



# Современные аудио кодеки

## Всесторонний обзор



- 1.1 Идея векторного квантования (VQ)
- 1.2 Идея остаточного векторного квантования (RVQ)
- 1.3 Возвращение к моделям с одной кодовой книгой
- 1.4 Значимость размера кодовой книги и других гиперпараметров
- 1.5 Решение коллапса кодовой книги
- 1.6 Современные подходы к обучению аудио кодеков
- 1.7 Предложенный подход к обучению своего аудио кодека

# Захар Варфоломеев

- ★ Делаю нейронки для музыки и речи
- ★ Создатель сервиса Audio2MIDI
- ★ Люблю делиться знаниями
- ★ Перешел в ML из backend разработки



# ЧТО И ЗАЧЕМ?

- Модели для сжатия/дискретизации аудио фичей
- Чтобы эффективно передавать данные через сеть, а также для обучения LLM
- Нет проблемы апостериорного коллапса, но есть другие :)

# непрерывное/дискретное

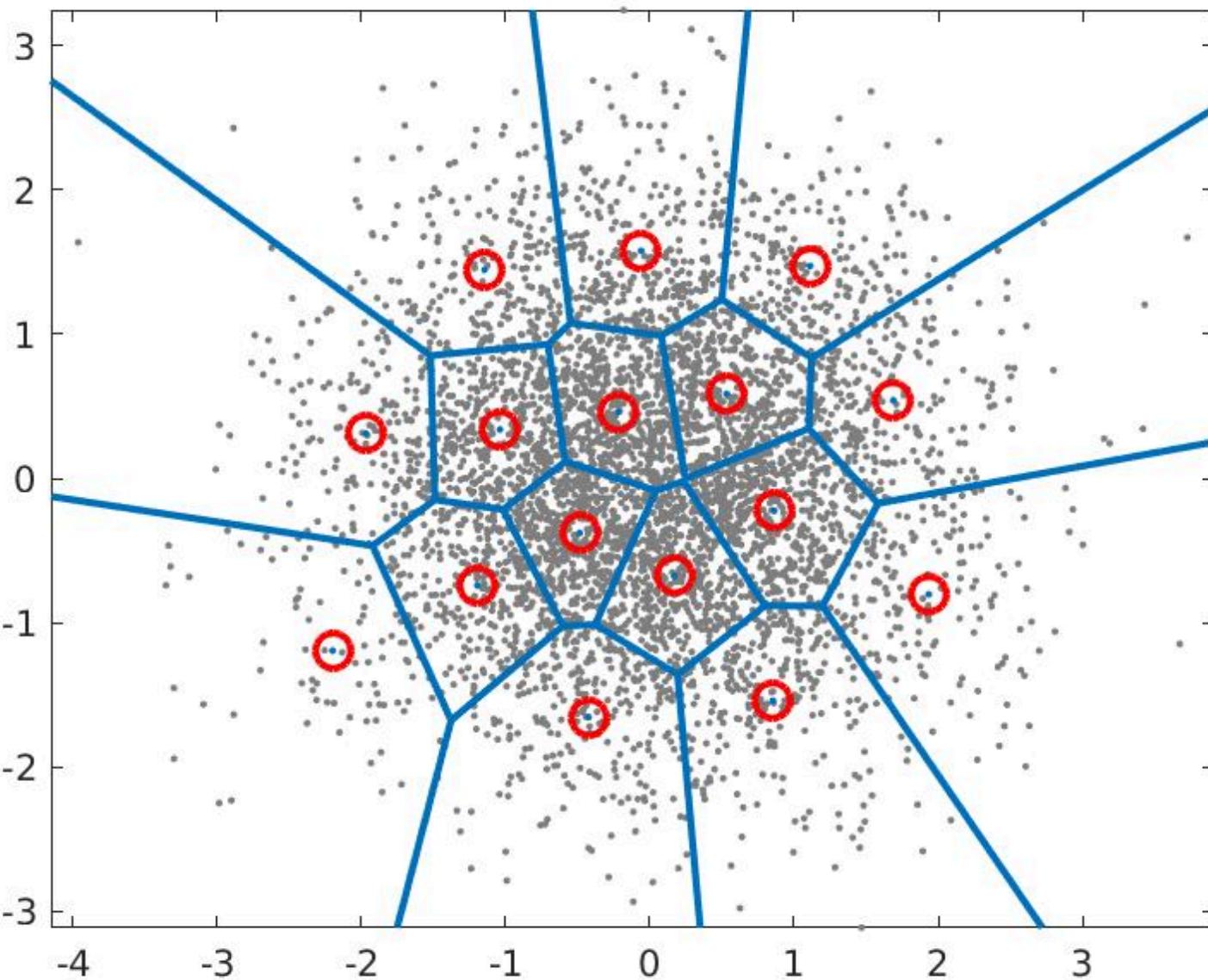
- Есть непрерывное латентное пространство, есть дискретное
- Можем сжимать сырое аудио или спектrogramмы (STFT, Mel, CQT...)
- Дистиллировать фичи из предобученных SSL моделей
- Добавлять другие модальности (текст, картинки, видео)
- Можете делать что угодно...
- Поверх фичей можно брать любые технологии
- Обе стороны могут давать отличное качество
- Выбирайте на основе ваших ресурсов и готовности к эмпирическим тестам

