# Machine Learning for Music

Module 4: Audio-Domain Music Generation

# Hi! Who am I?



- Zachary Novack
  - 3rd year CSE PhD Student
  - Undergrad @ CMU
  - https://zacharynovack.github.io/
  - Florida native! 🐊
- Research-Wise
  - Audio-Domain Generation
  - Controllable/Efficient Gen.
  - Text-Audio Reasoning
- Music / Hobbies
  - Percussionist (and bad piano)
  - Grad student volleyball (come join!)
  - Teach competitive indoor percussion (POW Percussion)



# Music Generation

Slides mostly based on

**“Generative AI for Music and Audio Creation” (Michigan):**

https://hermandong.com/teaching/pat498_598_fall2024/

(though lots of other references as we go)

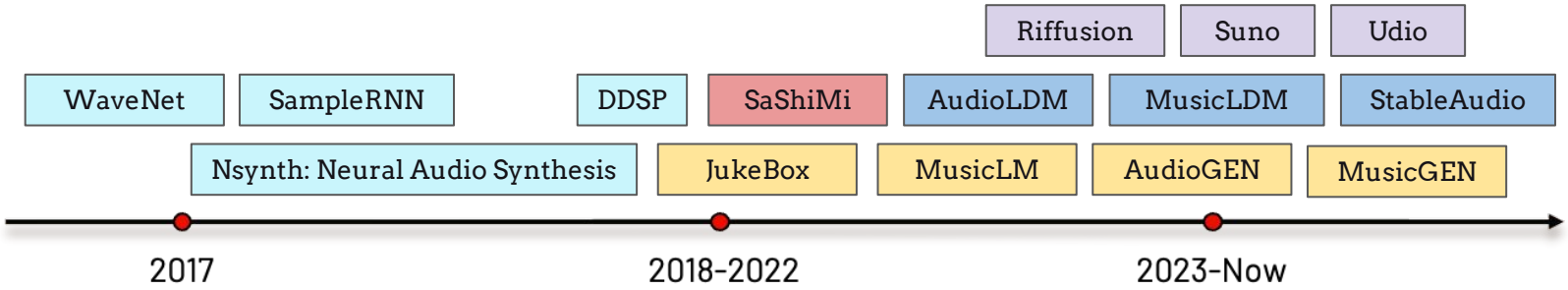
# Music Generation

We've spent the last ~two weeks discussing how music can be *generated* in the symbolic domain, but *listening* to our generations still requires running our outputs through a MIDI->Audio synthesizer

How can we generate musical audio **directly?**

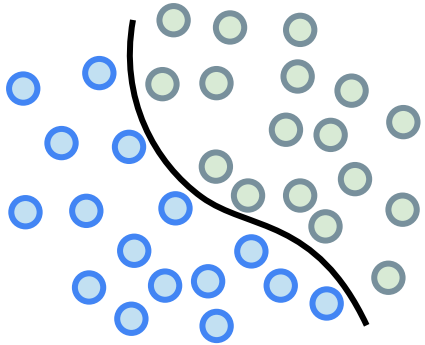
# Music Generation

## Audio Domain Music Generation



# Recall: *Discriminative* versus *Generative* AI

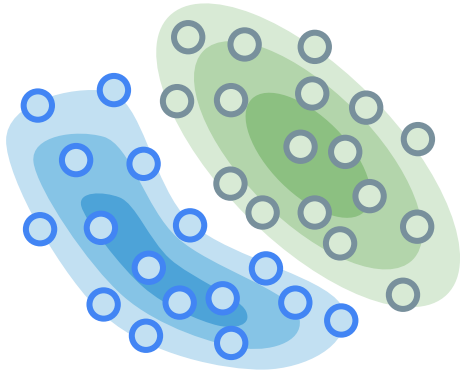
Discriminative



Discriminative models learn the **decision boundary**

$P(y|x)$

Generative



Generative models learn the underlying distribution

$P(x)$ or $P(x|y)$
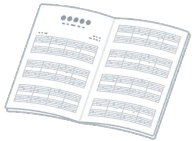
# Recall: *Discriminative* versus *Generative* AI

*Generative* AI learns the underlying distribution (x), so can “generate” new samples (x’s)



NVIDIA Fugatto (2024)
**(note: not symbolic)**

# Four paradigms of music generation



Symbolic music generation

(Module 3)

text

image

```
Program_change_0,
Note_on_60, Time_shift_2, Note_off_60,
Note_on_60, Time_shift_2, Note_off_60,
Note_on_76, Time_shift_2, Note_off_67,
Note_on_67, Time_shift_2, Note_off_67,
...
```

MIDI




piano roll



Audio-domain music generation

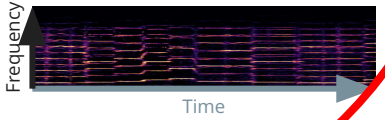
(Module 4)

time series

image



waveform




spectrogram

(certainly not a complete picture, but a rough way to characterize things for now)

# Module 4 Roadmap

- “Historical” Approaches to Audio Synthesis
  - More on this in Taylor’s guest lecture!
- “Old-School” AI Audio Generation
  - Models on Raw Audio, Models on Spectrograms
  - CNNs, RNNs, GANs, VAEs
- Audio **Representation Learning**
  - VAEs and GANs, but *not* for generation
  - Learning **quantized** vs. **continuous** representations
- Continuous-Modality Audio Generation
  - Intro to diffusion models (your homework!)
- Discrete-Modality Audio Generation
  - RNNs / Transformers on discrete representations

# How much DSP do I need to know??

- Is AI audio generation predicated on intense DSP knowledge??
- Am I cooked if I didn’t take signal processing classes??

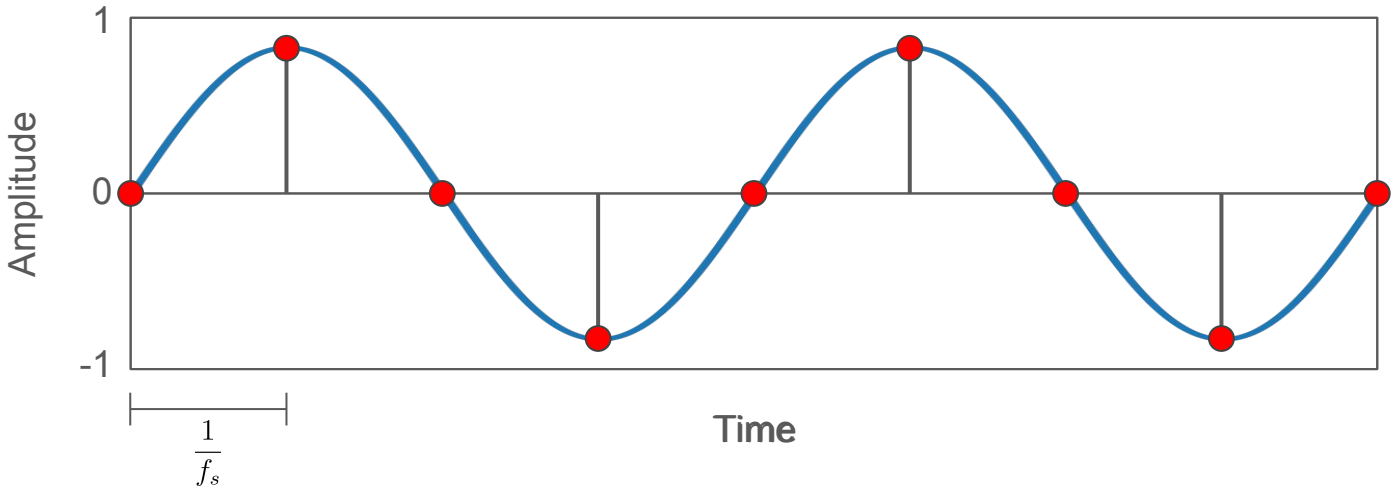
- Thankfully, NO!
- Audio-domain generation is driven largely by innovations in image and language, so as long as you understand the AI side of things, you can port things into audio and it mostly… just works
- You can mostly pick up the relevant DSP knowledge on the fly

# Audio-Domain Music Generation

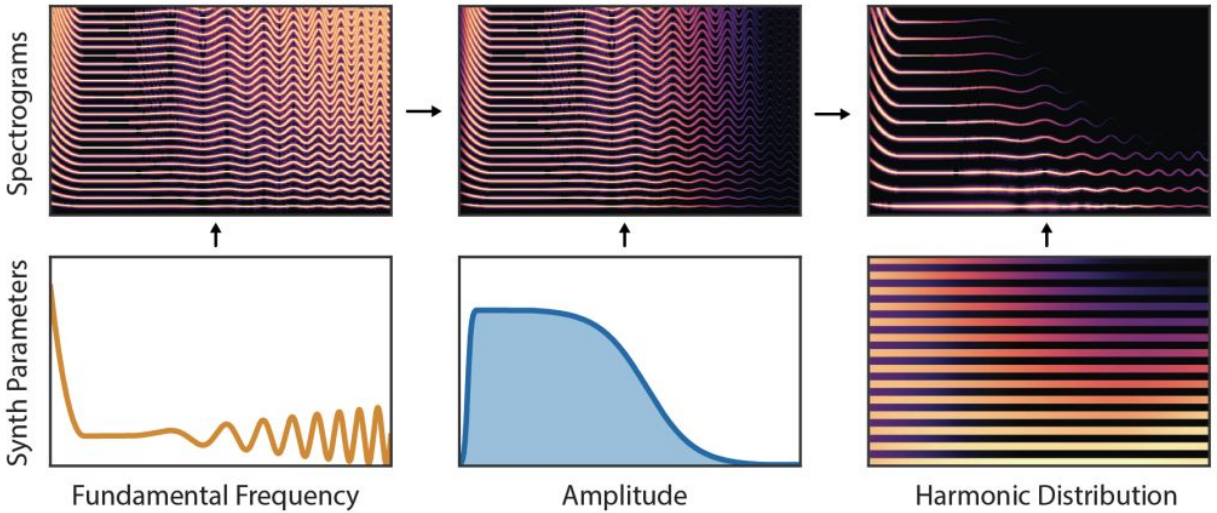
4.1: Historical Audio Synthesis

# We know how to synthesize a waveform directly...

- $a(t) = \sin(2\pi ft + \varphi)$
- To synthesize this at sampling rate $f_s$, we only need to calculate the value of this function at points $t = i / f_s$, where $i \in Z^+$



# Additive Synthesis

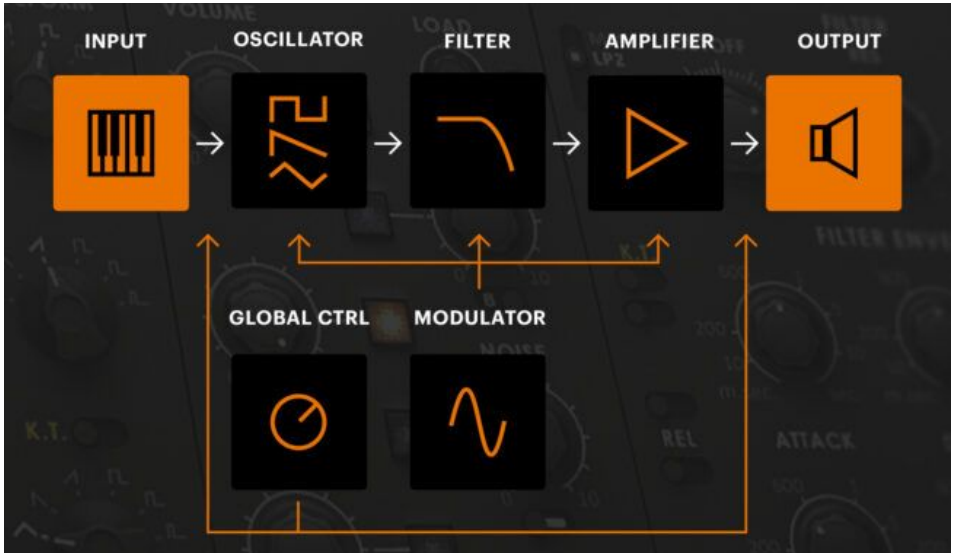



- We can stack sinusoids at harmonics (i.e. whole number) multiples of some fundamental frequency (pitch)!
- Then take amplitude envelope+harmonic dist. to shape the sound
- A lot of early synthesizers use this

*https://arxiv.org/pdf/2001.04643*

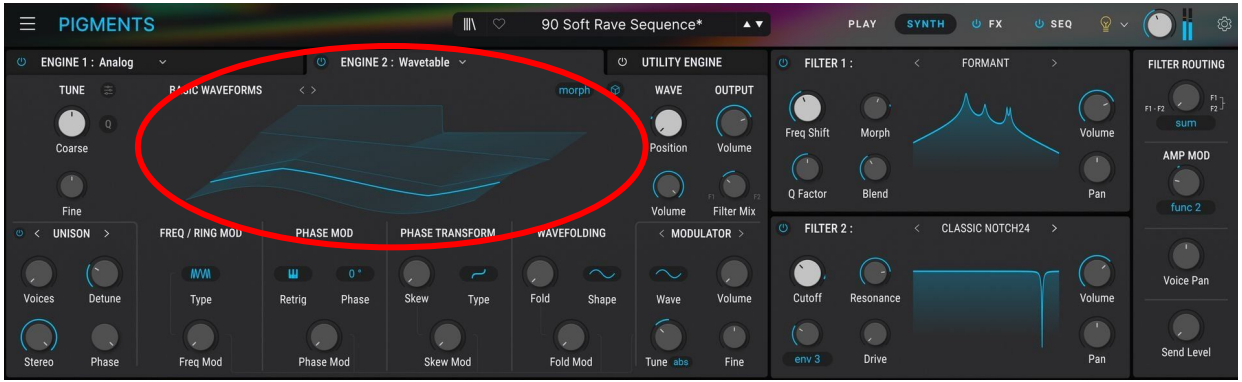
# Subtractive Synthesis

- Like additive, but backwards!
- Start w/complex waveform
  - (e.g. square, sawtooth, triangle)
- Use filter envelopes to attenuate spectral image of the waveform to your liking
- Cheaper than additive synthesis
- **VERY** popular in classical analog synthesis




*https://blog.native-instruments.com/subtractive-synthesis/*

# Wavetable Synthesis




- Lookup table of a bunch of single-period waveforms
- Synthesis by index into table and interpolating with waveforms nearby
- Super common, still used today!

*Arturia Pigments*

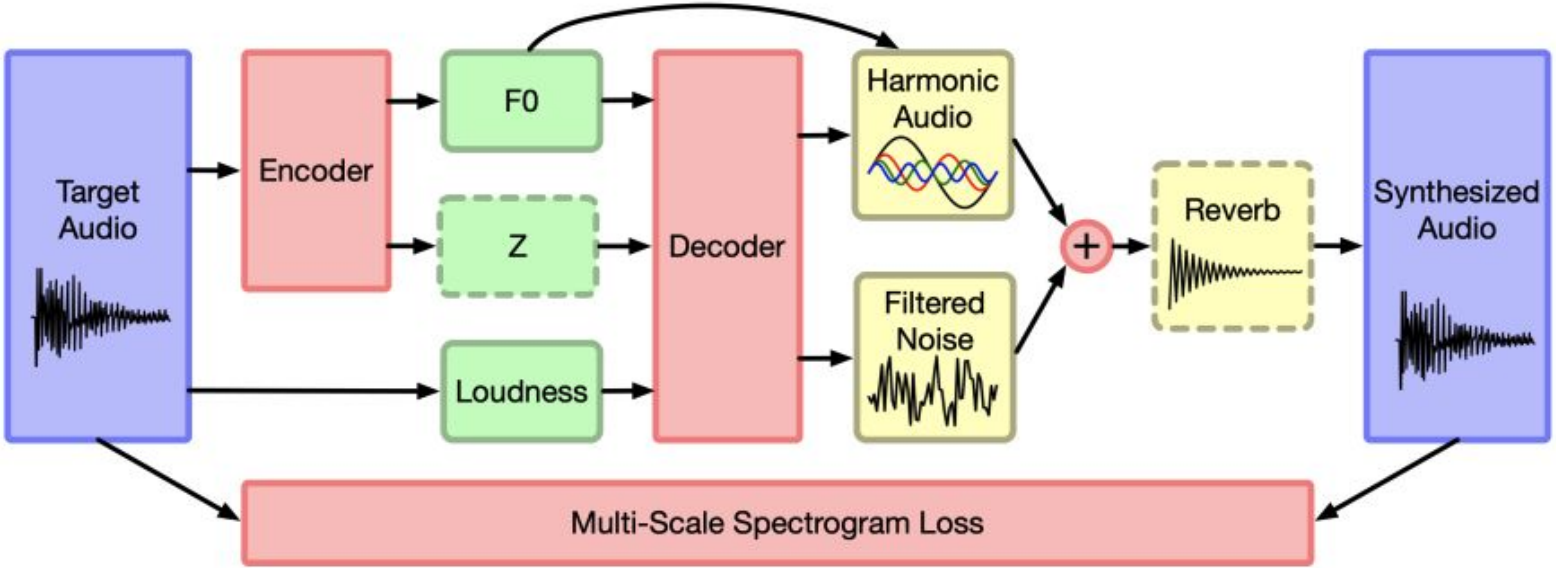
# Okay but...is this all relevant in 2025?

# Yes!

All of these type of synthesizers have some tunable parameters to them...

So we can learn these with the same methods we do for other AI tasks!

# Yes!

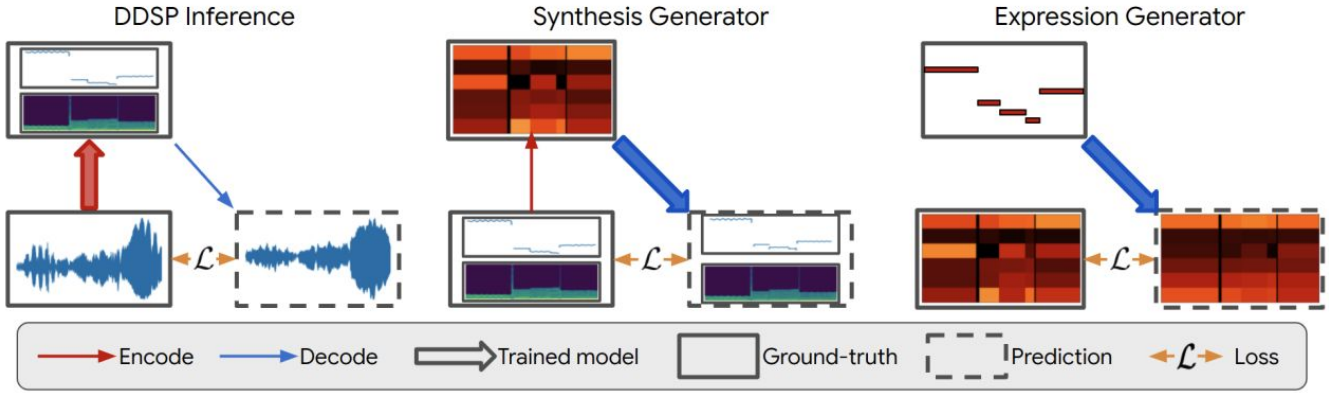



**DDSP:** *Differentiable* Digital Signal Processing

- Learn parameters of an additive synth to match input timbre

*https://arxiv.org/pdf/2001.04643*

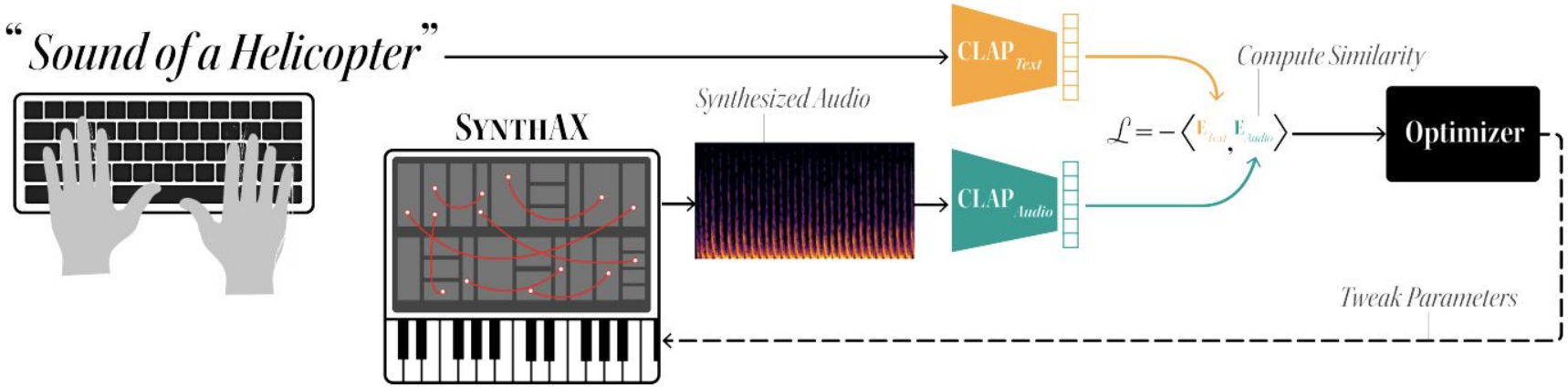
# Yes!




