [user_input]
```

```
The used prompt for reference is this: "Generate a prompt for image generation based
on the user input for each text separated by ---: {userInput}.
The prompt should be brief but complete, maximizing the total information contained
in the user input to provide the diffusion model a perfect prompt to generate a well
constructed image.
Be creative and surprising.
Try to keep the subject constant.
The key to achieving an optimal prompt for image generation is to use short
sentences that describe the main character and the context, and add details using a
single keyword. few keywords that should be always added are words like
"masterpiece" or "high quality" to improve the general prompt.
THE OUTPUT MUST BE ONLY THE PROMPT, AVOID STARTING WITH "Generate a prompt"" where
user input is the extracted chunk transcription.
```

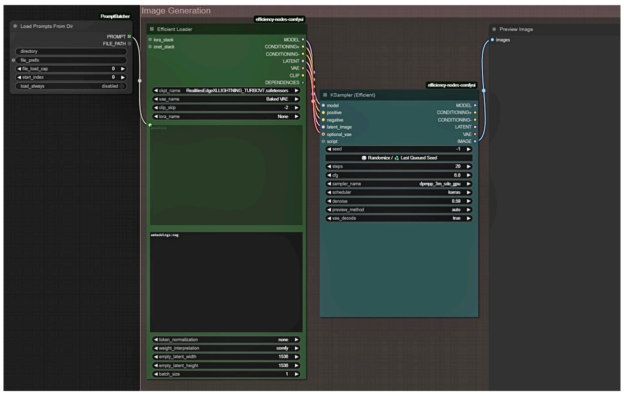
```
temperature            0.75
max_tokens             312
top_p                  1.00
seed                   1941818968
control after generate randomize
max_retries            2
stop
json_mode              false

Save Text File With Path
path                   [time(%Y-%m-%d)]/
prefix                 prompt_for_chunk_
counter_separator      _
counter_length         0
suffix
output_extension       txt
```

Figure 3: *Text2Prompt group node*

as input for the diffusion model in the subsequent stage.

Prompt Loading and Diffusion Synthesis Block
The final node group orchestrates the actual image generation. First, a "Load Prompts From Dir" node (often called PromptBatcher) reads each saved prompt string in turn and emits it as the positive conditioning input. These text prompts, together with a fixed "negative" embedding tensor containing a state-of-the-art blacklist of undesirable keywords, are then fed into an *Efficient Loader* node. This loader assembles all required components—SDXL checkpoint, CLIP encoder, optional LoRA stacks, and VAE decoder while routing the prompt into the `positive` port and the negative embedding into the `negative` port. Next, the combined conditioning vectors and an empty latent tensor travel through a *sd-perturbed-attention* module, which perturbs the model's attention maps according to a user-specified scale factor. The perturbed latents and conditioning are handed



```
Load Prompts From Dir
directory
file_prefix
file_load_cap          0
start_index            0
load_always            disabled

Efficient Loader
ckpt_name              RealitiesEdgeXLLIGHTNING_TURBOV7.safetensors
vae_name               Baked VAE
clip_skip              -2
lora_name              None
embeddings:neg
token_normalization    none
weight_interpretation  comfy
empty_latent_width     1536
empty_latent_height    1536
batch_size             1

KSampler (Efficient)
seed                   -1
steps                  20
cfg                    6.0
sampler_name           dpmpp_3m_sde_gpu
scheduler              karras
denoise                0.59
preview_method         auto
vae_decode             true
```

Figure 4: *Image generation group node*

off to a *KSampler (Efficient)* node, where the core diffusion sampling occurs: here you configure the sampler algorithm (e.g. Euler or Karras), number of steps, CFG scale, seed, scheduler type, and denoise strength. As the reverse diffusion loop unfolds, the latent representation converges toward an image that matches the positive prompt while avoiding the negative concepts. Finally, the sampler outputs a batched latent image tensor, which is automatically decoded by the integrated VAE and passed into a *Preview Image* node. This node renders the final RGB output on the ComfyUI canvas, completing the end-to-end transformation from spoken audio to visually rich imagery.

Results

The following sections illustrate a proof of concept using a simple fairy tail. For context, this is the exact textual representation of the audio:
"C'era una volta un gatto di nome Leo dallo splendente pelo rosso che viveva in un cortile polveroso, tra vasi di fiori colorati. Ogni mattina si stirava al primo chiarore e, con fare silenzioso, saliva sui muri per contare le ombre degli ulivi quando si accorse che in un angolo nascosto c'era una scatola di metallo lievemente arrugginita. Al suo interno giaceva un grosso libro polveroso in cui veniva raccontata una storia di un mondo lontano. Il gatto non sapendo leggere inizio a girare le pagine incuriosito quando si accorse di un bellissimo disegno di un castello incantato. Il gatto, affascinato, portò il quaderno dentro al suo riparo cosi che potesse ogni giorno sfogliava le pagine, immaginando i colori e i profumi di quel mondo lontano. Così, grazie a quel fantastico libro, il gatto Leo poteva sognava a occhi aperti: dal cortile alle torri del borgo, ogni angolo nascondeva un'idea e ogni pomeriggio si trasformava in un sogno da esplorare."