# Идея векторного квантования

Векторное квантование отображает  $k$ -мерные векторы из векторного пространства  $\mathbb{R}^k$  в конечное множество векторов  $Y = \{y_i: i = 1, 2, \dots, N\}$ . Каждый вектор  $y_i$  называется кодовым вектором или кодовым словом, а множество всех кодовых слов называется кодовой книгой.

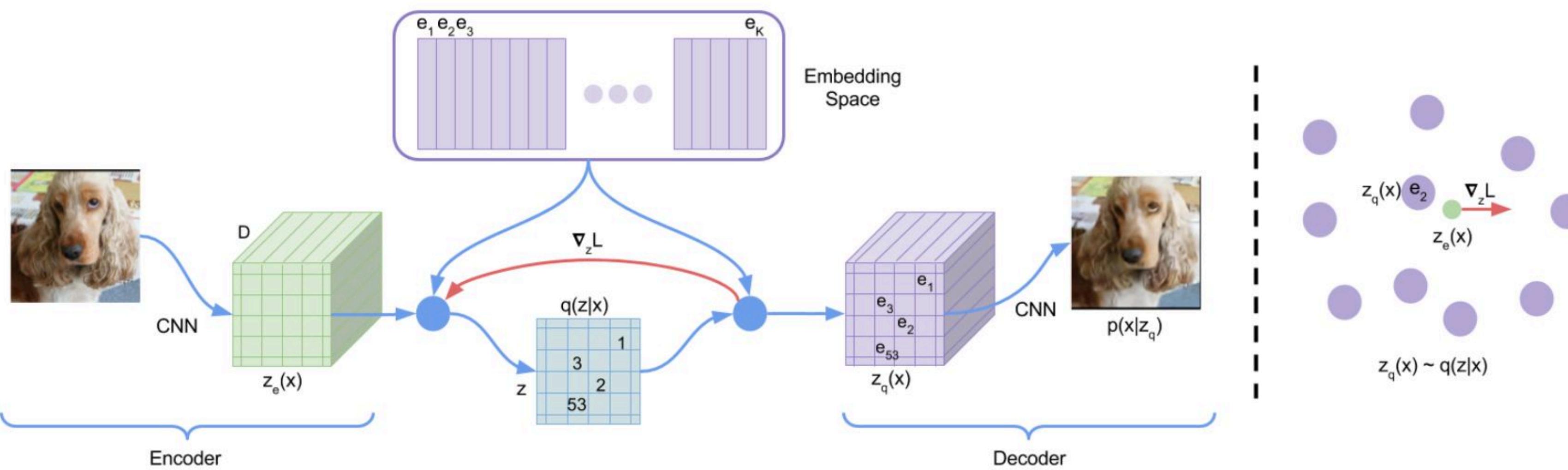
# Идея векторного квантования

Векторное квантование отображает  $k$ -мерные векторы из векторного пространства  $\mathbb{R}^k$  в конечное множество векторов  $Y = \{y_i: i = 1, 2, \dots, N\}$ . Каждый вектор  $y_i$  называется кодовым вектором или кодовым словом, а множество всех кодовых слов называется кодовой книгой.



