

GCT634/AI613: Musical Applications of Machine Learning

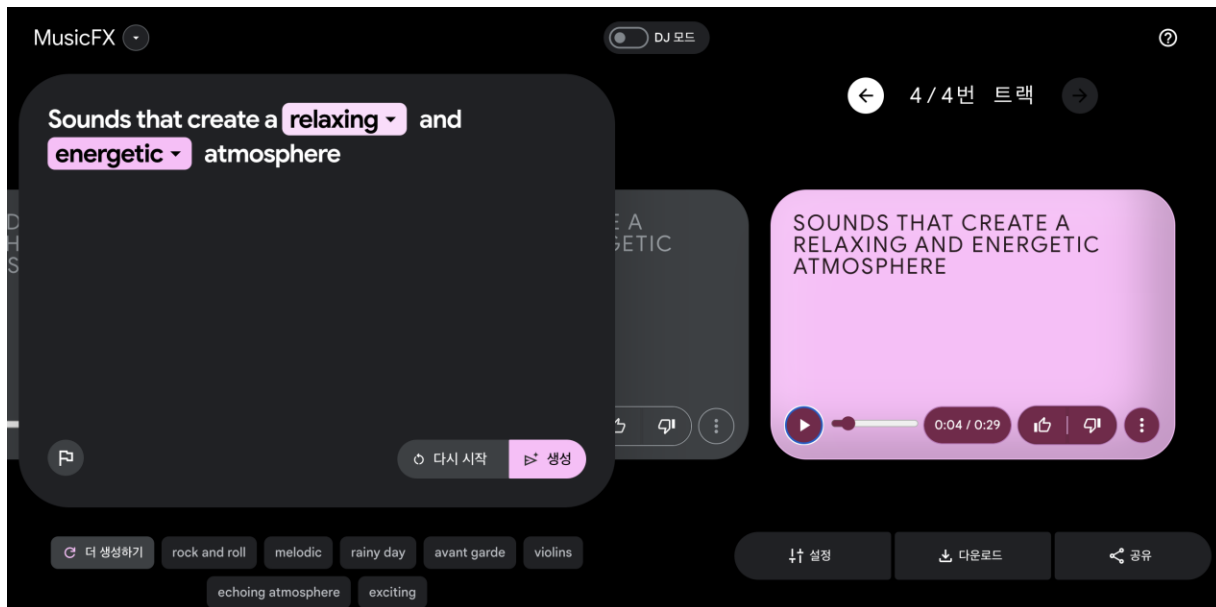
# Audio-Level Music Generation



**Juhan Nam**

# Recent Advances in Audio Generation AI

- Music generation



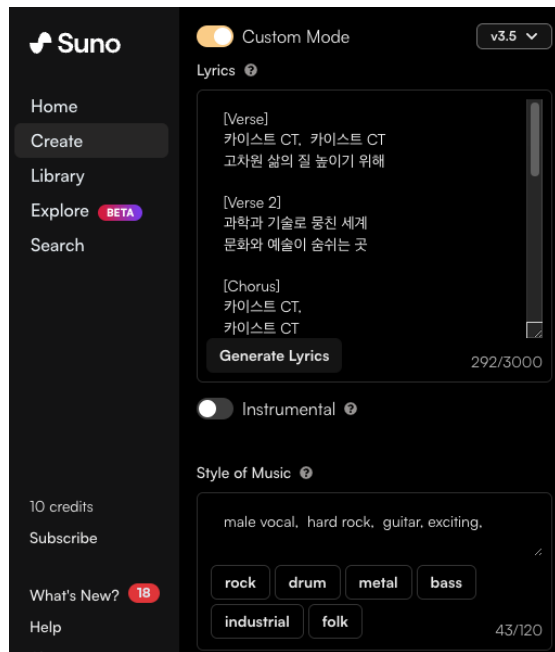
Google MusicFX



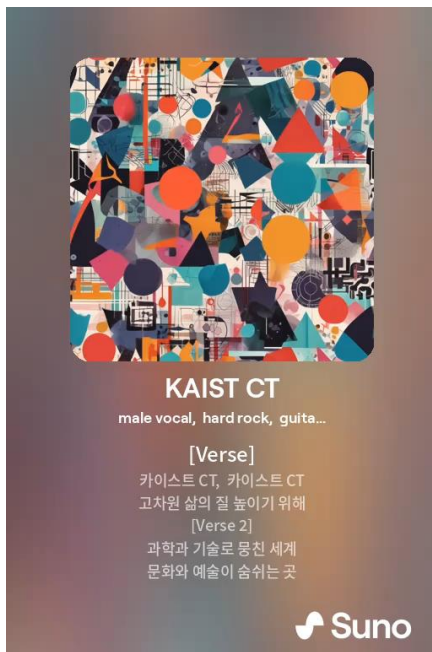
<https://aitestkitchen.withgoogle.com/ko/tools/music-fx>

# Recent Advances in Audio Generation AI

- Music generation



The screenshot shows the Suno AI web interface. On the left is a dark sidebar with navigation options: Home, Create, Library, Explore (with a BETA badge), Search, 10 credits, Subscribe, What's New? (with 18 notifications), and Help. The main area is titled 'Custom Mode' with a 'v3.5' dropdown. It features a 'Lyrics' section with a text area containing Korean lyrics for a song. Below the lyrics is a 'Generate Lyrics' button and a character count '292/3000'. There is also an 'Instrumental' toggle and a 'Style of Music' section with a text input 'male vocal, hard rock, guitar, exciting.' and several genre tags: rock, drum, metal, bass, industrial, and folk. A character count '43/120' is visible at the bottom right of the style section.

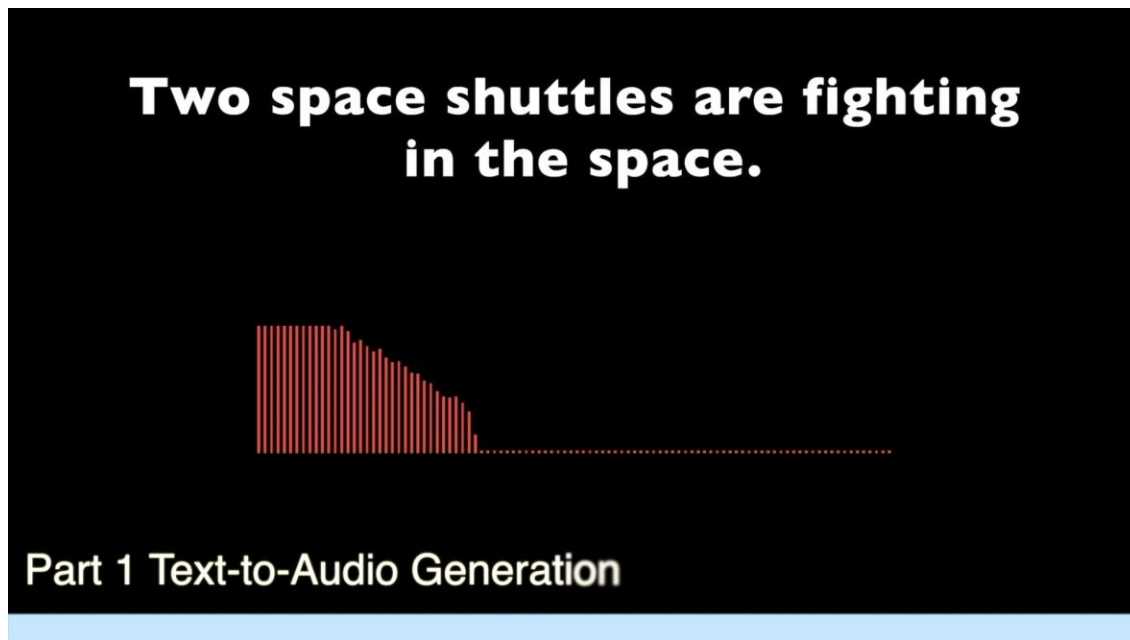


The image shows a generated music card for 'KAIST CT'. At the top is a colorful, abstract geometric pattern. Below it, the title 'KAIST CT' is displayed in bold, followed by the genre 'male vocal, hard rock, guita...'. The lyrics are shown in two sections: '[Verse]' and '[Verse 2]', both containing Korean text. The Suno logo is in the bottom right corner.