Impact of Audio Cleaning on Transcription Fidelity
To quantify the benefit of our audio-cleaning stage, we conducted an experiment comparing Whisper's transcription of (a) the original noisy recording versus (b) the same recording after background noise suppression. As ground truth, we used a manually verified transcript that matches the audio with 100 % fidelity. Both versions of the audio were fed to the Whisper "small" model under identical decoding settings. We then computed SimHash fingerprints on character 3-grams for each transcript and measured the Hamming distance and similarity against the reference fingerprint.

Filename	Hamming	Similarity
cleaned transcription.txt	8	0.8750
original transcription.txt	19	0.7031

Table 1: *Comparison of SimHash (3-gram) distances between Whisper outputs and the reference transcript.*

As shown in Table 1, the transcript produced after noise removal is substantially closer to the reference (Hamming distance=8, similarity=0.875) than the transcript obtained from the raw input (Hamming distance=19, similarity=0.703). This result demonstrates that our preprocessing block, by isolating and attenuating background interference, significantly reduces transcription errors and yields outputs that more faithfully reflect the original spoken content. In this case the chunking happens before the transcription but for the purpose of demonstrating the effectiveness of the audio preprocessing step the single chunks are then merged in a single file that allowed this evaluation.

Chunk to Prompt Generation
Prompt quality hinges critically on how the source audio is segmented: chunks that respect natural pauses and narrative boundaries tend to produce more coherent and contextually rich prompts. Unlike purely quantitative metrics, we assess each chunking strategy by its ability to yield semantically self-contained segments that a downstream LLM can readily transform into descriptive image prompts. Throughout development, ComfyUI proved indispensable: its node-based architecture caches intermediate results and only recomputes affected branches when parameters change, allowing rapid experimentation with different segmentation algorithms without reprocessing the entire audio. For the fairy-tale recording under study, silences deliberately inserted by the narrator provided clear demarcations between story beats. For the sake of simplicity, we will use our "Silence Chunker" to obtain te results and proof of concept discussed further. By configurint its node to detect pauses above a 300 ms threshold, we obtained clean, semantically coherent slices corresponding to complete narrative episodes. These transcription fragments were then concatenated, preserving chronological order, into a single text file that the LLM node ingested as its user prompt. Inside ComfyUI, the LLM node takes this unified transcript and applies a consistent system prompt template, yielding a series of finely tuned image-generation directives. Because each chunk was already aligned to a meaningful story unit, the resulting prompts capture both the visual motifs and the emotional tone of each scene, demonstrating that thoughtful audio segmentation is a key enabler of effective text-to-image workflows.

Generating the Images

Using ComfyUI's visual pipeline, we produced a series of stylized images that correspond directly to each narrative segment and its automatically generated prompt. To facilitate qualitative assessment, we present a composite grid in which each row aligns:

1. **Story Excerpt:** the original transcribed chunk of the fairy-tale,
2. **Generated Prompt:** the LLM's output that conditions the diffusion model,
3. **Resulting Image:** the stylized render produced by SDXL 1.5.

The resulting framework creates a bidirectional dialogue between text and image, enhancing reader engagement and reinforcing comprehension through multimodal reinforcement. Quantitative metrics aside, the qualitative juxtaposition of story excerpt, generated prompt, and rendered image provides a transparent evaluation protocol that highlights the contribution of each component to the emergent narrative. Beyond its analytical utility, this pipeline offers a tool to transform passive reading into an interactive visual experience.

Comparisons

In this section, we examine several works that are conceptually aligned with Hear-to-See. We highlight the differences in methodological approach, thereby situating Hear-to-See more precisely within the current state of the art and elucidating its potential strengths and weaknesses. We present a comparative analysis of the techniques employed in a corpus of ten papers, grouped into two main categories of five documents each: those using audio input and those using textual input. Even if Hear-to-See sticks with audio inputs, analyzing techniques used for textual generation make us understand better possibile future developments.

Audio Input

ACT Encoder-Decoder vs Hear-to-See
The first approach under consideration is that described by Lee et al. (see References section, even for other papers discussed). The two methods diverge primarily in their audio-to-text conversion strategies. Hear-to-See employs a modular pipeline comprising explicit stages for audio classification (speech versus music), source separation, and silence- and prosody-based segmentation, followed by transcription via Whisper and template-driven prompt engineering for Stable Diffusion. In contrast, Lee et al. introduce a unified, end-to-end Audio Captioning Transformer (ACT) that directly maps raw audio to descriptive captions, augmented by audio-attention and sentence-attention mechanisms to produce a latent representation without any intermediate processing steps. The ACT is an encoder–decoder Transformer pre-trained on AudioSet for audio tagging and subsequently fine-tuned on captioning datasets such as AudioCaps. Given an arbitrary "in-the-wild" audio waveform, ACT generates a richly detailed descriptive sentence, for example, "A person walking on a beach with waves", thereby transcending the limitations of fixed taxonomic categories. During generation, the audio-attention mechanism assigns weights to concepts according to their temporal prominence, while sentence-attention emphasizes salient nouns via part-of-speech tagging; these attention weights, combined with positional encodings, yield the latent vectors that directly condition the Stable Diffusion model through optimization. Hear-to-See offers fine-grained control over each module, which simplifies debugging and component replacement, and achieves superior computational efficiency by avoiding the costly latent-space optimizations required by ACT yet still delivers high visual quality. Moreover, our use of customizable templates renders the prompt-engineering process transparent and amenable to human-in-the-loop modification. By contrast, the Lee et al. approach operates as a "black-box," offering less interpretability. Furthermore, leveraging a large language model for textual prompt generation enables flexible adaptation to different stylistic requirements without necessitating additional retraining. Although the ACT framework can achieve semantically precise captions, its audio-attention and sentence- attention mechanisms endow the textual output with auxiliary embeddings that guide the diffusion model's focus, its inherent opacity and computational demands present clear trade-offs relative to our modular design.