# vq-vae

Квантуем непрерывное латентное пространство VAE. По сути мы ограничиваем количество эмбеддингов, которые могут пройти в декодер теми, что есть в кодовой книге.



# vq-vae

Операция VQ не дифференцируемая, поэтому нужны вспомогательные лоссы 3-4. Мы взаимно приближаем фичермапы и коды ПОСЕМПЛОВО (это проблема, почему?).

$$\mathcal{L} = \mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{codebook}} + \beta \mathcal{L}_{\text{commit}} \quad (1)$$

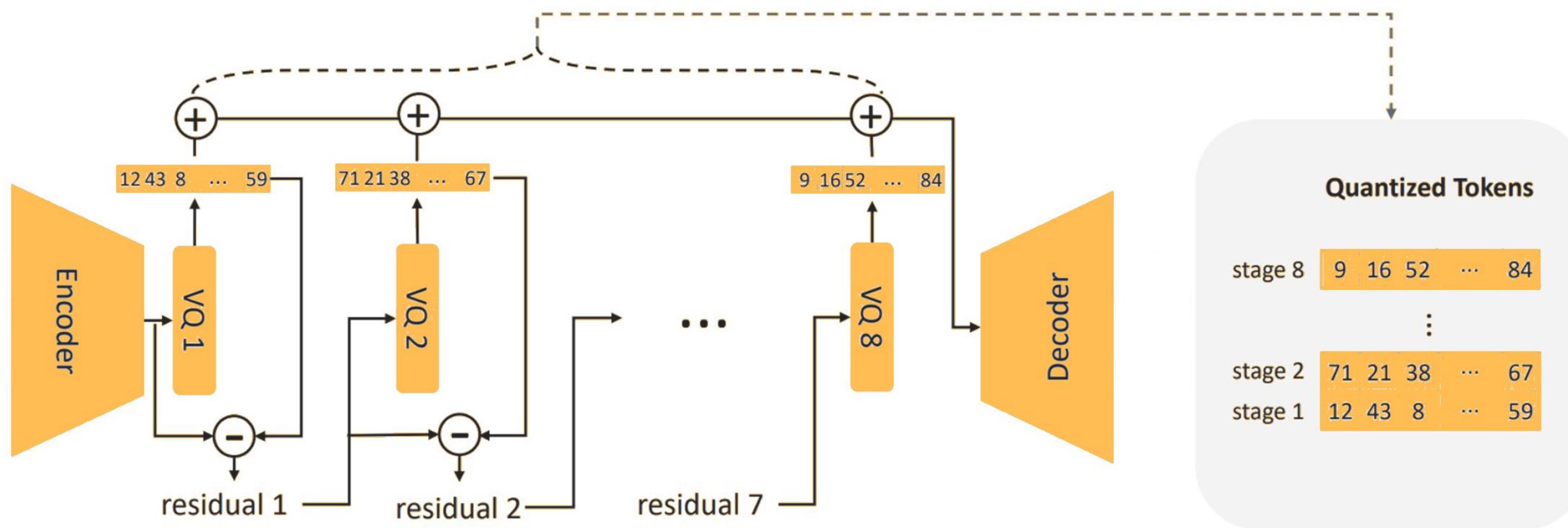
$$\mathcal{L}_{\text{recons}} = \frac{1}{T} \sum_t \|\mathbf{x}_t - D(\mathbf{e}_{z_t})\|_2^2 \quad (2)$$

$$\mathcal{L}_{\text{codebook}} = \frac{1}{S} \sum_s \|\text{sg}[\mathbf{h}_s] - \mathbf{e}_{z_s}\|_2^2 \quad (3)$$

$$\mathcal{L}_{\text{commit}} = \frac{1}{S} \sum_s \|\mathbf{h}_s - \text{sg}[\mathbf{e}_{z_s}]\|_2^2 \quad (4)$$

