

VIDEO-TO-MUSIC GENERATION FOR GAMEPLAY VIDEOS

Felipe Marra[↓]

Lucas N. Ferreira[↓]

Universidade Federal de Viçosa (UFV)
Departamento de Informática
Viçosa, Minas Gerais - Brasil

Universidade Federal de Minas Gerais (UFMG)
Departamento de Ciência da Computação
Belo Horizonte, Minas Gerais - Brasil

UFMG MOSAIC Group[↓]

ABSTRACT

Video-to-music models have advanced considerably in the last few years, particularly in film and music video applications. In this paper, we investigate this problem in the video game domain, which introduces new challenges for these models: video frames are rendered graphics, music is mostly synthetic audio, and soundtracks loop across entire levels rather than following on-screen events. We introduce a new dataset of 217.6 hours of Super Nintendo (SNES) gameplay video paired with 485 hours of clean soundtracks, free of sound effects and voice-overs, matched to gameplay audio via audio fingerprinting. With this dataset, we train a simple encoder-decoder transformer that passes video features directly to a MusicGen decoder, comparing different encoding strategies: textual descriptions (T5), independent frames (ViT), or spatiotemporal patches (ViViT). Each encoder is tested both frozen and fine-tuned, while the decoder is always fine-tuned. Frozen encoders match or outperform their fine-tuned counterparts on every metric, and the frozen ViViT achieves the best overall results. We compare this model with state-of-the-art baselines using both objective metrics and a listening study ($N = 96$). Despite having up to 18% fewer parameters, our model outperforms all baselines on objective metrics, surpasses GVMGen in the listening study, and performs comparably to OSSL.

Index Terms— multimedia, video-to-music, video games.

1. INTRODUCTION

Video-to-music (V2M) research primarily focuses on film [1, 2] and music videos [3, 4, 5], domains with real footage where the generated music needs to be temporally aligned with the video. Video games are another promising application: V2M models could help independent developers, who often lack the resources for custom soundtracks, create music that fits their game’s visuals. However, video games pose a different challenge: both graphics and audio are synthetic, and soundtracks often loop across entire levels rather than following on-screen

events. It remains unclear whether V2M models, built on encoders pre-trained on real-world data, transfer to this domain.

We extend V2M to retro video games, a style that remains popular among independent titles. We introduce a Super Nintendo (SNES) dataset that pairs gameplay videos with clean soundtracks, free of voice-overs and sound effects. By fingerprinting 10-second YouTube clips against the original soundtracks and downsampling across 11 genres for balance, we obtain 217.6 hours of video paired with 485 hours of music.

With this dataset, we train a simple encoder-decoder transformer that passes video features directly to a pre-trained MusicGen [6] decoder via cross-attention. To find the best video encoder for this domain, we compare three approaches common in the V2M literature: describing the video textually (T5) [7], encoding it frame by frame (ViT) [8], or fusing frames across time (ViViT) [1]. Each encoder is initialized with pre-trained weights and tested both frozen and fine-tuned, while the decoder is always fine-tuned. Frozen encoders match or outperform their fine-tuned counterparts on every metric, and the frozen ViViT achieves the best overall results, with the lowest FAD [9] and competitive KL divergence and ImageBind [10] scores.

We compare this model with state-of-the-art (SOTA) baselines: Herrmann-1 [7], OSSL [1], and GVMGen [8]. Our model outperforms all baselines on objective metrics. In a listening study ($N = 96$), it outperforms GVMGen and performs comparably to OSSL, despite having up to 18% fewer parameters. These results suggest that frozen Transformer encoders pre-trained on real-world data can generalize to out-of-distribution domains such as retro video games. Our code and dataset (video URLs and clean soundtrack mappings) are available¹.

2. RELATED WORK

Video-to-Music Models. V2M generation conditions music on a sequence of video frames. Given its proximity to text-to-music (T2M) generation, V2M models largely build on

¹<https://felipemarra.github.io/demo-v2m-4-gameplay-videos-v1/>

pre-trained T2M models, which generate music from textual descriptions. MusicGen [6] is one of the most widely T2M models adopted as a base for V2M generation. It leverages an neural audio codec [11] to compress waveforms into compact discrete tokens, and predicts interleaved codebook streams in parallel, conditioned on T5 [12] embeddings. Given this relation to T2M generation, we categorize V2M models by how they use text to bridge visual and musical modalities: *text as interface*, *text embedding as interface*, and *direct mapping*.

