DJNet: Diffusion Models for Generating Seamless Transitions

August 14, 2025

Soykat Amin

Abstract

Creating seamless DJ transitions requires aligning musical compatibility, rhythmic coherence, and timbral detail. I present DJNet, the first diffusion-based model for generating DJ transitions directly from raw audio. The system operates on mel-spectrograms of two source tracks to synthesize a novel transition segment. To address the absence of annotated transition data, I constructed a synthetic dataset of 5,000 examples across multiple mixing styles. The model, a U-Net–based architecture, was trained using gradient accumulation and optimized for spectrogram reconstruction. A subjective listening test against baselines (DJTransGAN, crossfade, hard cut) showed that while the model produces coherent transitions, limited training data led to noticeable noise, highlighting the need for larger and more diverse datasets to improve quality.

1. Introduction

Creating seamless DJ transitions is a complex task that combines musical compatibility, rhythmic alignment, and creative mixing. While prior work has explored recommendation-based track selection, rule-driven mixing, and generative approaches using GANs or Transformers, there remains a gap in methods that can directly generate realistic audio transitions while capturing both structural and timbral nuances.

In this work, I present DJNet, the first diffusion-based model for generating DJ transitions from raw audio. My approach operates on mel-spectrogram representations of two source tracks and synthesizes a novel transition segment. To address the lack of annotated data, I constructed a synthetic dataset of 5,000 transitions spanning multiple mixing styles, enabling supervised training.

Email: Soykat Amin <amin.1985500@studenti.uniroma1.it>.

Deep Learning and Applied AI 2025, Sapienza University of Rome, 2nd semester a.y. 2024/2025.

Code. The project source code is available at: https://github.com/SoykatAmin/DJNet-StableDiffusion

2. Related work

Selecting the next track in DJing is primarily a recommendation problem, where the goal is to find songs that are musically compatible. (Mowery et al., 2025) address this by vectorizing playlists into a 48-dimensional feature space (tempo, timbre, key, energy) and framing selection as an optimization problem that minimizes a custom "song norm." This yields coherent track suggestions but does not produce the transition audio itself.

Other systems automate the entire mixing workflow through Music Information Retrieval (MIR) and handcrafted rules. For example, (Vande Veire & De Bie, 2018) developed a fully automated Drum and Bass DJ system, combining beat and key detection, style-based track selection, and rule-driven crossfading. While robust, such approaches are limited to predefined mixing logic and traditional transitions.

Recent work treats transitions as a generative task. A significant advancement in audio-based generation is demonstrated by (Chen et al., 2022) Their method uses a Generative Adversarial Network (GAN) to learn from a collection of real-world DJ mixes without needing perfectly aligned, paired data. The generator's core is a novel, differentiable DJ mixer composed of an equalizer (EQ) and fader layers. Instead of arbitrarily generating spectrogram pixels, the generator learns to set the parameters of these audio effects to produce a mix that the discriminator cannot distinguish from a real human mix.

Symbolic approaches, such as (Hsu & Chang, 2021), apply Transformers to generate MIDI-based transitions using a dual-encoder architecture. These produce musically structured results but cannot capture the timbral detail of audio.

3. Method

3.1. Dataset

To train my model, I generated a synthetic dataset of 5,000 transitions using FMA Small (Defferrard et al., 2017). For each track, I extracted the following features:

- **Tempo Detection**: BPM (beats per minute) analysis.
- **Beat Tracking**: Precise beat and downbeat timing.
- **Key Detection**: Musical key identification for harmonic compatibility.
- **Audio Processing**: Standardized sample rate conversion (22kHz) and mono conversion.

I then determined track compatibility based on two rules:

- **Tempo compatibility**: The tempo difference is within a threshold of 10.0 BPM.
- **Key compatibility**: Keys are represented as integers from 0 to 11. Two keys are compatible if their difference is 0, 1, or 11 (wrap-around).