- **MIDI-DDSP**: predict expression parameters given MIDI notes, then predict synthesizer parameters given expression

*https://arxiv.org/pdf/2112.09312*

# Yes!




- **CTAG**: let text-audio similarity models (CLAP) assess "fitness" of synth parameters
- Use genetic algorithm to optimize synth parameters

https://arxiv.org/pdf/2406.00294

# Yaboi Hanoi – Entering Demons & Gods

**(2022)**




youtu.be/PbrRoR3nEVw



soundcloud.com/yaboihanoi/enter-demons-and-gods

# TLDR

We’re going over this briefly as it’s not the focus of the class, but important to understand there’s a long history of non-neural approaches to audio synthesis.

AND there are clear ways these techniques can be married with advances in AI

Taylor will talk about this more!

# Audio-Domain Music Generation

4.2: “Old-School” Audio Generation

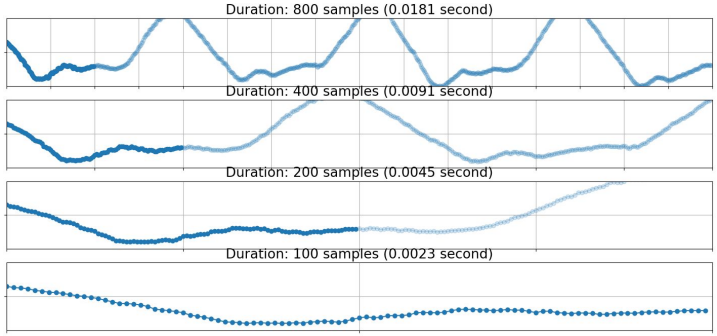
# The 2016-2020(ish) Era of Generative Audio

We've seen how CNNs, RNNs, VAEs, and GANs can be applied to generative models in the *symbolic* domain

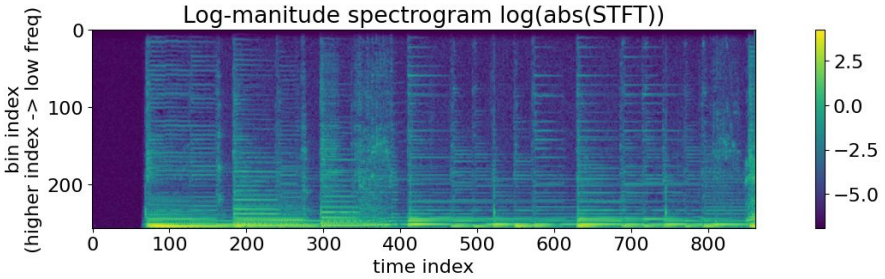
We've also seen the two most common representations for audio: **waveforms** (time-domain) and **spectrograms** (frequency domain)

This era is typified by the combination of these two broad ideas

# Waveform vs. Spectrograms




- Pros: Lossless, direct synthesis
- Cons: high-dimensional, unaligned with perception




- Pros: Compact, perceptual in nature, easy to port techniques from images
- Cons: Phase issues, reconstruction artifacts

# Direct Waveform Generation

# (Recap) Language Models (Mathematically)

- A class of machine learning models that learn the next **token probability** (e.g. word tokens, midi events, any *discrete* object)

$$p(\mathbf{x}_i \mid \mathbf{x}_{<i})$$

Next token

Previous tokens

# Autoregressive Models (Mathematically)

- A class of machine learning models that learn the probability of the **next value** given previous **values** (could be anything!)

$$p(\mathbf{x}_i \mid \mathbf{x}_{<i})$$

Next thing

Previous things

# Autoregressive Models

- So then there's no difference between autoregressive modeling of discrete vs. continuous objects?
- **NO!**

Discrete Data

$$p(\mathbf{x}_i \mid \mathbf{x}_{<i})$$

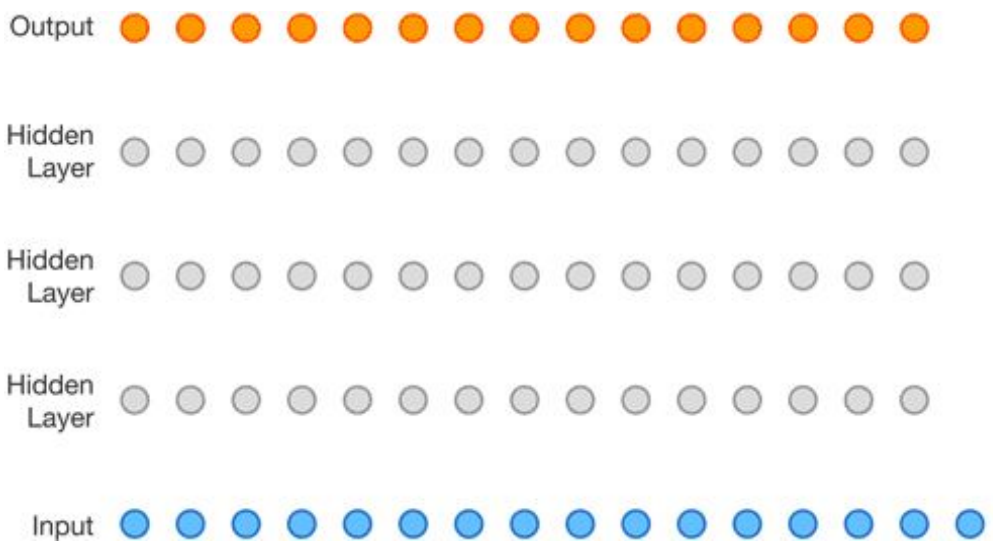
- Model outputs **probabilities** over **fixed vocabulary**
- To generate, (1) estimate probabilities, (2) **sample** from distribution

Continuous Data

$$p(\mathbf{x}_i \mid \mathbf{x}_{<i})$$

- Model can outputs **values directly** OR **probabilities** over **quantized values** (fixed vocab)
- Generation is either sampling (like discrete) OR regression (i.e. the direct output of the model *is* the prediction)

# Example: WaveNet (van den Oord et al., 2016)



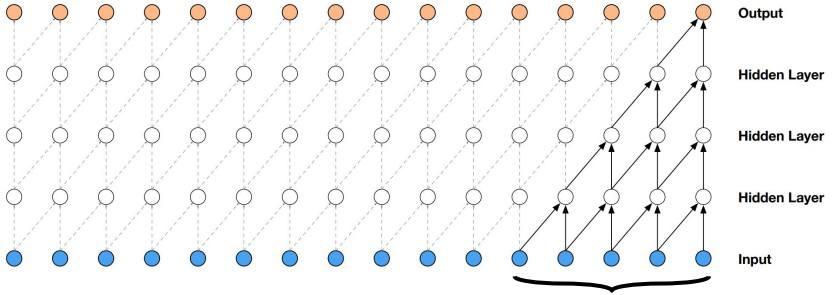

(Source: van den Oord et al., 2016)

**A convolutional neural network for raw waveform generation**

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *ICML*, 2016.

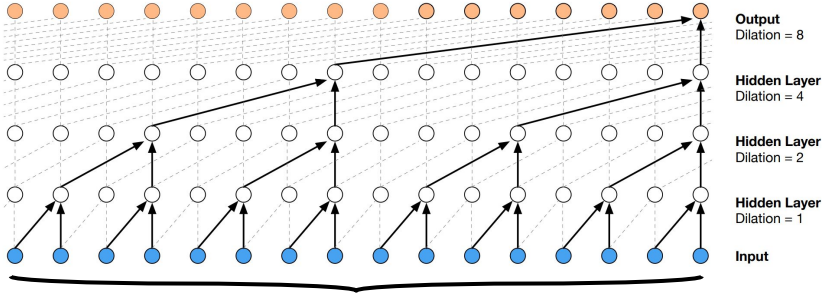
# Example: WaveNet (van den Oord et al., 2016)

**Standard convolution**




**Smaller "receptive field"**

**Dilated convolution**



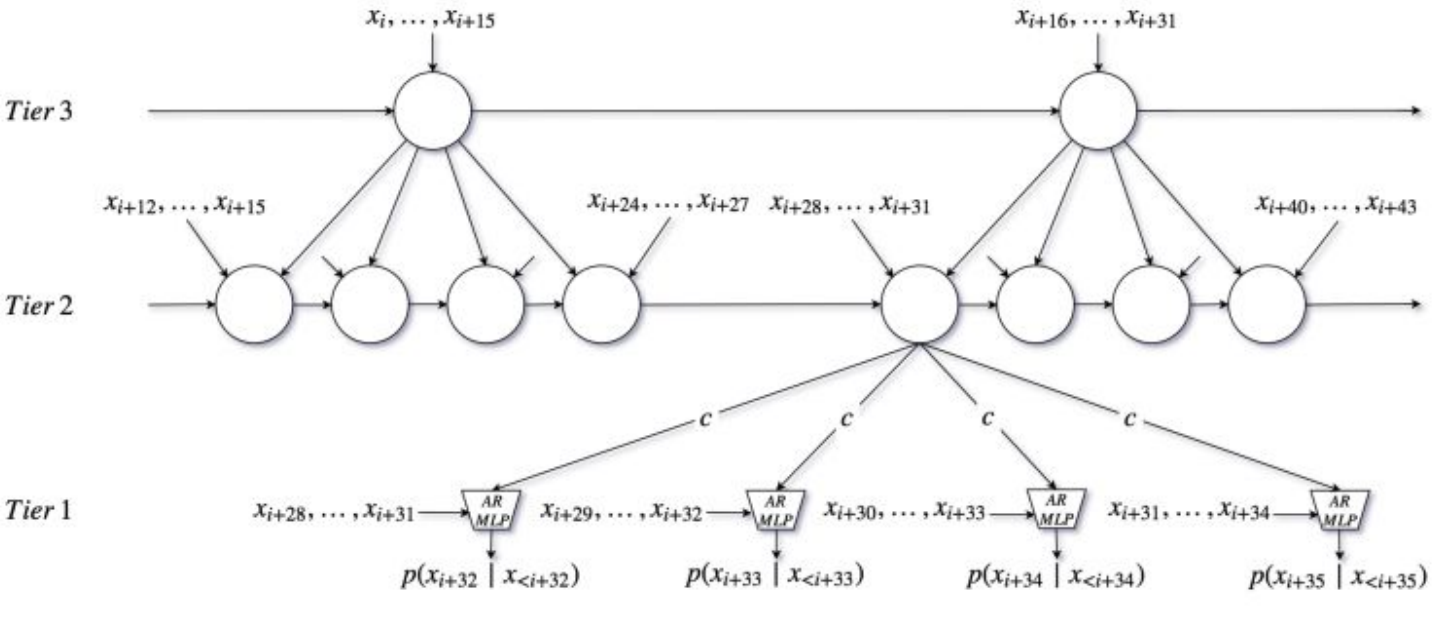

**Larger "receptive field"**

deepmind.google/discover/blog/wavenet-a-generative-model-for-raw-audio

(Source: van den Oord et al., 2016)

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *ICML*, 2016.

# Example: SampleRNN (Mehri et al., 2017)




**Aggregate "context" from preceding samples into fixed-sized hidden state**

**Predict next sample given last "frame" of samples + current hidden state**

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., ... & Bengio, Y. (2017). SampleRNN: An unconditional end-to-end neural audio generation model. ICLR 2017.

# Example: SampleRNN (Mehri et al., 2017)

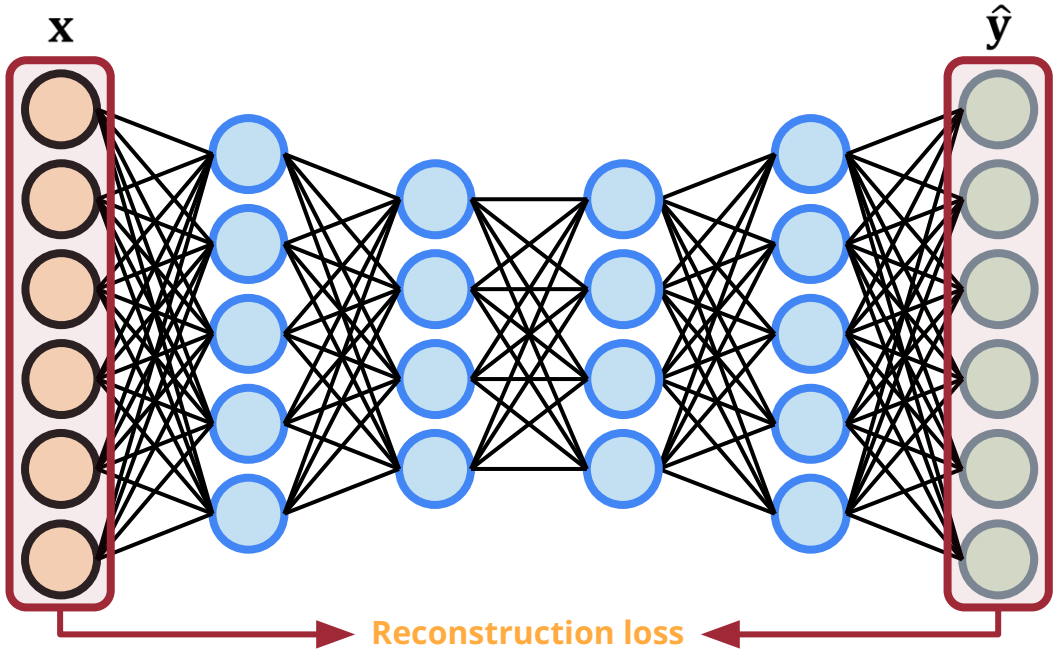
**Nuns in a Moshpit (AI Song Contest 2022 Runner-up)**



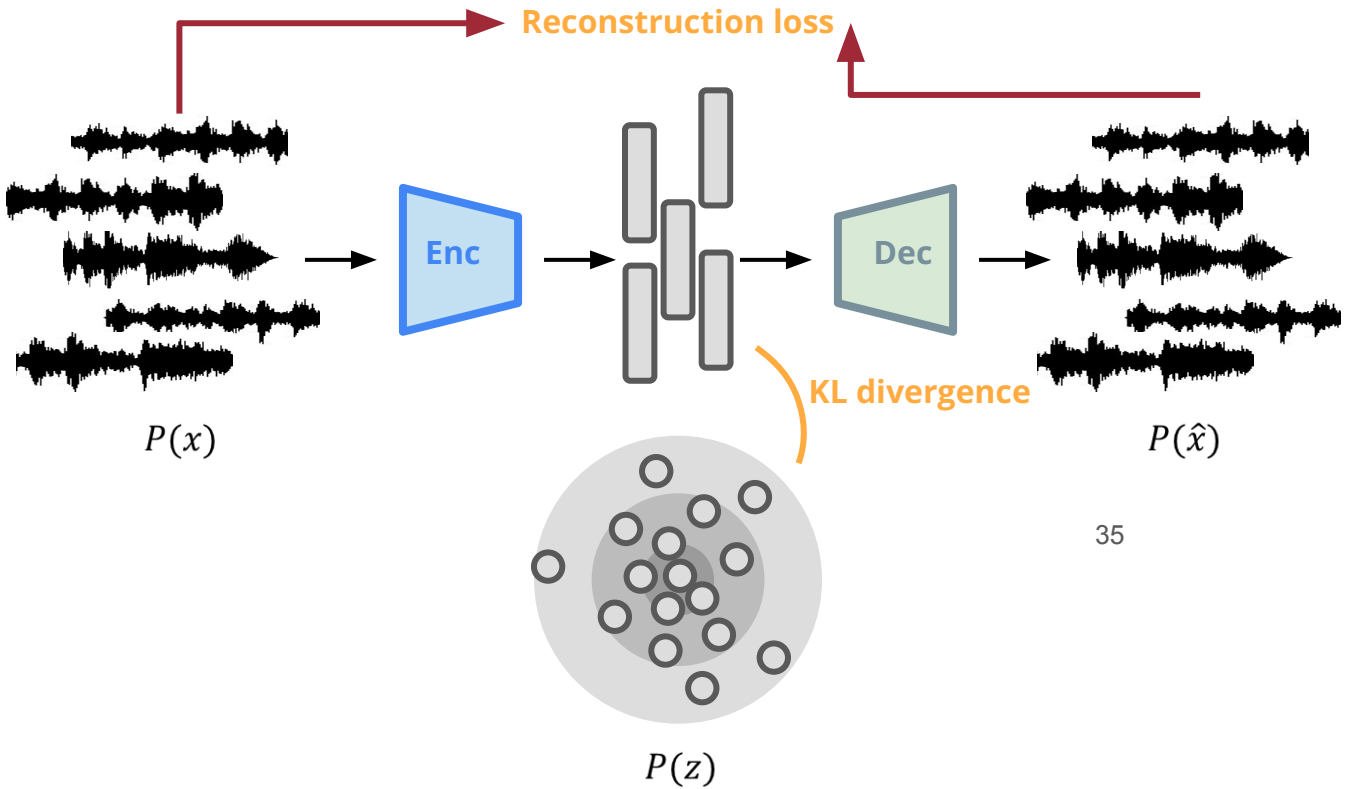
Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., ... & Bengio, Y. (2017). SampleRNN: An unconditional end-to-end neural audio generation model. ICLR 2017.

# (Recap) Autoencoders

- A neural network where the **input and output are the same**



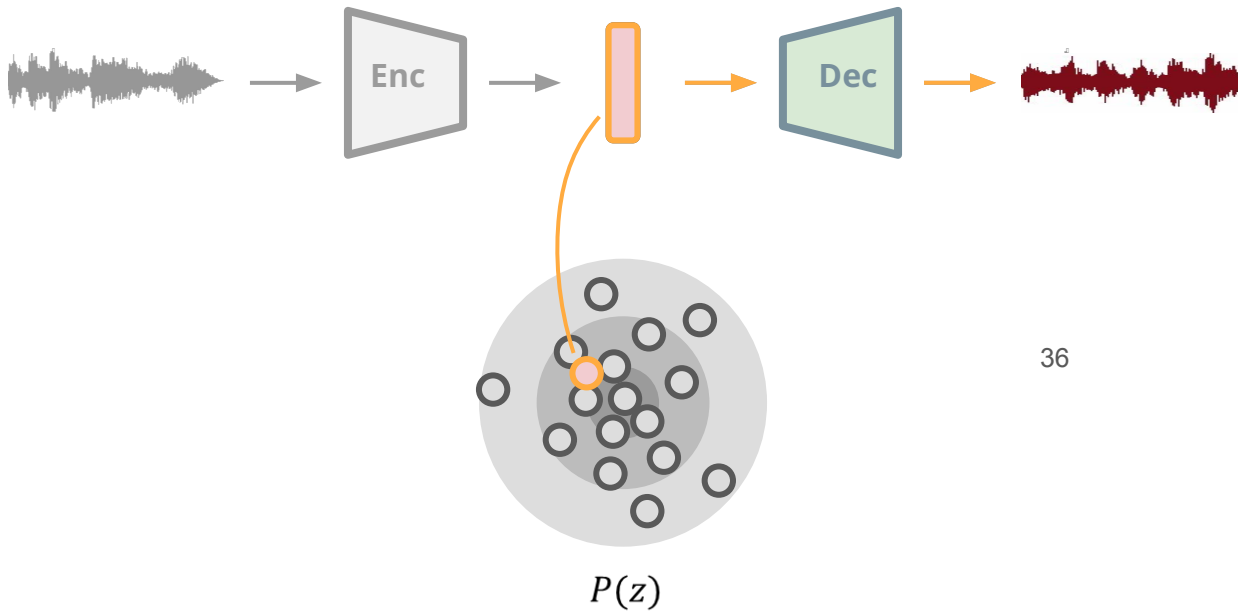
# (Recap) Variational Autoencoders (VAEs)
## – Training






Diederik P Kingma and Max Welling, "Auto-Encoding Variational Bayes," *ICLR*, 2014.

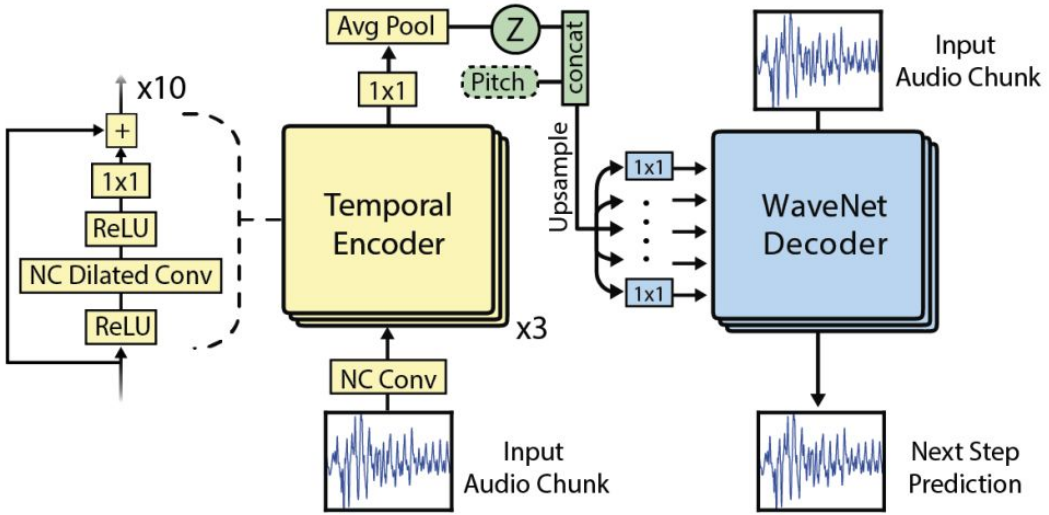
# (Recap) Variational Autoencoders (VAEs)
## – Generation






Diederik P Kingma and Max Welling, "Auto-Encoding Variational Bayes," *ICLR*, 2014.