SonicDiffusion vs Hear-to-See
Both Hear-to-See and SonicDiffusion seek to bridge audio and imagery, yet they do so from markedly different vantage points. Hear-to-See is engineered to produce visual summaries of extended narrative streams by segmenting audio into semantically coherent "chunks" and rendering each fragment as a stylistically consistent illustration. SonicDiffusion, by contrast, treats sound itself as the direct steering signal for both image generation and editing, building on a diffusion model originally trained on text. Unlike Hear-to-See's selective preprocessing, SonicDiffusion ingests audio "as-is" by converting it into log-mel spectrograms and projecting these directly into a shared embedding space with text via the CLAP encoder. CLAP is a neural network designed to map audio spectrograms into the same latent space as textual embeddings, thereby yielding semantic vectors that can be directly compared with, and integrated into, text-based models. Architecturally, SonicDiffusion employs an "adapter" strategy: it freezes the pre-trained weights of Stable Diffusion and augments them with an Audio Projector module. This projector transforms the CLAP vectors into discrete tokens, which are then injected into the UNet decoder through gated cross-attention layers. This design supports both the synthesis of wholly new images and the sound-guided refinement of existing photographs. In terms of outputs, Hear-to-See delivers a sequence of cartoon-style vignettes arranged to follow the narrative flow—ideal for converting fables or podcast episodes into visual storyboards. SonicDiffusion, in contrast, can produce entirely original images that embody the timbral, dynamic, and textural qualities of an audio signal (for example, "crackling fire" or "gentle rainfall") and can even perform targeted edits on real photographs based on an audio clip. Although SonicDiffusion shares with the work of Lee et al. a higher computational footprint, it differentiates itself and Hear-to-See by bypassing any intermediate textual generation and operating entirely within the latent space. While not expressly designed for storytelling in the manner proposed by Can Biner et al., its fundamental notion of deriving a text-aligned vector to inform image content renders it readily adaptable to narrative applications with the proper tuning.

Sound2Scene vs Hear-to-See
Sound2Scene is designed to generate images of natural scenes directly from "in-the-wild" audio signals, without any textual labels or supervision. This approach closely mirrors the framework proposed by Lee et Al. that we discussed earlier. The implementation builds on a fixed Stable Diffusion backbone, which interacts with the input from:

1. **A pre-trained audio encoder** (an audio-classification network) that extracts multi-layer feature representations
2. **A lightweight embedder** that projects these features into the same dimensional space as text embeddings, followed by an attentive pooling layer that condenses the sequence into a single "audio token".
3. **Concatenation** of this audio token with the tokens of a predefined base prompt, which is then passed to the diffusion model.

The key novelty of Sound2Scene is that, unlike SonicDiffusion, which operates on log-mel spectrograms, relies on a direct audio-encoder representation. As with SonicDiffusion, it produces standalone static images rather than videos or temporally coherent sequences; achieving full storytelling would require temporal extensions and chunking strategies similar to those used in Hear-to-See. Nevertheless, Sound2Scene is the first among the alternatives we surveyed to incorporate an a priori textual prompt: while Hear-to-See derives its text prompt directly from the audio, one could instead transcribe the incoming narrative with Whisper and chunk it appropriately to drive a comparable storytelling pipeline. In this way, Sound2Scene can perform as a storytelling tool in the way we intended it in Hear-to-See, even if the main difference relies on the fact that, here, the audio should give more 'contextual' informations than the textual input itself, which gives what the image should represent, and so what we in Hear-to-See tried to model with the chunking and salience mechanisms.

