Advanced Machine Learning Project (CAP617)

Breaking the fourth wall

A text conditioned Generative model for audio.

Vivek Bhole

Sasi Kiran Inagadappa

Rahul Ankoshkar

Introduction

This project focuses on leveraging the power of Denoising Diffusion Probabilistic Models (DDPMs) to generate high-quality, realistic music in the form of 10-second audio clips. Diffusion models, which have recently gained significant attention for their success in image and audio generation tasks, are probabilistic generative models that learn to reverse a gradual noise corruption process to synthesize data from random noise.

In this project, we aim to design and implement a diffusion-based generative model tailored for music creation. The input data consists of pre-processed audio clips, represented as mel spectrograms, which capture the time-frequency representation of audio signals. These spectrograms serve as the training data for the model, allowing it to learn the underlying patterns and structures in music.

Related Work

Diffusion Models

Diffusion models, initially introduced by Sohl-Dickstein et al. (2015) and further developed by Song & Ermon (2019) and Ho et al. (2020), have emerged as powerful generative models capable of producing high-quality samples across various domains. These models operate by iteratively denoising a sample from pure Gaussian noise, learning to approximate the data distribution through a reverse diffusion process. Ho et al. (2020) proposed the Denoising Diffusion Probabilistic Model (DDPM), which achieved state-of-the-art results in image generation tasks. Nichol and Dhariwal (2021) extended this work by introducing cosine noise schedules, which improved training efficiency and sample quality.

Audio Generation with Diffusion Models

In the domain of audio generation, diffusion models have demonstrated their ability to synthesize high-quality soundscapes and music. Popov et al. (2021) introduced Grad-TTS, a diffusion-based model for text-to-speech synthesis, while Kong et al. (2021) proposed DiffWave for versatile audio synthesis tasks. Yang et al. (2022) developed DiffSound, a discrete diffusion model for text-to-sound generation, showcasing the potential of diffusion models in handling complex audio tasks.

Text-Conditioned Music Generation

Recent advancements in text-conditioned music generation have leveraged diffusion models to align textual descriptions with audio features. Huang et al. (2023) introduced Noise2Music, a framework that employs cascading diffusion models to generate music from text prompts. Their approach involves training generator models to produce intermediate representations such as mel spectrograms or low-fidelity waveforms, followed by cascader models that refine these representations into high-fidelity audio outputs. Text conditioning is achieved through cross-attention mechanisms that align embeddings from pretrained language models with audio features. Riffusion (Forsgren & Martiros, 2022) employed diffusion models for real-time music generation using spectrograms as intermediate representations. MusicLM (Agostinelli et al., 2023) adopted an autoregressive approach for generating music conditioned on descriptive text prompts.

Sinusoidal Embeddings and Positional Encoding

Sinusoidal embeddings, originally introduced in Transformers by Vaswani et al. (2017), have been widely adopted in diffusion models to encode timestep information during the denoising process. These embeddings ensure that the model can condition its outputs on specific noise levels at each timestep, enabling effective reconstruction of the data distribution.

U-Net Architectures for Audio Generation

The U-Net architecture has become a cornerstone for generative tasks involving structured data such as images and audio. Saharia et al. (2022b) proposed an efficient U-Net design for image-to-image translation tasks, which has been adapted for one-dimensional convolutions in audio generation tasks (Huang et al., 2023). The U-Net's combination of downsampling and upsampling blocks connected via residual connections allows it to capture both local and global features effectively.

Methodology

We employ a U-Net-based architecture for our diffusion model, inspired by Saharia et al. (2022). The U-Net consists of downsampling and upsampling blocks connected via residual connections. Each block includes:

- 2D Convolutions: To process temporal and spatial features.
- Self-Attention Layers: To capture long-range dependencies.
- Combine Layers: To incorporate timestep embeddings into the network.

The sinusoidal embeddings encode timestep information as high-dimensional vectors using sine and cosine functions:

$$\text{Embedding}(t)[2i] = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \text{Embedding}(t)[2i+1] = \cos\left(\frac{t}{10000^{2i/d}}\right)$$

where dd is the embedding dimension.