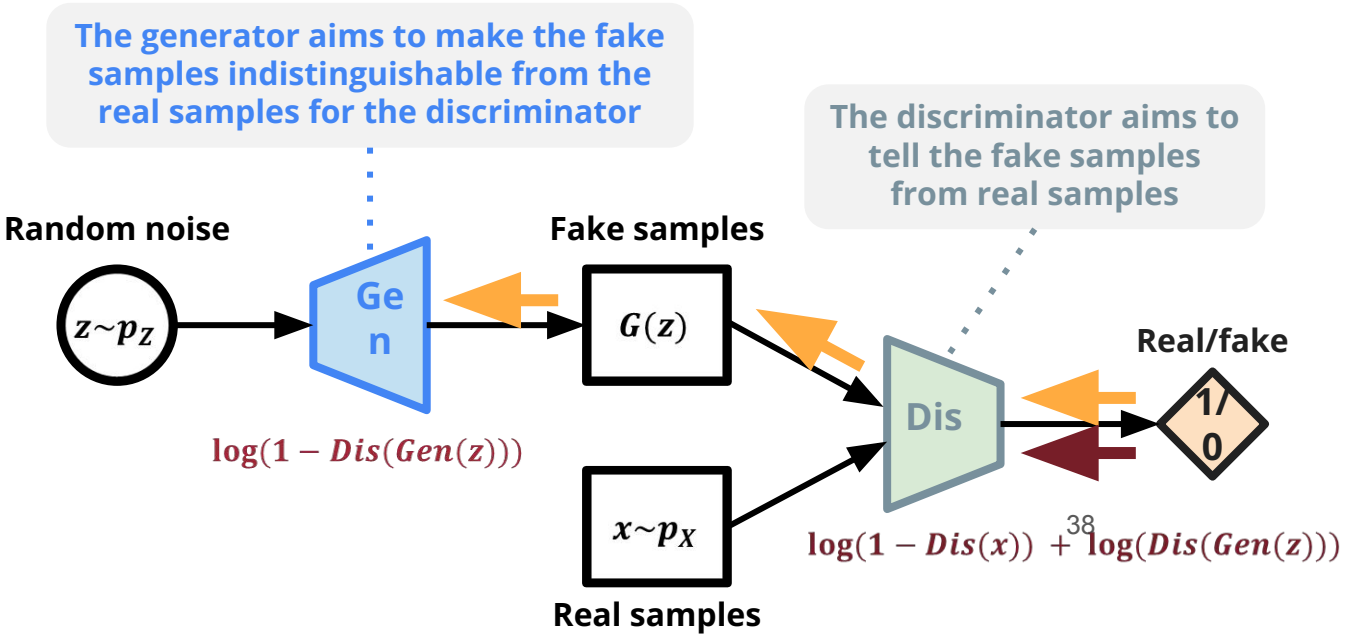
# Example: WaveNet Autoencoder (Engel et al., 2017)




***Autoregressive* Decoder:**
Generate samples by:
1) Drawing some latent representation from a gaussian
2) Conditioning the decoder on this latent to generate next sample
3-∞) feed generated sampe + latent back into model to generate subsequent samples

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., & Simonyan, K. (2017). Neural audio synthesis of musical notes with WaveNet autoencoders. ICML 2017.

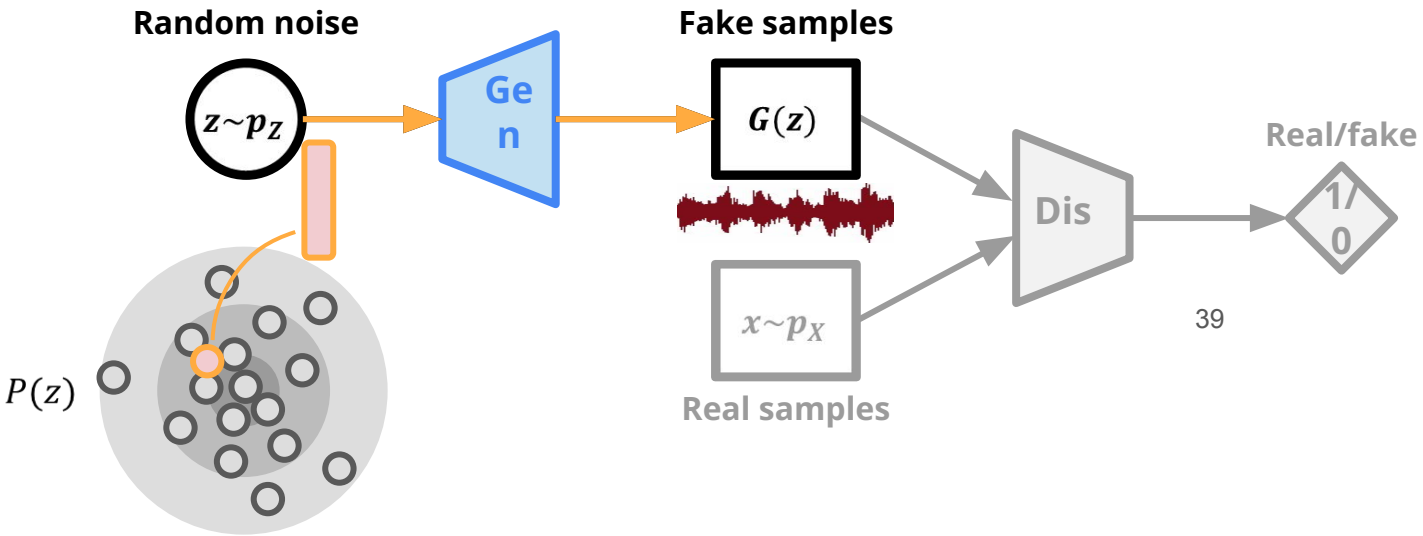
# (Recap) Generative Adversarial Nets (GANs) – Training






Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative Adversarial Networks," *NeurIPS*, 2014.

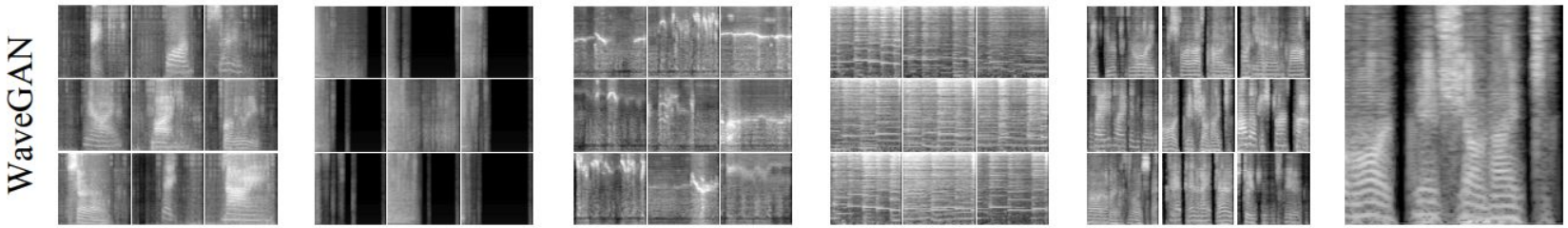
# (Recap) Generative Adversarial Nets (GANs) – Generation






Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative Adversarial Networks," *NeurIPS*, 2014.

# Example: WaveGAN (Donahue et al., 2018)




**Like GAN for images, but 1D and over waveforms**

Donahue, C., McAuley, J., & Puckette, M. (2018). Adversarial audio synthesis. ICLR 2018.

# Direct Spectrogram Generation

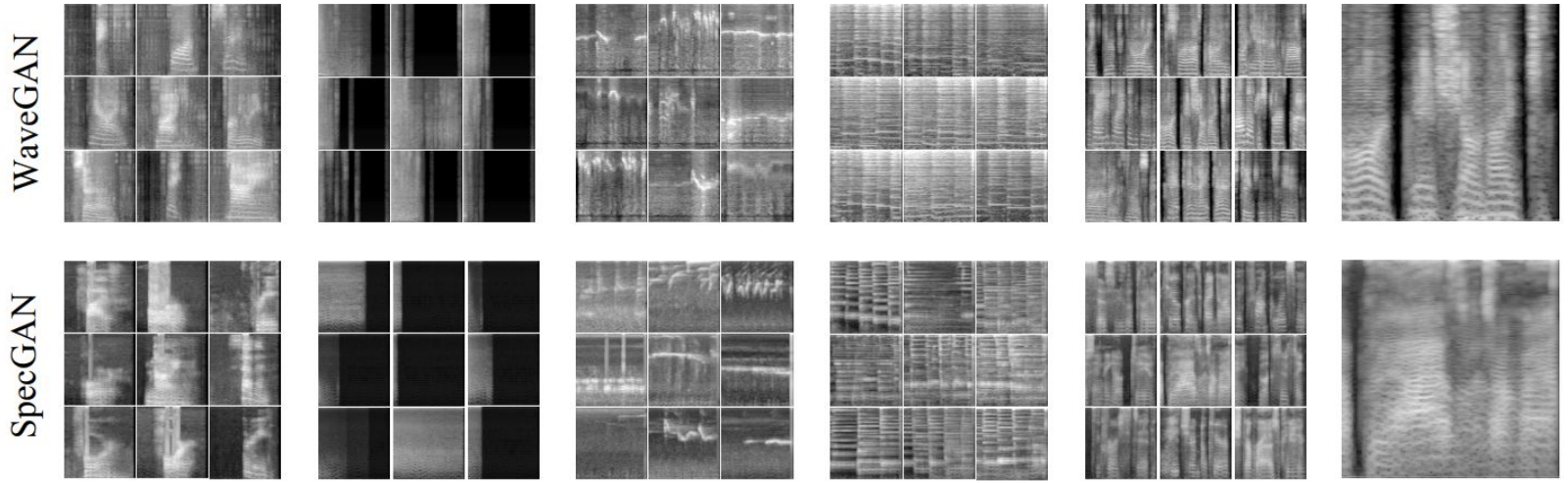
# Spectrograms

**Spectrograms are just like images, so we can borrow generative techniques from vision...but what kind of spectrograms should we use?**

- Complex: Exact, but how to deal with modeling complex data?
- Magnitude: Lossy, non-perceptually aligned, how to reconstruct?
- Mel: Lossy, perceptually aligned, how to reconstruct?

We'll get into this more when we talk about representations more broadly.

# Example: SpecGAN (Donahue et al., 2018)




**Like GAN for images, but on magnitude spectra!**

**Reconstruction: Griffin-Lim Algorithm (quite slow, unpopular)**

Donahue, C., McAuley, J., & Puckette, M. (2018). Adversarial audio synthesis. ICLR 2018.

# Example: SpecGAN (Donahue et al., 2018)



Non GAN-activated cat | WaveGAN activated cat | SpecGAN activated cat

Figure 7: Compared to resting state, this cat's level of alertness increased when presented bird vocalizations synthesized by WaveGAN and SpecGAN.

**But it worked! (according to cat)**

Donahue, C., McAuley, J., & Puckette, M. (2018). Adversarial audio synthesis. ICLR 2018.

# Spectrograms

**Direct Spectrogram Synthesis wasn’t massively popular until the diffusion era...**

# Audio-Domain Music Generation

4.3: Representations for Audio

# What's a good representation for audio generation?

We've talked a lot implicitly about audio representations (waveforms, different kinds of spectrograms), but are these optimal for generative modeling?

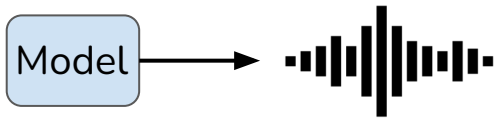
We'll see that we can repurpose our classical generative models (VAEs, GANs), for **learning good representations**, rather than **generation**

Zachary Novack
Ke Chen

# Diffusion TTM - Representation

## What is $\boldsymbol{x}_{\text{data}}$ ?

### Direct Waveform Modeling

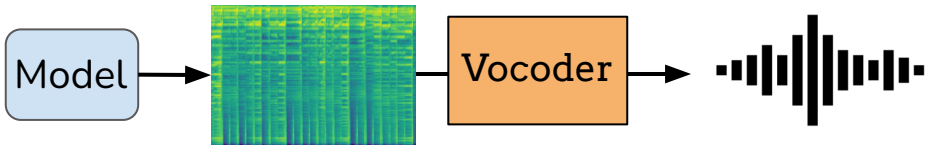



$$\boldsymbol{x}_{\text{data}} \in \mathbb{R}^{f_s T \times C}$$

$$C = 1 \text{ (mono) or } 2 \text{ (stereo)}$$

- Waveform at given sampling rate and time length
- SampleRNN, WaveNet

### (mel)-Spectrogram Modeling




$$\boldsymbol{x}_{\text{data}} \in \mathbb{R}^{H \times W \times C}$$

$$C = 1 \text{ (mel) or } 2 \text{ (complex)}$$

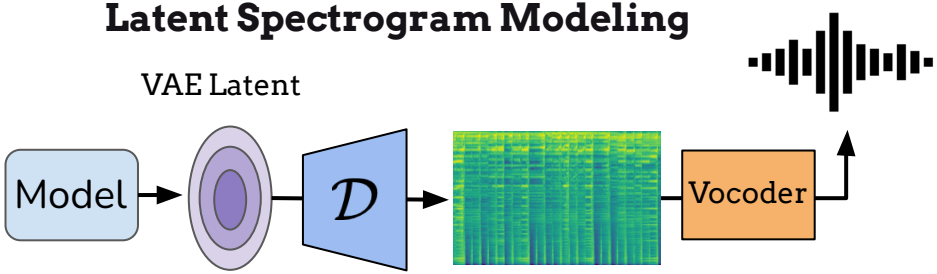
- Spectrogram, audio “image”
- Needs vocoder for mel-spec modeling
- DITTO, SpecGAN, EDMSound

Zachary Novack
Ke Chen

# Diffusion TTM - Representation

## What is $\boldsymbol{x}_{\text{data}}$ ?

### Latent Spectrogram Modeling

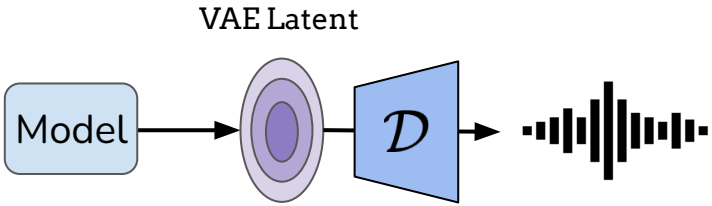



$$\boldsymbol{x}_{\text{data}} \in \mathbb{R}^{D_h \times D_w \times D_c}$$

$$D_h, D_w \ll H, W \quad D_c > C$$

- Continuous 2D latent from spectrogram VAE
- MusicLDM , Riffusion v1

### DAC-Style[1] Latent Waveform Modeling




$$\boldsymbol{x}_{\text{data}} \in \mathbb{R}^{D_T \times D_c}$$

$$D_T \ll f_s T \quad D_c > C$$

- 1D *continuous* latent from **waveform** VAE
- Like DAC/EnCodec (VQGAN), **but no quantization** (only KL reg.)
- **SAO**

[1] Rithesh Kumar et al., “High-Fidelity Audio Compression with Improved RVQGAN”

# Direct Waveform Modeling

**Pros:**

- Lossless! We have phase!
- 1-stage modeling

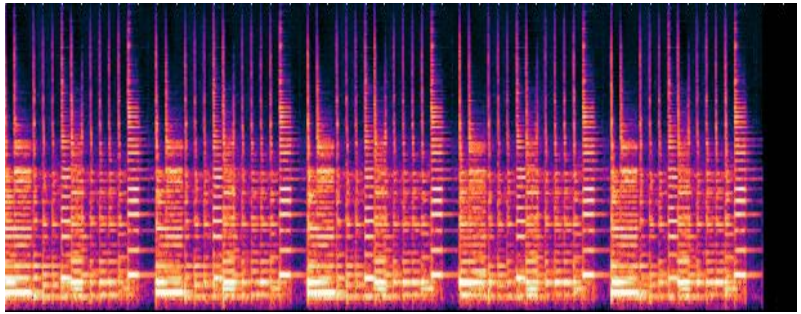
**Cons:**

- **SUPER** high-dimensional
  - e.g. 47.55 seconds of 44.1 kHz (professional grade) audio is >2M values
  - You are never going to generate this quickly
- Long-range dependencies
  - Relevant concepts in data are far apart
- Modeling fine-grained details
  - Waveforms include **everything**, perceptual *fidelity* (texture) + all semantic content (structure)
  - **Can we disentangle texture and structure?**

# Spectrogram Modeling (Does this disentangle)?

**Pros:**

- Biologically plausible!
- Compressed, shorter-range dependencies
- Semantically focused
    - Broadly, we can **see** musical concepts here



**Cons:**

- Lossy / We can't reconstruct this easily!
    - Complex: hard to model
    - Magnitude: need vocoder / griffin-lim, less bio correlated
    - Mel: need vocoder
- Bad scaling
    - Longer audio -> multiplicative modeling cost (depending on spectral resolution)
- Structure vs. Texture
    - Is this **optimal?**
    - Nothing about spectral decomposition implies ease of generative modeling

# Homework 4 is cancelled!

This was too hard / pain to setup and wasn't fully tested enough

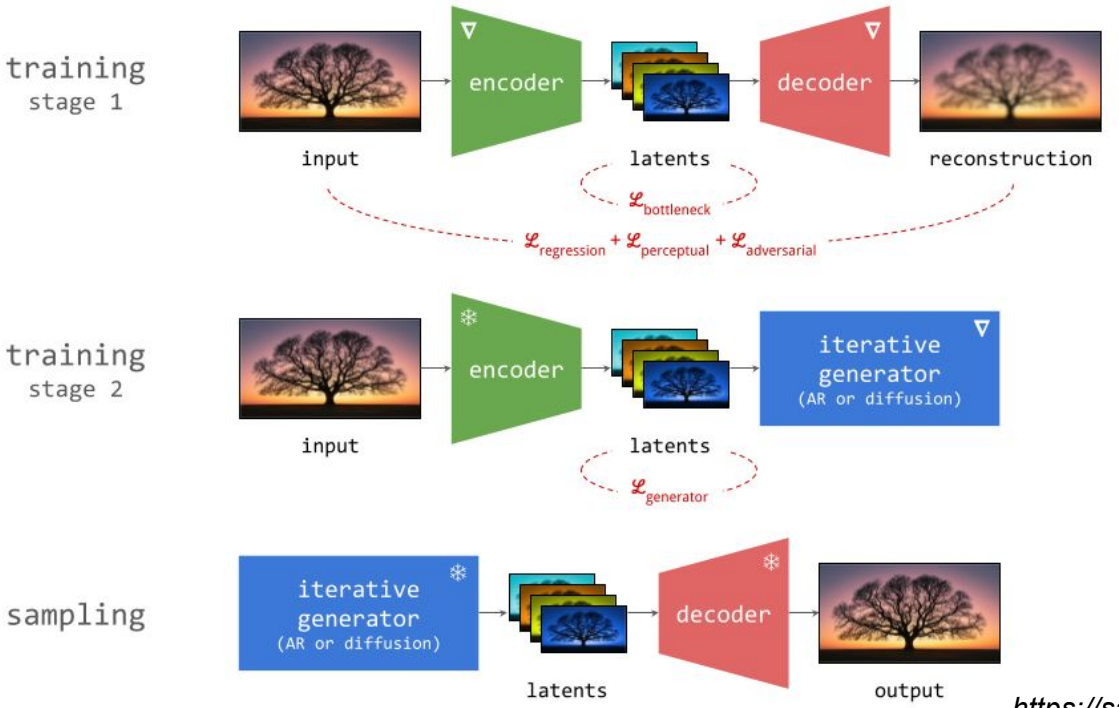
We'll post the solutions soon for those interested

Sorry for the pain on this one :)

Now we can just learn about audio and music in peace

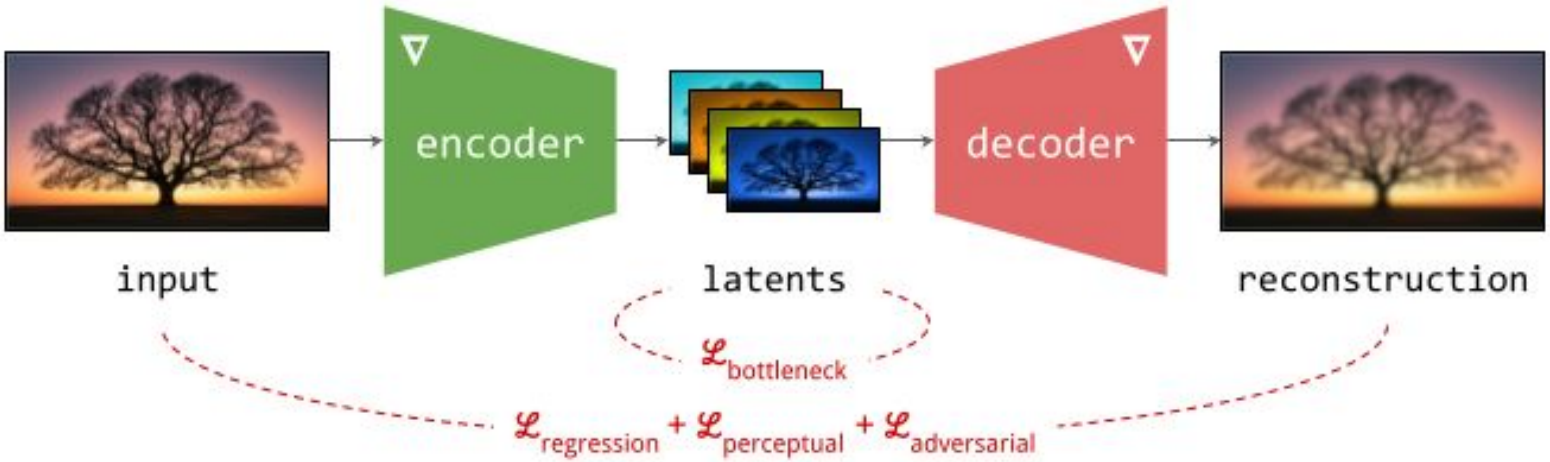
# Can we *learn* a good representation?