Text as Interface. These approaches extract visual features, convert them into a musical description, often via an LLM, and pass that description to a T2M generator. Because these components can remain frozen, this paradigm can use unpaired video and music data. Herrmann-1 [7] tackles the film domain by transforming scene descriptions, detected emotions, and speech into music prompts for MusicGen. CoT-VTM [13] is similar, but distills the pipeline into a single model that outputs directly into the T5 embedding space. SONIQUE [14] extracts visual tags to condition a pre-trained Stable Audio model.

Text Embedding as Interface. These methods map visual features into the text embedding space of a music decoder via an adapter, offering greater representational flexibility than discrete text. OSSSL [1] uses a LoRA adapter to bridge a frozen ViViT [15] video encoder with the T5 text encoder of MusicGen. MuMu-LLaMA [3] employs modality-specific adapters to build a multimodal LLM whose outputs are mapped to T5 tokens for synthesis. VidMusician [16] extends this paradigm with time alignment, using CLIP-based [CLS] tokens for semantic conditioning and high-frequency patch-level tokens for rhythmic synchronization.

Direct Mapping. Bypassing text entirely, these models map visual features directly to the music decoder. This typically requires training the decoder, as visual features lie outside the text encoder’s distribution, and suits tasks that require temporal alignment. GVMGen [8] uses a frozen ViT [17] with a trainable Transformer adapter, arguing that spatiotemporal fusion, as in ViViT, degrades frame-level alignment. VidMuse [2] employs an LSTM adapter with a sliding window to provide both global guidance and local adaptation. V2Meow [5] extracts features via 3D-CNN, CLIP, and ViT-VQGAN, training a MusicLM-based decoder on a large-scale music video dataset filtered from YouTube-8M [18].

Video Game Music. Music generation for video games has mostly been studied in the symbolic domain. NES-MDB [19] provides multi-instrumental NES soundtracks with expressive performance attributes, and LakhNES [20] generates NES-style music with a Transformer pre-trained on the Lakh MIDI dataset and fine-tuned on NES-MDB, without conditioning on gameplay. Closest to our work, NES-MVDB [21] pairs NES gameplay videos with symbolic music, using rule-based video features for MIDI generation. In contrast, we generate audio and target the SNES, whose 8-channel sample-based audio is considerably richer than

the NES’s 5-channel synthesized sound and whose library contains more games.

3. SNES VIDEO-TO-MUSIC DATASET

Collection & Pre-Processing. We started by scraping the original SNES soundtracks from Zophar’s Domain,² an online repository of crowd-sourced video game assets. Following [21], we filtered out all audio files shorter than 8 seconds and, for each of the 1,604 resulting games, queried YouTube for a gameplay video using “{*GAME_NAME*} *Nintendo SNES Longplay*”, where {*GAME_NAME*} is the game’s title on Zophar’s Domain, keeping the top result. We partitioned the videos into 10-second clips to match the ViViT encoder’s input while minimizing the likelihood of capturing track transitions within a single clip.

Noisy to Clean Soundtracks. To map noisy gameplay audio to the clean original soundtracks from Zophar’s Domain, we built game-specific fingerprint databases using Dejavu [22], which ranks candidate tracks by a confidence score, defined as the fraction of successful fingerprint matches. We queried each clip’s audio against its game’s database and assigned it the top-ranked clean track if its confidence score was above 0.01, a value chosen empirically. To evaluate the quality of this mapping, we manually annotated 10 hours of video, balanced by genre (approximately 0.9 hours per genre), with the start and end timestamps of each music track. Compared against these annotations, the assigned tracks achieved an average accuracy of 77.36%.

Downsampling. To prevent genre dominance, we mapped titles to an 11-genre taxonomy [21]: Action, Adventure, Fighting, Platform, Puzzle, RPG, Racing, Shooter, Simulation, Sports, and Strategy. We identified each title’s genre by querying its name on Wikidata and using an LLM (DeepSeek R1-70B) to map the retrieved metadata to the most appropriate genre. We then downsampled clips from majority genres to balance them with the minority genres, while preserving at least one clip of every unique background track to ensure diversity. The final dataset comprises 78,344 clips, totaling 217.6 hours of video, paired with 16,808 unique background tracks, totaling 485 hours of clean audio.