3.2. Model Architecture

The model operates on mel-spectrograms with dimensions 128×512 (frequency×time), corresponding to approximately 12 seconds of audio at a 22.05 kHz sampling rate.

Core Architecture. The U-Net consists of a 4-layer encoder–decoder structure with progressive channel expansion (64→128→256→512) and symmetric skip connections. The encoder performs spatial downsampling through strided convolutions, reducing the input from 128×512 to 8×32 at the bottleneck, while preserving critical frequency–time relationships through batch normalization and ReLU activations.

Input Processing. The model accepts 3-channel input tensors representing: (1) source track A spectrogram, (2) source track B spectrogram, and (3) structured noise for transition variability. Each channel undergoes mel-spectrogram transformation using a 2048-point FFT with a 512-sample hop length, followed by logarithmic scaling and normalization to [-1,1] range.

Decoder Architecture. The decoder employs transposed convolutions for upsampling, with skip connections concatenating corresponding encoder features to preserve spatial detail. Each decoder block includes convolutional layers, batch normalization, and dropout (0.1–0.2) for regularization. The final layer outputs a single-channel transition spectrogram.

Loss Function. I optimized the system using MSE loss between generated and target spectrograms, with the AdamW optimizer (1e-4 learning rate) and cosine annealing scheduling. Gradient accumulation enabled effective large-batch training on limited hardware, while gradient clipping (max norm 1.0) ensured training stability.

3.3. Training and Inference

The model was trained on the 5,000 synthetic DJ transitions using gradient accumulation with an effective batch size of 16 over 30 epochs. I implemented automatic checkpointing with best model selection based on validation loss. Mixed-precision training and memory optimization techniques allowed efficient training on consumer GPUs, with TensorBoard logging providing real-time monitoring of loss curves, learning rates, and model convergence metrics.

During inference, the system loads pre-trained checkpoints and processes 12-second audio segments through the optimized AudioProcessor pipeline. Input audio files are transformed into mel-spectrograms with precise cropping to maintain the target 128×512 dimensionality, followed by logarithmic scaling and normalization. The ProductionUNet model performs single forward passes in evaluation mode with gradient computation disabled for efficiency. Reconstruction is performed using the Griffin–Lim algorithm (Griffin & Lim, 1984) with 32 iterations for high-quality spectrogram-to-audio conversion.

4. Experimantal results

Since DJ transitions can be considered a form of art, objective evaluation alone may not accurately reflect the quality of the output. For this reason, I asked human listeners to judge the best transition. I compared my model against the following baselines:

- DJTransGAN (Chen et al., 2022)
- Simple crossfade
- Hard cut

The survey involved ten participants. Each participant was asked to rank the transitions in order of personal preference. The songs and cue points were carefully selected by me. I included five groups of transitions. Unlike the approach in (Chen et al., 2022), where users were simply asked to choose the one they liked most, my ranking system allows not only the identification of the most preferred method but also a comparison of the others.

To embed the results, I assigned 0.7 points to the first-ranked track, 0.2 points to the second, and 0.1 points to the

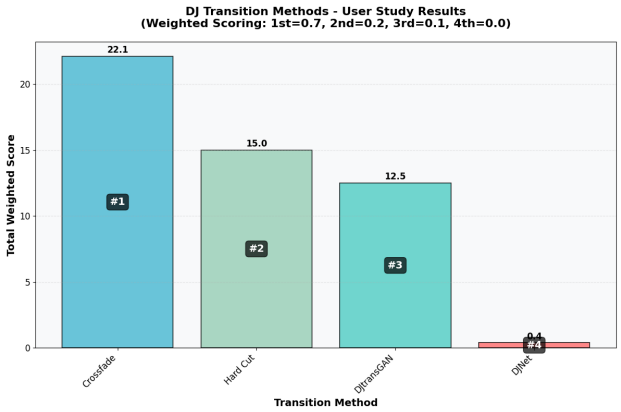

Figure 1. Results of the survey.

third. The last-ranked track received 0 points. The tracks were randomized so that the participants were not aware of the method used.