***All modern Generative Audio Systems follow this paradigm***



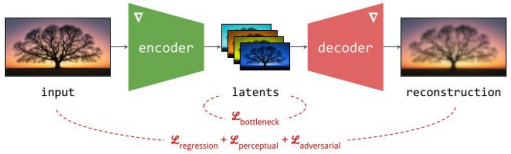

https://sander.ai/2025/04/15/latents.html

# ***How*** can we ***learn*** a good representation?




*https://sander.ai/2025/04/15/latents.html*

# ***How*** **can we** ***learn*** **a good representation?**

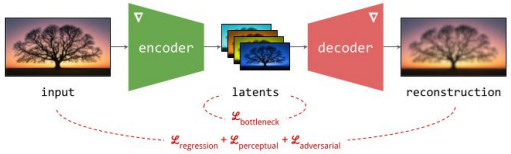



## Regression Loss:

- This is just reconstruction! make output = input (i.e. D(E(x)) = x)
  - Should we do this on waveforms? On spectrograms?
- Instead of directly enforcing D(E(x)) = x, we can make the outputs match inputs along different **views** (i.e. f(D(E(x))) = f(x))
- Treat f as a collection of spectrogram functions (w/different params)
  - Multi-scale spectral loss

*https://sander.ai/2025/04/15/latents.html*

# ***How*** can we ***learn*** a good representation?

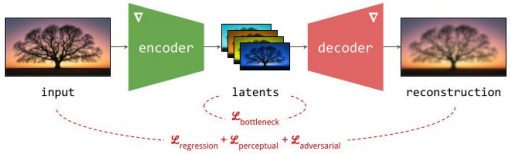



## Perceptual Loss:

- What if f wasn't just a spectrogram, but *any* fixed embedding model?
  - What if this pre-trained embedding model M() encoded "perceptual" features well (i.e. high frequency details, fidelity)
- "Make the extracted features of the output match the extracted features of the input"
- Note: For audio, unlike images, this *isn't* that common
  - Why? We don't have great "perceptual embedders" like in image (LPIPS, DINO)

*https://sander.ai/2025/04/15/latents.html*

# ***How*** can we ***learn*** a good representation?

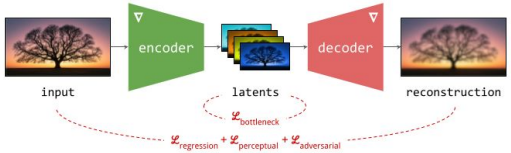



## Adversarial Loss:

- What if we **learned** f *at the same time* as learning out autoencoder?
  - How can we learn a feature extraction that is useful for representation at the same time?
- We can do it with a discriminator!
- Identical to normal GANs, but replace the generator (i.e. mapping from random z->data x) with an autoencoder (i.e. mapping from data x->latent z->data x)
  - We can apply the same idea of “views” (from reconstruction) to this as well
  - Multi-Period / Multi-Resolution Discriminators

*https://sander.ai/2025/04/15/latents.html*

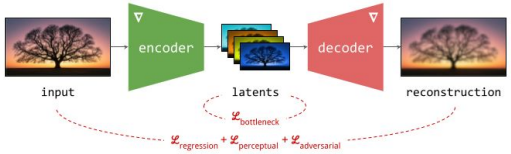
# ***How*** can we ***learn*** a good representation?




## Adversarial Loss:

*https://sander.ai/2025/04/15/latents.html*

# ***How*** **can we** ***learn*** **a good representation?**

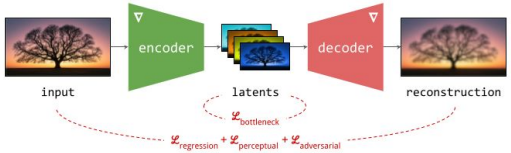



## Bottleneck Loss:

- So this defines all the losses on the *output* of our model decoder, but what about losses on the **encoded** latents?
  - How do we regularize the latent space to be compact and simple?
- Option 1: Standard KL Loss!
  - Make latents more Gaussian, standard VAE (or VAEGAN) formulation
  - Yields continuous-valued latents (like a VAE)
  - The main workhorse behind diffusion models
- Option 2: **Vector Quantization (VQ)**
  - Replace KL loss with a quantization layer, VQVAE (VQGAN)
  - Yields **discrete-valued** latents (like in symbolic music!)
  - We'll talk more about this later

*https://sander.ai/2025/04/15/latents.html*

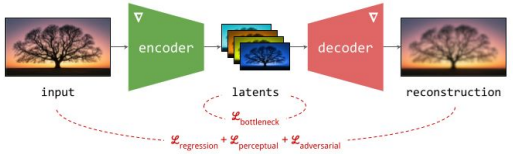
# ***How*** can we ***learn*** a good representation?




## Bottleneck Loss:

*https://sander.ai/2025/04/15/latents.html*

# ***How*** can we ***learn*** a good representation?



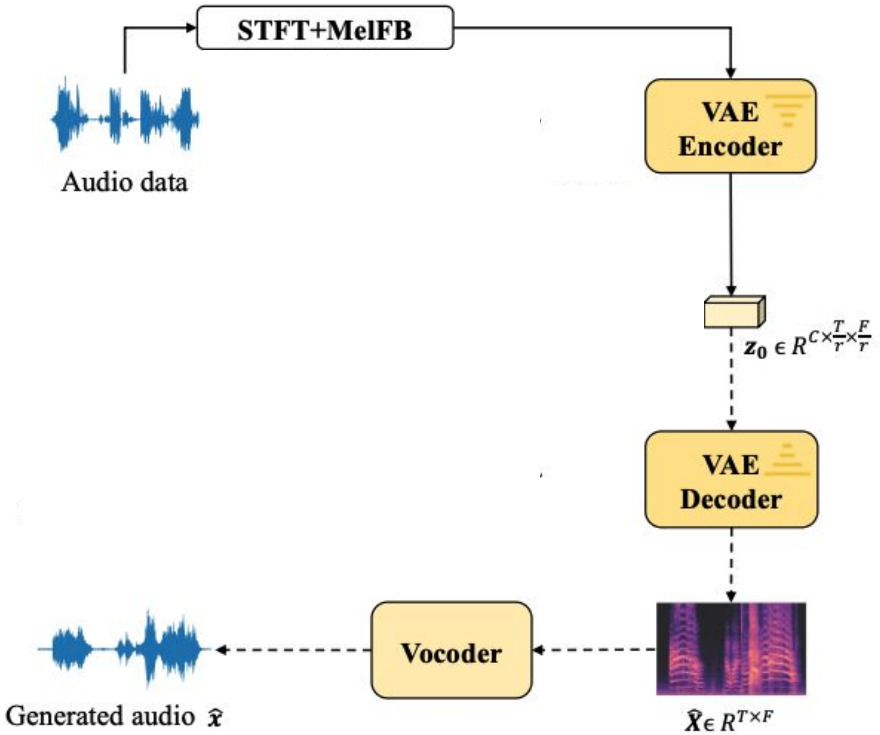

Texture vs. Structure (or Stuff vs. Things)

- If we want to reconstruct this image, what's more important:
  - Making sure every blade of grass is the same? (**texture, stuff**)
  - Making the dog have eyes in the right place? (**structure, things**)
- **A good representation should abstract away texture, but not structure**
  - Keep the dog a dog, but it's okay if the grass is different on reconstruction



*https://sander.ai/2025/04/15/latents.html*

# Latent Spectrogram Modeling




Pros:

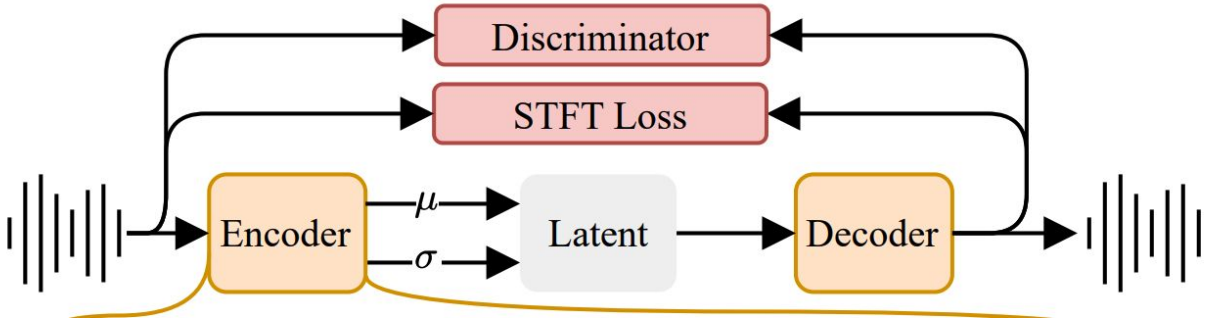
- (Hopefully) Informative latent space for modeling
- Can reuse code from images (since specs are 2D)

Cons:

- We still need a vocoder, AND we need to convert latents->specs after generation

https://arxiv.org/pdf/2301.12503

# Latent Waveform Modeling




Pros:

- (Hopefully) Informative latent space for modeling
- 1D latents! More efficient to model
- No vocoder needed!

Cons:

- Decoding can get a bit slow form long sequences (but not much slower than other approaches
- This is the state-of-the-art right now

*https://arxiv.org/pdf/2404.10301v2*

# Are VAEs / GANs dead?

As **generators**: Mostly yes

As **representation learning methods**: Definitely not!

Is there anything else they’re good for?

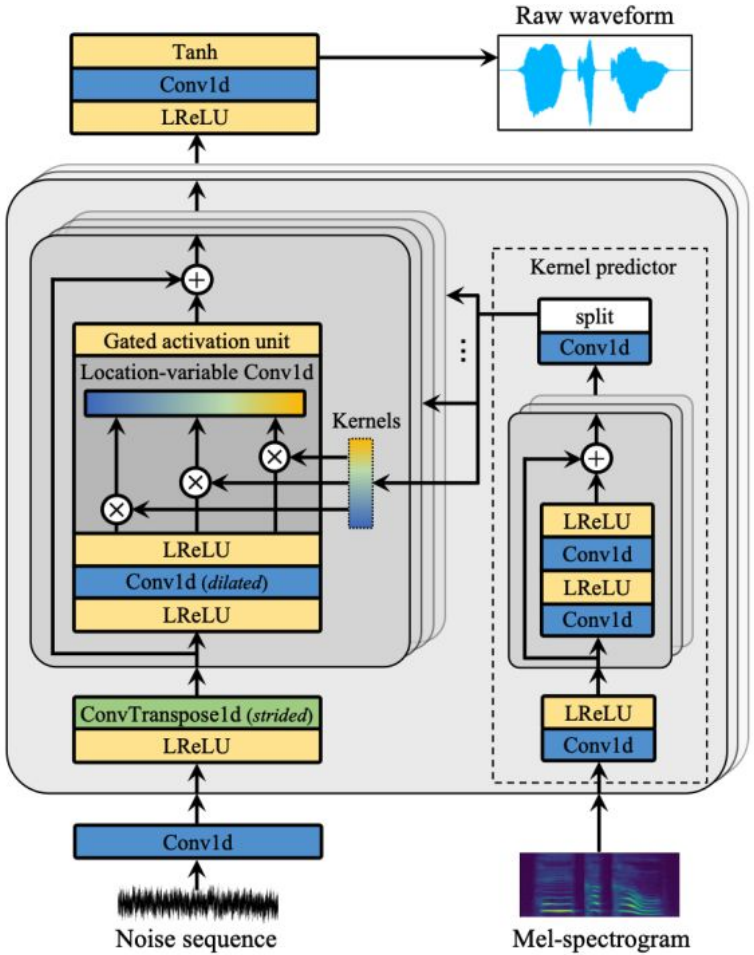
# Yes: Vocoders!

Vocoder: Map spectrogram -> Waveforms!

- Train as conditional GANs: generate waveforms **conditioned** w spectrograms
- Most modern vocoders (Hifi-GAN, MusicHIFI, MelGAN) are GANs!

Why can GANs do this well, but not generation?

- Adversarial training is hard from scratch
- *But*, good if task is *strongly conditioned*




https://arxiv.org/pdf/2106.07889

# Yes: RAVE!



RAVE: Real-time VAE

- Train on *specific* instruments
- Encoding+decoding normal audio->style transfer

# ~~Next time...~~ This time...

So we've learned about classical models (that are okay) and good representations...

we'll see how to train *good* models using these representations



# Audio-Domain Music Generation

4.4: Diffusion Models for Music Generation

# Diffusion Models are GenAI's modern workhorse

Flow and Diffusion Models: state-of-the-art models for generating images, videos, proteins!



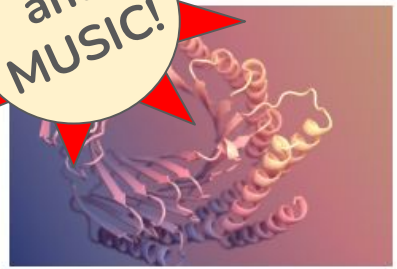
*Stable Diffusion*

*DALEE*



*OpenAI Sora*

*Meta MovieGen*



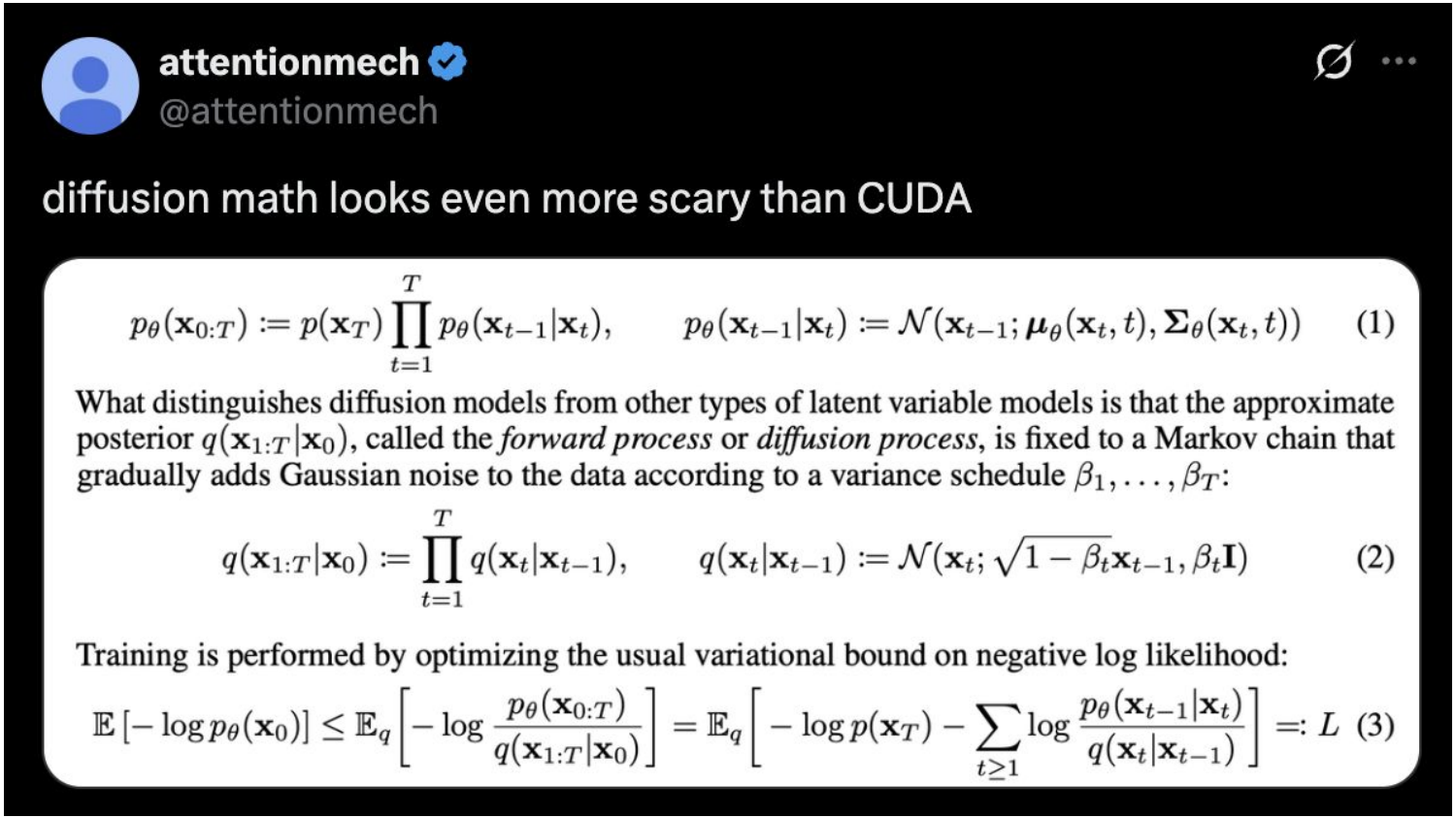

*AlphaFold3*

*RFDiffusion*

**Most SOTA generative AI models for images/videos/proteins/robotics: Diffusion and Flow Models**

*https://diffusion.csail.mit.edu/docs/slides_lecture_1.pdf*

# However, the math for diffusion...



attentionmech
@attentionmech

diffusion math looks even more scary than CUDA

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \quad (1)$$

What distinguishes diffusion models from other types of latent variable models is that the approximate posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$, called the *forward process* or *diffusion process*, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \ldots, \beta_T$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad (2)$$

Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L \quad (3)$$

# People Overcomplicate This

- A lot of diffusion tutorials online suck, but this is because diffusion only *recently* converged on better notation / messaging
  - The early papers are a grind to read
- Often, people describe diffusion using latent variables / hierarchical denoising autoencoders, and this is **BAD**
  - While connected to VAEs, the math is way more complex, and requires strong comfort with VAE math
  - While this helps explain training (somewhat), this approach is *very* unintuitive for **sampling** from DMs (and if you look at the code inspired by the approach, you'll agree)
- Some may disagree, but they're not the ones making these slides :)

# Diffusion as Continuous-Time Differential Equations

- Instead, we will view diffusion as continuous-time DEs
- Why?
  - More models are following this parameterization now
  - It generalizes to arbitrary flow matching (also popular, Stable Diffusion 3, Flux)
  - The simplest open-source *music* model uses this (Stable Audio Open)
  - Connects much more to **sampling** from DMs
- Do I need DE knowledge for this?
  - Not really, introductory calculus will suffice

# Our two guiding lights for diffusion

## Denoising:

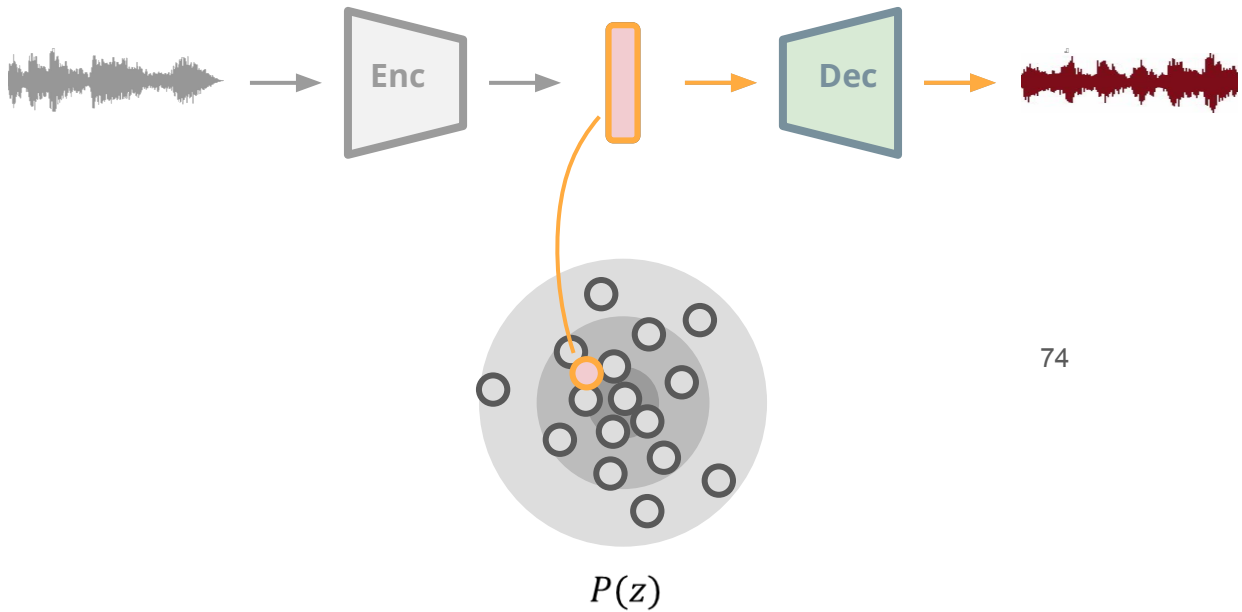
We want a model that takes some noisy version of audio, and removes the noise

