

Generating Rock Music from Spectrogram Images Using a Fine-Tuned Stable Diffusion Pipeline

Bekzat Ongdassynov, Temirlan Tursynbekov, Sultan Kasenov
Nazarbayev University, Astana, Kazakhstan

Authors' email: bekzat.ongdassynov@nu.edu.kz, temirlan.tursynbekov@nu.edu.kz, sultan.kasenov@nu.edu.kz

Abstract—This project investigates an image based approach for generating short rock music audio samples. Instead of training a model directly on waveform audio, the proposed pipeline converts audio clips into mel spectrogram images and uses a fine-tuned Stable Diffusion model to generate new spectrograms. These generated spectrogram images are then converted back into audio using inverse mel reconstruction and Griffin-Lim phase estimation. The system uses the GTZAN music genre dataset as the source dataset, extracts the rock genre, resamples all audio files to 16 kHz, chops them into 4-second segments, and converts each segment into a 512 by 512 RGB spectrogram image. These image-text pairs are then used to fine-tune a generative image model using a custom identifier token. During generation, the trained model produces new spectrogram-like images from text prompts. The final stage decodes the generated images into audio, merges several generated clips and applies simple post-processing such as crossfading and clipping distortion.

The project shows that spectrogram-based music generation can produce recognizable audio textures from an image generation model. However, the output quality is strongly limited by the loss of phase information, the approximate nature of Griffin-Lim reconstruction, and the fact that the model learns visual spectrogram structure instead of waveform-level sound properties. The result is an experimental pipeline that connects audio processing, image generation, and audio reconstruction in one workflow.

Index Terms—audio generation, spectrogram, Stable Diffusion, DreamBooth, LoRA, Griffin-Lim, GTZAN, rock music, generative AI

I. INTRODUCTION

Recent generative models have shown strong results in image synthesis, text-to-image generation, and style adaptation. Stable Diffusion is one of the most widely used latent diffusion models for generating images from text prompts [4]. At the same time, many audio processing tasks can represent sound as time-frequency images, such as spectrograms or mel spectrograms. This creates a simple bridge between audio and image generation. If audio can be represented as an image, then an image generation model can be trained to produce new audio-like spectrograms.

This project explores that idea using rock music as the target domain. The main goal is to build a pipeline that generates short rock-style audio clips by first generating spectrogram images. The project does not train a waveform audio model. Instead, it uses a Stable Diffusion-based workflow. The model is fine-tuned on spectrogram images generated from rock music clips. Then, new generated spectrogram images are converted back to audio.

The project follows five main stages. First, rock music audio files are collected from the GTZAN dataset [1]. Second, the audio files are resampled and split into 4-second chunks. Third, each chunk is converted into a 512 by 512 mel spectrogram image. Fourth, these spectrograms are used to fine-tune the image generation model. Fifth, generated spectrograms are reconstructed back into audio and merged into a final track.

The motivation for this approach is practical. Stable Diffusion and DreamBooth tools are easier to run in colab environment than training a full audio generation model from scratch. Spectrograms also provide a visual representation of audio structure. This makes it possible to use image-based generative models for an audio-related task.

The main limitation is also clear. A spectrogram image does not preserve all information from the original waveform. In particular, phase information is lost. During reconstruction, the Griffin-Lim algorithm estimates the missing phase [2]. This usually introduces artifacts such as smearing, noise, and metallic sound. Therefore, this project should be understood as an experimental image-to-audio generation pipeline.

II. PROJECT SOURCES AND TEAM CONTRIBUTIONS

This project used existing opensource Colab notebooks as base environments. The original notebooks were not written by our team. Our team used them to set up training and generation runtimes, then added project-specific code for audio preparation, spectrogram conversion, model training setup, generation, reconstruction, and final audio post-processing.