Figure 1 shows the results. A simple crossfade emerged as the preferred method. Surprisingly, the hard cut method ranked second, likely because the cue points were carefully selected rather than randomized. My model did not perform well—some participants reported ranking the tracks produced by DJNet last because they sounded noisy.

Appendix

Motivating the idea Transformers (Copet et al., 2023; Borsos et al., 2023) and diffusion models (Evans et al., 2024; Ning et al., 2025) currently represent the state of the art in music generation. Some recent works combine the two. For example, ACE-Step (Gong et al., 2025) employs a transformer for token prediction and a diffusion model for decoding, while AudioX (Tian et al., 2025) integrates a multi-modal transformer with diffusion for the final audio output. Given that transformers have already been extensively explored in previous research, I decided to focus on diffusion models. My architecture was inspired by Riffusion (Forsgren & Martiros, 2022), a text-to-music model based on Stable Diffusion v1.5 fine-tuned on a music dataset represented as mel-spectrogram images. I chose not to use the pre-trained Riffusion model due to several challenges: adapting its architecture to our task, mitigating potential overfitting, and managing the inference latency caused by the multiple denoising steps required.

Other attempts At first, I tried a Transformer-based approach, replicating the architecture from (Hsu & Chang, 2021). My idea was to train on a large dataset to generate mel-spectrograms instead of REMI notes. However, since this approach was not particularly innovative, I decided to abandon it.

I then attempted a more ambitious diffusion model with explicit conditioning, where the conditioning included tempo embeddings, transition type embeddings, and transition length embeddings. Unfortunately, the results were disappointing—it produced noise rather than music. At that point, I was ready to give up on the project entirely.

Training I used Kaggle for training since I do not have an NVIDIA GPU, and it offers 30 hours of GPU time per week. Initially, I generated a dataset of 20,000 transitions, each with segments lasting 30 seconds, which resulted in about 38 GB of data. However, I faced several issues: my upload speed is very slow, my internet connection is limited, and Kaggle GPUs have 16×2 GB of VRAM. To address this, I cropped the segments to 15 seconds and reduced the dataset to 5,000 transitions. I trained the model for 30 epochs, but after epoch 6 it began to overfit. Therefore, the checkpoint I uploaded corresponds to epoch 6.

Dataset Anyone attempting this project must deal with the lack of an annotated dataset of DJ transitions. To address this issue, I decided to create a synthetic dataset using some simple, well-known techniques. I initially considered using Medley2k since, as far as I know, it is the only annotated dataset containing real DJ transitions. However, the download link they provided is expired, and my attempt to contact the authors received no response.

I implemented the following transition types:

- **Linear Fade**: Smooth crossfade between tracks (30%).
- **Exponential Fade**: More natural-sounding fade curves (25%).
- **Bass Swap with EQ**: Frequency-selective transitions (25%).
- **Filter Sweep**: Gradual frequency filtering effects (10%).
- **Hard Cut**: Abrupt transitions for specific musical contexts (5%).
- **Echo Fade**: Reverb-enhanced transitions (5%).

These transitions range from 0.1 to 8 seconds in duration.

Each generated transition sample includes:

- **Source A/B**: 20-second segments from each track
- **Target**: Generated transition with natural duration (0.8–8 seconds)

- **Metadata**: Complete conditioning information in JSON format
- **CSV Index**: File mapping all samples to their metadata

You can find the code here: https://github.com/SoykatAmin/DJNet-Dataset

Evaluation The Fréchet Audio Distance (FAD) (Kilgour et al., 2018) evaluation was conducted to assess the perceptual quality of the generated DJ transitions compared to baseline methods. FAD provides an objective metric for measuring the similarity between distributions of audio features, making it particularly suitable for evaluating generative audio models. Lower FAD values indicate that the feature distributions are more similar.

FAD evaluation uses VGGish, a pre-trained audio classification model developed by Google, to extract high-level audio features. The baselines for comparison were the synthetic dataset and a simple crossfade. I used 300 transitions for reliable statistical analysis.