## Score Modeling:

We want a model that captures the *gradient* (i.e. score/derivative/direction) of the map from noise->audio

**The optimal denoiser = The optimal score model**

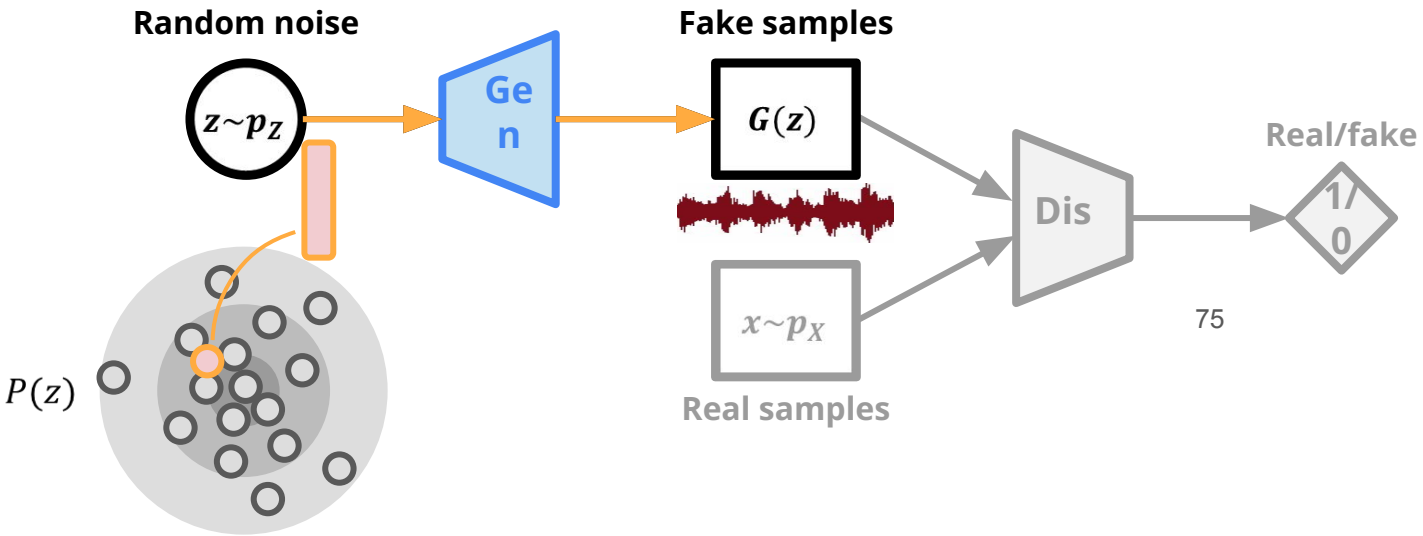
# (Recap) Variational Autoencoders (VAEs)
## – Generation






Diederik P Kingma and Max Welling, "Auto-Encoding Variational Bayes," *ICLR*, 2014.

# (Recap) Generative Adversarial Nets (GANs) – Generation



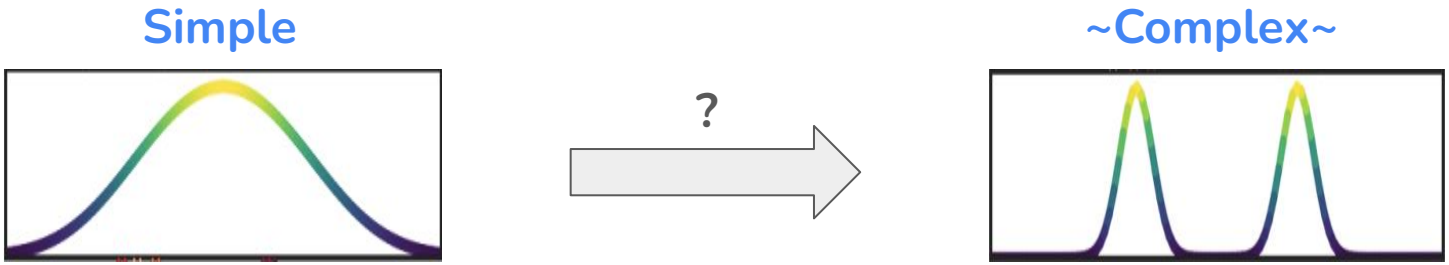



Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative Adversarial Networks," *NeurIPS*, 2014.

At its core, diffusion is just yet another way to map simple distribution->hard distribution

# Let’s start with a simple 1D example...

How could we sample from this complex distribution, or rather, how can we learn to transition from our simple distribution to the complex one (and thus sample)?



# Let's start with a simple 1D example...

How would a GAN / VAE do this?

- Learn some model **G**(z) = x, i.e. a **direct map** from simple->complex
- Broadly, trying to get **G** to simulate **p(x)**

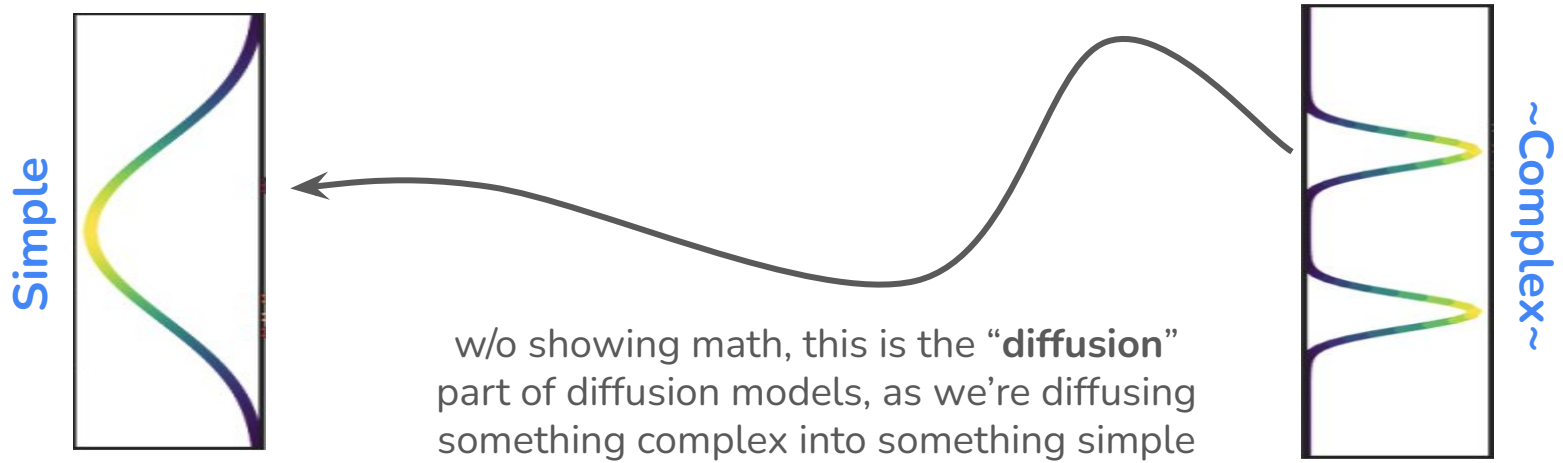



This is asking a lot, right? We're trying to transfer (possibly) wildly different distributions in 1 shot!

# Let's start with a simple 1D example...

Instead, let's try something crazy: let's find an ***easy*** way to go in the other direction, from complex->simple

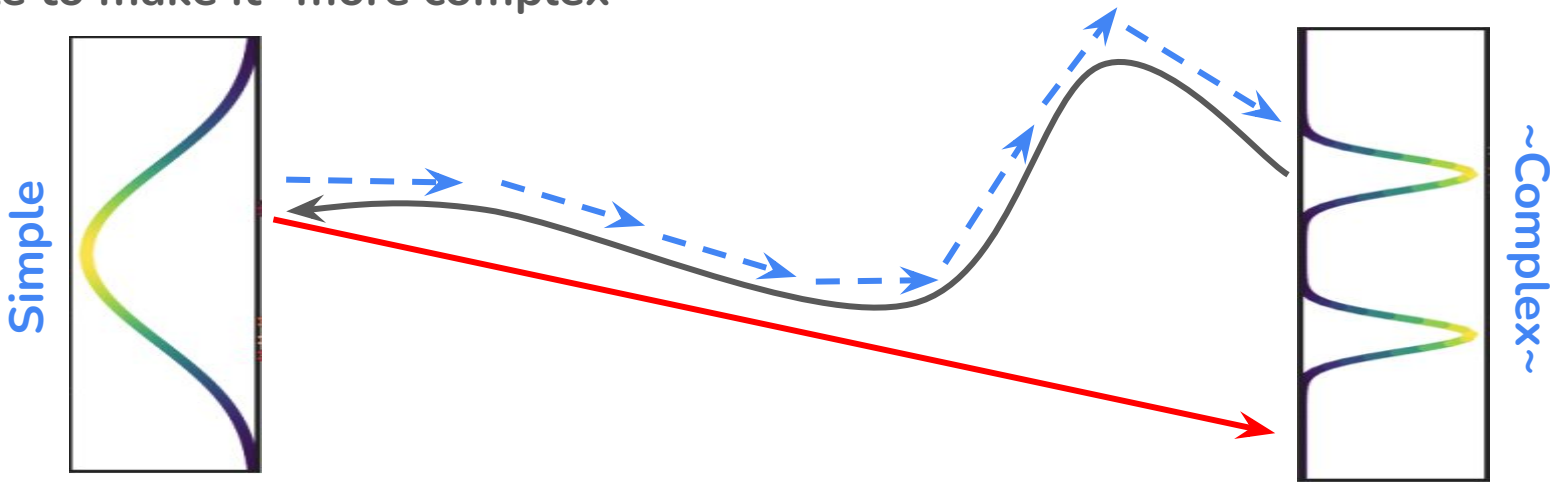
- Note: this doesn't have to be *simple*, just easy (closed-form) to calculate



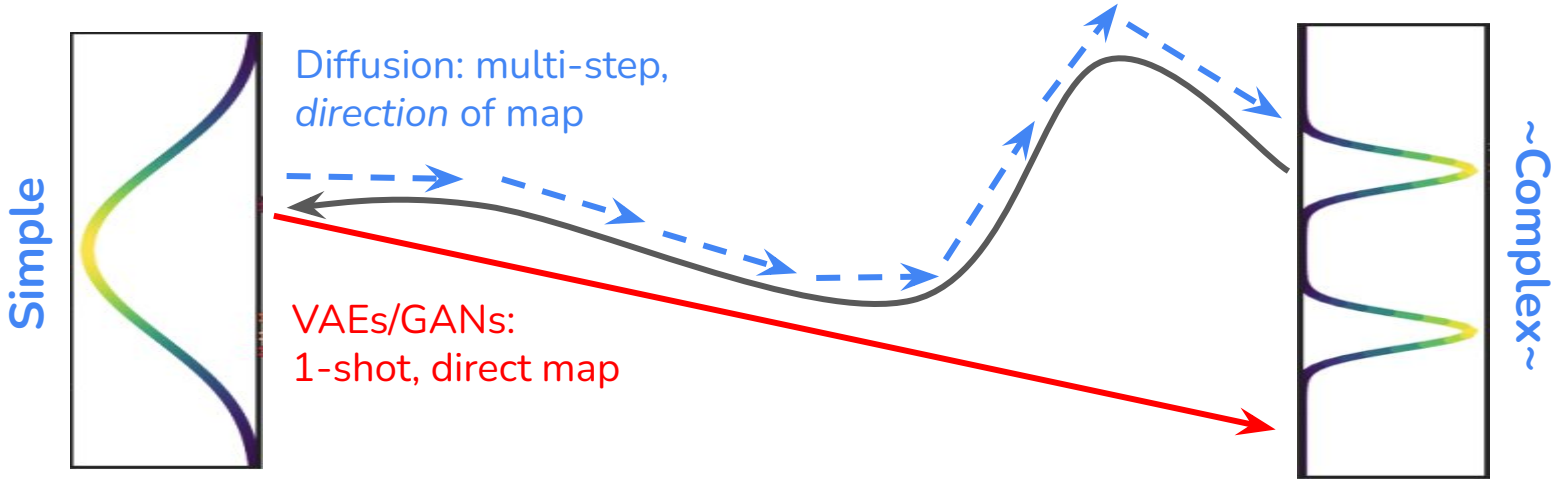
# Let's start with a simple 1D example...

Why was that useful at all? You just made this more complex!

What if instead of learning a **map G** from simple->complex, we learn the ***derivative*** of such a map **s**(z) ≈ dG(z)/dz, i.e. **the *direction* to move anything simple to make it "more complex"**



# This is the most important of diffusion!




Diffusion literally just gives us more chances to make our output better

# Ok but...

- Where did you pull this diffusion process from?
- Are we *really* modeling the derivative of something? How is that possible?
- How does this extend from 1D to high-dimensional audio signals?

- You've said what it *does*, but how do we train this??

# What is this *diffusion* process?

**It's a Stochastic Differential Equation (SDE)!**

$$\underbrace{\mathrm{d}\mathbf{x}_t}_{\text{As time increases, x changes according to...}} = \underbrace{\sqrt{2t}}_{\text{scaled by the increasing time}} \underbrace{\mathrm{d}\mathbf{w}_t}_{\text{some Gaussian noise (weiner process)...}}$$

$\mathbf{x}_0$ is our normal data (i.e. $\mathbf{x}$)

$\mathbf{x}_T \approx \mathcal{N}(0, \mathbf{I})$

# What is this *diffusion* process?

**And why *this* specific SDE?**

We can *reverse*[1] this SDE and move *backwards in time* (i.e. **from noise to data**) following a simpler *deterministic* Ordinary DE (ODE):

$$\underbrace{\mathrm{d}\mathbf{x}_t}_{\text{As time decreases, x changes according to...}} = \underbrace{-t}_{\text{scaled by the decreasing time}} \underbrace{\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}_{\text{The gradient w.r.t. x of the marginal log prob. of x}} \mathrm{d}t$$

[1] *Brian DO Anderson. Reverse-time diffusion equation models. Stochastic Processes and their Applications, 12(3):313–326, 1982.*

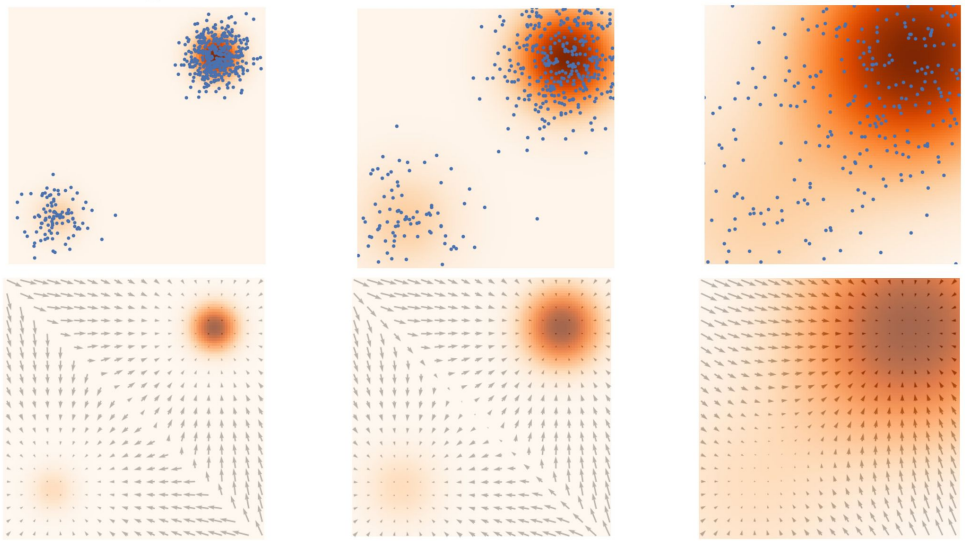
# Are we modeling a derivative? What is this RH Side?

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

This is called the **score function** of our marginal probability distribution

A “vector field”, describing the *direction* to move (in data space) such that our sample is more “data-like”

*Broadly*, like a derivative, but for high dimensional data



# Modeling Scores - To Reiterate

**So if we learn a model to match the score...**

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

**Then we can *numerically* integrate this ODE...**

$$\mathrm{d}\mathbf{x}_t = -t\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\mathrm{d}t$$

**And map noise to data!**

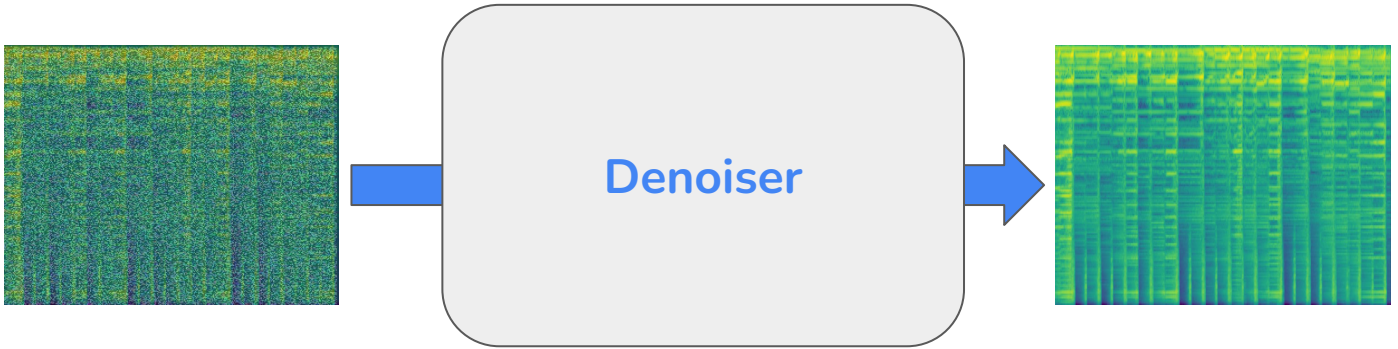


# How do we train this?

- For training, one *can* still view this as matching scores…
- …but a clever results ties this to the idea of *denoising* instead, which admits a much cleaner objective!

# How do we train this?

Denoising:




- Remove noise from an image
- Quite popular in “old-school” era as Denoising Autoencoders (DAEs)

# How do we train this?

Training denoisers:

$$\mathbb{E}_{t\sim p(t),\epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\mathbf{x}-D_\theta(\mathbf{x}+t\epsilon,t)\right\|_2^2\right]$$

- Given some random gaussian noise, and some random “level” of noise t
- Minimize MSE b/w clean data and denoiser running on noisy data
  - This is just a regression task!
  - We are learning to predict a high-dimensional set of continuous values given our “features”, which are randomly noisy versions of our target

# How do we train this?

So let's say we did this optimally, what would that mean?

**Our model *maximizes* the conditional expectation of clean given noisy**

$$D_\theta \approx \mathbb{E}_t[\mathbf{x} \mid \mathbf{x}_t]$$

I.e. what is the *average* clean data given any noisy data a particular level

Drumroll please...