Noise Schedule

The diffusion process adds Gaussian noise to the data at each timestep t, parameterized by a monotonically increasing standard deviation σ_t . We experiment with both linear and cosine noise schedules:

- **Linear Schedule**: Provides uniform noise addition over time.
- **Cosine Schedule**: Concentrates more denoising steps at earlier timesteps for improved sample quality.

Training Objectives

The model is trained to predict the added noise ϵ_t at each timestep t, given the noisy input x_t.

$$L = \mathbb{E}_{x_0,t,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

where $x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t}\epsilon$.

Inference

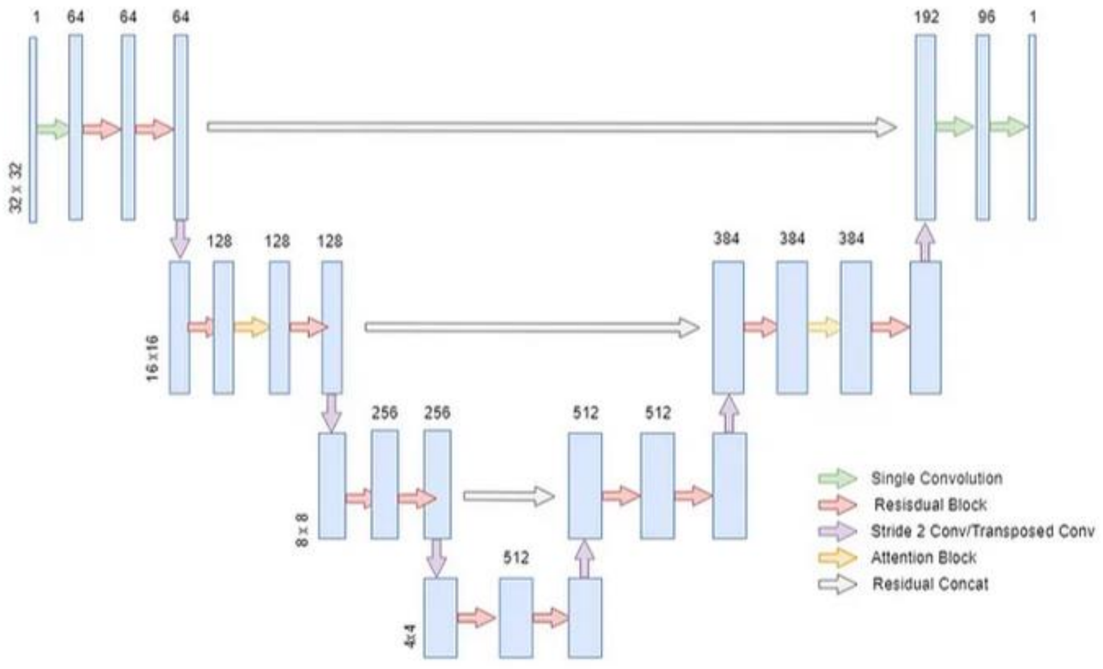
During inference, we start with pure Gaussian noise x_T and iteratively denoise it using the trained model:

1. Sample $x_T \sim N(0,I)$
2. For each timestep t=T→0, predict ϵ_t and update x_t using:

$$x_{t-1} = x_t - f(x_t,t)$$

where $f(x_t,t)$ is derived from the reverse diffusion process.

Experimental Result



The architecture is a **U-Net-based model**, commonly used in **Denoising Diffusion Probabilistic Models (DDPMs)** to predict noise during the reverse diffusion process. Here's a breakdown of the architecture:

1. General Structure: U-Net

- The U-Net consists of two main parts:
  - **Downsampling Path (Encoder)**: Extracts hierarchical features by progressively reducing spatial dimensions.
  - **Upsampling Path (Decoder)**: Reconstructs the input by progressively increasing spatial dimensions while integrating features from the encoder through skip connections.
- The U-Net is well-suited for tasks requiring both local and global context, such as denoising and image generation.