MACS vs Hear-to-See
This is probabily the tool we can get the mostful insights of. MACS is the very first tool that addresses audio to image generation by adopting a separation-before-generation paradigm: it first disentangles a mixed audio clip into latent source components, ranks them by contextual salience, aligns them semantically with language, and then conditions a diffusion image generator on the resulting structured audio representation. In **Stage 1** (**Multi-source Sound Separation, MSS**), the input spectrogram is passed through a UNet that predicts a fixed number of masks whose products with the mixture yield putative source estimates. Training follows a MixIT-like "mixture of mixtures" self-supervised reconstruction loss, supplemented by CLAP-based audio-text contrastive alignment to tie separated sources to semantic labels. A ranking loss encourages the model to order sources by their contribution to the original mixture, emphasizing salient events over background noise. **Stage 2** injects the ordered source embeddings into a largely frozen Stable Diffusion backbone through a lightweight Decoupled Cross-Attention Adapter. A small MLP, then, maps audio embeddings to the conditioning space required by the diffusion UNet; training uses the standard denoising objective, updating only the adapter and projection layers. This design yields efficient adaptation without full model fine-tuning. Evaluated on single, mixed, and explicitly multi-source benchmarks, MACS consistently outperforms prior A2I approaches, like the Sound2Scene we discussed before, across visual and semantic metrics. Like we said before, MACS is the first audio to image model that genuinely separates the incoming audio stream into interpretable components instead of treating it as a coarse text surrogate. In our domain, MACS would isolate the speech track from the background ambience and, crucially, assumes that every separated component may carry semantic weight. Hear-to-See, by contrast, has so far assumed that background sources are mere noise to be attenuated, focusing exclusively on vocal salience. MACS therefore invites a new question: what if the background is itself meaningful? Story-driven contexts, think film narration or theatrical performance where music underscores the scene, demand that we treat background music not as clutter but as a first-class semantic cue. Thanks to mature music/speech separation models such as the Demucs we used, it is possible to isolate musical layers with high fidelity and use that stem for further processing. Incorporating such separated musical embeddings into Hear-to-See would force us to revise its core assumptions and could unlock richer cross-modal understanding, enabling the pipeline to infer narrative tone, emotional arc, or genre cues directly from accompaniment.

Topic-Segmented Podcast Player vs Hear-to-See
Our main idea of 'storytelling' could be seen, in a specific context of application, also as filling the gap in podcast consumption, namely the absence of visual scaffolding that allows listeners to preview content, skip low-value portions and pinpoint specific topics, which was our objective in Hear-to-See. The tool we're discussing now highlights this interviewing twelve habitual podcast listeners and confirmed three core needs: fast access to a desired segment, effortless bypassing of advertisements or lengthy introductions, and an overview of the episode's structure before committing listening time. To meet these needs, the authors devised a four-stage generative-AI pipeline that first converts the audio into a time-stamped transcript, then segments the text into six to eight coherent topics using a large language model, generates a representative illustration for each topic with a diffusion model, and finally synchronizes these assets within an Android player that presents a segmented timeline, clickable transcript and thumbnail

strip. A within-subject usability study comparing this interface with a baseline keyword-search player showed that the topic-based designs doubled episode-comprehension scores and were roughly four times faster than keyword search when retrieval queries were vague; eleven of the twelve participants preferred a segmented mode, with the image-augmented variant judged most intuitive. Participants also reported lower cognitive load and higher satisfaction when using the segmented views. The authors recommend exposing the topic map at once, fusing text and imagery for immediacy, retaining keyword search for fine-grained look-ups, and visualizing segment length to convey topical salience. Although the study is limited by its small Korean-language sample, it offers the first empirical evidence that multimodal, generative-AI-driven segmentation markedly enhances podcast browsing efficiency and user experience, suggesting fruitful directions for broader cross-lingual validation and improved visual coherence of generated thumbnails, all aspects that deeply connect with the goals of Hear-to-See, even if in a different context of application. The authors' tool differs from prior solutions in that, among all the systems reviewed, it is the first, together with Hear-to-See, to produce not a single illustration but an entire carousel of images that are mutually coherent from a semantic standpoint. As in our approach, the prompts fed to the diffusion model are generated by a large language model acting on the transcripts of the previously segmented audio. The overall pipeline is therefore close to Hear-to-See: it assumes spoken word-centric input and treats background noise as semantically negligible. We contend, however, that Hear-to-See would outperform the authors' tool under comparable conditions, because their chunking strategy rests solely on temporal segmentation and ignores additional salient information that can be inferred directly from the audio signal. In this respect, their method remains aligned with other baselines discussed in this section, which effectively reduce audio to a mere textual surrogate. Nonetheless, the user-experience findings reported by the authors reinforce our own conclusions: sampling listener feedback confirms that visually enriched output substantially improves content navigation and overall satisfaction when consuming spoken-word media.

Textual Input

StoryGAN vs Hear-to-See

In our work, the StoryGAN architecture served as the fundamental inspiration for the development of Hear-to-See: whereas StoryGAN shows how to translate existing textual narratives into coherent, high-quality image sequences via a story encoder, a hierarchical contextual encoder ("Text2Gist"), and two-level discriminators, Hear-to-See extends this paradigm to the audio domain by adding a layer of complexity for spoken language processing. A GAN (Generative Adversarial Network) is a deep learning framework composed of two competing modules: the generator, which produces synthetic data (e.g., images) and the discriminator, which attempts to distinguish real samples from generated ones, thereby forcing the generator to progressively improve the realism of its outputs. StoryGAN begins directly with the written narrative, harnessing the text's latent structure to guide GAN based generation and ensure both local and global consistency between the produced images and the storyline. In contrast, Hear-to-See starts from raw audio input and requires an extensive preprocessing pipeline: speech detection and isolation, intelligent segmentation based on pauses and pitch changes, accurate transcription, and temporal alignment, before any visual synthesis can commence. Only after this audio-to-text conversion does the visual-generation phase begin. A crucial aspect of Hear-to-See is its ability to distinguish two levels of semantics within the original audio, as discussed in previous sections and emphasized here: the first is the "conceptual" semantics of the story, corresponding to the objective content of the text; the second is the "narrator's" semantics, namely the nuances of salience, prosody, and emphasis with which the user delivers the narrative. By analyzing acoustic features such as intonation, rhythm, and stress, Hear-to-See weights each segment of text in the generation process, visually accentuating the portions deemed most relevant or emotionally charged. Examining works like StoryGAN reveals that one of Hear-to-See's performance limitations often lies in its omission of attention mechanisms: in addition to the audio-domain methods outlined earlier, we could have implemented attention over the textual content itself, an approach that, given that our background noise carries no semantically relevant information, might have been more effective.