# Tweedie's Formula

**The optimal denoiser = The optimal score model**

Rearranging terms of our ODE from before:

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

$$\underbrace{\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t}}_{\text{The } \textit{change} \text{ in x over time (i.e. derivative)...}} = \underbrace{-t\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}_{\text{...is related to the score of the distribution}} = \underbrace{\frac{\mathbf{x}_t - \mathbb{E}_t[\mathbf{x} \mid \mathbf{x}_t]}{t}}_{\text{...AND is related to our optimal denoiser!}}$$

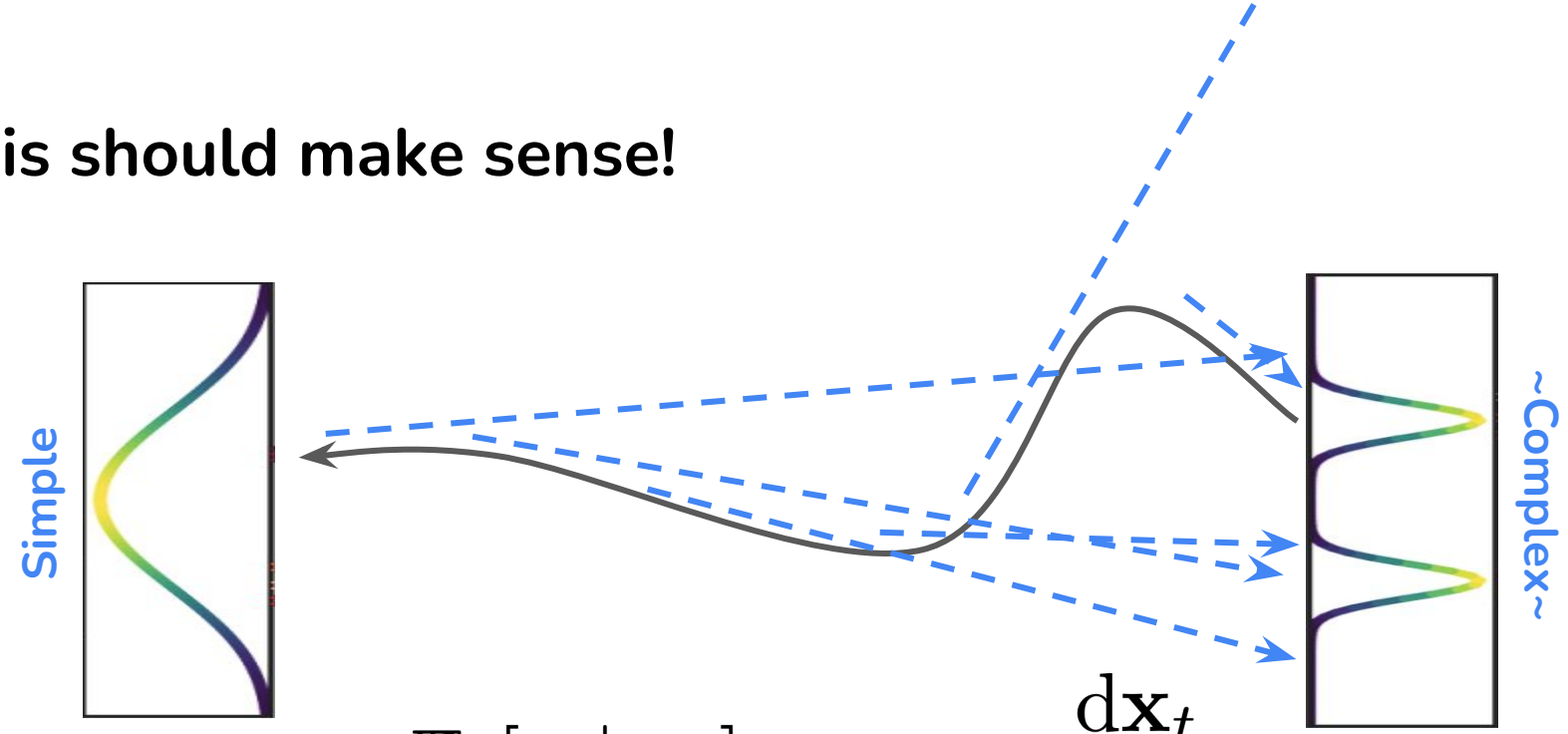
# Tweedie's Formula

$$\underbrace{\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}_{\text{sampling}} = \frac{\mathbf{x}_t - \overbrace{\mathbb{E}_t[\mathbf{x} \mid \mathbf{x}_t]}^{\text{training}}}{t}$$

- We can **train** our model with our simple regression MSE denoising loss
- We can **sample** by converting our denoised estimate to the **score/derivative**, and solving the reverse-time ODE!



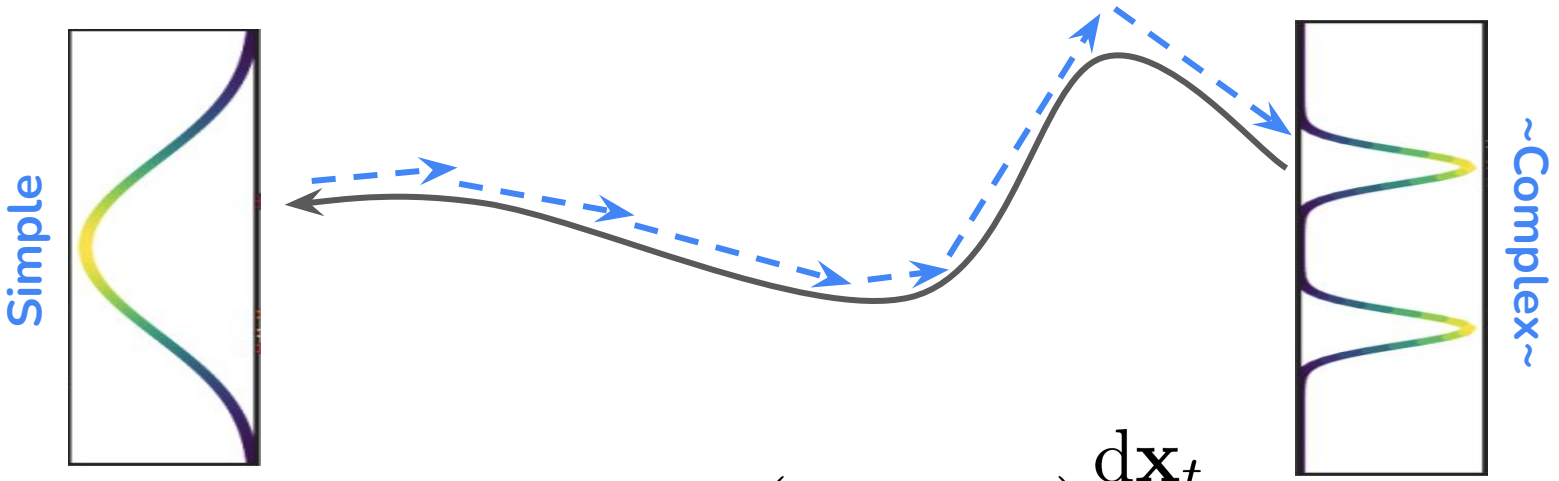
# This should make sense!




$$\mathbb{E}_t[\mathbf{x} \mid \mathbf{x}_t] = \mathbf{x}_t - t\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t}$$

Our “best guess” (expectation) at any point is taking where we currently are (x_t), getting the derivative, and stepping the distance remaining to clean data along that straight line (t)

# This should make sense!




$$\mathbf{x}_{t_{\text{next}}} = \mathbf{x}_t + (t_{\text{next}} - t)\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t}$$

**This gives us our simplest sampling algorithm: Euler's method! Given predefined timesteps, get derivative at current point, and step *distance* from current time to next time**



# So to recap

- Diffusion models equivalently learn to **denoise** data, and also to model the **score** (gradient) of the map of noise to data
- Denoising gives us a clean, simple, scalable **training** method
- Score matching gives us a flexible way of **sampling** from the model
  - We can use *any* ODE/SDE solver to sample from a diffusion model!

# Next time...

Hopefully we have a good sense of how diffusion works now...

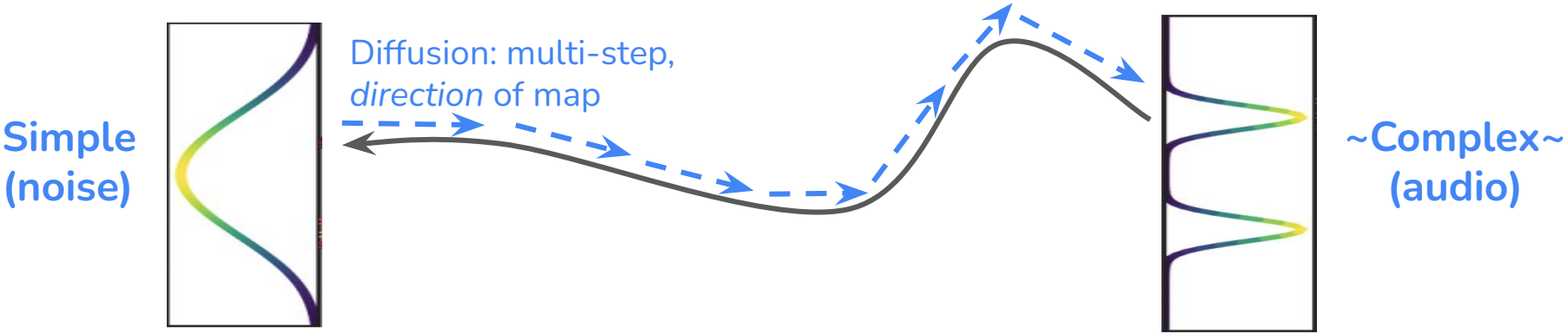
1. How does this apply to musical audio?
2. Is this the only approach for SOTA systems?



# Audio-Domain Music Generation

4.4.5: Diffusion Models for Music Generation (part 2)

# We've seen how diffusion works broadly



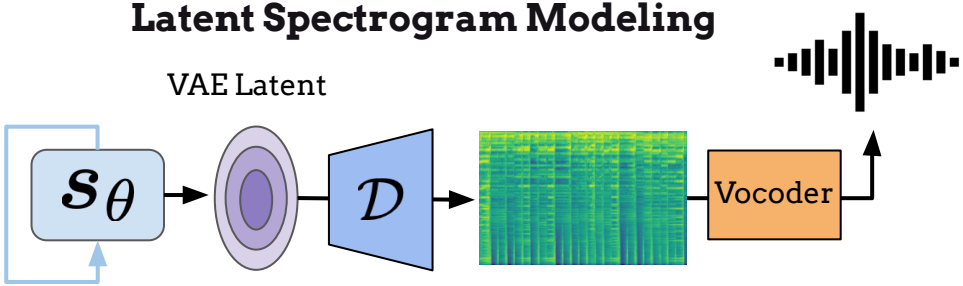

But how do we use this for text-conditioned audio-domain music generation?

Zachary Novack
Ke Chen

# Diffusion TTM - Representation

## What is $\boldsymbol{x}_{\text{data}}$ ?

### Latent Spectrogram Modeling

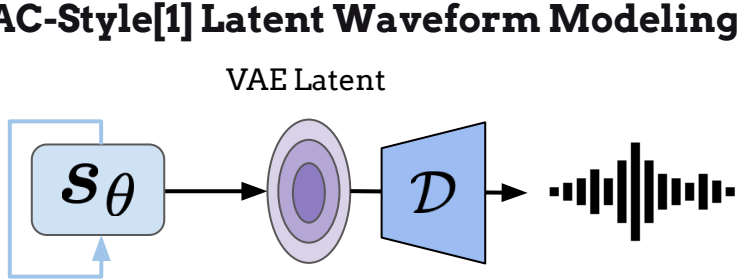



$$\boldsymbol{x}_{\text{data}} \in \mathbb{R}^{D_h \times D_w \times D_c}$$

$$D_h, D_w \ll H, W \quad D_c > C$$

- Continuous 2D latent from spectrogram VAE
- AudioLDM-1/2, Riffusion

### DAC-Style[1] Latent Waveform Modeling



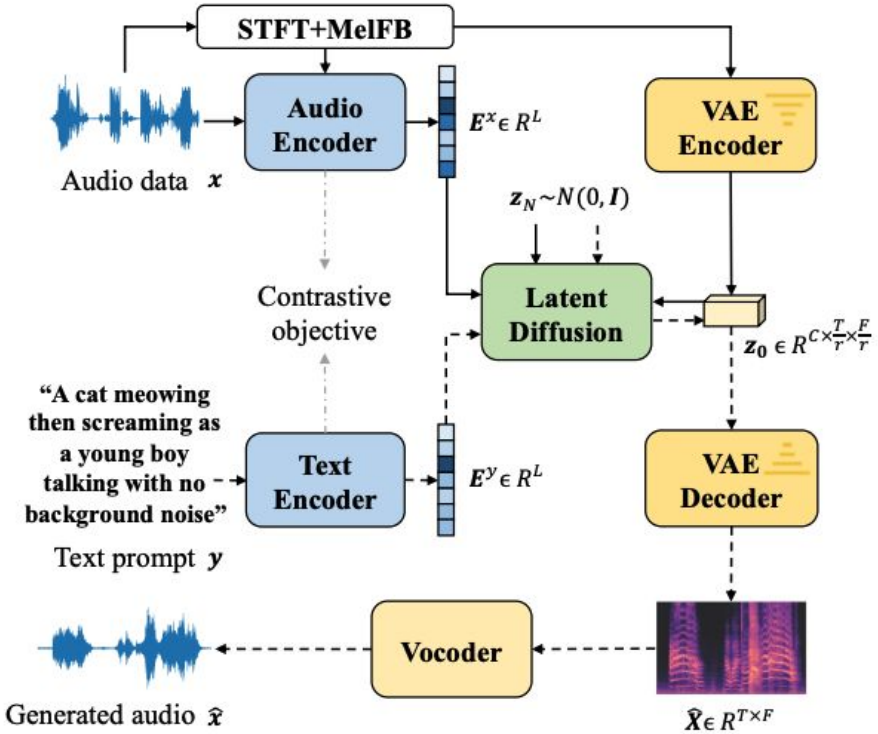

$$\boldsymbol{x}_{\text{data}} \in \mathbb{R}^{D_T \times D_c}$$

$$D_T \ll f_s T \quad D_c > C$$

- 1D *continuous* latent from **waveform** VAE
- Like DAC/EnCodec (VQGAN), **but no quantization** (only KL reg.)
- **SAO**

[1] Rithesh Kumar et al., "High-Fidelity Audio Compression with Improved RVQGAN"

# Latent Spectrogram Modeling

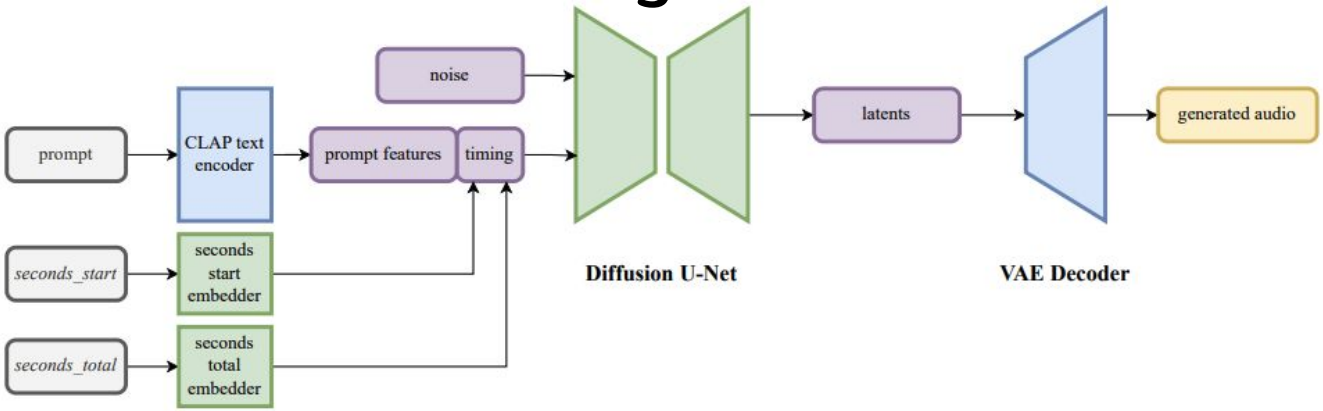



It's just like images!

- Diffusion happens *inside* VAE latent space
- Start w/gaussian noise the same shape as our VAE latents, and gradually denoise them into "clean" VAE latents
- After generating, Decoder->spectrogram->Vocoder->Audio

*https://arxiv.org/pdf/2301.12503*

# Latent Waveform Modeling




Same thing as before, but with 1D VAE latents

- Equivalent to a “1D image”, like a single pixel row of an image
- Decoder->audio

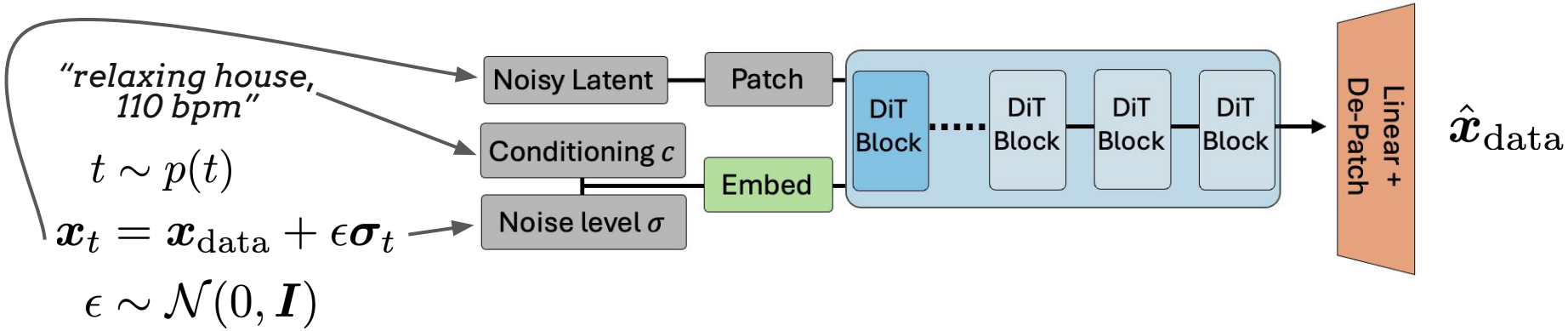
*https://arxiv.org/pdf/2404.10301v2*

But how can we make these models text-conditioned?

Zachary Novack
Ke Chen

# Diffusion TTM - Architecture

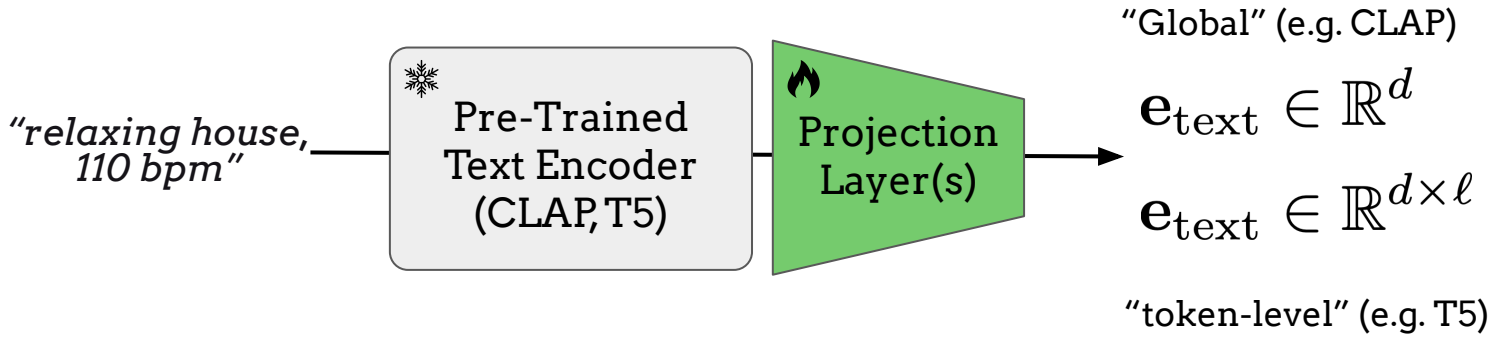
## Diffusion Transformer (DiT) [1]



Zachary Novack
Ke Chen

# Diffusion TTM - Conditioning

## Step (1): Embedding

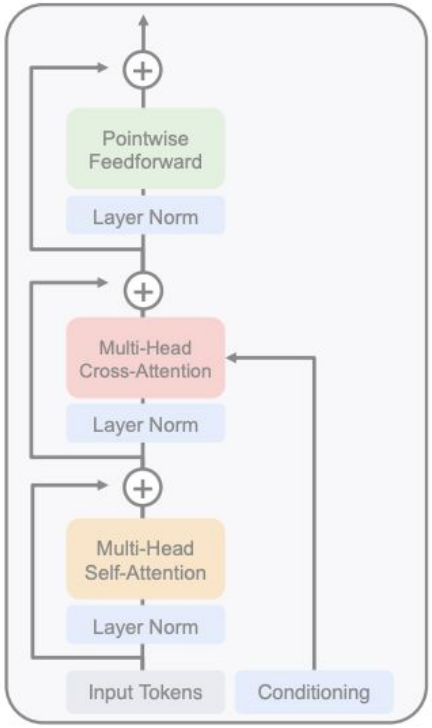


Zachary Novack
Ke Chen

# Diffusion TTM - Conditioning

Options:

- **Cross Attention**
  - Perform attention operation of latent with text embeddings
  - Pros: Most performant
  - Cons: Expensive




DiT Block with Cross-Attention

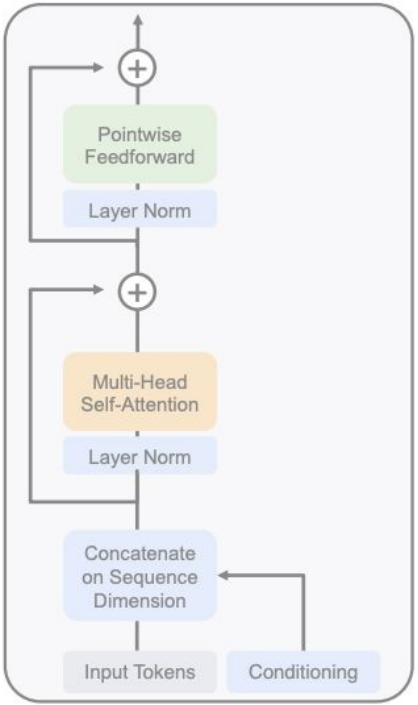
[1] Peebles and Xie, “Scalable Diffusion Models with Transformers”.