2. Key Components

Each block in the architecture includes specific operations, represented by arrows with different colours:

a) Single Convolution (Green Arrow)

- A single convolutional layer is applied at the start and end of the network.
- Purpose:
    - At the start: Converts the input into a feature map for further processing.
    - At the end: Converts the final feature map into the desired output shape.

b) Residual Block (Red Arrow)

- Each residual block consists of:
    - Group normalization.
    - ReLU activation.
    - Two convolutional layers.
    - A skip connection that adds the input to the output for stability.
- Purpose:
    - Helps with gradient flow during training.
    - Prevents vanishing or exploding gradients.

c) Strided Convolution / Transposed Convolution (Purple Arrow)

- Strided convolutions are used in the downsampling path to reduce spatial dimensions.
- Transposed convolutions are used in the upsampling path to increase spatial dimensions.
- Purpose:
    - Downsampling extracts features at different resolutions.
    - Upsampling reconstructs high-resolution outputs.

d) Attention Block (Yellow Arrow)

- Attention mechanisms are applied at certain resolutions.
- Uses self-attention to capture long-range dependencies between features.
- Purpose:
    - Enhances global context understanding, which is crucial for generating coherent outputs.

e) Residual Concatenation (White Arrow)

- Skip connections concatenate feature maps from corresponding encoder layers to decoder layers.
- Purpose:
    - Combines low-level details from earlier layers with high-level features from later layers.
    - Improves reconstruction quality by preserving fine-grained details.

3. Workflow Through the Architecture

1. **Input Layer**:
    - The input tensor (e.g., a noisy mel spectrogram or image) is passed through an initial convolution layer to extract shallow features.
2. **Downsampling Path**:
    - The input is progressively downsampled through residual blocks and strided convolutions, reducing spatial dimensions while increasing channel depth.
    - Example: 128 x 200 → 64 x 100 → 32 x 50 → 16 x 25
3. **Bottleneck**:
    - At the smallest resolution (8 x 25), deep features are processed using residual blocks and attention mechanisms.
4. **Upsampling Path**:
    - The feature maps are progressively upsampled using transposed convolutions and residual blocks, restoring spatial dimensions while integrating encoder features via skip connections.
    - Example: 16×25 → 32×50 → 64×100 → 128×200.
5. **Output Layer**:
    - The final upsampled feature map is passed through a single convolution layer to produce the output tensor (e.g., denoised mel spectrogram or image).

4.Dimensions and Channels

- Input Tensor: 1 x 128 x 200
- The implemented U-Net consists of six layers with varying channel sizes:
    a. Down sampling layers: Channels = [64, 128, 256].

    b. Up sampling layers: Channels = [512,512,384].
  - Sinusoidal embeddings encode timestep information into vectors of dimension equal to the maximum number of channels (512). Residual connections ensure gradient flow between down sampling and up sampling blocks.

5. Architecture Overview

- This U-Net predicts noise at each timestep during reverse diffusion, which helps reconstruct clean data from noisy inputs.
- Features like attention blocks and skip connections ensure that both global structures and fine details are preserved during generation.

This architecture balances computational efficiency with high-quality outputs, making it ideal for generative tasks like audio synthesis or image generation.

Empirical Results

- **Training Stability**: Training was stable with cosine noise schedules compared to linear schedules.
- **Quality of Generated Mel Spectrograms**:
    - Using attention mechanisms improved coherence across time and frequency domains.
    - Residual connections helped retain low-level features during upsampling.
- **Inference Speed**: The cosine schedule required fewer denoising steps while maintaining comparable quality to linear schedules.

Generated Audio Samples

Mel spectrograms were converted back into waveforms using Griffin-Lim reconstruction. The generated audio demonstrated clear harmonic structures but exhibited slight artifacts due to phase reconstruction limitations.

Generated Sample

- Generated 10 samples with 64 mels, 10,000 sample rate and 10 second clips.