Table 1. FAD Score Comparison

Comparison	FAD Score
DJNet vs Dataset	15978.262
Crossfade vs Dataset	307.186
DJNet vs Crossfade	13365.340

I then performed a rhythmic consistency evaluation to assess the musical coherence and tempo stability of the generated DJ transitions. Unlike perceptual metrics such as FAD, rhythmic analysis focuses specifically on beat-level timing, tempo variations, and cross-segment rhythmic alignment. The baseline was the synthetic dataset.

Four primary metrics quantify rhythmic consistency:

- **Tempo Stability**

  $$\text{Tempo\_Stability} = max(0.0, min(1.0, 1.0 - \frac{\text{tempo\_std}}{10.0}))$$

  Where $tempo\_std$ is the standard deviation of the tempos.

- **Cross-Segment Consistency**

  $$\text{Cross\_Consistency} = 1.0 - \frac{\text{max\_tempo\_diff}}{20.0}$$

  where max_tempo_diff is the largest tempo difference between any two segments, normalized by 20 BPM.

- **Beat Alignment Quality**

  $$\text{Beat\_Alignment} = 1.0 - \frac{\text{mean\_phase\_deviation}}{\text{expected\_interval}}$$

- **Beat Strength Consistency**

  $$\text{Beat\_Strength} = 1.0 - \frac{\sigma_{\text{strength}}}{\text{mean\_strength}}$$

  Where $\sigma_{strength}$ and $mean\_strength$ represent beat onset strength statistics.

The overall score was calculated as:

$$Overall\_Score = 0.4 \cdot Tempo\_Stability + 0.3 \cdot Cross\_Consistency + 0.2 \cdot Beat\_Alignment + 0.1 \cdot Beat\_Strength$$

The weighting rationale was:

- **Tempo Stability** (40%): Most critical for maintaining dancefloor energy
- **Cross-Consistency** (30%): Essential for smooth transitions between tracks
- **Beat Alignment** (20%): Important for avoiding rhythmic artifacts
- **Beat Strength** (10%): Secondary metric for energy consistency

For the subjective evaluation, I decided not to include the work from (Hsu & Chang, 2021), since MIDI-based approaches cannot reproduce real audio characteristics such as vocal timbre, reverberation, or audio artifacts present in actual DJ transitions.

Web App To try the model, I included a web application in the `/app` folder of the repository. It is a simple Flask-based app that allows you to upload two songs, select the cue points, and generate transitions. For comparison, you can choose between my model, a simple crossfade, or a hard cut. You can also listen to the samples of some transitions that I uploaded.

Considerations The model produces transitions with noticeable noise, likely due to the limited size of the training dataset—5,000 transitions are probably insufficient for the model to learn to generate smooth musical segments. This

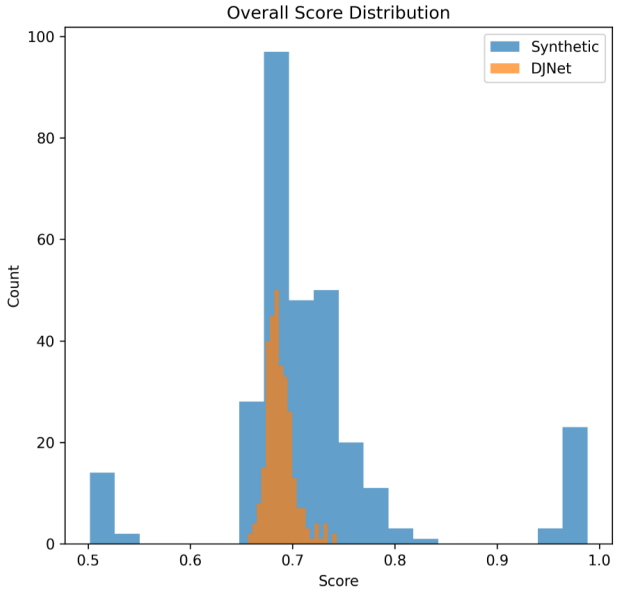

Figure 2. This chart represents the overall score for the rhythmic consistency.