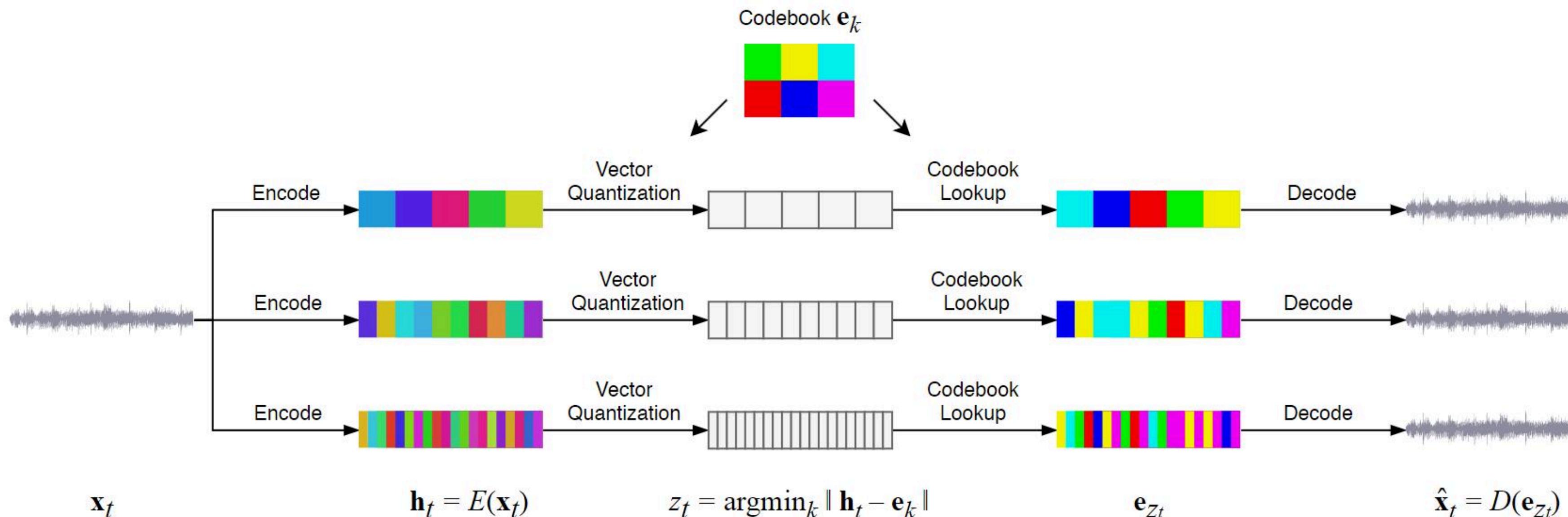
# ХОТИМ БОЛЬШОЙ КОДБУК

Для большинства задач размер кодовой книги не мог превышать  $2^{12}$ - $2^{13}$ . Однако нужен кодбук больше, иначе не сможет хорошо покрыть домен! Решение есть: использовать остаточное векторное квантование (RVQ):