<https://suno.com/>

# Recent Advances in Audio Generation AI

- General audio (environmental sound, speech, music)

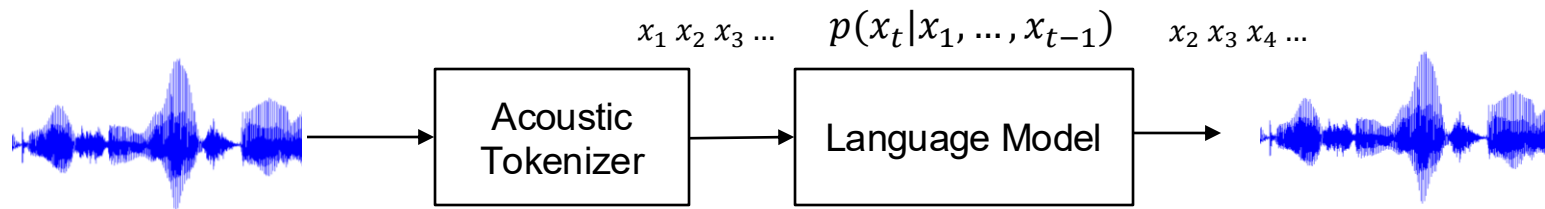


AudioLDM

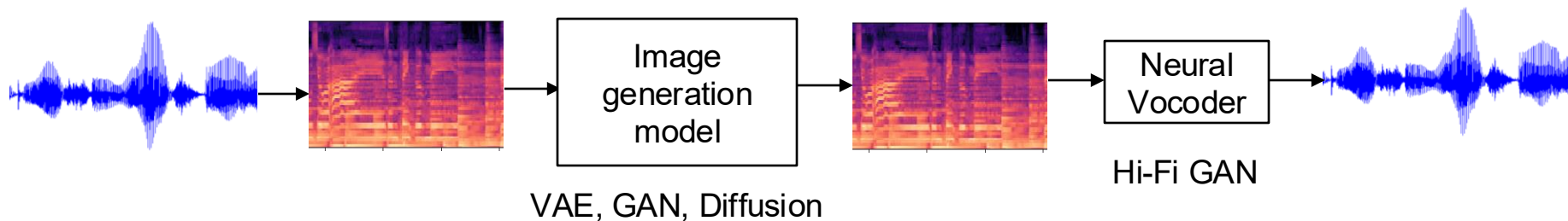
<https://audioldm.github.io/>

# Overview of Audio Generation AI Models

- Audio language model (1D approach)
  - Discrete acoustic tokens + Language Model (Transform Decoder)



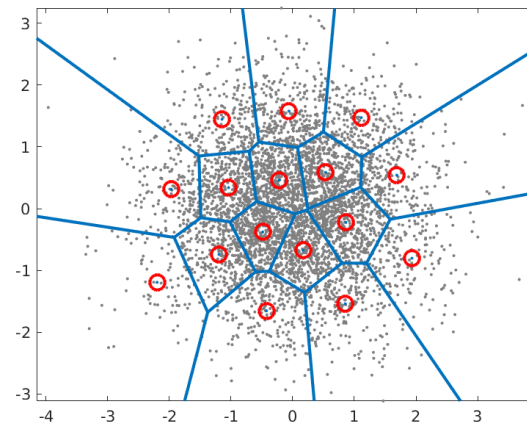
- Spectrogram generation model (2D approach)
  - Mel-spectrogram + Image Generation Model



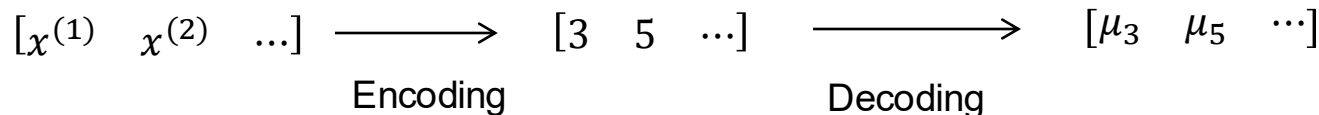
# Audio Discretization Using K-means (Week 2 Slides)

- Vector Quantization

- The set of cluster centers is called “codebook”:
- Encoding a sample vector to a single scalar value of “codebook index” (membership index)
- The compressed data can be reconstructed using the codebook
- Example: CELP (Code-Excited Linear Prediction)
  - A component of speech sound is vector-quantized and the codebook index is transmitted in the speech communication



Example of a codebook for a 2D Gaussian with 16 code vectors



- Variational auto-encoder with a vector-quantized latent space

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \text{sg}[e]\|_2^2$$

### Reconstruction loss

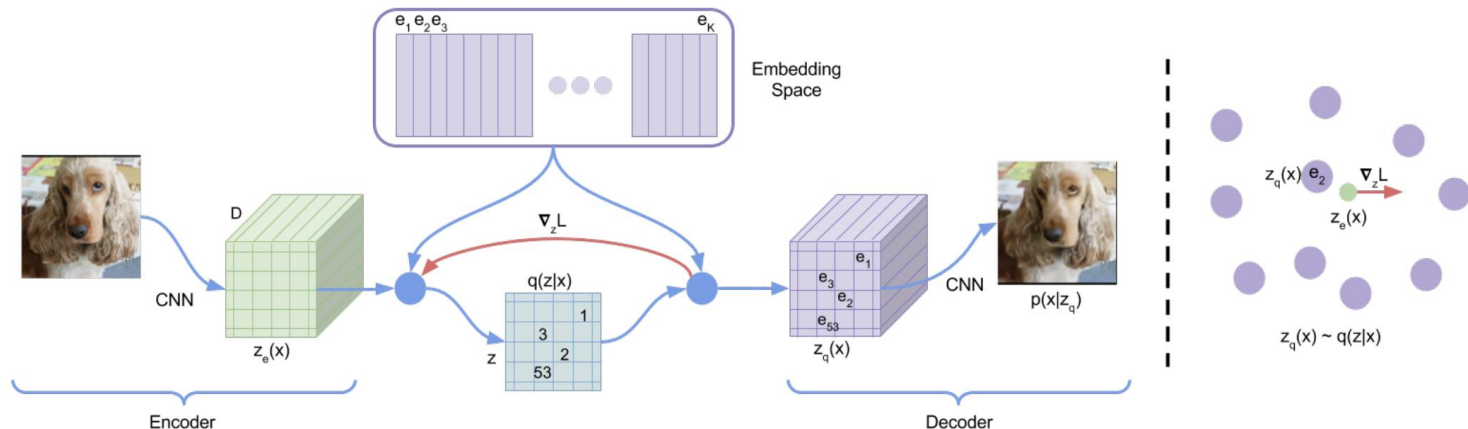
Update the encoder  
And decoder

### VQ loss

Update the codebook  
towards the encoder output

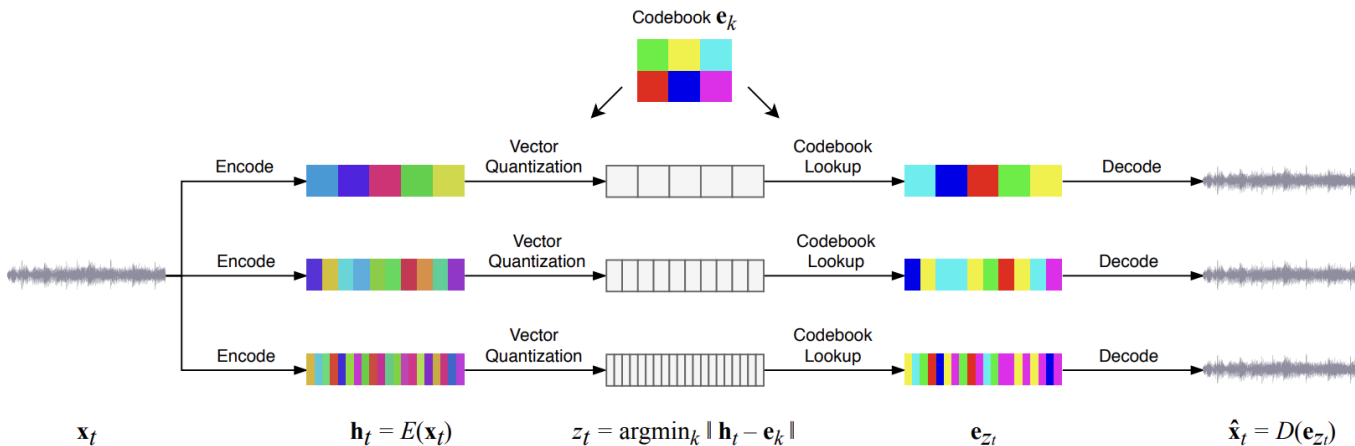
### Commitment loss

Move the encoder outputs  
towards the codebook  
for stable training



# JukeBox



- Use three levels of hierarchical VQ-VAE for music generation
  - The codebook sizes of the top, middle, bottom VQ-VAEs are the same
  - Upper levels take longer audio segments for quantization, learn more musical information, while having worse reconstruction audio quality
  - About 7,234 tokens per second