Neural Storyboard Artist vs Hear-to-See

In **Neural Storyboard Artist**, the input consists of a multi-sentence textual narrative: each sentence corresponds to the description of a scene or action, exactly as in traditional cinematic storyboards, and serves as the driver for selecting and generating the illustrative images. The retrieval-plus-refinement pipeline unfolds in two sequential stages. In the retrieval stage, the Story-to-Image Retriever module employs Contextual-Aware Dense Matching (CADM) to compare each textual description against a large database of film frames. By means of a hierarchy of intra- and inter-sentence attention mechanisms, CADM enriches word representations with surrounding context and computes a fine-grained similarity between textual concepts and visual regions, thereby retrieving one or more highly relevant inspirational images for each sentence. In the subsequent refinement stage, the Storyboard Creator transforms these "raw" images into cohesive storyboard panels through three principal operations: (1) semantic region segmentation, which removes elements deemed irrelevant; (2) stylistic unification via CartoonGAN, which applies an adversarial style transfer to achieve a homogeneous, professional cartoon aesthetic; and (3) semi-automatic replacement of characters with pre-existing, mutually coherent 3D models (selected rather than generated de novo). By grounding each panel in authentic visual material, this approach ensures both semantic fidelity to the original descriptions and the high stylistic quality characteristic of professional storyboards. This methodology stands in stark contrast to StoryGAN, which generates each image entirely from scratch based solely on textual input and relies on learned text–image similarity. In our initial efforts, we attempted to follow a strategy analogous to that of Neural Storyboard Artist: we assembled a dataset of cartoon-style images and applied extensive data augmentation, then fine-tuned a diffusion model via Low-Rank Adaptation (LoRA), aiming to integrate the principles of CADM and CartoonGAN into a single LoRA-based process. However, we ultimately abandoned this avenue due to computational resource limitations that produced an insurmountable bottleneck. We found discussing Neural Storyboard Artist as main importance because, with StoryGAN, has been one of the articles that inspired the idea that led to Hear-to-See.

RCDM vs Hear-to-See

The tool discussed in "Boosting Consistency in Story Visualization with Rich-Contextual Conditional Diffusion Models" introduces a two-stage framework for improving the generation of narrative image sequences from captions. In the first stage, a **Frame-Prior Transformer** predicts semantic embeddings for all missing frames by jointly attending to (i) the complete set of captions and (ii) any "known" frames supplied a priori both modalities having been encoded via a CLIP encoder. In the second stage, a **Frame-Contextual 3D Diffusion Model** fuses these semantic embeddings with the original textual inputs to produce the entire sequence in a single inference pass, thereby preserving both stylistic coherence and temporal continuity. Empirically, this unified inference strategy outperforms contemporary autoregressive models and GAN-based approaches in terms of accuracy metrics and human evaluation scores. The **CLIP encoder** (Contrastive Language–Image Pre-training) itself comprises two parallel branches: a visual backbone (e.g., ResNet or Vision Transformer) and a textual Transformer. Both branches are co-trained with a contrastive objective over hundreds of millions of image–caption pairs, driving the model to embed semantically matching image–text pairs close together in latent space while pushing non-matching pairs apart. This alignment yields robust, semantically rich representations that facilitate seamless cross-modal comparison and fusion. Within the Frame-Prior Transformer, the CLIP-derived embeddings of known frames and of all captions are merged to construct:

1. A global context embedding, which enforces stylistic and semantic continuity across the story
2. Frame-Specific Hidden Tokens, which ensure that each generated frame remains tightly coupled to its corresponding caption, respecting positional relationships and incrementally modeling the evolution of spatial and temporal dynamics.

The combined outputs of these two components are reshaped into a unified tensor serving as the input to the Frame-Contextual 3D Diffusion Model. By generating all frames in one shot, RCDM (Rich-Contextual Diffusion Model) avoids the error accumulation typical of step-wise methods. By contrast, in our "Hear-to-See" pipeline we first segment Whisper's transcript into textual chunks and then employ an LLM to transform each chunk into a separate diffusion prompt. Here, semantic consistency across frames relies entirely on the LLM's linguistic capabilities, its contextual understanding, syntactic coherence, and lexical richness. In practice, this simple text-only strategy (e.g. concatenating all chunks so the model knows which text segment corresponds to each prompt, plus global instructions such as "keep the subject constant") can yield surprisingly coherent image sequences even without a sophisticated context-aware diffusion backbone like RCDM. However, the LLM-based approach tends to dilute detail when stories involve many characters or intricate settings. A promising hybrid strategy would be to generate an initial subset (e.g. two-thirds) of frames via LLM-derived prompts to capture global semantics, and then leverage RCDM's stylistic and temporal coherence guarantees to complete the remaining frames. This combined scheme could marry the flexibility of pure LLM prompting with the consistency benefits of rich-context conditional diffusion.