A. Referenced Open-Source Notebooks

For model training, our team used TheLastBen's Colab Pro ComfyUI notebook as the base training environment:

- TheLastBen fast-stable-diffusion ComfyUI Colab notebook [6]

For generation, our team used a separate runtime where the trained model was loaded and used to generate new spectrogram images:

- TheLastBen fast-stable-diffusion AUTOMATIC1111 Colab notebook [7]

B. Our Own Work

The project-specific code and modifications added by our team include:

- downloading and organizing the GTZAN rock dataset
- resampling audio files to 16 kHz

Algorithm 1: GTZAN Rock Audio Chunking

Input: GTZAN rock audio folder
Output: Folder of 4-second Wav chunks
 $SR \leftarrow 16000$
 $T \leftarrow 4.0$
 $L \leftarrow SR \cdot T$
Create output directory `rock_chunks`
 $files \leftarrow$ all WAV files in input folder
 $total \leftarrow 0$
if $files$ is empty **then**
 print “No audio files found”
 stop
foreach $file \in files$ **do**
 $name \leftarrow$ filename without extension
 $y \leftarrow \text{load}(file, \text{sample rate } SR, \text{mono}=\text{True})$
 $K \leftarrow \lfloor \text{length}(y)/L \rfloor$
 for $i \leftarrow 0$ **to** $K - 1$ **do**
 $start \leftarrow i \cdot L$
 $end \leftarrow start + L$
 $chunk \leftarrow y[start : end]$
 $out \leftarrow name + \text{“_chunk”} + i + \text{“.wav”}$
 write WAV($out, chunk, SR$)
 $total \leftarrow total + 1$
 print $total$ created chunks

- chopping 30-second tracks into 4-second audio chunks
- converting audio chunks into mel spectrogram images
- preparing image-text pairs for DreamBooth or LoRA training
- designing captions with the custom identifier token `rockspectro`
- loading the trained model for spectrogram image generation
- converting generated spectrogram images back into audio
- merging reconstructed audio clips into one track
- applying crossfade and distortion post-processing

III. DATASET PREPARATION

The dataset preparation stage converts the original GTZAN rock recordings into short waveform samples. These samples are later converted into mel spectrogram images for training. Only the rock genre is used, since the goal of the project is to generate rock-style spectrograms and reconstruct them into audio.

The original GTZAN tracks are approximately 30 seconds long. They are divided into 4 second chunks to keep the spectrograms compact and consistent. All files are resampled to 16 kHz and converted to mono before chunking. Therefore each 4 second sample contains 64000 samples.

Only complete chunks are saved. Any leftover audio shorter than 4 seconds is discarded. This keeps every training sample the same length and avoids inconsistent spectrogram sizes. The pre processing procedure is summarized in Algorithm 1.

The output of this stage is a folder of fixed-length 4-second WAV files. These files are not used directly by Stable Diffusion. They are used in the next step, where each chunk is converted into a 512 by 512 mel spectrogram image and paired with a text caption.

IV. SPECTROGRAM ENCODING

A. Mel Spectrogram Representation

After the audio files are divided into 4-second chunks, each chunk is converted into a mel spectrogram using Librosa [3]. This step changes the audio from a one-dimensional waveform into a two-dimensional time-frequency representation. Instead of storing only amplitude over time, the spectrogram shows how audio energy is distributed across frequency bands over time.

The mel spectrogram is used because it gives a compact representation of sound that is close to human frequency perception. Lower frequencies are represented with finer resolution, while higher frequencies are compressed. This is useful for music signals, where bass, drums, guitar texture, and high-frequency noise occupy different regions of the spectrogram.

In this project, the spectrogram is the main training representation. The diffusion model does not learn directly from waveform audio. It learns from spectrogram images that visually encode rock music structure.

The sampling rate is fixed at 16 kHz, matching the dataset preparation stage. The FFT size controls the window used for frequency analysis. The hop length controls the time spacing between neighboring spectrogram frames. The number of mel bins is set to 256, so each spectrogram has 256 frequency bands before image resizing.