# JukeBox


- Demo: <https://openai.com/index/jukebox/>

Unseen lyrics | T = 0.995 Share on Twitter | Facebook | Soundcloud | Copy link

 **OpenAI**  
Classic Pop, in the style of Frank Sinatra - Jukebox  SOUNDCLLOUD



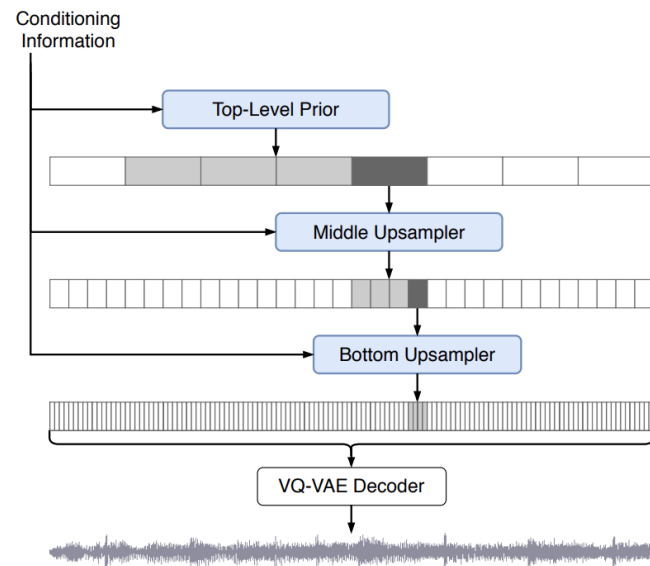
Privacy policy ▶ 443



When I  
Wake up  
And look up  
I dreamed that my model  
could reason  
When I  
Go out  
And look around  
I really hope that my model  
Reasons

# JukeBox

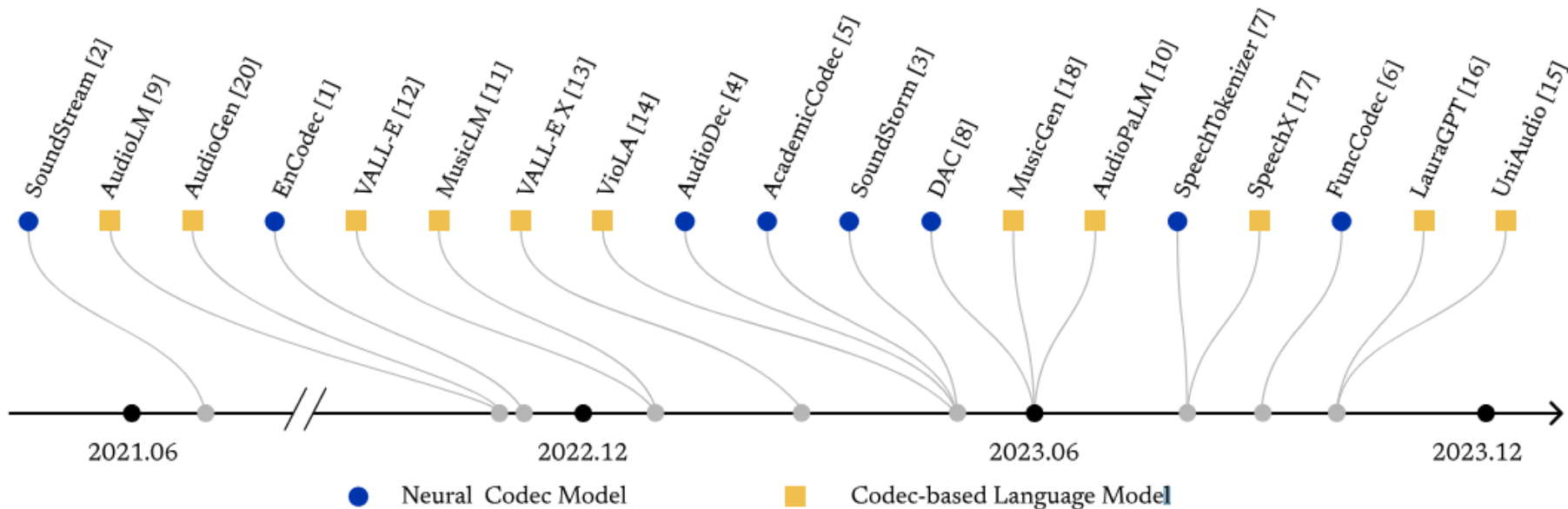
- Hierarchical language model for music generation
  - Each level has its own language modeling: transformer decoder
  - Upper-level tokens are upsampled (via residual WaveNet+ transposed Conv) and used for condition lower-level modeling
- Input conditioning
  - Artist., Genre, Time duration
  - Lyrics
    - Vocal separation + audio-to-text alignment
    - Lyrics embedding
    - Cross-attention to top-level



- Issues

- 1.2 M songs as training data
- Much resource in training
  - 512 V100s, total training time >6 weeks (around >60 years of single V100 time)
- Slow inference time
  - 1 hour to generate 1 min of top level tokens
  - 8 hours to upsample one minute of top level token
- Limited controllability

# Audio language model: Timeline



# Audio Codec

- Compress audio: wav → mp3
  - Fidelity vs bitrate, model complexity, computational efficiency
- Traditional audio codec
  - Signal processing: filterbanks + Modified DCT
  - Frequency-wise SNR based on psychoacoustic model (masking)  
→ bit allocation
  - Lossless compression (e.g. Huffman coding)
- Neural audio codec
  - Use neural networks to convert raw waveform into discrete acoustic tokens

# Soundstream

- Neural Audio Codec: AE + residual vector quantization (RVQ)
  - Recursive VQ for residuals: coarse quantization → fine quantization

Discriminator loss

$$\mathcal{L}_{\mathcal{D}} = E_x \left[ \frac{1}{K} \sum_k \frac{1}{T_k} \sum_t \max(0, 1 - \mathcal{D}_{k,t}(x)) \right] + E_x \left[ \frac{1}{K} \sum_k \frac{1}{T_k} \sum_t \max(0, 1 + \mathcal{D}_{k,t}(\mathcal{G}(x))) \right]$$

Adversarial loss

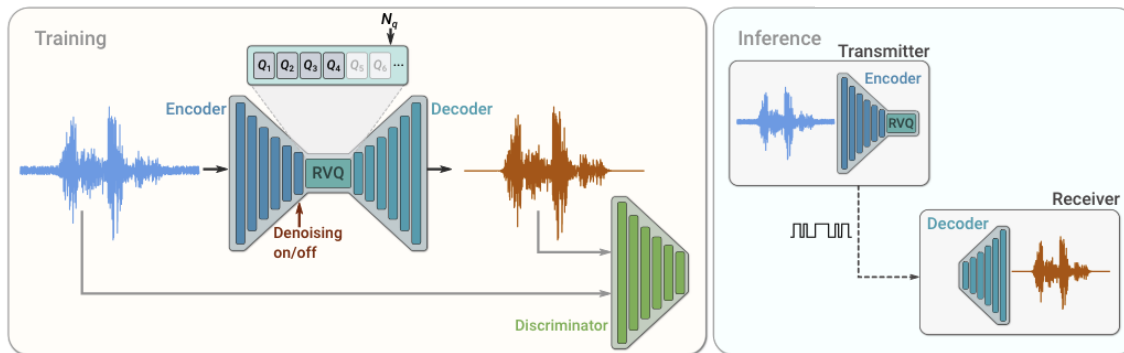
$$\mathcal{L}_{\mathcal{G}}^{\text{adv}} = E_x \left[ \frac{1}{K} \sum_{k,t} \frac{1}{T_k} \max(0, 1 - \mathcal{D}_{k,t}(\mathcal{G}(x))) \right]$$

Feature loss

$$\mathcal{L}_{\mathcal{G}}^{\text{feat}} = E_x \left[ \frac{1}{KL} \sum_{k,l} \frac{1}{T_{k,l}} \sum_t \left| \mathcal{D}_{k,t}^{(l)}(x) - \mathcal{D}_{k,t}^{(l)}(\mathcal{G}(x)) \right| \right]$$