MARTT vs Hear-to-See

In their 2021 paper "Integrating Visuospatial, Linguistic and Commonsense Structure into Story Visualization," the authors tackle the same challenge we address: converting a sequence of (primarily textual) captions into a coherent series of images. To this end, they propose a hybrid encoder (MARTT) that leverages each sentence's tree-structured syntactic representation, propagates it across the story via a recurrent memory module, and enriches it with commonsense knowledge extracted from ConceptNet. A two-stage GAN generator then uses these multimodal embeddings to render frames that preserve both character consistency and narrative continuity. Evaluated on PororoSV and FlintstonesSV, their approach yields substantial gains in FID and in maintaining character coherence over prior models. This very architecture—combining fine-grained linguistic structure, external knowledge, and visual feedback—informed our design: whereas in 2021 the inclusion of explicit symbolic structures enhanced text–image alignment, today we can delegate much of that complexity to off-the-shelf large language models. At the time, such models were neither truly multimodal nor narratively reliable, forcing the authors to rely on parse trees, knowledge-base graphs, and contrastive losses as stand-ins for a unified reasoning engine. In our Hear-to-See system, the single LLM we employ subsumes the roles of parse-tree extraction, commonsense graph querying, and recurrent memory: it produces precise prompts for the diffusion model while ensuring global coherence, retrieves world knowledge on the fly, recognizes and tracks character references, and fills in implicit details all in one pass. This obviates the need for explicit syntactic parsing and external ontology queries, reduces pipeline failure points, and accelerates development. (Should users require it, an external knowledge base can still be integrated simply by allowing them to append domain-specific facts directly into the LLM prompt, as we demonstrate in our paper.) When viewed purely through the lens of image generation, our pipeline is both simpler and more robust: the LLM natively encodes syntax and commonsense; the audio input injects temporal dynamics and expressive intensity; and the diffusion model delivers sharper, more stable frames than a GAN. The result is a compact workflow that generates storyboards that are not only coherent and expressive but also precisely attuned to the rhythm of the spoken narrative.

StoryAgent vs Hear-to-See

StoryAgent employs a hybrid two-phase architecture that marries "top-down" story planning with "bottom-up" asset generation. In the first phase, a network of LLM based agents decomposes the user's textual prompt into a hierarchical JSON structure. Specialized subgroups of agents take ownership of discrete narrative components, defining the story arc, characters, settings, story beats and individual scenes and operate in an expert-critic loop, iteratively comparing and refining these components until the overall structure converges on a coherent outline. In the second phase, the text descriptions produced by the LLMs drive dedicated generative pipelines text-to-image, 2D animation, speech synthesis and sound-effect generation which automatically create and assemble multimedia assets, ensuring tight integration and semantic consistency across text, image, audio and animation. The integration of LLMs within this architecture yields significant benefits, many of the same that motivated our use of an LLM in Hear-to-See for textual prompt generation. By automatically extracting parameters from natural-language descriptions and decomposing complex concepts into simpler subcomponents, LLMs greatly simplify the workflow compared to more monolithic architectures that fuse multiple attention mechanisms to preserve stylistic and narrative coherence. This approach strikes a favorable balance between the expressiveness of the representation and computational efficiency, while also supporting the injection of user specific directives at any point in the pipeline. Although Hear-to-See and StoryAgent serve different end goals, both systems recognize the LLM as a pivotal tool for abstracting and structuring raw text into intermediate formats consumable by specialized generative modules. In Hear-to-See, the LLM ingests a unified transcription of all audio chunks and produces a descriptive prompt that conditions a diffusion model for image synthesis. Likewise, in StoryAgent the LLM parses input text into asset-specific JSON objects (for characters, environments, sounds, SFX), orchestrates expert critic reviews, and schedules generation tasks so that every component remains narratively and stylistically aligned throughout the story. Moreover, StoryAgent's methods can enhance Hear-to-See when the goal shifts from static image sequences to more sophisticated storytelling. For example, StoryAgent's audio asset workflow, where an LLM transforms semantic text descriptions into calls to AudioGen or MusicGen, could be applied to each Hear-to-See chunk to automatically generate synchronized soundtracks and sound effects, treating music itself as an informative narrative layer. Similarly, StoryAgent's CC2D animation and clip assembly pipeline suggests a path for Hear-to-See to segment image outputs into sub clips, assign camera movements and character animations based on acoustic-signal derived keyframes, and then compile a continuous video complete with dynamic transitions and audio. By incorporating the expert-critic review loops and iterative reflection mechanisms of StoryAgent, Hear-to-See could transcend its inherently static storytelling model to deliver a fully automated, cinematic narrative experience.