4. VIDEO ENCODERS

With our SNES dataset, we compare three of the most common encoders in autoregressive V2M approaches: T5 [12], ViT [17], and ViViT [15]. We build three encoder-decoder models, one per encoder, while keeping the same MusicGen decoder in all of them. In each model, the encoder output is linearly projected to the decoder’s dimension and passed to the decoder via cross-attention.

²<https://www.zophar.net/music/nintendo-snes-spc>

T5. This model follows a text-as-interface approach, where visual information is translated into text before generation. We use VideoLLaMA 3 [23] to generate detailed captions for each 10-second clip, with a prompt that asks the model to describe interface options and backgrounds for menu scenes, and actions, environments, movement speed, and mechanics for gameplay. A T5 encoder then maps each caption to a sequence of contextual embeddings $E = (e_1, \dots, e_k)$, where k is the number of T5 tokens. Notably, we omit the intermediate step of using an LLM to “translate” scene descriptions into musical prompts, forcing T5 to represent the raw visual semantics and allowing a direct comparison with the purely visual encoders.

ViT. The remaining two models follow a direct mapping approach. The ViT-based model represents a video as a temporal sequence of independent frames. From each clip, we sample 32 evenly spaced frames $F = (f_1, \dots, f_{32})$, resize the short size to 224 and center crop to 224×224 , and encode each one with ViT-B/16, producing 196 patch embeddings and one prepended [CLS] token. Since the [CLS] token summarizes the frame, we keep only the sequence of [CLS] vectors $C = (c_1, \dots, c_{32})$, yielding a compact frame-level temporal representation.

ViViT. The ViViT-based model uses the same 32 frames, but projects them into non-overlapping $2 \times 16 \times 16$ tubelets spanning the spatiotemporal volume, capturing spatial and temporal information jointly in a single projection. This produces a sequence of spatiotemporal embeddings $V = (v_1, \dots, v_n)$, with $n = 3,136$, where each token represents a local region of the gameplay. Unlike in ViT, we omit the global [CLS] token to preserve local spatiotemporal detail rather than a single clip-level summary.

5. EXPERIMENTS AND RESULTS

Experimental Setup. We initialized each encoder with pre-trained weights: ViViT-B [15] on Kinetics-400 action classification, ViT-B [17] on ImageNet classification, and T5-base [12] on the Colossal Clean Crawled Corpus. Each encoder was trained in two regimes: frozen and fully fine-tuned on our dataset. The decoder is initialized with pre-trained MusicGen medium weights and is always fine-tuned. The projection layer is always randomly initialized. We split the dataset by genre into 50% for training, 40% for validation, and 10% for testing; the large validation split is used to fine-tune the ImageBind [10] model used in our objective metrics. Following [6], we trained all models for 75 epochs of 2,000 updates each, with a batch size of 6 as in GVMGen [8], using AdamW with a learning rate of 10^{-4} and a cosine scheduler. Training used Distributed Data Parallel across three NVIDIA A100 GPUs (80 GB). At inference, we generate 30-second audio at 32 kHz using top- k sampling with $k = 250$ and a classifier-free guidance scale of 3.

Objective Metrics. Following standard V2M prac-

	KLD _{PaSST} ↓	FAD _{VGGish} ↓	IB _{tuned} ↑
 MGen	0.85 ± 0.64	13.90 ± 4.09	0.45 ± 0.02
 T5	0.80 ± 0.66	11.32 ± 4.06	0.47 ± 0.02
 T5	0.71 ± 0.68	11.07 ± 4.27	0.47 ± 0.02
 ViT	0.93 ± 0.98	10.98 ± 3.95	0.49 ± 0.05
 ViT	0.73 ± 0.63	10.57 ± 3.97	0.49 ± 0.02
 ViViT	0.76 ± 0.55	11.57 ± 4.09	0.47 ± 0.03
 ViViT	0.72 ± 0.60	10.53 ± 3.60	0.49 ± 0.02

Table 1. Encoders comparison. Metrics are averaged across genres.  denotes a fine-tuned encoder while  denotes a frozen pre-trained one.  MGen is MusicGen Base.