Multi-scale spectral  
Reconstruction loss

$$\mathcal{L}_{\mathcal{G}}^{\text{rec}} = \sum_{s \in 2^0, \dots, 2^{11}} \sum_t \|\mathcal{S}_t^s(x) - \mathcal{S}_t^s(\mathcal{G}(x))\|_1 + \alpha_s \sum_t \|\log \mathcal{S}_t^s(x) - \log \mathcal{S}_t^s(\mathcal{G}(x))\|_2,$$



$$\mathcal{L}_{\mathcal{G}} = \lambda_{\text{adv}} \mathcal{L}_{\mathcal{G}}^{\text{adv}} + \lambda_{\text{feat}} \cdot \mathcal{L}_{\mathcal{G}}^{\text{feat}} + \lambda_{\text{rec}} \cdot \mathcal{L}_{\mathcal{G}}^{\text{rec}}$$

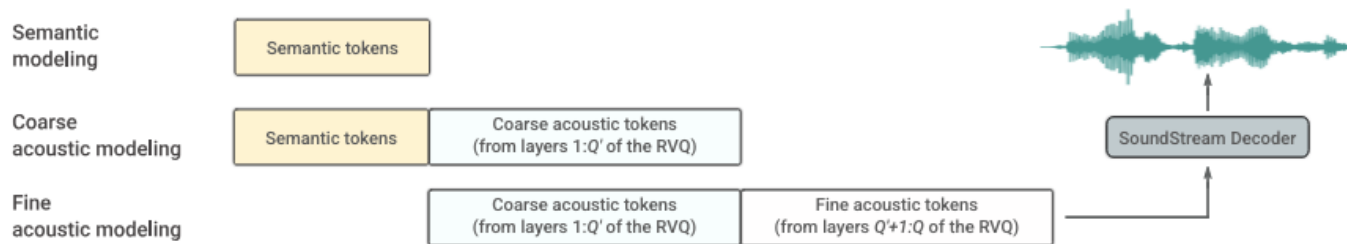
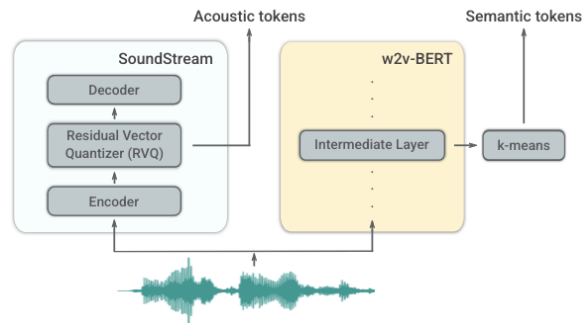
# AudioLM

- Two levels of tokens
  - Semantic token: long-term coherence
  - Acoustic token: audio compression

- Hierarchical language model

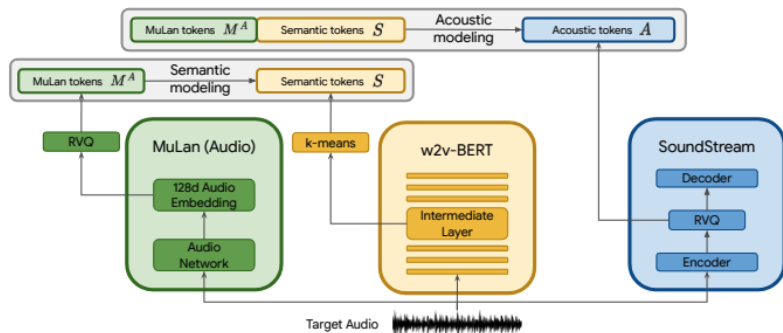
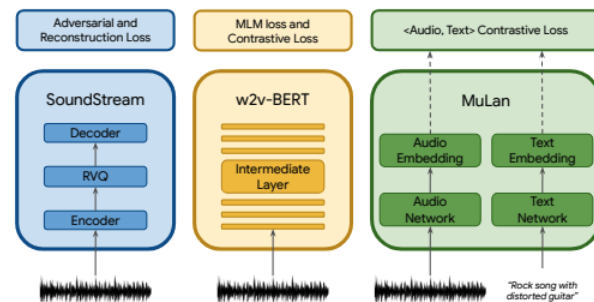
- Semantic modeling:  $p(z_t | z_{<t})$
- Acoustic modeling:

- Coarse  $p(y_t^q | z, y_{<t}^{\leq Q'}, y_t^{< q})$  ( $q \leq Q'$ ) + fine  $p(y_t^q | y^{\leq Q'}, y_{<t}^{> Q'}, y_t^{< q})$  ( $q > Q'$ )

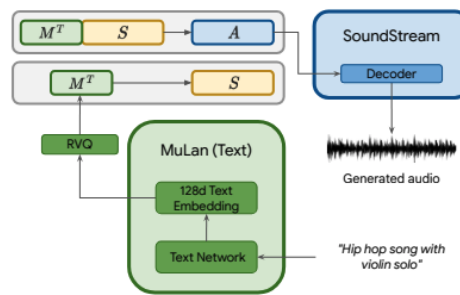


# MusicLM

- Add “audio-text joint embedding” (MuLan) for text conditioning
- Hierarchical autoregressive generation
  - Semantic modeling:  $p(S_t|S_{<t}, M_A)$
  - Acoustic modeling:  $p(A_t|A_{<t}, S, M_A)$  (RVQ: course + fine)



Training (with audio)



Inference (with text)

<https://google-research.github.io/seanet/musiclm/examples/>

<https://aitestkitchen.withgoogle.com/ko/tools/music-fx>

- Evaluation

- MusicCaps dataset: 5.5k high-quality music and text description pairs
- Fréchet Audio Distance (FAD): the distance between two Gaussian distributions given a pretrained audio embedding model (e.g., VGGish, PANN)

evaluation set embeddings  $\mathcal{N}_e(\mu_e, \Sigma_e)$  (model)

background embeddings  $\mathcal{N}_b(\mu_b, \Sigma_b)$  (ground truth)

  $\mathbf{F}(\mathcal{N}_b, \mathcal{N}_e) = \|\mu_b - \mu_e\|^2 + \text{tr}(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e})$

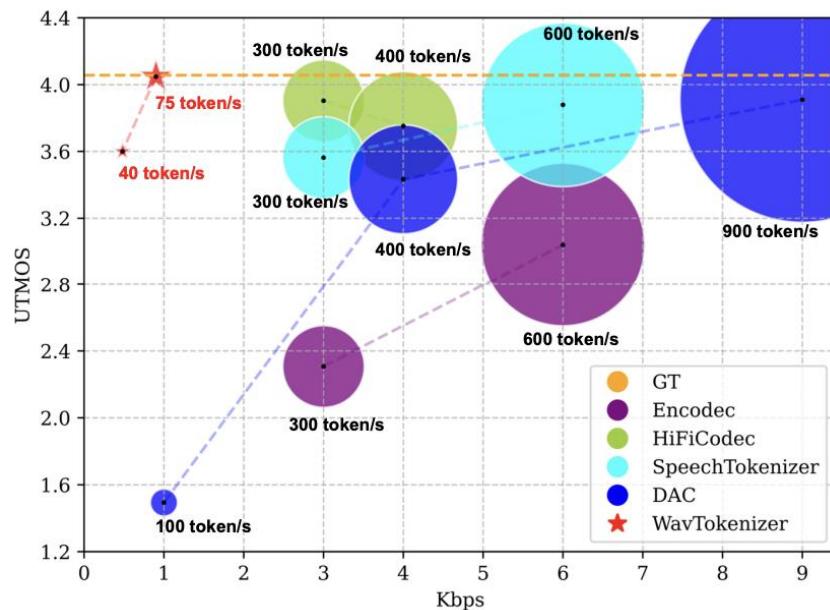
- Other objective metrics: Inception Score (IS)  $\rightarrow$  diversity

- Issues

- Slow inference: 625 tokens per second  
(acoustic tokens: 600, semantic tokens: 25)
- Limited controllability: text prompt, melody (need an additional module)
- High training cost

# Challenges in Audio language model

- More efficient acoustic tokenizer
  - This will become more important for the data storage issue
  - Encodec, DAC, WavTokenizer, Finite Scalar Quantization (FAQ)