After computing the mel spectrogram, the power values are converted into decibels. This is done because raw power values have a large numerical range. Decibel scaling makes both loud and quiet parts of the audio visible in the same image. In the implementation, the maximum value of each spectrogram is used as the reference level. The dynamic range is limited to 80 dB, so very low-energy values are clipped.

B. Image Normalization

The decibel spectrogram is normalized into image pixel values. The range $[-80, 0]$ dB is mapped into the image range $[0, 255]$:

$$S_{\text{img}} = 255 \cdot \frac{S_{\text{dB}} + 80}{80}. \quad (1)$$

After normalization, the values are clipped to the valid image range and converted to unsigned 8-bit integers. Brighter pixels correspond to stronger audio energy, while darker pixels correspond to weaker energy.

The image is then flipped vertically so that low frequencies appear at the bottom and high frequencies appear at the top. Although the original spectrogram is grayscale, it is duplicated across three channels to form an RGB image. This is done because Stable Diffusion models expect RGB inputs. The final image is resized to 512 by 512 pixels, which matches the training format used in the diffusion workflow.

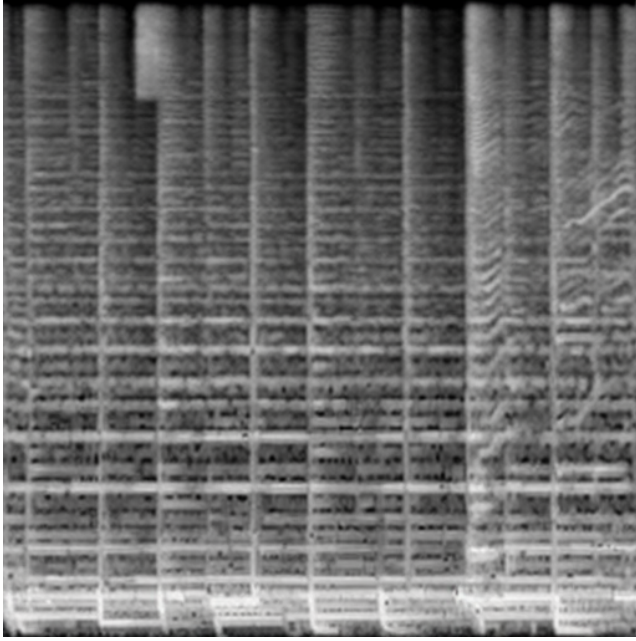


Fig. 1. Example mel spectrogram image generated from a 4-second rock music chunk.

C. Output of Spectrogram Encoding

The output of this stage is a set of PNG spectrogram images (see Fig. 1). Each image corresponds to one 4-second WAV chunk from the previous stage. These images are later paired with text captions and used as the training dataset for the generative model.

This encoding step strongly affects the final result. The model learns the visual structure of rock music spectrograms, not the waveform itself. Therefore, consistent spectrogram parameters are needed so that the generated images can later be decoded back into audio.

V. MODEL TRAINING PIPELINE

A. Training Format

The spectrogram dataset is prepared as image-text pairs. Each spectrogram image has a corresponding caption file with the same base name. The file naming format follows a DreamBooth-style structure:

```
rockspectro (1).png
rockspectro (1).txt
rockspectro (2).png
rockspectro (2).txt
```

Each caption contains the custom identifier token and a short description of the target sound:

```
rockspectro, full rock band, distorted guitars,
heavy drums, complex audio spectrogram
```

The identifier `rockspectro` is used to teach the model a specific visual concept associated with rock music spectrograms.

TABLE I
DREAMBOOTH FINE-TUNING CONFIGURATION.

Parameter	Value
Base model	Stable Diffusion v1.5
Trainable component	UNet only
Resolution	512×512
UNet training steps	300
UNet learning rate	2×10^{-6}
Text-encoder steps	0 (frozen)
Optimizer	8-bit Adam
Mixed precision	fp16
External captions	Yes (per-image .txt)
Offset noise	Disabled
Instance images	≈ 700
Identifier token	<code>rockspectro</code>
Effective batch size	1

B. Training Environment

The training stage uses TheLastBen’s fast-stable-diffusion ComfyUI Colab notebook as the base environment [6]. Our team loaded the prepared spectrogram image-text dataset into this environment and trained a model using the project-specific spectrogram data.