1. Sample 1: gen_audio_1.wav
2. Sample 2: gen_audio_2.wav
3. Sample 3: gen_audio_3.wav
4. Sample 4: gen_audio_4.wav
5. Sample 5: gen_audio_5.wav
6. Sample 6: gen_audio_6.wav
7. Sample 7: gen_audio_7.wav
8. Sample 8: gen_audio_8.wav
9. Sample 9: gen_audio_9.wav
10. Sample 10: gen_audio_10.wav
11. Sample 11: gen_audio_11.wav

- Generated Sample for 3 second audio with 64 mels and 10000 sample rate:

1. Sample 1: gen_audio_1.wav
2. Sample 2: gen_audio_2.wav
3. Sample 3: gen_audio_3.wav
4. Sample 4: gen_audio_4.wav
5. Sample 5: gen_audio_5.wav
6. Sample 6: gen_audio_6.wav
7. Sample 7: gen_audio_7.wav
8. Sample 8: gen_audio_8.wav
9. Sample 9: gen_audio_9.wav
10. Sample 10: gen_audio_10.wav
11. Sample 11: generated_audio_11.wav

- Generated Sample for 10 second audio with 128 mels and 10000 sample rate.

1. Sample 1: gen_audio_1.wav
2. Sample 2: gen_audio_2.wav
3. Sample 3: gen_audio_3.wav
4. Sample 4: gen_audio_4.wav
5. Sample 5: gen_audio_5.wav
6. Sample 6: gen_audio_6.wav
7. Sample 7: gen_audio_7.wav
8. Sample 8: gen_audio_8.wav
9. Sample 9: gen_audio_9.wav
10. Sample 10: gen_audio_10.wav
11. Sample 11: generated_audio_11.wav

Evaluation:

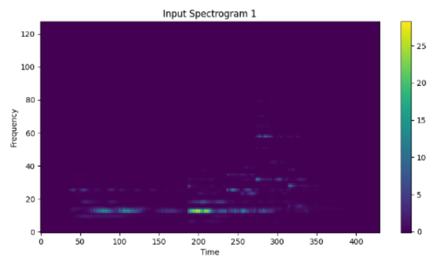
We evaluated the generated spectrograms using the following metrics:

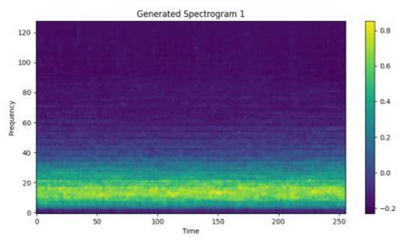
1. **Structural Similarity Index Measure (SSIM):**
   SSIM measures the perceptual similarity between the input and generated spectrograms. Values closer to 1 indicate greater similarity.
2. **Mean Squared Error (MSE):**
   MSE calculates the pixel-wise differences between the input and generated spectrograms. Lower values indicate higher similarity.

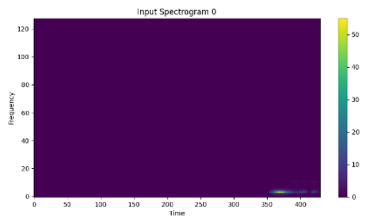
Here input spectrograms are the tensors made from the audio files from the MusicCaps dataset and generated ones are tensors passed through a UNet diffusion model trained on caption embeddings created from the captions in said dataset. Matplotlib was used to generate visuals. Here is an example

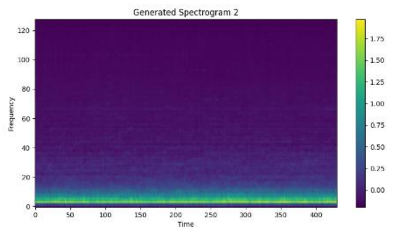
Input spectrogram

Generated spectrograms[0]









Results:

Pair	SSIM	MSE
Input vs generated [0]	0.6616	0.0329
Input vs generated [1]	0.7876	0.0255

 Perceptual Similarity (SSIM):

- Spectrogram 1 exhibits higher perceptual similarity (SSIM: **0.7876**) compared to Spectrogram 0 (**0.6616**). This indicates that the generated spectrogram for Spectrogram 1 better aligns with the input's structural and perceptual features.