tice [8, 1, 2], we report FAD_{VGGish} [9, 24] for audio quality, KLD_{PaSST} [25] for semantic similarity to the reference, and the ImageBind [10] (IB) score for video-music alignment. To compute these metrics, each model generates one music piece for every clip in the test split, then scores are calculated and averaged per genre. FAD and KLD compare each generated piece with the clip’s mapped ground-truth soundtrack (Section 3); IB is the cosine similarity between the embeddings of each clip and its generated piece. Since IB was not trained on SNES music, we fine-tuned its audio and video encoders contrastively on our validation split, following a linear-probing-then-fine-tuning strategy [26]: 5 epochs tuning only the last layer, followed by 5 epochs of LoRA with the last layer frozen. We denote this tuned version as IB_{tuned}. The test split is never used for tuning.

Encoders Comparison. Table 1 reports the mean and standard deviation across genres of FAD_{VGGish}, KLD_{PaSST}, and IB_{tuned} for the T5, ViT, and ViViT encoders, with fine-tuned ones denoted by  and frozen ones by . We compare all model variants against the original MusicGen model ( MGen) [6], which is not fine-tuned on our dataset and is conditioned on the same captions as our T5 model. All  variants outperform it on every metric, confirming the benefit of training the MusicGen decoder on our dataset. For every metric,  models perform better than or equal to their  counterparts. We hypothesize that tuning only the decoder lets the cross-attention adapt to stable pre-trained features, whereas tuning both at once distorts these features [26]. KLD differences among  models are subtle, with T5, ViViT, and ViT scoring 0.71 ± 0.68 , 0.72 ± 0.60 , and 0.73 ± 0.63 , respectively. IB scores are also similar across models, with  ViT and ViViT tied for the highest mean and lowest variance (0.49 ± 0.02), suggesting that all encoders extract useful semantic information for music generation. In FAD,  ViViT achieves the lowest mean and variance (10.53 ± 3.60), making it the only model ranked in the top two on all metrics.

SOTA Comparison. We compare  ViViT, our best model, against one SOTA approach per category in Section 2: Herrmann-1 [7] for text as interface, OSSSL [1] for text embedding as interface, and GVMGen [8] for direct mapping.

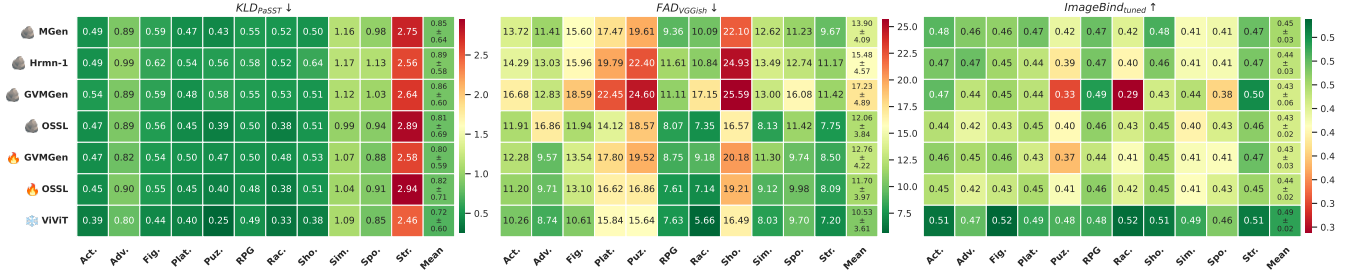


Fig. 1. Objective performance comparison of the best proposed architecture against baseline models.

For OSSL and GVMGen, we consider both their base (🐼) versions, with their original pre-trained weights, and fine-tuned (🔥) versions, whose adapters are trained from scratch on our dataset with the same hyperparameters as our models. For fine-tuned OSSL, all weights are frozen except the LoRA adapter [1]; for fine-tuned GVMGen, the ViT encoder is frozen, while the Feature Transformation module and MusicGen decoder are trained [8]. Herrmann-1 uses only frozen models, with descriptions generated as in [7], except that we omit the speech module and replace GPT-4 with Qwen2.5-7B-Instruct. We also report MusicGen Base, which, together with OSSL, uses the same captions as our T5 model.