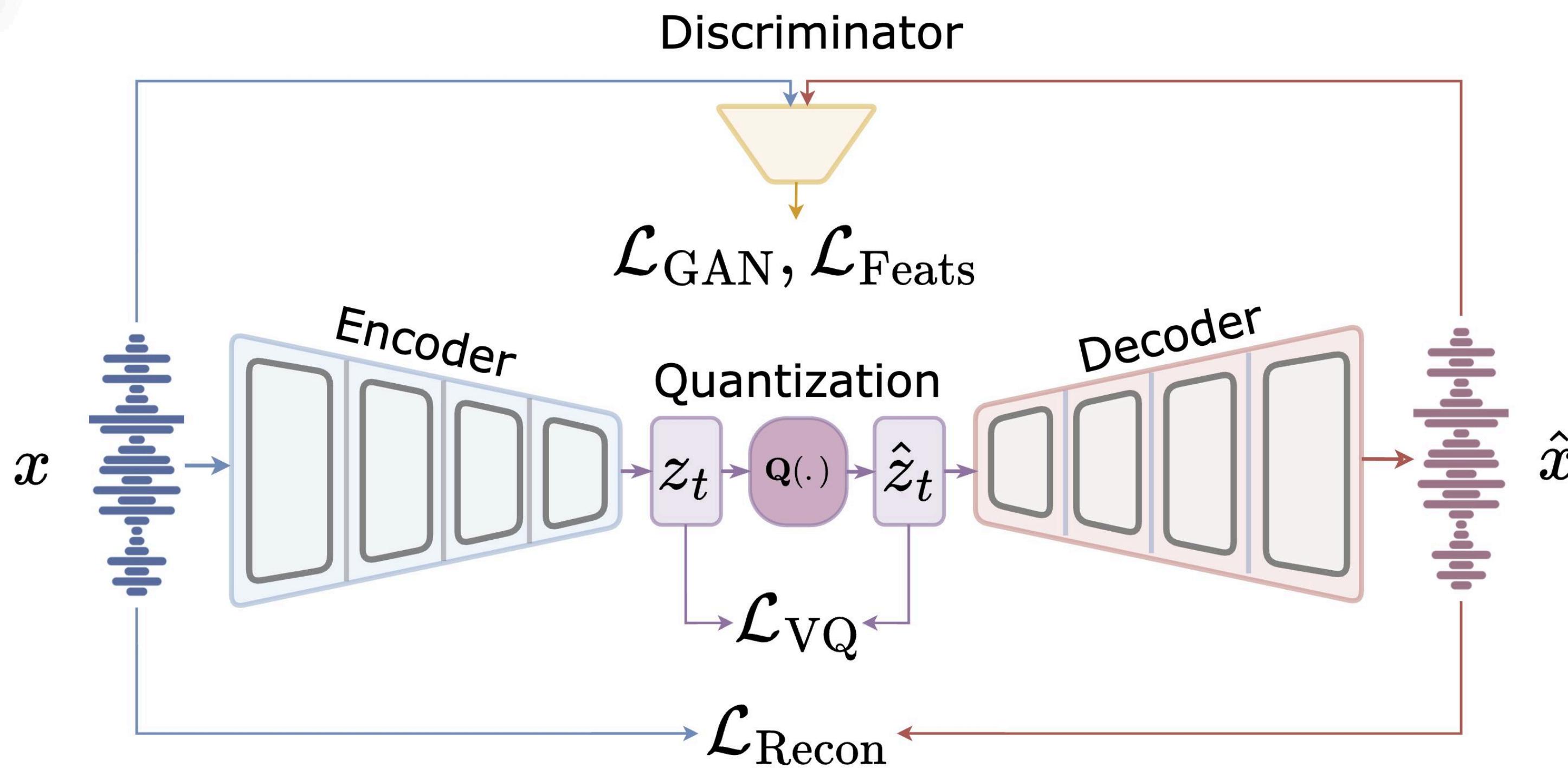
# VQ-VAE внутри JukeBox

Параллельно обучаем 3 уровня детализации и последовательно генерируем токены следующего уровня на основе текущего