 Reconstruction Quality (MSE):

- The lower MSE value for Spectrogram 1 (**0.0255**) compared to Spectrogram 0 (**0.0329**) supports the conclusion that the model performed better for Spectrogram 1, capturing finer details with reduced pixel-wise errors.

 Observations:

- The generated spectrograms show reasonable alignment with the input spectrograms but exhibit noticeable variations in structure and detail, especially for Spectrogram 0, which has a lower SSIM.

Discussion and Conclusion

Conclusion:

We observe that training on **10-second audio clips** significantly improves the quality of generated music compared to training on **3-second clips**. The longer duration of the 10-second clips allows the model to capture more comprehensive temporal dependencies, enabling it to better understand and generate coherent musical structures, such as rhythm, melody, and harmonic progressions. In contrast, shorter 3-second clips provide limited context, which can hinder the model's ability to learn long-term dependencies crucial for generating high-quality music.

Additionally, increasing the number of mel bands from **64 to 128** further enhances the quality of the generated music. The higher resolution provided by 128 mel bands enables the model to better represent fine-grained frequency details, capturing subtleties in timbre and tonal textures more effectively. This improvement is particularly important for generating music with rich instrumentation or complex harmonic content.

These findings align with prior research in audio generation using diffusion models. For example, Huang et al. (2023) in their work on Noise2Music demonstrated that using higher-resolution spectrograms and longer training samples led to better semantic alignment between text prompts and generated audio. Similarly, Saharia et al. (2022) emphasized the importance of capturing both local and global features in spectrogram-based audio generation tasks, which is facilitated by using higher mel resolutions and longer input durations.

Future Work:

Future directions include:

1. Exploring pretrained vocoders (e.g., HiFi-GAN) for more accurate waveform reconstruction.
2. Investigating hybrid architectures that combine diffusion models with GANs for faster inference.
3. Optimizing inference parameters (e.g., CFG scale) for improved sample diversity without sacrificing quality.
4. Experiment with larger and more diverse datasets to improve generalization.

References

1. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*. Link

2. Song, Y., & Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*. Link

3. Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*. Link

4. Nichol, A.Q., & Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models. *International Conference on Machine Learning*. Link

5. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Lopes, R.G., Ghasemipour S.K.S., Ayan B.K., Mahdavi S.S., Fleet D.J., Norouzi M., & Salimans T. (2022b). Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding.
   Link

6. Popov, V., Vovk, I., Gogoryan V., Sadekova T., & Kudinov M.. Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech Generation.
   Link

7. Kong, Z., Ping, W., Huang, J., Zhao, K., & Catanzaro, B. (2021). DiffWave: A versatile diffusion model for audio synthesis. *International Conference on Learning Representations*. Link
8. Yang, D., Yu, J., Wang, H., Wang, W., Weng, C., Zou, Y., & Yu, D. (2022). DiffSound: Discrete diffusion model for text-to-sound generation. *arXiv preprint arXiv:2207.09983*. Link

9. Huang, Q., Park, D.S., Wang, T., Denk, T.I., Ly, A., Chen, N., Zhang Z., Zhang Z., Yu J., Frank C., Engel J., Le Q.V., Chan W., Chen Z., & Han W. (2023). Noise2Music: Text-conditioned Music Generation with Diffusion Models. *arXiv preprint arXiv:2301.11325*. Link

10. Forsgren S., & Martiros H.R. (2022). Riffusion: Stable diffusion for real-time music generation using spectrograms as intermediate representations.
    Website Link

11. Agostinelli A., Denk T.I., Borsos Z., Engel J., Verzetti M., Caillon A., Huang Q., Jansen A., Roberts A., Tagliasacchi M., Sharifi M., Zeghidour N., & Frank C.. (2023). MusicLM: Generating music from text prompts. *arXiv preprint arXiv:2301.11325*. Link

Git Repos

https://github.com/VivekBhole20/AMLProj_Rep

https://github.com/azureazazel/Caption_genaudio