Fig. 1 shows one heatmap per objective metric, where rows correspond to models, columns to genres, and the last column to the mean and standard deviation across genres; greener cells indicate better scores. 🌸 ViViT achieves the best mean on all metrics while using fewer parameters (approximately 1.93B, compared with 2.35B for OSSL and 2.22B for GVMGen), and it ranks first or tied in at least 9 of the 11 genres on every metric. Genre difficulty is consistent across models: Strategy yields the highest KLD, likely because its top-down map views are more “open to interpretation,” while Platform, Puzzle, and Shooter yield the highest FAD.

Two comparisons isolate the effect of our design. First, OSSL uses the same frozen ViViT encoder but maps it into the T5 embedding space of a frozen decoder; our direct mapping with a trained decoder outperforms it on all metrics. Second, GVMGen, like our 🌸 ViT model, pairs a frozen ViT with a trained decoder, but adds a Feature Transformation module; our simpler model again performs better. Moreover, fine-tuning substantially improves GVMGen, whose decoder is trained, but barely changes OSSL, whose decoder stays frozen, reinforcing that adapting the decoder is the key step. Finally, base GVMGen, trained on real-world videos, yields the worst FAD, even below MusicGen Base, highlighting the domain gap addressed by our dataset, while Herrmann-1, which translates descriptions into musical prompts with an LLM, underperforms MusicGen Base conditioned on raw captions, supporting our choice to omit this step.

Listening Test. Since 🌸 ViViT outperformed the other encoders on the objective metrics, we compared it against the

	🔥 OSSL	🔥 GVMGen	🌸 ViViT
Q1 ↑	2.82 ± 1.14^a	2.53 ± 1.28^b	2.87 ± 1.21^a
Q2 ↑	2.99 ± 1.15^a	2.76 ± 1.23^b	3.06 ± 1.17^a
Q3 ↑	3.38 ± 1.09	3.22 ± 1.16	3.26 ± 1.11
Q4 ↑	3.05 ± 1.14	2.88 ± 1.22	2.84 ± 1.23
Q5 ↑	3.29 ± 1.13^{ab}	3.06 ± 1.25^a	3.40 ± 1.15^b

Table 2. Average scores (in 1–5) for each metric in our listening test. Values with shared letter superscripts^{ab} in a given row have p-value > 0.05 in pairwise post-hoc comparisons.

fine-tuned OSSL and GVMGen in an online listening test, following the within-subjects methodology used to evaluate OSSL [1]. Each participant was randomly assigned two gameplay clips from a pool of 122 test clips. For each clip, participants first watched the original gameplay to become familiar with its visual dynamics and background music. They then watched the same clip three more times, each with background music generated by a different model in randomized order, and rated each generated soundtrack on five 5-point Likert scales: Q1) Video-Music Alignment, Q2) Game Genre Alignment, Q3) Audio Quality, Q4) Compositional Professionalism, and Q5) SNES Aesthetic Alignment. This procedure was then repeated for the second clip. We recruited participants via university mailing lists and public gaming-related Discord channels, allowing further snowball sampling. In total, $N = 96$ participants completed the test.

Table 2 shows the average score of each method for each question. For each question, we ran a Kruskal-Wallis test across methods and, if significant, pairwise Wilcoxon rank-sum tests with Holm correction. The methods differ significantly ($p \leq 0.05$) only on Q1, Q2, and Q5. On video-music alignment (Q1) and game genre alignment (Q2), ViViT and OSSL perform comparably, and both significantly outperform GVMGen. On SNES aesthetic alignment (Q5), ViViT achieves the highest score, significantly outperforming GVMGen. Given that ViViT is nearly 18% smaller than OSSL, these results suggest that frozen spatiotemporal representations are an effective choice for retro-game music generation.

6. CONCLUSION

We introduced an SNES video-to-music dataset and used it to compare visual encoding strategies for autoregressive video-to-music generation. Among T5, ViT, and ViViT encoders, ViViT achieved the best overall results, and frozen encoders matched or outperformed fine-tuned ones, suggesting that real-world pre-trained features transfer to rendered game graphics when only the decoder is adapted. Our simple direct-mapping model also achieved the best objective results against SOTA baselines and performed comparably to OSSSL in a listening study, with up to 18% fewer parameters.

7. ACKNOWLEDGMENTS

We would like to thank the Universidade Federal de Viçosa (UFV) for access to the DGX GPU Cluster. In particular, we are grateful for the long-term technical support given by Eliab Venancio, head of the Scientific Development Support Service at the Diretoria de Tecnologia de Informação (DTI).