Zachary Novack
Ke Chen

# Diffusion TTM - Conditioning

Options:

- **In-Context Conditioning**
  - Controls are concatenated to input as “extra features”, and perform all operations w/input throughout the model
  - Pros: Strong performance, no added parameters vs. cross attention
  - Cons: Slow and Expensive




DiT Block with In-Context Conditioning

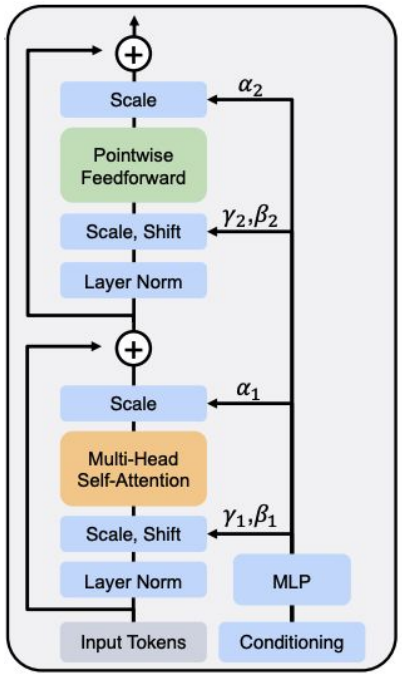
[1] Peebles and Xie, “Scalable Diffusion Models with Transformers”.

Zachary Novack
Ke Chen

# Diffusion TTM - Conditioning

Options:

- **Adaptive Layer Norm** (AdaLN):
  - learn per-channel **Shift** and **Scale** parameters based on conditions
  - Pros: cheapest / fastest to run
  - Cons: Significant added parameter count, less expressive




DiT Block with adaLN-Zero

[1] Peebles and Xie, “Scalable Diffusion Models with Transformers”.

# What is Classifier-Free Guidance?

- Let's say we have our model, but its outputs are ~meh~ quality and don't follow our prompt well, can we improve it w/o training?
  - Can we sample from a distribution that is "heightened" towards our prompt w/strength gamma?

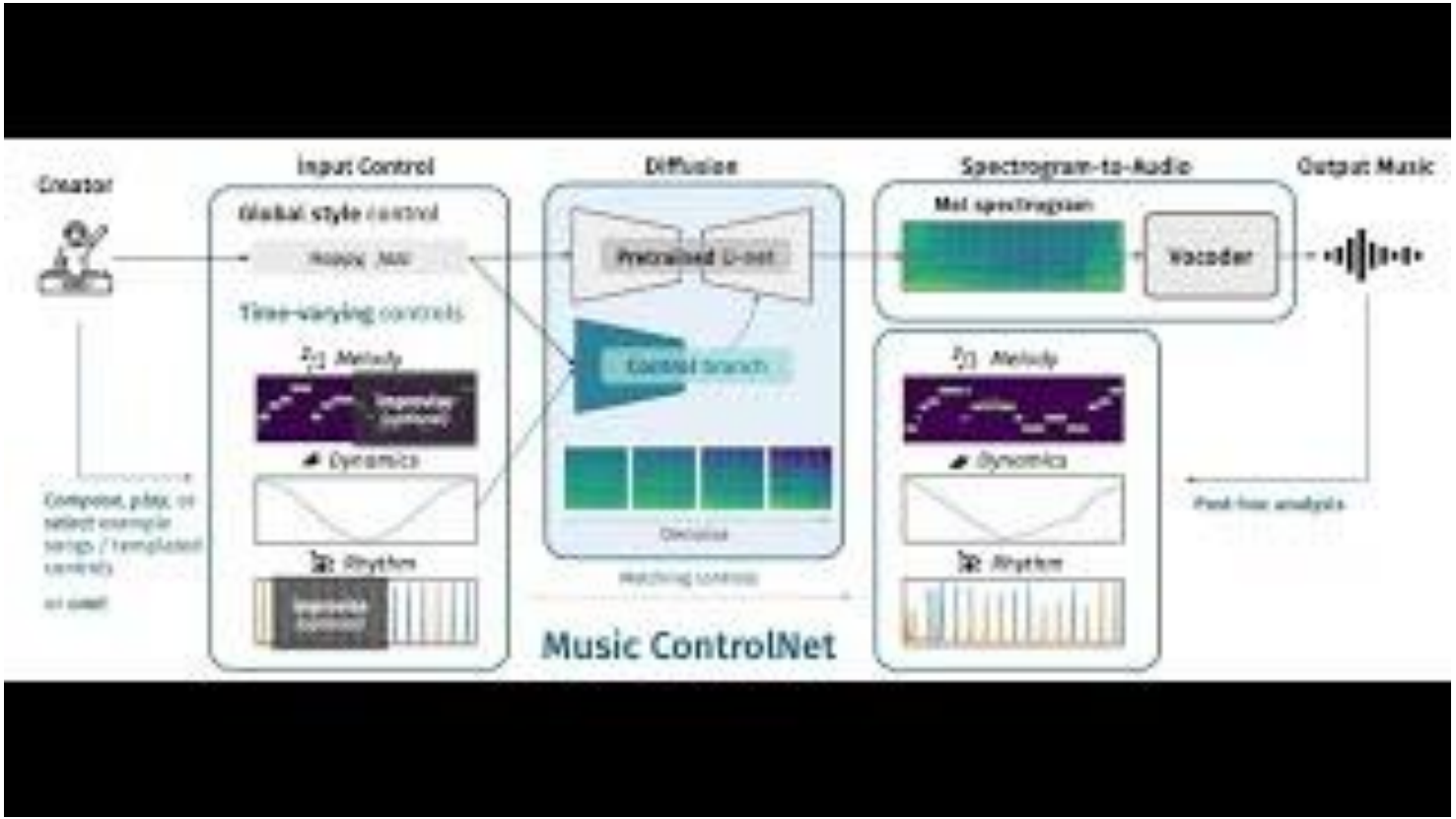
$$\nabla_{x_t} \log p^{\gamma}(x_t|y) = \nabla_{x_t} \log p(x_t) + \gamma(\nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t))$$

**Tldr; we use the *difference* of our model w/ and w/o text conditions, pushing outputs "more towards" our text prompt**

# Applications



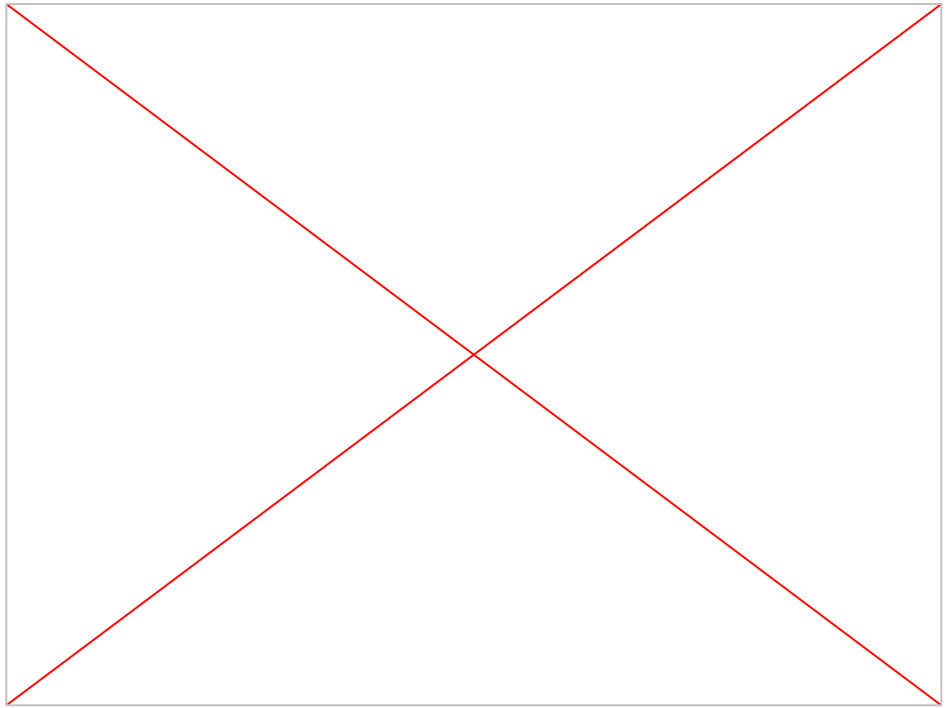
# Applications



# Applications

Stable Audio Open Small! Optimized for 11s CPU inference!

“Accelerated diffusion”
Open source here:



# Audio-Domain Music Generation

4.5: Discrete Audio-Domain Music Generation

# How can we model a continuous modality discretely?

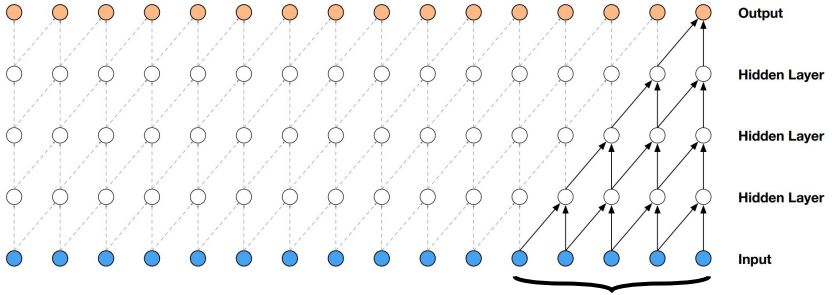
We've seen discrete models of discrete modalities (symbolic music),
And continuous models of continuous modalities (audio-domain music)...

How can we have **discrete** models of **continuous** modalities?

**Quantization!** If we can find a *quantized* representation of continuous signals, we can train a model over these signals like we would in text or symbolic music!

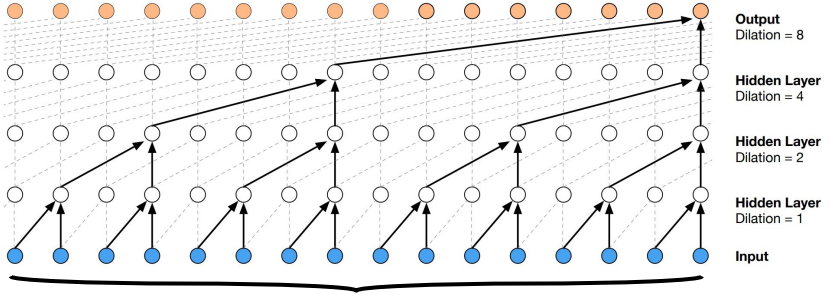
# Example: WaveNet (van den Oord et al., 2016)

**Standard convolution**




**Smaller "receptive field"**

**Dilated convolution**



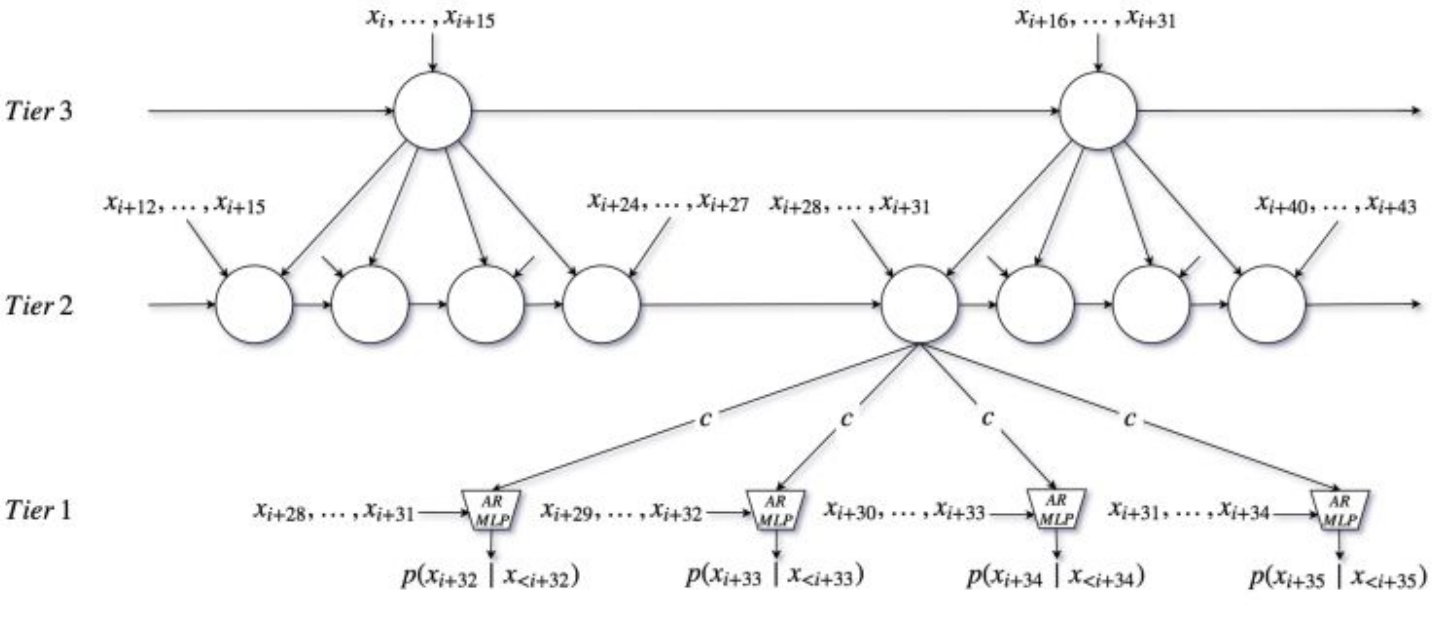

**Larger "receptive field"**

deepmind.google/discover/blog/wavenet-a-generative-model-for-raw-audio

(Source: van den Oord et al., 2016)

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *ICML*, 2016.

# Example: SampleRNN (Mehri et al., 2017)




**Aggregate "context" from preceding samples into fixed-sized hidden state**

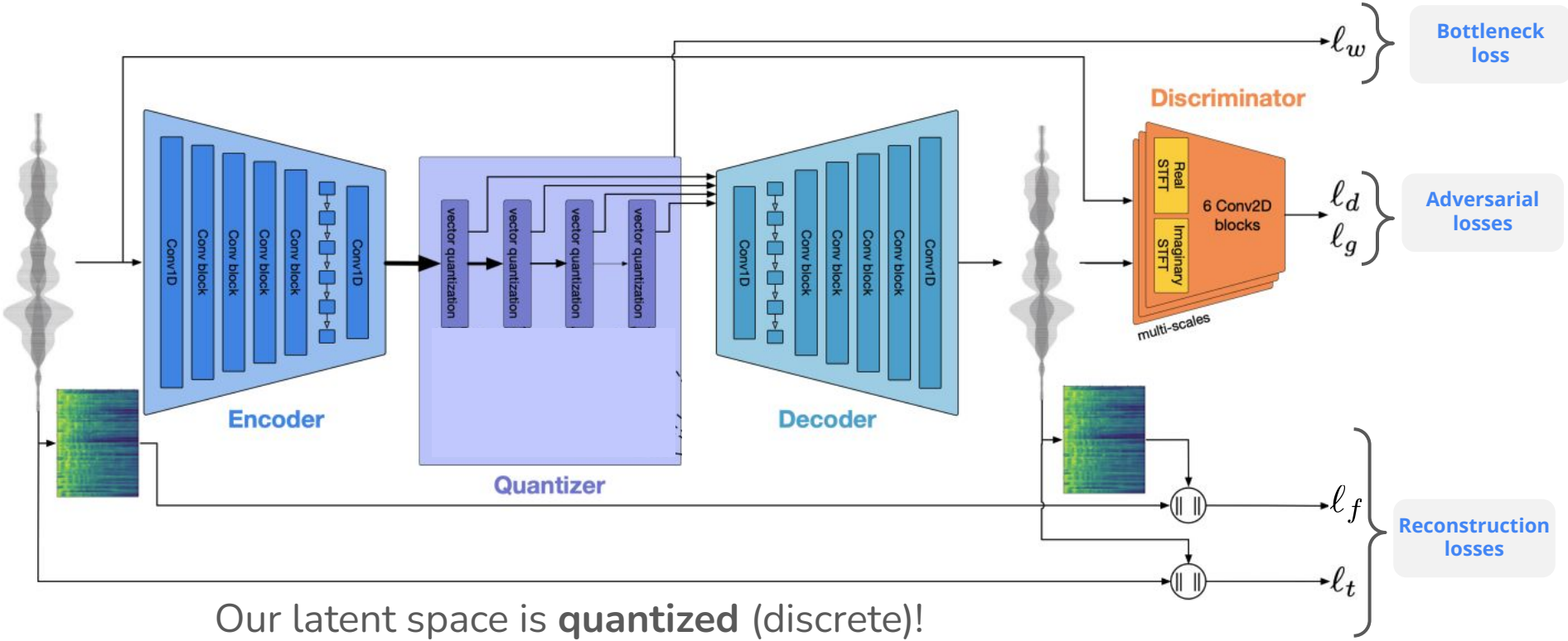
**Predict next sample given last "frame" of samples + current hidden state**

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., ... & Bengio, Y. (2017). SampleRNN: An unconditional end-to-end neural audio generation model. ICLR 2017.

# These were discrete (quantized) models!

# But how do we get *better* representations?

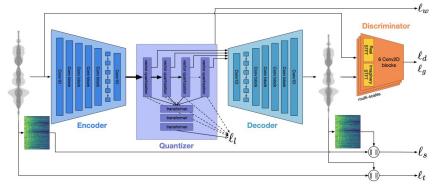
# (Residual) Vector Quantization!



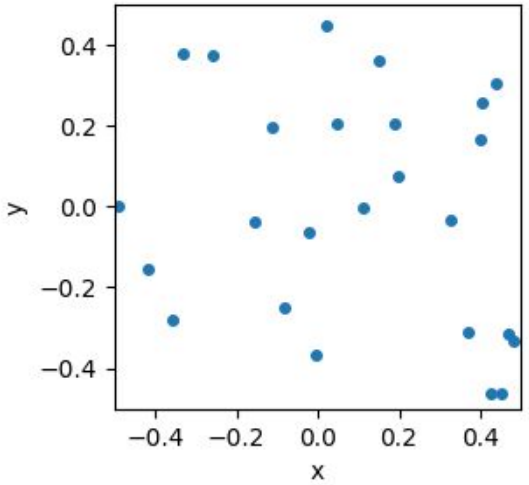

Our latent space is **quantized** (discrete)!