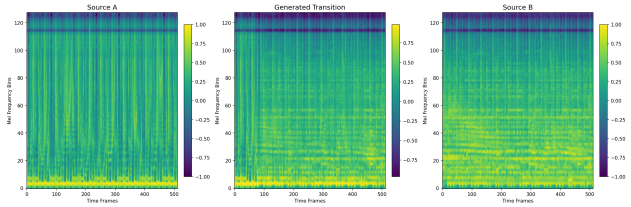

Figure 3. Mel spectrogram transition generated using the best checkpoint.

limitation is also reflected in the subjective evaluation results, where participants ranked my model as the lowest-performing option. You can see the comparison of mel-spectrograms in 3 and 4. The spectrogram in 4 appears less detailed, which explains the noisy transition. Training the model with a larger dataset will hopefully improve the quality.

References

Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM transactions on audio, speech, and language processing*, 31:2523–2533, 2023.

Chen, B.-Y., Hsu, W.-H., Liao, W.-H., Ramírez, M. A. M., Mitsufuji, Y., and Yang, Y.-H. Automatic dj transitions with differentiable audio effects and generative adversarial networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 466–470. IEEE, 2022.

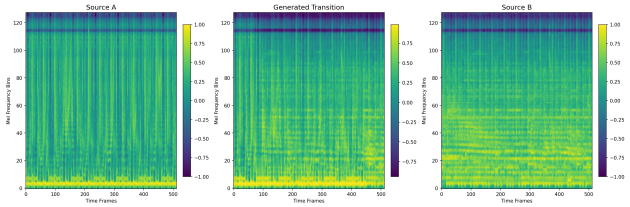

Figure 4. Mel spectrogram transition generated using the model trained with only 200 transitions.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. Simple and controllable music generation. *Advances in Neural Information Processing Systems*, 36:47704–47720, 2023.

Defferrard, M., Benzi, K., Vandergheynst, P., and Bresson, X. FMA: A dataset for music analysis. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017. URL https://arxiv.org/abs/1612.01840.

Evans, Z., Parker, J. D., Carr, C., Zukowski, Z., Taylor, J., and Pons, J. Long-form music generation with latent diffusion. *arXiv preprint arXiv:2404.10301*, 2024.

Forsgren, S. and Martiros, H. Riffusion - Stable diffusion for real-time music generation. 2022. URL https://riffusion.com/about.

Gong, J., Zhao, S., Wang, S., Xu, S., and Guo, J. Ace-step: A step towards music generation foundation model. *arXiv preprint arXiv:2506.00045*, 2025.

Griffin, D. and Lim, J. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984. doi: 10.1109/TASSP.1984.1164317.

Hsu, J.-L. and Chang, S.-J. Generating music transition by using a transformer-based model. *Electronics*, 10(18), 2021. ISSN 2079-9292. doi: 10.3390/electronics10182276. URL https://www.mdpi.com/2079-9292/10/18/2276.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. Fr\'echet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

Mowery, W., Zhang, K., and Zhang, M. A mathematical approach to dj transitions. 2025.

Ning, Z., Chen, H., Jiang, Y., Hao, C., Ma, G., Wang, S., Yao, J., and Xie, L. Diffrhythm: Blazingly fast and embarrassingly simple end-to-end full-length song generation with latent diffusion. *arXiv preprint arXiv:2503.01183*, 2025.

Tian, Z., Jin, Y., Liu, Z., Yuan, R., Tan, X., Chen, Q., Xue, W., and Guo, Y. Audiox: Diffusion transformer for anything-to-audio generation. *arXiv preprint arXiv:2503.10522*, 2025.

Vande Veire, L. and De Bie, T. From raw audio to a seamless mix: creating an automated dj system for drum and bass. *EURASIP Journal on Audio, Speech, and Music Processing*, (13):1–21, 2018.