Concluding the Comparative Analysis

The core idea behind our Hear-to-See project is to capture the narrator's subjective emphasis on particular segments of the incoming audio and find those segments. Conventional summarization models focus almost exclusively on semantic importance, overlooking the primary information conveyed by audio inputs: what the speaker actually experiences and how is intended what he's eventually saying. While the quality of the generated images could certainly be enhanced by incorporating more powerful encoders and attention mechanisms, in line with the alternatives we reviewed, doing so would sacrifice the computational efficiency of our overall architecture, like we discussed. But it's not just a matter of computational efficiency: when encoders and neural based models are used in the same way we saw in the previous comparisons, we would lose the key concept of Hear-to-See itself. Even if, although the models we benchmarked Hear-to-See against were not originally designed for storytelling per se, this limitation is in fact secondary: each aims to generate images from audio inputs, and so a rudimentary form of storytelling ia achievable simply by issuing multiple calls to these models using chunking and cleaning techniques to give inputs iteratively. In our view, all of these approaches treat audio as nothing more than a surrogate for a text representation and that's what we were claiming before: while that may suffice for many applications, it ignores very important information for storytelling purposes that are directly in the audio input itself. Specifically, variations in pitch and, in particular, the prosodic inflections present in spoken audio convey vital information about what the narrator 'hears' and 'feels.' By neglecting these cues, one loses a rich source of interpretative data, which we have instead sought to model explicitly in Hear-to-See. Possible future developments could involve leveraging more complex neural models still based on encoder–decoder architectures; however, rather than reducing audio to a mere "text" in another form, the use of these models would focus on producing chunks that are semantically meaningful for the narrator as outputs, not basing the approach on the traditional way to intend semantic meaningfulness. As noted in MACS, future work could introduce even a dedicated branch in the pipeline for extracting information from the musical stem. This entails revising our original assumption that music is simply noise and, instead, treating it as a semantically informative signal. While image generation should still be guided primarily by the core semantic content of each audio chunk, background music can supply additional contextual cues, such as mood, genre, or narrative tone, that are not otherwise accessible. Incorporating these musical embeddings explicitly would allow the model to leverage accompaniment-driven information to enrich its cross-modal representations; for a better understanding, see the 'MACS vs Hear-to-See' discussion.

About the References
First 10 articles regard what we discussed before in the Comparisons section, the others you find are general resources we took insipiration from during the redaction of this article.

References

[1] T. Lee, J. Kang, H. Kim, T. Ki (2024). Audio Captioning Transformer: End-to-end generation of descriptive captions from raw audio.

[2] B. Can Biner, S. Farrin Marouf, K. Umur Berkay, D. Ceylan, E. Erdem, A. Erdem (2024). SonicDiffusion: Audio-Driven Image Generation and Editing with Pretrained Diffusion Models

[3] K. Sung-Bin, A. Senocak, H. Andrew Owens, T. Oh (2023). Sound to Visual Scene Generation by Audio-to-Visual Latent Alignment

[4] H. Zhou, X. Guo, Y. Zhu, A. Wai-Kin Kong (2025). MACS: Multi-source Audio-to-image Generation with Contextual Significance and Semantic Alignment

[5] J. Park, C. Lee, E. Cho, U. Oh (2024). Enhancing the Podcast Browsing Experience through Topic Segmentation and Visualization with Generative AI

[6] Y. Li, Z. Gan, Y. Shen, J. Liu, Y. Cheng, Y. Wu, L. Carin, D. Carlson, J. Gao (2019). StoryGAN: A Sequential Conditional GAN for Story Visualization.

[7] Shizhe Chen, Bei Liu, Jianlong Fu, Ruihua Song, Qin Jin, Pingping Lin, Xiaoyu Qi, Chunting Wang, Jin Zhou (2019). Neural Storyboard Artist: Visualizing Stories with Coherent Image Sequences.

[8] F. Shen, H. Fei Shen, S. Liu, J. Zhang, C. Wang, X. Han, W. Yang (2024). Boosting Consistency in Story Visualization with Rich-Contextual Conditional Diffusion Models

[9] A. Maharana, M. Bansal (2021). Integrating Visuospatial, Linguistic and Commonsense Structure into Story Visualization

[10] S. Sohn, D. Li, S. Zhang,C. Chang, M. Kapadia (2024). From Words to Worlds: Transforming One-line Prompt into Immersive Multi-modal Digital Stories with Communicative LLM Agent Human Effort.

[11] S. Davis and P. Mermelstein (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(3), 357–366.

[12] S. F. Boll (1979). Suppression of Acoustic Noise in Speech Using Spectral Subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2), 113–120.

[13] TensorFlow Authors (2023). YAMNet: A Pre-trained Audio Event Classification Model. Accessible by `https://www.tensorflow.org/hub/tutorials/yamnet`.

[14] A. Défossez, N. Usunier, L. Bottou, and F. Bach (2019). Music Source Separation in the Waveform Domain. *arXiv preprint arXiv:1911.13254*.

[15] J. Foote (2000). Automatic Audio Segmentation Using a Measure of Audio Novelty. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 452–455.