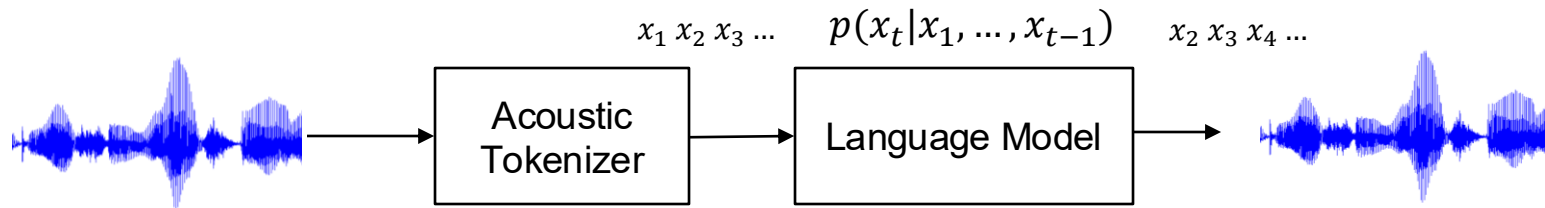


# Challenges in Audio language model

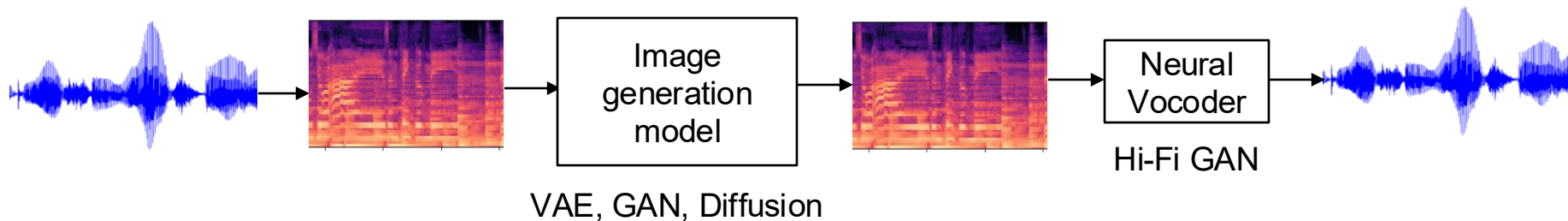
- More efficient acoustic tokenizer
  - This will become more important for the data storage issue
- Slow inference: autoregressive and complexity in transformer
  - Coarse-to-grained hierarchical modeling
  - Efficient attention modules: linear, local attention, pruning
  - Non-auto-regressive model: integration with diffusion models
- Limited conditioning by the 1D modeling: time-alignment and long-term dependency
  - Improved positional encoding
  - Coarse-to-grained hierarchical modeling

# Overview of Audio Generation AI Models

- Audio language model (1D approach)
  - Discrete acoustic tokens + Language Model (Transform Decoder)

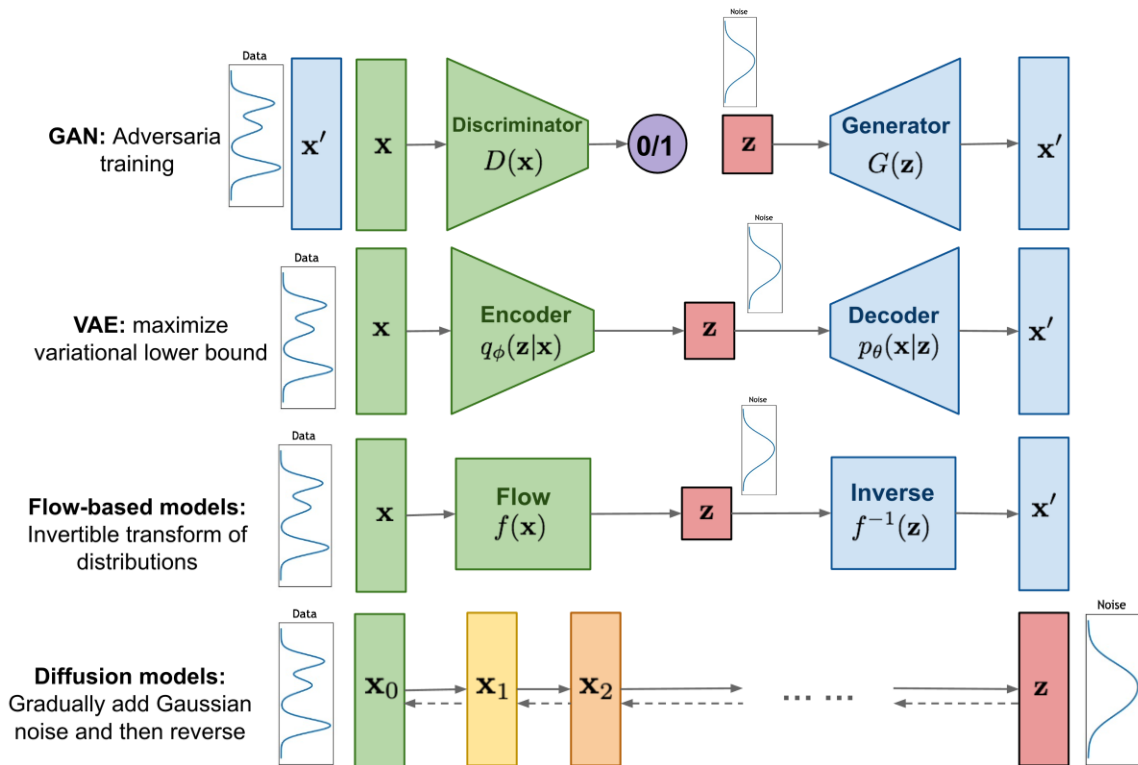


- Spectrogram generation model (2D approach)
  - Spectrogram + Image Generation Model



# Generative Models for Images (Parallel Inference)

- Models
  - Generative GAN
  - VAE
  - Flow-based models
  - Diffusion models
- Use spectrogram as an image



# Diffusion Models

- Powerful generative model, outperforming GANs

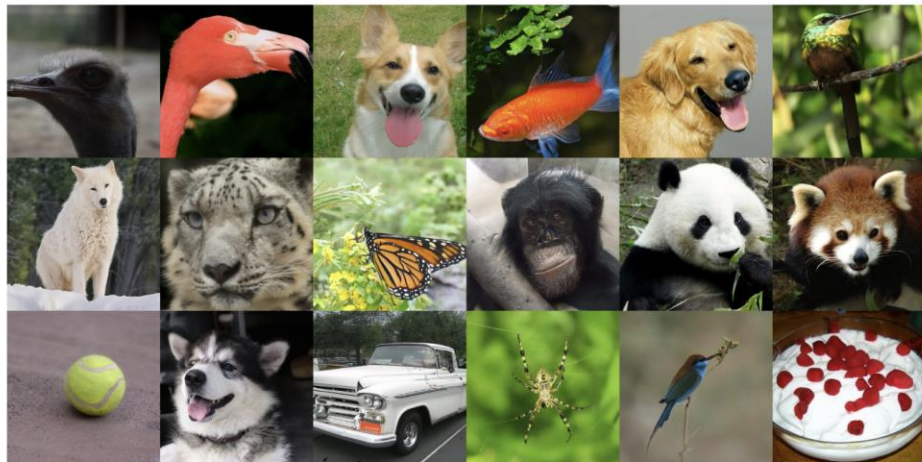


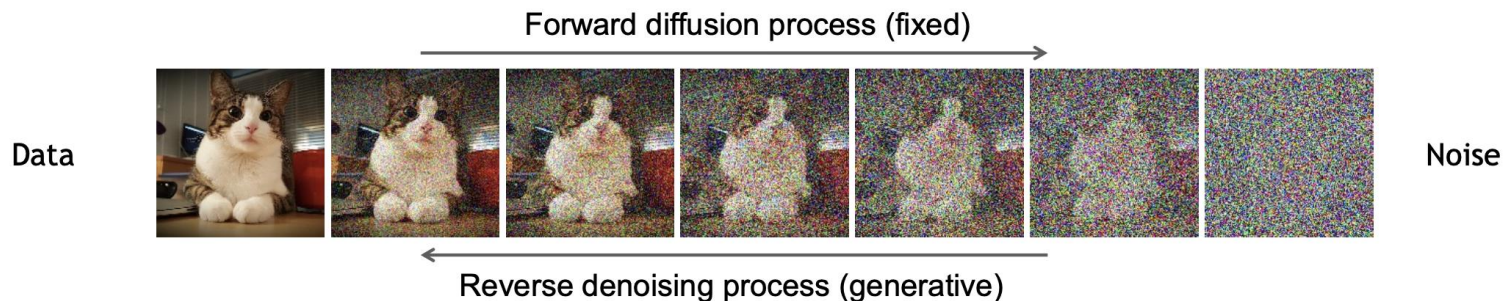
Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)