The original notebook was used for environment setup and access to the training workflow. The project-specific dataset creation and spectrogram processing code was added separately by our team.

C. Training Hyperparameters

The fine-tuning configuration follows the DreamBooth recipe in TheLastBen’s training notebook. The base model is Stable Diffusion v1.5; only the UNet is updated, while the CLIP text encoder is frozen because the rock-spectrogram identifier is paired with a fixed caption template, so additional text-encoder capacity offers no benefit. Mixed-precision training in fp16 and the 8-bit Adam optimizer are used to fit the 512×512 training resolution into a single Colab T4 GPU. Table I lists the values used in the final run.

The number of training steps is deliberately conservative. Longer schedules were tested during development but produced spectrograms that visibly snapped to a small set of memorised training images, indicating mode collapse onto the limited dataset. Three hundred steps consistently retained the broad image priors of Stable Diffusion while clearly responding to the `rockspectro` token.

D. Generation Environment

After training, generation was performed in a different runtime using TheLastBen’s fast-stable-diffusion AUTOMATIC1111 Colab notebook [7]. In that runtime, the trained model was loaded and used to generate new spectrogram images from text prompts.

Example prompt:

```
rockspectro, full rock band, distorted guitars,
heavy drums, complex audio spectrogram
```

The generated image is then treated as a predicted mel spectrogram and passed into the reconstruction pipeline.

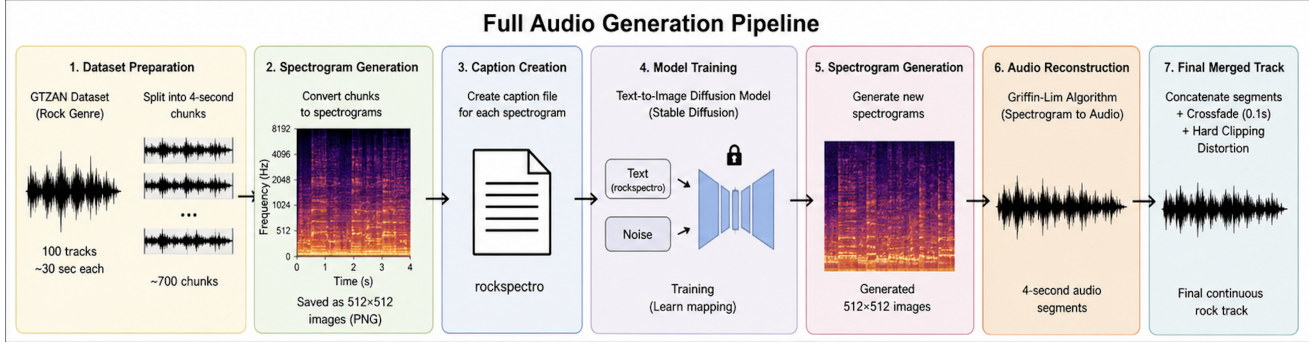


Fig. 2. Template figure. Full project pipeline from GTZAN audio files to generated rock-style audio. Replace this with a diagram showing dataset preparation, spectrogram conversion, training, generation, reconstruction, and post-processing.

E. Inference Settings

Generation is performed in TheLastBen’s AUTOMATIC1111 runtime with the trained checkpoint loaded from Google Drive. Each spectrogram is sampled with the Euler-Ancestral scheduler at 30 denoising steps and a classifier-free guidance scale of 7.0; the negative prompt is left empty. Seeds are randomised across the batch so that the merged final track is built from genuinely distinct generations rather than seed-shifted variants of one image. For the demonstration track, twelve images were generated in a single batch, exported as 512×512 PNGs, and decoded individually before being concatenated.