8. REFERENCES

- [1] H. Kim, Z. Novack, W. Xu, J. McAuley, and H. Dong, “Video-guided text-to-music generation using public domain movie collections,” *ISMIR*, 2025.
- [2] Z. Tian, Z. Liu, R. Yuan, J. Pan, X. Huang, Q. Liu, et al., “Vidmuse: A simple video-to-music generation framework with long-short-term modeling,” *CVPR*, 2025.
- [3] S. Liu, Q. Wu, C. Sun, A. S. Hussain, and Y. Shan, “Mumu-llama: Multi-modal music understanding and generation via large language models,” *Expert Syst. Appl.*, 2025.
- [4] L. Zhuo, Z. Wang, B. Wang, Y. Liao, C. Bao, S. Peng, et al., “Video background music generation: Dataset, method and evaluation,” in *ICCV*, 2023.
- [5] K. Su, J. Y. Li, Q. Huang, D. Kuzmin, J. Lee, C. Donahue, et al., “V2meow: Meowing to the visual beat via video-to-music generation,” in *AAAI*, 2024.
- [6] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, et al., “Simple and controllable music generation,” *NeurIPS*, 2023.
- [7] M. T. Haseeb, A. Hammoudeh, and Gus X., “Gpt-4 driven cinematic music generation through text processing,” in *ICASSP. IEEE*, 2024.
- [8] H. Zuo, W. You, J. Wu, S. Ren, P. Chen, M. Zhou, et al., “Gvmgen: A general video-to-music generation model with hierarchical attentions,” in *AAAI*, 2025.
- [9] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet audio distance: A metric for evaluating music enhancement algorithms,” *arXiv:1812.08466*, 2018.
- [10] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. Alwala, A. Joulin, and I. Misra, “Imagebind: One embedding space to bind them all,” in *CVPR*, 2023.
- [11] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, “High fidelity neural audio compression,” *arXiv:2210.13438*, 2022.
- [12] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, et al., “Exploring the limits of transfer learning with a unified text-to-text transformer,” *JMLR*, 2020.
- [13] X. Guan, Z. Gu, J. Huo, T. Ding, and Y. Gao, “CoT-VTM: Visual-to-music generation with chain-of-thought reasoning,” 2025, *ACL*.
- [14] L. Zhang and M. Fuentes, “Sonique: Video background music generation using unpaired audio-visual data,” in *ICASSP. IEEE*, 2025.
- [15] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, “Vivit: A video vision transformer,” in *ICCV*, 2021.
- [16] S. Li, B. Yang, C. Yin, C. Sun, Y. Zhang, W. Dong, et al., “Vidmusician: Video-to-music generation with semantic-rhythmic alignment via hierarchical visual features,” *arXiv:2412.06296*, 2024.
- [17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” in *ICLR*, 2021.
- [18] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, et al., “Youtube8m: A large-scale video classification benchmark,” *arXiv:1609.08675*, 2016.
- [19] C. Donahue, H. Mao, and J. McAuley, “The NES music database: A multi-instrumental dataset with expressive performance attributes,” in *ISMIR*, 2018.
- [20] C. Donahue, H. Mao, Y. Li, G. Cottrell, and J. McAuley, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training,” in *ISMIR*, 2019.
- [21] I. Cardoso, R. Moraes, and L. Ferreira, “The nes video-music database: A dataset of symbolic video game music paired with gameplay videos,” in *FDG*, 2024.
- [22] W. Drevo, “Dejavu: Audio fingerprinting and recognition in python,” June 2020.

- [23] Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, et al., “Videollama 3: Frontier multimodal foundation models for image and video understanding,” *arXiv:2501.13106*, 2025.
- [24] S. Hershey, S. Chaudhuri, D. Ellis, J. Gemmeke, A. Jansen, R. Moore, et al., “Cnn architectures for large-scale audio classification,” in *ICASSP*. IEEE, 2017.
- [25] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer, “Efficient training of audio transformers with patchout,” in *Interspeech*, 2022.
- [26] A. Kumar, A. Raghunathan, R. Jones, T. Ma, and P. Liang, “Fine-tuning can distort pretrained features and underperform out-of-distribution,” in *ICLR*, 2022.