Stable Diffusion

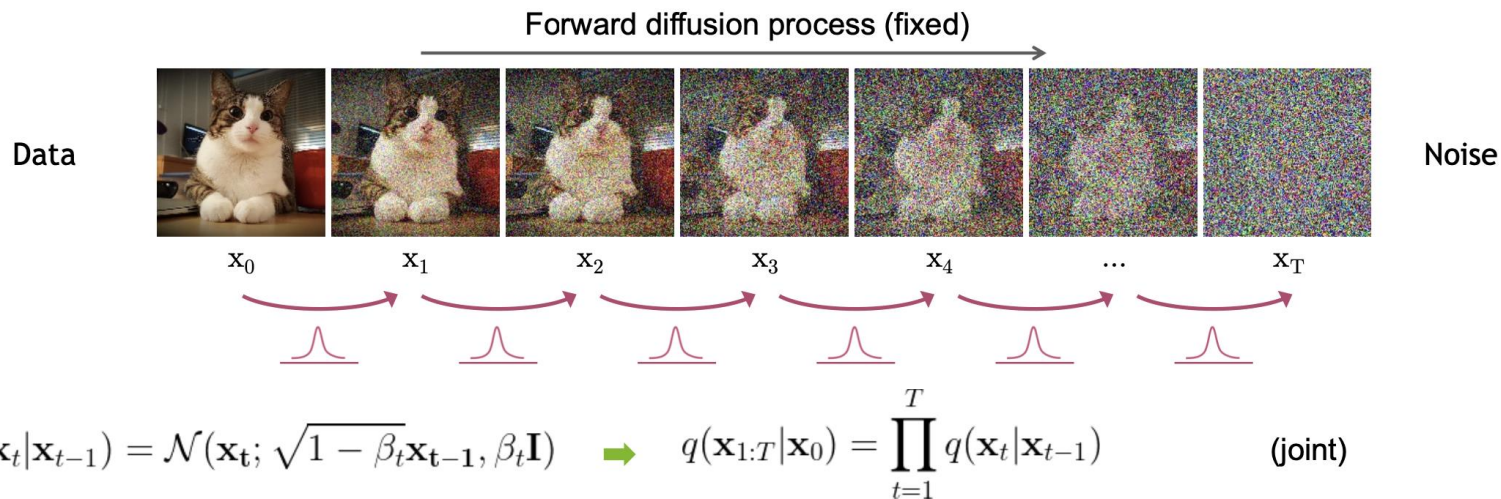
# Diffusion Models

- Learning to generate by denoising
  - Forward diffusion process: gradually add noise to input
  - Reverse denoising process: learn to generate data by denoising



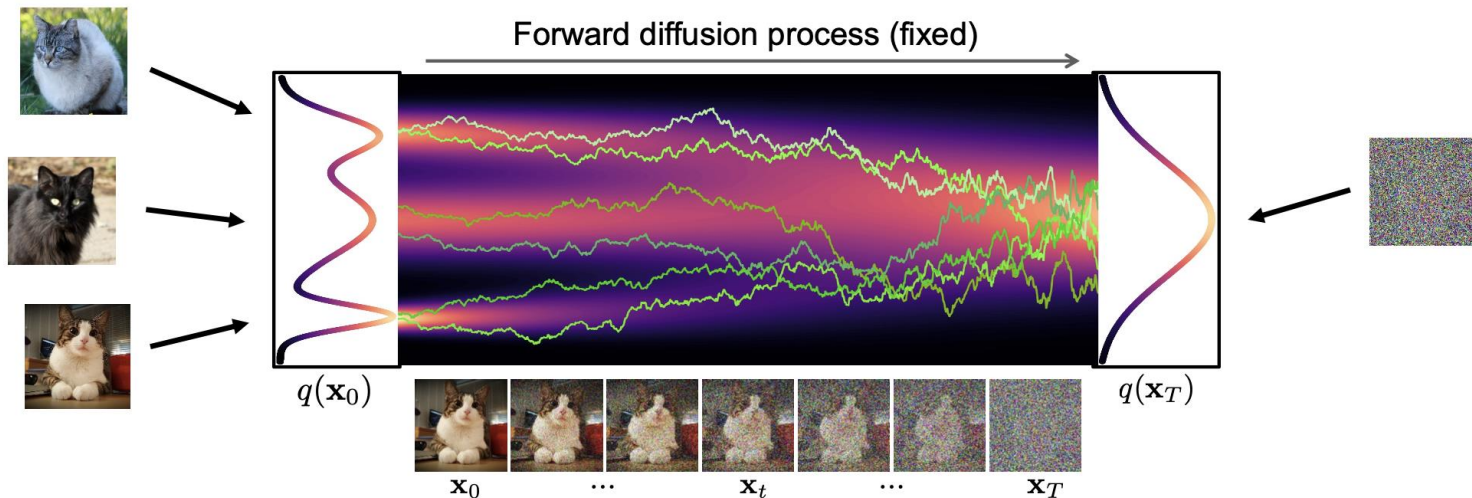
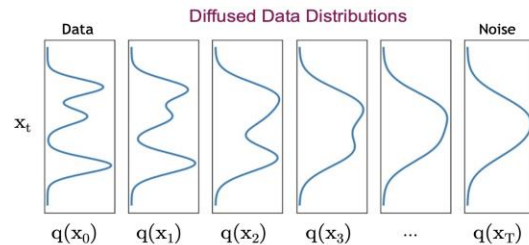
# Forward Diffusion Process

- Gradually add Gaussian noises to input in T steps
  - The result approximates to a Gaussian distribution



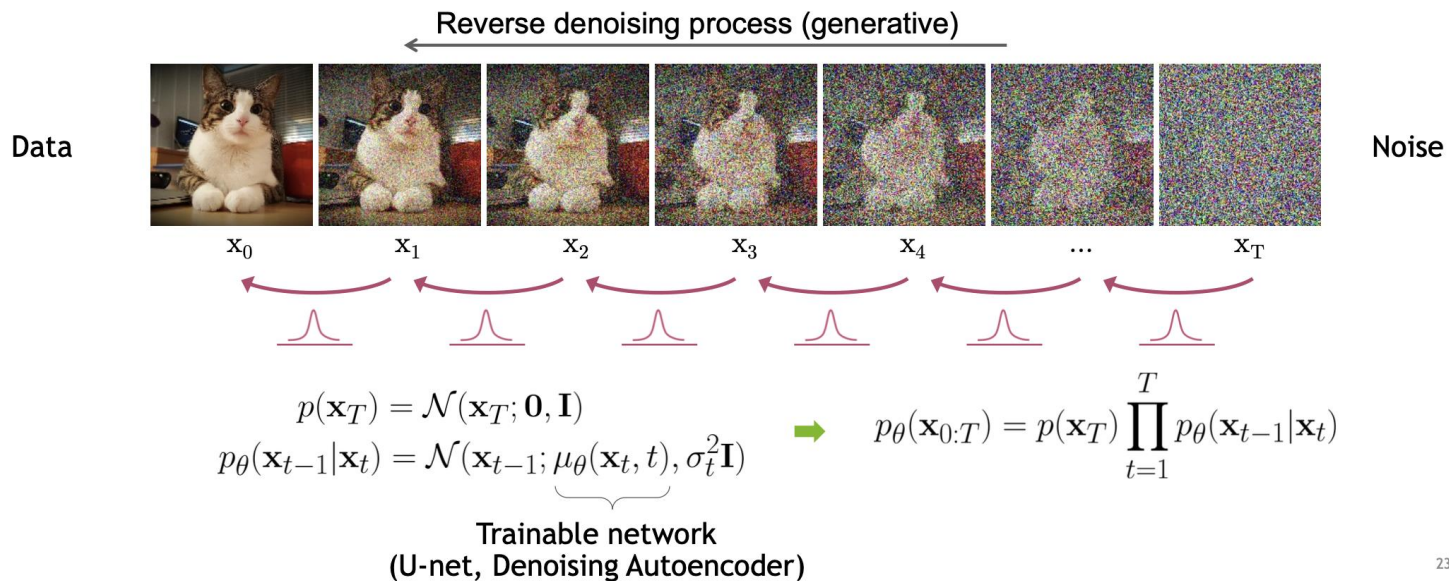
# Forward Diffusion Process

- Gradually add Gaussian noises to input in  $T$  steps
  - The result approximates to a Gaussian distribution



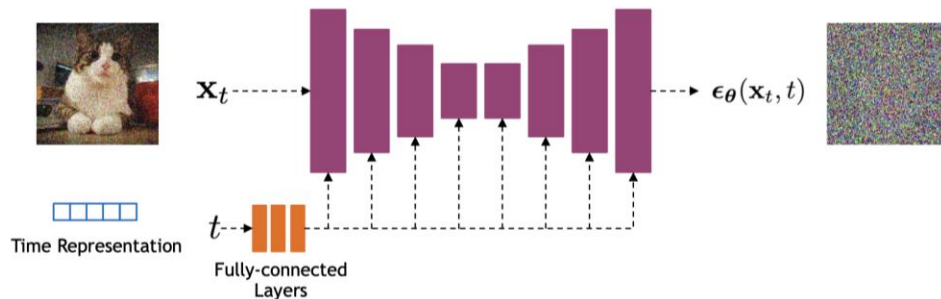
# Reverse Diffusion Process

- Learn to generate data by denoising
  - Start from Gaussian noise



# Algorithm

- Denoising auto-encoder
  - Time features (positional embeddings) are added to the residual blocks using simple spatial addition or using adaptive group normalization layers.