VI. AUDIO RECONSTRUCTION FROM GENERATED IMAGES

A. Reverse Image Processing

The generated Stable Diffusion output is a 512 by 512 image. To convert it back into audio, the image is first converted to grayscale. Then it is resized back to the original spectrogram dimensions:

$$251 \times 256, \quad (2)$$

where 251 is the approximate number of time frames for a 4-second chunk and 256 is the number of mel bins.

The image is then flipped vertically to restore the frequency ordering expected by Librosa. Pixel values are mapped back into decibels:

$$S_{\text{dB}} = 80 \cdot \frac{S_{\text{img}}}{255} - 80. \quad (3)$$

The decibel spectrogram is then converted back into power:

$$S_{\text{power}} = 10^{S_{\text{dB}}/10}. \quad (4)$$

Two strategies were tested for mapping the 512×512 generated image back to the native mel grid of 256×251 . Bicubic down-resizing preserves the overall energy distribution but smears narrow harmonic bands. The alternative is to *crop* the bottom-left 256×251 region directly, treating the diffusion canvas as if the active spectrogram occupies that exact sub-rectangle. Cropping yields sharper transients and harmonic edges but is sensitive to drift in where the model places the

spectrogram on the canvas. The final pipeline uses bicubic resizing as the default because it degrades more gracefully when the generated image is not perfectly aligned, with cropping retained as a fallback for cases where the model centers its output cleanly.

B. Griffin-Lim Reconstruction

The spectrogram image stores magnitude-related information, but it does not store the original phase. Therefore, the reconstruction step estimates the missing phase using Griffin-Lim reconstruction [2].

The reconstructed waveform can be written as:

$$\hat{y} = \text{MelToAudio}(S_{\text{power}}). \quad (5)$$

This reconstruction is approximate. It usually introduces audible artifacts because the phase is guessed rather than recovered exactly.

VII. POST-PROCESSING AND FINAL TRACK GENERATION

A. Spectrogram Noise Gating

Before reconstruction, the generated image is passed through a hard pixel gate that suppresses low-amplitude background regions. These regions, after inverse mel and Griffin-Lim, otherwise manifest as broadband hiss in the reconstructed audio. Given the 8-bit image S_{img} , the gated image is

$$S'_{\text{img}}[t, f] = \begin{cases} S_{\text{img}}[t, f], & S_{\text{img}}[t, f] \geq \tau, \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

with $\tau = 50$ on the 0-255 image range. The threshold is intentionally aggressive: relaxing it restores more of the reverb tail at the cost of audible noise, while raising it produces a stuttering effect on sustained chords. The chosen value gave the cleanest attack on percussive regions across the test set without removing audible body from the reconstructed audio.

After several generated spectrogram images are reconstructed into audio clips, the clips are merged into one longer waveform. Direct concatenation can create clicks at clip boundaries. To reduce this issue, the project uses short crossfades between neighboring clips.

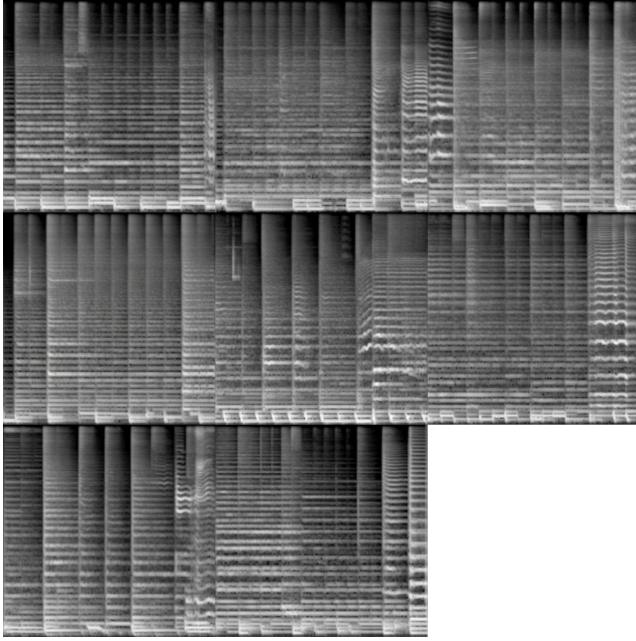


Fig. 3. Examples of generated spectrogram images from the trained model

For two neighboring audio segments x_1 and x_2 , the overlapping region is merged using:

$$x_{\text{fade}}[n] = (1 - \alpha[n])x_1[n] + \alpha[n]x_2[n], \quad (7)$$

where $\alpha[n]$ increases linearly from 0 to 1 over the crossfade duration.

In the final version, a short crossfade of 0.1 seconds is used. After merging, hard clipping distortion is applied to make the result more intense:

$$y_{\text{drive}} = D \cdot y, \quad (8)$$

where D is the drive parameter. Then the waveform is clipped:

$$y_{\text{final}} = \text{clip}(y_{\text{drive}}, -1, 1). \quad (9)$$

In the implementation, the drive value is set to:

$$D = 3.0. \quad (10)$$

This increases loudness and creates distortion.

VIII. EXPERIMENTAL RESULTS

A. Dataset Output

The dataset construction process transformed raw audio recordings from the GTZAN dataset into a structured multimodal dataset suitable for generative modeling. Specifically, the rock genre subset was selected, consisting of 100 audio tracks, each approximately 30 seconds long. These tracks were segmented into shorter, fixed-length audio chunks of 4 seconds to increase the number of training samples and improve model generalization.

TABLE II
DATASET STATISTICS.

Item	Value
Original dataset	GTZAN
Selected genre	Rock
Number of original tracks	100
Track length	Approximately 30 seconds
Chunk length	4 seconds
Number of generated chunks	700
Spectrogram image size	512 by 512
Caption identifier	rockspectro

After segmentation and filtering of incomplete segments, approximately 700 valid audio chunks were retained. Each chunk was converted into a corresponding spectrogram representation using Short-Time Fourier Transform. These spectrograms were stored as grayscale images with a resolution of 512×512 pixels, preserving the time-frequency characteristics of the original signals.

In parallel, each spectrogram was paired with a simple textual caption, enabling multimodal training. The final dataset consisted of aligned image-text pairs, where each spectrogram image corresponded to a descriptive label. This format allowed the use of diffusion-based text-to-image models for learning the distribution of rock music spectrograms.

The constructed dataset reflects both the constraints of the original GTZAN dataset and the preprocessing design choices. Starting from 100 rock tracks, segmentation into 4-second windows yielded approximately 700 usable samples after removing edge cases and incomplete segments.

Each spectrogram image has a consistent resolution of 512×512 pixels, ensuring uniformity during model training. The use of fixed-length audio chunks simplifies the reconstruction process and allows batch processing during training.

Overall, the dataset can be characterized as a relatively small but structured multimodal dataset, designed specifically for experimentation with generative models. While sufficient for proof-of-concept results, the dataset size remains a limiting factor for achieving high-fidelity generation.

B. Training Results

The generative model demonstrated the ability to learn key visual patterns present in spectrogram representations of rock music. During training, the model progressively captured high-energy frequency regions, temporal transitions, and vertical structures associated with percussive events such as drum hits.

Generated spectrograms exhibited several realistic characteristics, including broadband energy distribution and repetitive temporal structures that resemble rhythmic components of rock music. In some cases, the outputs showed convincing patterns similar to distorted guitar textures, which are common in the genre.

However, the model also produced noticeable artifacts. These included overly smooth regions lacking detail, inconsistent frequency bands, and unnatural transitions in time-frequency space. Such imperfections indicate that the model

struggled to fully capture fine-grained spectral details, likely due to the limited dataset size and the complexity of audio representations.

Despite these limitations, the results confirm that diffusion-based models can learn meaningful representations of audio when expressed as spectrogram images.