*https://arxiv.org/pdf/2210.13438*

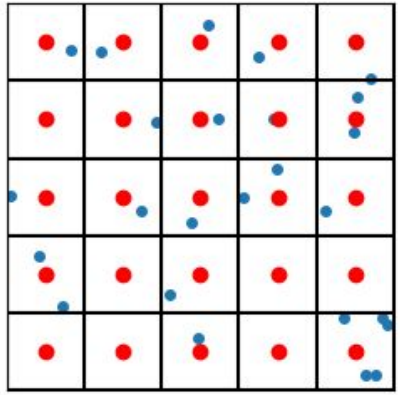
# (Residual) Vector Quantization



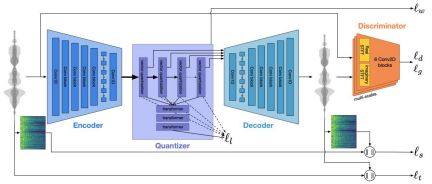

## Vector Quantization:




- Given fixed number of K codes:
- Partition data into K “regions”, each defined by a centroid
  - e.g. K-means
- Map any **continuous** data point (blue) to **index** (discrete) denoting nearest centroid (red)
- Even if 2D, we collapsed infinite possible points into 25 values!
- Gradients propagate with ST-Estimator

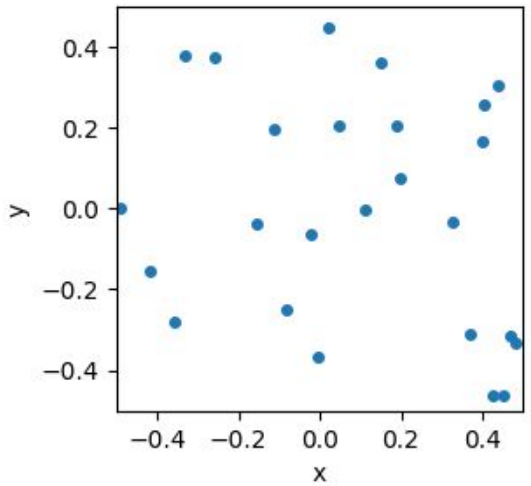


# (Residual) Vector Quantization

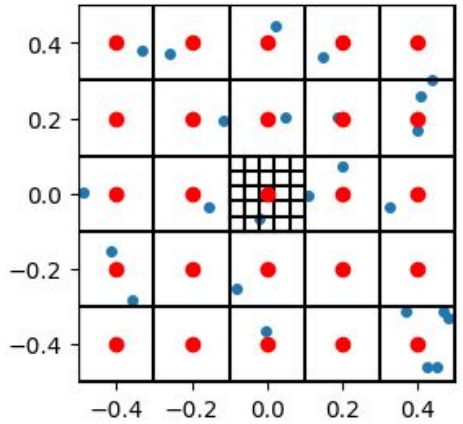



## Residual Vector Quantization:

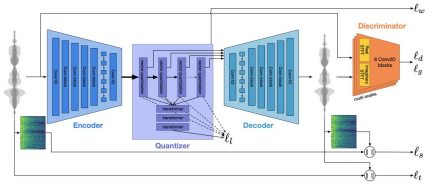
- These approximations are pretty bad, can we do any better?
- Naïve idea, more codes at finer resolution! But expensive :(
- Better: store **residual** codes (i.e. codes at different **levels** of fineness)




- Think about how we write numbers:
  - 253 is 3*1 + 5*10 + 2*100
  - Way better than 253 1s!
- **Ordered levels** of codebooks (coarse->fine)



# (Residual) Vector Quantization



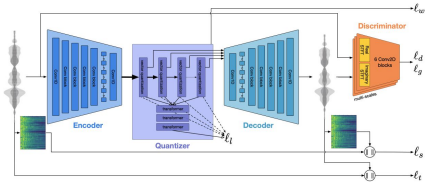

**Reconstruction Loss:**

- Reconstruct outputs based on raw waveform
- Reconstruct outputs based on multi-resolution spectrogram
- “Learn a **discrete** representation that reconstructs many views of the input”

**Adversarial Loss:**

- Multi-resolution spectrogram discriminator
- “Learn a **discrete** representation that is ‘real enough’ according to the discriminator”

# (Residual) Vector Quantization




**Bottleneck Loss:**

- How do we learn good **quantized codes**, if we have to learn the **unquantized embeddings** at the same time???
- **Commitment loss:**

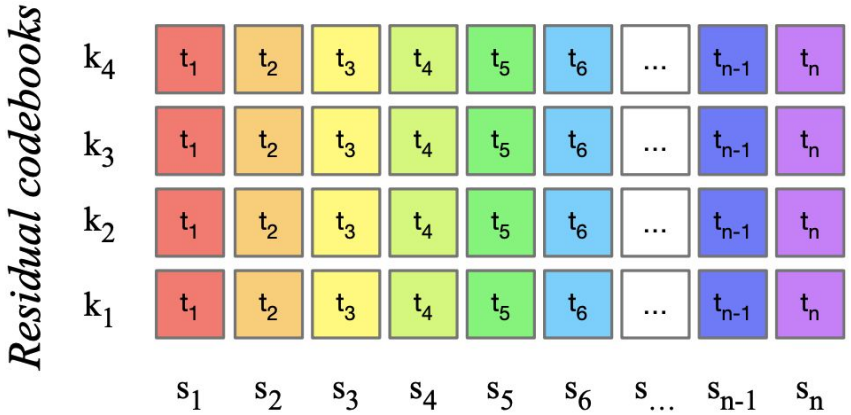
$$l_w = \sum_{c=1}^{C} \|\boldsymbol{z}_c - q_c(\boldsymbol{z}_c)\|_2^2.$$

- Learn an encoder such that *the embeddings are close to their respective centroids* (i.e. our data clusters well by the number of codes)

# Generative Models in Quantized Spaces

Now that we have some (discrete) latent space, we can just train a language model / transformer / RNN over it right?

Wrong!

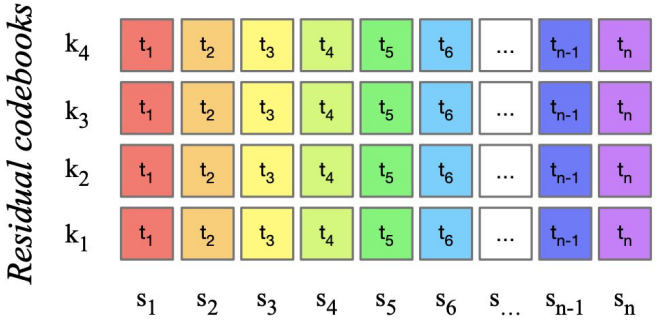



Our data is **# of compressed timesteps x # of codes x # number of RESIDUAL LEVELS**, so how do we predict multiple levels at the same time??

# Generative Models in Quantized Spaces

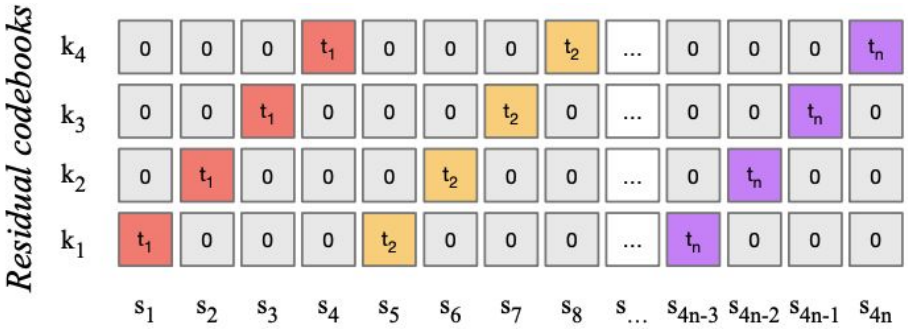
Should we predict each level in **parallel** at each step?

- No! Implies independence across levels at each time, but they're residual, and thus very **dependent!**




Should we **unravel** the levels, and predict each codebook level **separately**?

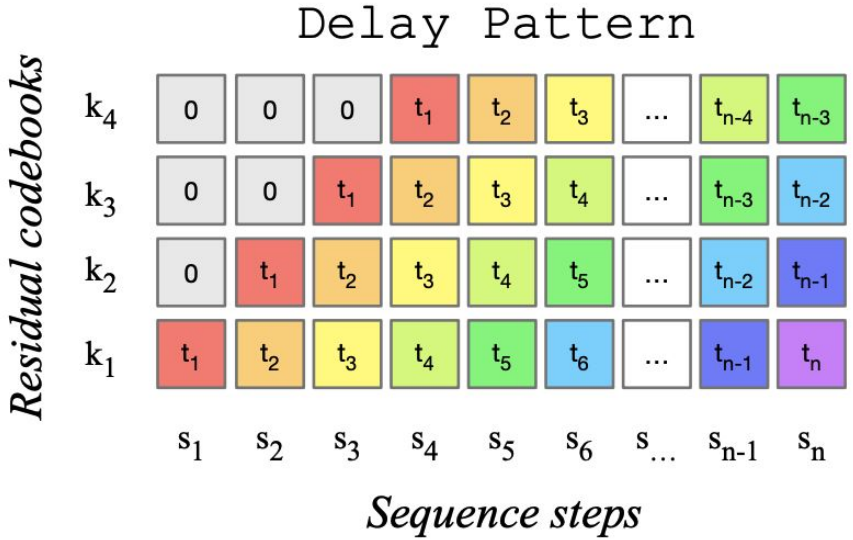
- No! This might be a good model, but it **multiplicatively** increases sequence length!
- More levels->slower and slower



# Generative Models in Quantized Spaces

How about we compromise?

- First predict **coarsest** codebook at time 1
- Then predict coarsest codebook at time 2 AND next-coarsest at time 1
- Then predict coarsest at time 3, next-coarsest at time 2, and next-next-coarsest at time 1,
- And so on...
- Predict multiple **levels** at once to not increase sequence length
- Have “some amount” of conditioning on previous **levels** to model dependence



Zachary Novack
Ke Chen

# MusicGEN - Generation



## Autoregressive Modeling [1]

Architecture:

[Input]: audio codec tokens $\{v_1, v_2, ..., v_{T_v}\}$

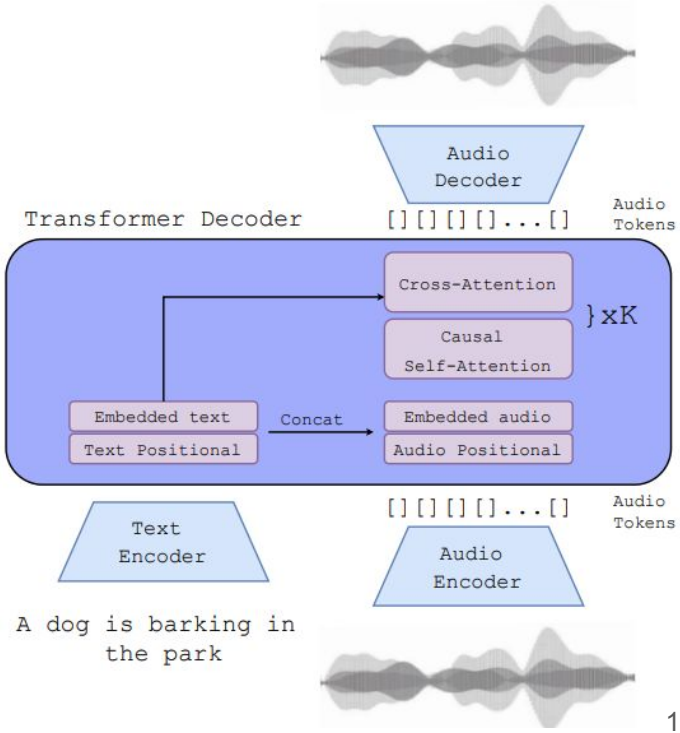
[Model]: transformer decoder

$$p_\theta(v_t | v_1, v_2, ..., v_{t-1})$$

[Output]: generated audio codec tokens

Objectives: cross-entropy loss

$$L = -\sum_{i=1}^{N} \sum_{t=1}^{T_v} p_\theta(v_t^i | v_1^i, v_2^i, ..., v_{t-1}^i)$$




[1] Felix Kreuk et al., "AudioGEN: Textually Guided Audio Generation", ICLR 2023.

Zachary Novack
Ke Chen

# MusicGEN - Conditioning



## Time-domain Concatenation / Prefix

Architecture:

[Input]:

Audio codec tokens $\{v_1, v_2, \ldots, v_{T_v}\}$

T5 text tokens $\{u_1, u_2, \ldots, u_{T_u}\}$

[Model]: Transformer decoder

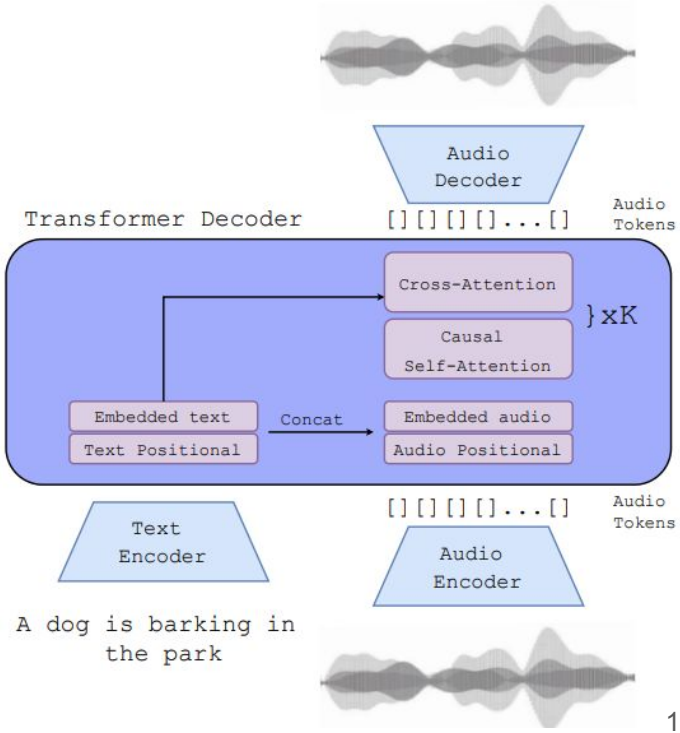
$$p_\theta(v_t | u_1, u_2, \ldots, u_{T_u}, v_1, v_2, \ldots, v_{t-1})$$

[Output]: generated audio codec tokens

## Cross-Attention

Keys and Values (K,V) come from the text encoder states

Queries (Q) come from the audio encoder states

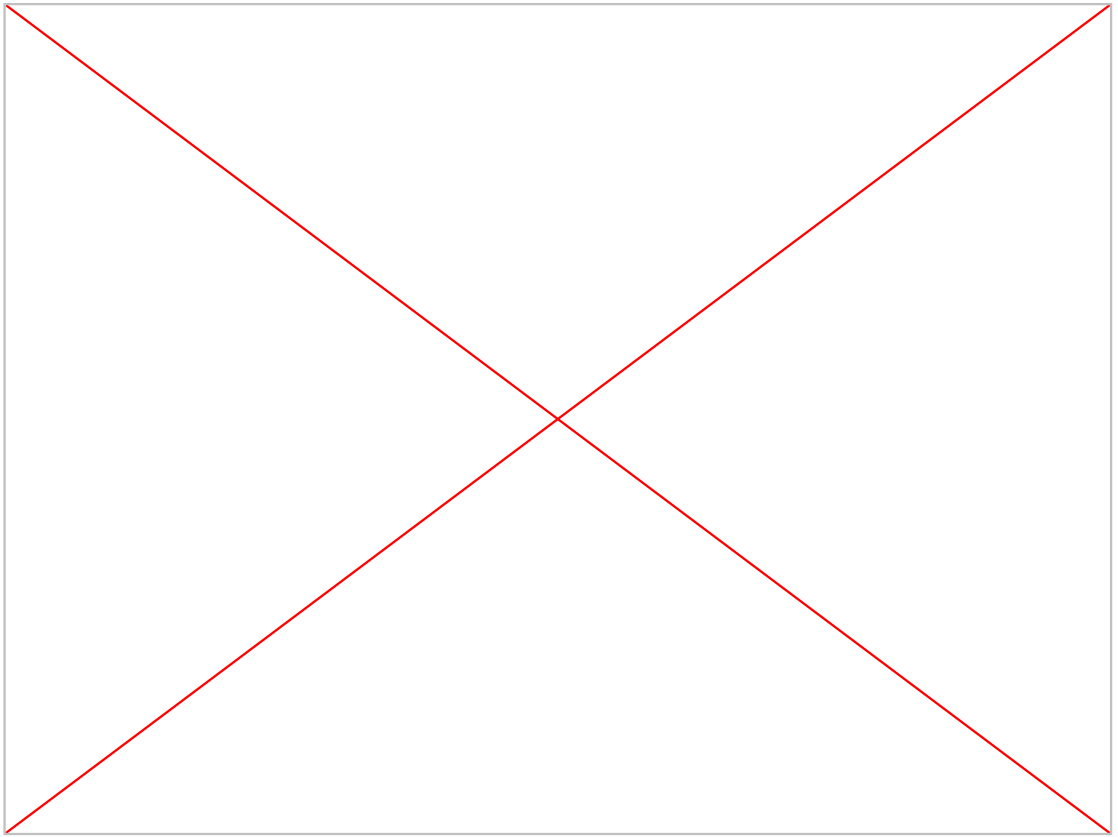



[1] Felix Kreuk et al., "AudioGEN: Textually Guided Audio Generation", ICLR 2023.

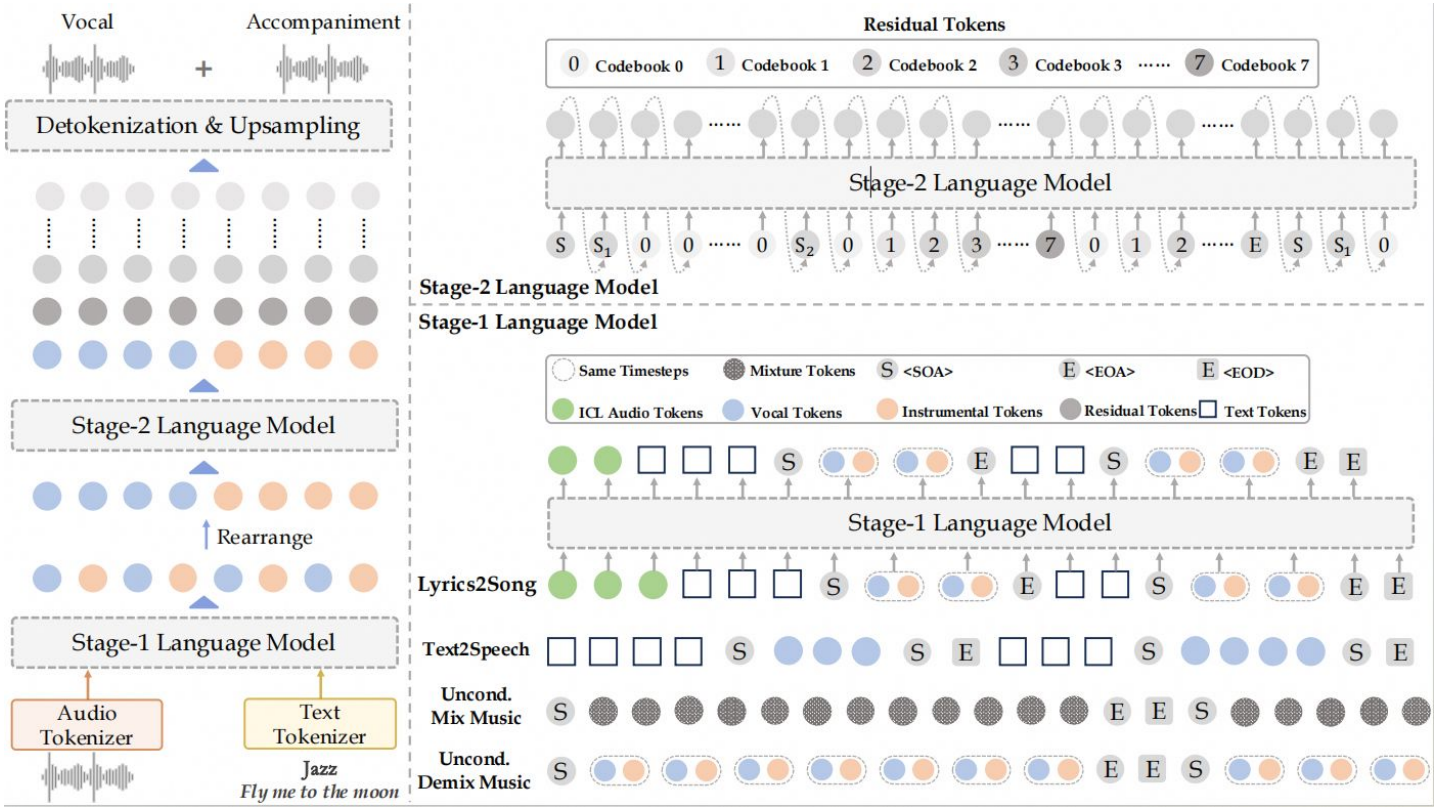
Zachary Novack
Ke Chen

# MusicGEN - DEMO





# Applications: YuE



# Applications: MusicFX DJ (I think)



# Audio-Domain Music Generation

4.6: Evaluation and Final Thoughts

# How do we evaluate such models?

Zachary Novack
Ke Chen

# Subjective Listening Test



## MOS (Mean Opinion Score)

Purpose:

[Rate the overall quality of a single audio stimulus]

Method:

[Testers are asked to rate audio stimulus from 1 (bad) to 5 (excellent) via their perception of its quality]

## MUSHRA (Multiple Stimuli with Hidden Reference and Anchor)

Purpose:

[Rate the overall quality among multiple audio stimulus]

Method:

[Testers are asked to rate audio stimulus from 0 (bad) to 100 (excellent)]

[A hidden reference (high-quality version) and an anchor (low-quality version) will also be presented]

## Rating Metrics

- Overall Quality (OVL)
- Relevance to the Input Text (REL)
- Control Liability on Musical Elements (Chord, Melody, Energy, Rhythmic Pattern, etc.)
- Others…

Zachary Novack
Ke Chen

# Audio Diversity



## Inception Score (IS)

Preparation:

[Reference networks on creating audio features]

Purpose:

[Assess the distinctiveness and diversity of audio generation]

Calculation:

$$\mathbf{IS}(G) = \exp\left(\frac{1}{N}\sum_{i=1}^{N} D_{KL}\left(p(y|\mathbf{x}^{(i)})||\hat{p}(y)\right)\right)$$

Averaged Features of N audios (diversity)

Feature of the i-th audio (distinctiveness)

Alternatives:

Recall, Coverage, Vendi Score

Zachary Novack
Ke Chen

# Audio Quality / "Realism"



## Fréchet Inception Distance (FID / FAD)

Preparation:

[Reference networks on creating audio features (VGGish, PANN, CLAP, etc.)]

Purpose:

[Assess the similarity (quality) of audio to the reference dataset]

Calculation:

$$\mathbf{FID}(G) = ||\mu_x - \mu_g||_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{0.5})$$

Averaged audio features by reference/generator

ConVar. of audio features by reference/generator

Alternatives:

Maximum Mean Discrepancy (MMD), Precision, Density, CLAP

Zachary Novack
Ke Chen

# Text Relevance



## CLAP Score

Purpose:

[Assess the correspondence between the generated audio and the reference text description]

Calculation:

[Dot-product / Cosine-similarity of the generated audio embedding and the reference text embedding ]

[Dot-product / Cosine-similarity of the generated audio embedding and the reference audio embedding]

[The embedding comes from CLAP[1] (contrastive language audio pre-training module)]

[1] Yusong Wu*, Ke Chen* et al., “Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation”, ICASSP 2023

# Closing thoughts

We've seen how generative models of audio used to be built...

How we can learn good representations that enable good generative models...

And how we can train both **continuous** and **discrete** generative models...

Lots of these models (MusicGen, SAO, SAO-Small, YuE) are open source, have fun!

Now for guest lectures :)