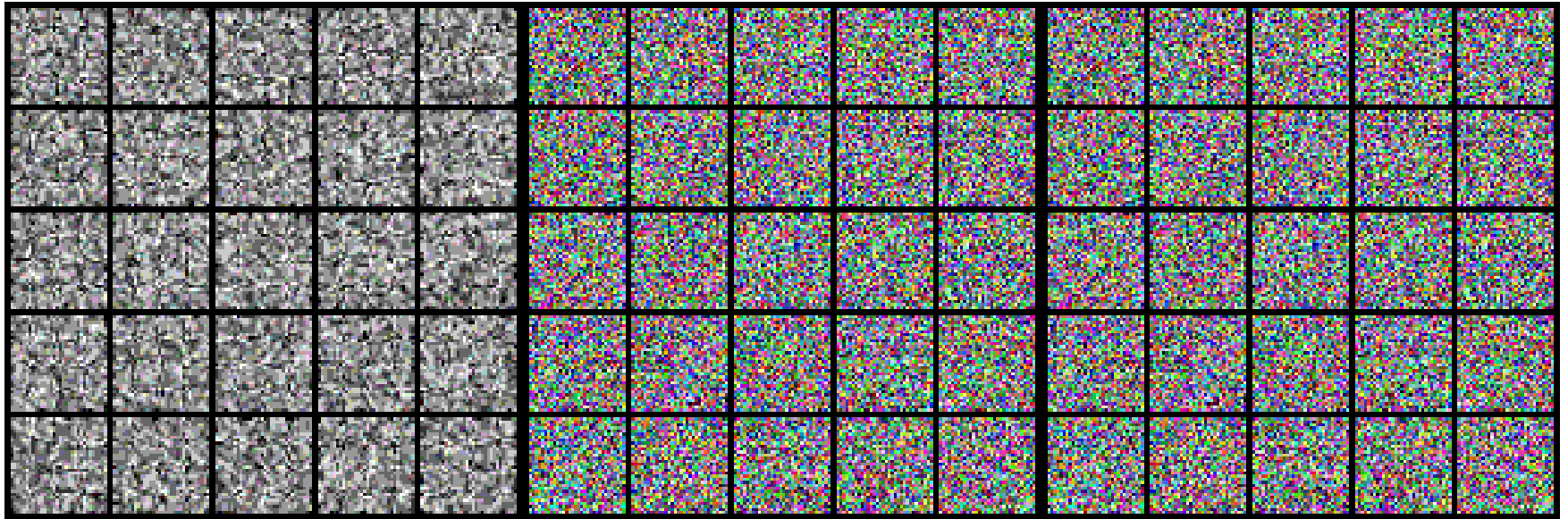
## Algorithm 1 Training

- 1: **repeat**
- 2:  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3:  $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4:  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on  
$$\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2$$
- 6: **until** converged

## Algorithm 2 Sampling

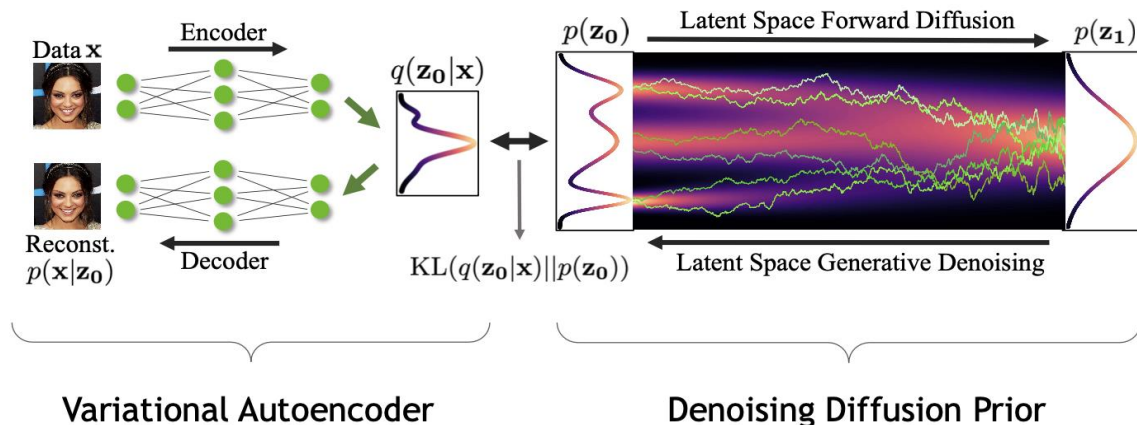
- 1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 2: **for**  $t = T, \dots, 1$  **do**
- 3:  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 4:  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return**  $\mathbf{x}_0$

# Examples



# Latent Diffusion Models (LDM)

- Use VAE to map the input data to an embedding space
- Denoising diffusion models are applied to the latent space
  - The distribution of embedding space is close to Normal distribution
  - Denoising becomes simpler and fast synthesis
  - The VAE can be tailored to the type of data (graphs, text, 3D data)



# AudioLDM

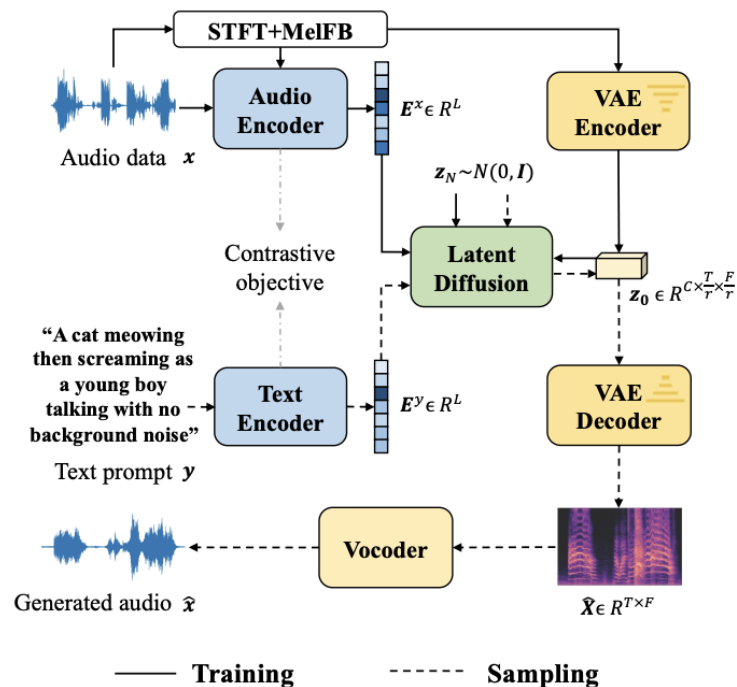
- Text-to-Audio generation model using LDM
  - Use VAE on spectrogram to obtain latent space and HiFi-GAN for vocoder
  - CLAP (audio-text joint embedding model) for text conditioning
  - The denoising AE takes the audio encoder output of CLAP additionally ( $E^x$ )

$$L_n(\theta) = \mathbb{E}_{z_0, \epsilon, n} \|\epsilon - \epsilon_\theta(z_n, n, E^x)\|_2^2$$

- The reverse process starts with the text encoder output of CLAP

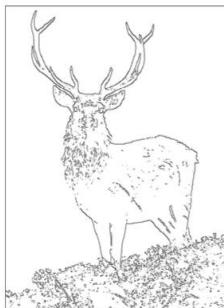
$$p_\theta(z_{n-1} | z_n, E^y) = \mathcal{N}(z_{n-1}; \mu_\theta(z_n, n, E^y), \sigma_n^2 I)$$

<https://audioldm.github.io/>



# ControlNet

- Allow pixel-level control (sketch, pose) for diffusion-based generation



Input Canny edge

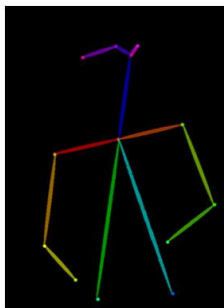


Default



“masterpiece of fairy tale, giant deer, golden antlers”