C. Audio Reconstruction Results

To evaluate the generated spectrograms, they were converted back into audio signals using the Griffin-Lim algorithm. This reconstruction process enabled qualitative assessment of the generated outputs.

The reconstructed audio exhibited several characteristics associated with rock music, including high energy, noise-like textures, and rhythmic fluctuations. In some cases, the outputs resembled distorted guitar sounds or percussive elements, suggesting that the model successfully captured certain structural aspects of the genre.

However, the overall audio quality was limited. The absence of phase information in the generated spectrograms led to reconstruction artifacts, including metallic tones, phase inconsistencies, and temporal smearing. Additionally, some outputs contained irregular or chaotic noise patterns that reduced perceptual coherence.

These limitations highlight a fundamental challenge in spectrogram-based generation: while magnitude information can be modeled effectively, accurate phase reconstruction remains difficult, significantly impacting audio fidelity.

D. Final Merged Track

To produce a continuous audio output, multiple reconstructed audio segments were concatenated into a single track. Short crossfades of approximately 0.1 seconds were applied between consecutive segments to reduce abrupt transitions and improve continuity.

Post-processing techniques were also applied to enhance the stylistic qualities of the output. In particular, hard clipping distortion was introduced to emulate the aggressive tonal characteristics commonly found in rock music. This step helped mask some reconstruction artifacts while reinforcing the perceived intensity of the signal.

The resulting merged track presented a continuous, noise-heavy audio sequence with recognizable stylistic elements of rock music. Although the output lacked the clarity, structure, and musical coherence of real recordings, it demonstrated the feasibility of generating audio through spectrogram-based diffusion models.

Overall, the final result can be considered an experimental approximation of rock audio, capturing its general texture and energy while highlighting the challenges of high-quality audio synthesis.

IX. DISCUSSION

The project demonstrates that a text-to-image model can be adapted to generate spectrogram-like images and that these

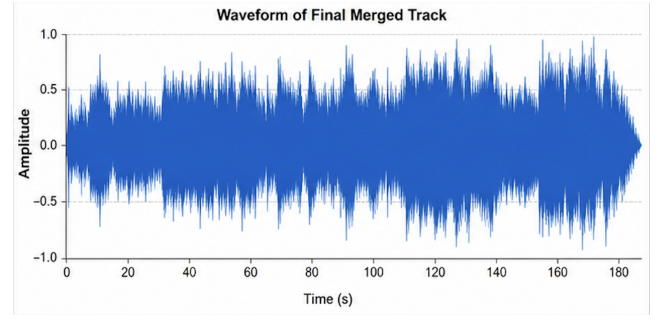


Fig. 4. Template figure. Waveform of the final generated audio track after merging and distortion. Replace this with a waveform plot from your final WAV file.

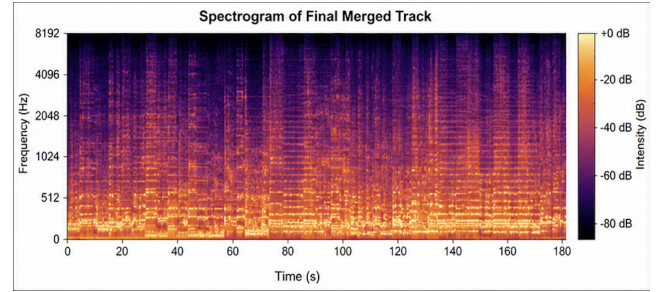


Fig. 5. Template figure. Spectrogram of the final reconstructed audio track. Replace this with the spectrogram of your final output file.

images can be converted into audio. This confirms that spectrograms can act as an intermediate representation between image generation and audio synthesis.

However, the method has several limitations. First, the image representation loses phase information. This makes exact reconstruction impossible. Second, resizing the spectrogram to 512 by 512 and then resizing it back introduces interpolation artifacts. Third, the generated spectrograms are optimized visually by the image model. They are not directly optimized for audio quality. A generated image may look like a spectrogram but still decode into poor audio.