[16] A. Radford, J. W. Kim *et al.* (2022). Whisper: Robust Speech Recognition via Large-Scale Weak Supervision. *arXiv preprint arXiv:2212.04356*.

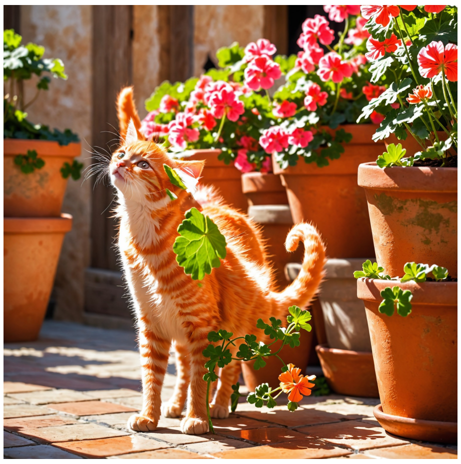
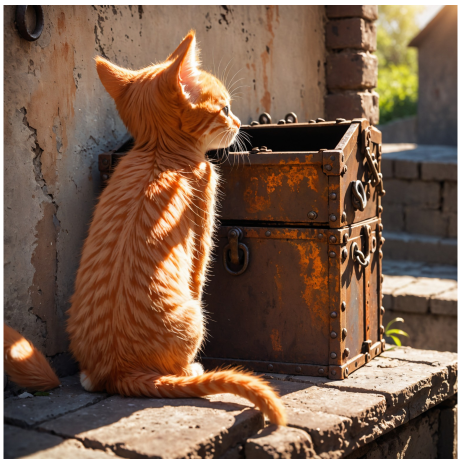
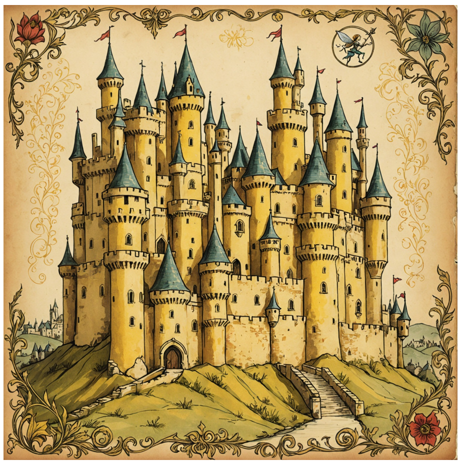
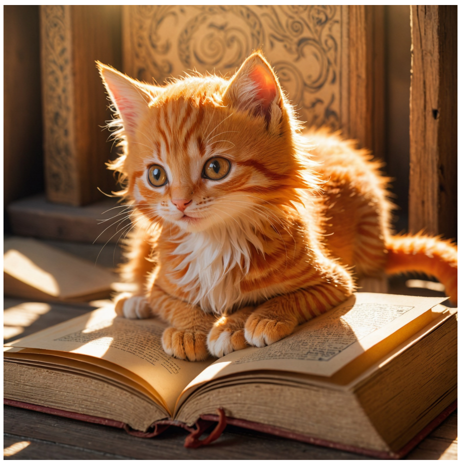
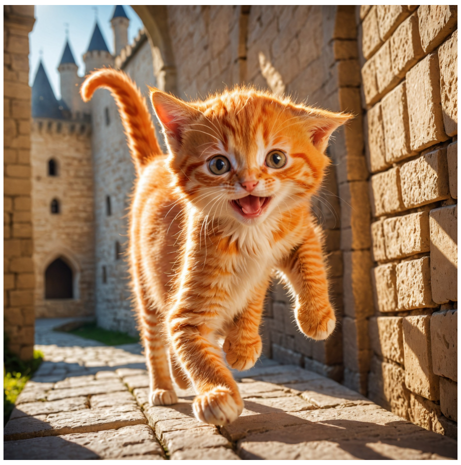
[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 5998–6008.

[18] J. Ho, A. Jain, and P. Abbeel (2020). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 6840–6851.

[19] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer (2022). High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10684–10695.

[20] D. P. Kingma, D. Podell *et al.* (2023). SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. *arXiv preprint arXiv:2307.01952*.

[21] Comfyanonymous (2025). ComfyUI: The Most Powerful and Modular Stable Diffusion GUI. Accessible by `https://github.com/comfyanonymous/ComfyUI`.

 **Prompt** → Orange kitten, dusty courtyard with colorful pots, sunlit, rustic texture, serene curiosity, nostalgia, vibrant masterpiece, fairytale	 **Prompt** → Orange kitten looking inside a rusted iron chest, from behind, sharp focus, nostalgia, vibrant, masterpiece, fairytale.
 **Prompt** → Yellow page, medieval castle illustration, well-painted, ancient book style, masterpiece, high quality.	 **Prompt** → Orange kitten lying on a dusty book indoors, rustic sunlit texture, serene curiosity, nostalgia, vibrant, fairytale masterpiece.
 **Prompt** → Orange kitten running through the streets of a medieval castle, dreamy light and shadow, rustic nostalgia, vibrant concept art, fairytale masterpiece.	