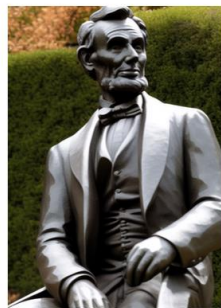
“..., quaint city Galic”



Input human pose



Default

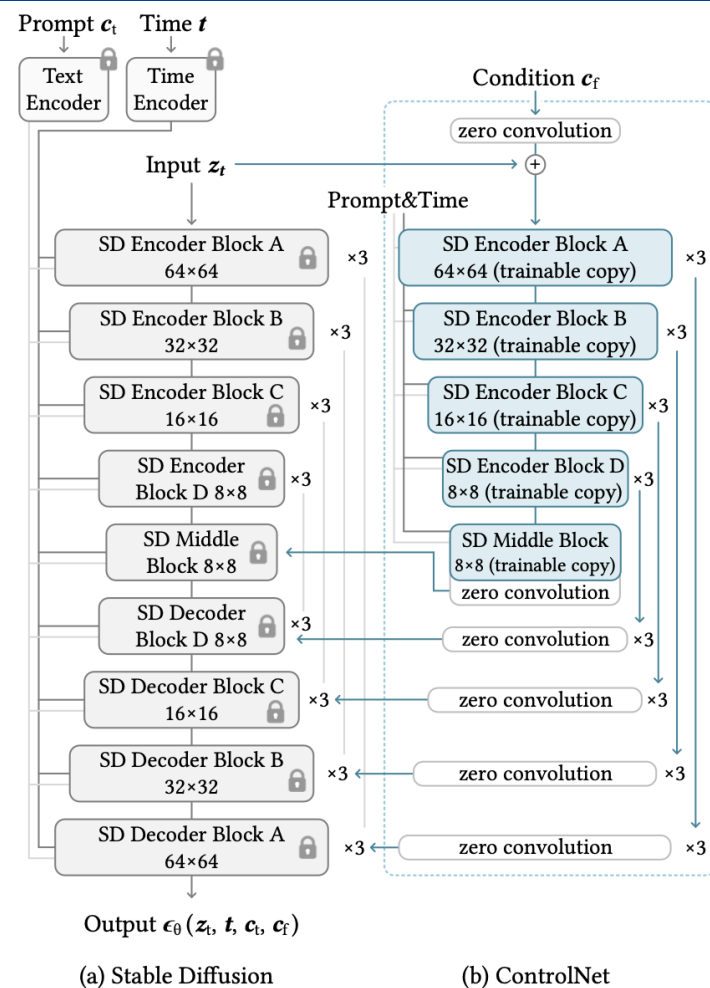
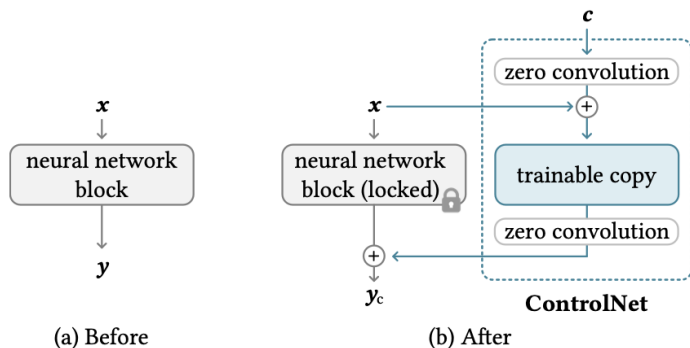


“chef in kitchen”

“Lincoln statue”

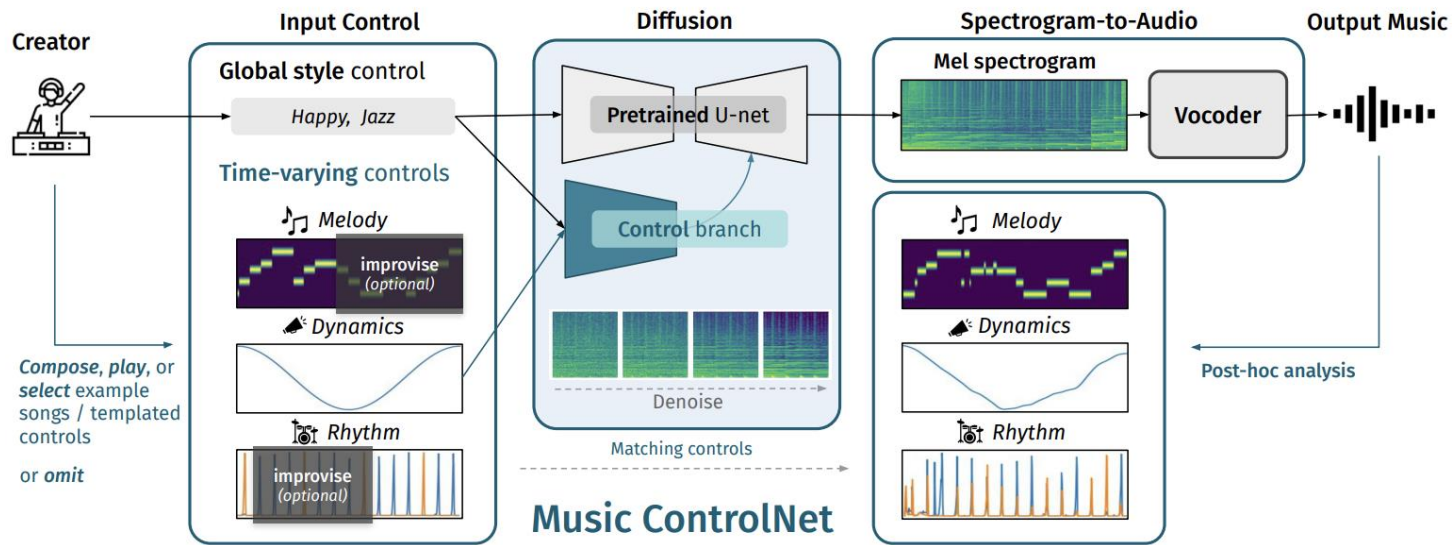
# ControlNet

- Use a pretrained diffusion model
  - E.g., Stable Diffusion (SD)
  - Copy the encoder to ControlNet
  - Build the decoder with “zero convolution” and use it to control the SD decoder
  - Zero convolution: 1x1 conv initialized with zero weight and bias ( $y_c = y$  at  $t = 0$ )



# Music ControlNet

- Allow temporal music features as a time-varying control
  - Melody (chroma), dynamic curve, beat position



## Music ControlNet: Multiple Time-varying Controls for Music Generation

Demo Video

Shih-Lun Wu,<sup>1,2\*</sup> Chris Donahue,<sup>1</sup> Shinji Watanabe,<sup>1</sup> & Nicholas J. Bryan<sup>2</sup>

<sup>1</sup>School of Computer Science, Carnegie Mellon University

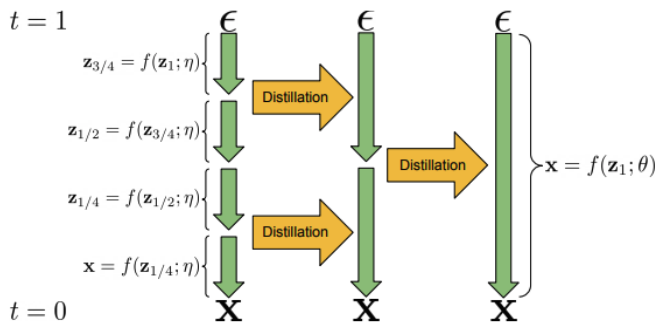
<sup>2</sup>Adobe Research

\*Work done during an internship at Adobe Research.

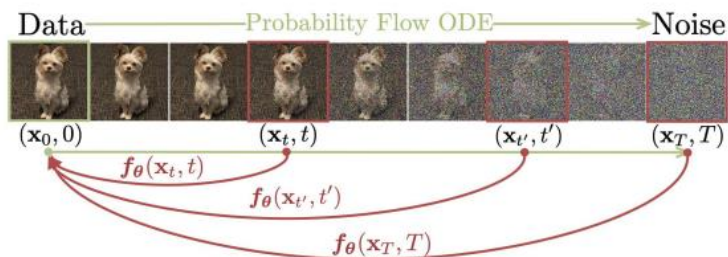
<https://musiccontrolnet.github.io/web/>

# Challenges in Diffusion Models

- Slow inference by the iterative denoising process
  - Speed up the sampling: Denoising Diffusion Implicit Models(DDIMs)
  - Reduce the number of iteration
    - Progressive Distillation, Consistency Model
    - DITTO/ DITTO2: Diffusion Inference-Time T-Optimization



Progressive Distillation



Consistency Model