Another limitation is the caption design. In the current version, most captions are similar. This helps the model associate the identifier token with rock spectrograms, but it does not give detailed control over musical features such as tempo, vocals, guitar riffs, or drum patterns. More diverse captions could improve controllability.

The reconstruction stage is also a major bottleneck. Griffin-Lim is useful for approximation, but it cannot fully recover natural-sounding audio from magnitude only mel spectrograms. Future versions could use a neural vocoder instead of Griffin-Lim. This would likely improve sound quality.

X. LIMITATIONS AND FUTURE WORK

This current system has a few drawbacks. For starters, we lose the phase information when we create the spectrogram images. While the images capture the magnitude aspects of the audio, they do not keep the original waveform's phase. This means the reconstruction process relies on Griffin-Lim

for phase estimation, which can unfortunately add noise, blurriness, and a metallic sound in the reconstructed audio. Because of this, a spectrogram can look visually correct, but after converting it back to sound, the result may still be noisy or not very musical.

Another limitation is that the model generates images, not physically constrained audio representations. Stable Diffusion learns the visual appearance of rock spectrograms, but it does not directly optimize for sound quality. The final result also depends strongly on post-processing steps such as crossfading, clipping, and distortion. In addition, the current caption set is limited, so the model has weak control over specific musical features such as tempo, vocals, drum patterns, or guitar style. The method was also tested only on one genre, so its generality across other types of music is not yet verified.

Future work could improve the data processing pipeline by replacing Griffin-Lim reconstruction with a neural vocoder. This vocoder would be trained to convert mel spectrograms into waveform audio signals. It would probably reduce some reconstruction artifacts and make the generated audio sound more natural. The dataset could also be expanded with more music genres and more detailed captions, not just one general description. This can give better control over what kind of output the model generates.

Another possible improvement is to use overlapping fragments during dataset preparation. It would increase the number of training samples, and maybe make transitions between generated clips smoother. The project could also test different spectrogram settings, for example hop length, FFT size, image resolution, and number of mel bins. Finally, the evaluation part can be improved too. It could include objective audio metrics and also listening tests, because in the end the sound quality is still judged by how people hear it.

XI. CONCLUSION

This project presents a spectrogram-based pipeline for generating rock-style audio using a fine-tuned Stable Diffusion model. The system converts rock music audio into mel spectrogram images, trains a generative image model on those images, generates new spectrograms, and reconstructs audio from the generated images. The final output is created by merging reconstructed clips and applying post-processing.

The experiment shows that image generation models can produce spectrogram-like structures that can be decoded into audio. At the same time, the project also shows the main weakness of this approach. Since the spectrogram image does not preserve phase, reconstruction quality is limited. The resulting audio is recognizable as a generated sound texture, but it is not clean music synthesis. The pipeline is useful as an experimental method for connecting audio processing and image generation.

REFERENCES

- [1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293-302, 2002.
- [2] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236-243, 1984.
- [3] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in Python," in *Proceedings of the 14th Python in Science Conference*, 2015, pp. 18-25.
- [4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10684-10695.
- [5] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22500-22510.
- [6] TheLastBen, "fast-stable-diffusion ComfyUI Colab notebook," GitHub repository. [Online]. Available: https://github.com/TheLastBen/fast-stable-diffusion/blob/main/fast_stable_diffusion_ComfyUI.ipynb
- [7] TheLastBen, "fast-stable-diffusion AUTOMATIC1111 Colab notebook," GitHub repository. [Online]. Available: https://github.com/TheLastBen/fast-stable-diffusion/blob/main/fast_stable_diffusion_AUTOMATIC1111.ipynb
- [8] AUTOMATIC1111, "Stable Diffusion Web UI," GitHub repository. [Online]. Available: <https://github.com/AUTOMATIC1111/stable-diffusion-webui>
- [9] ComfyUI, "The most powerful and modular stable diffusion GUI and backend," GitHub repository. [Online]. Available: <https://github.com/comfyanonymous/ComfyUI>