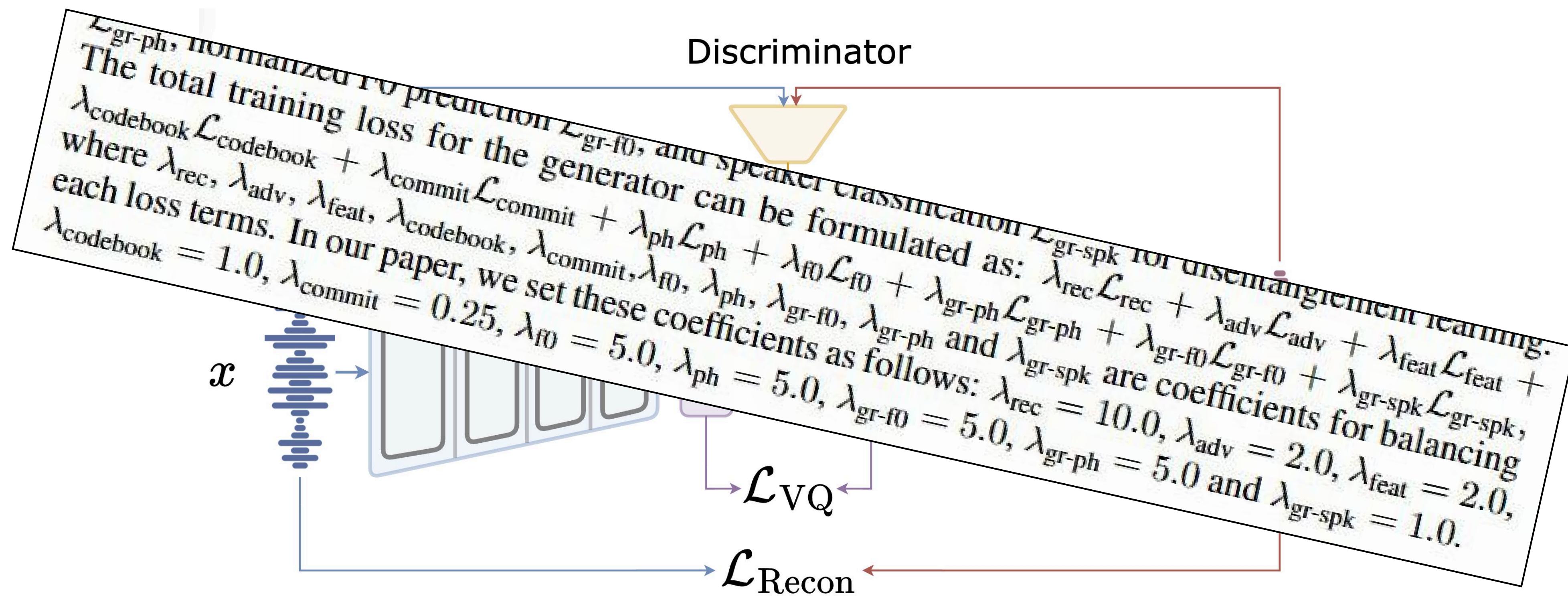
# возвращение к одному кодбуку

Добавляем мощный декодер и дискриминатор. Порой оптимизируется 10 лоссов одновременно! Испугался?



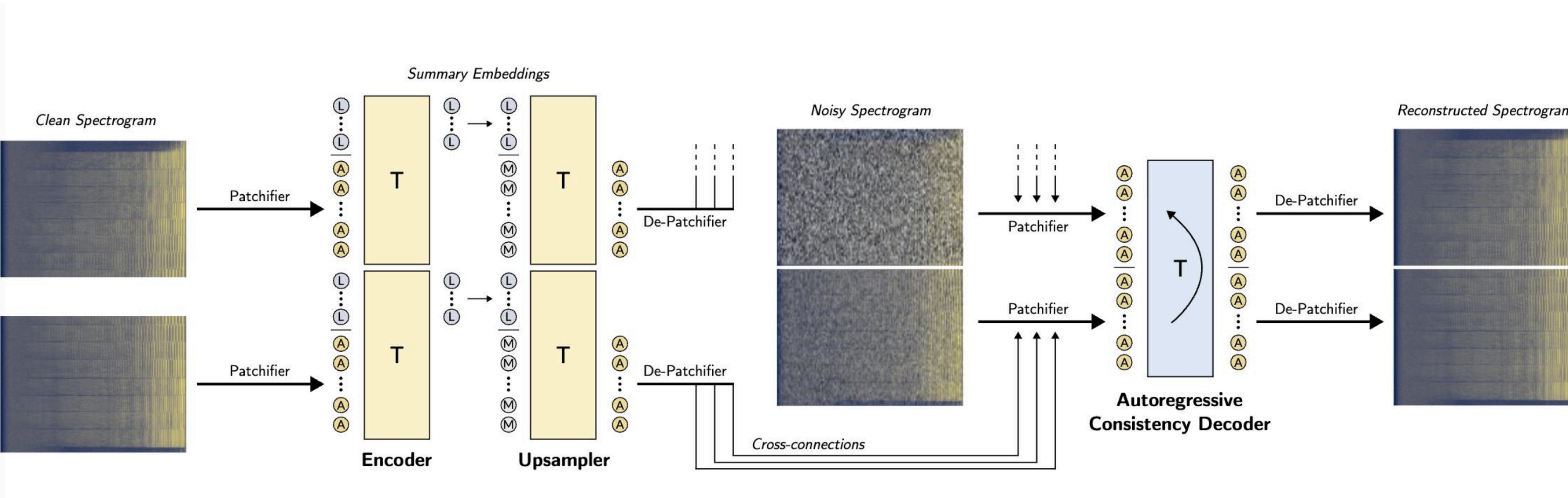
# возвращение к одному кодбуку

Добавляем мощный декодер и дискриминатор. Порой оптимизируется 10 лоссов одновременно! Испугался?



# диффузионный кодек

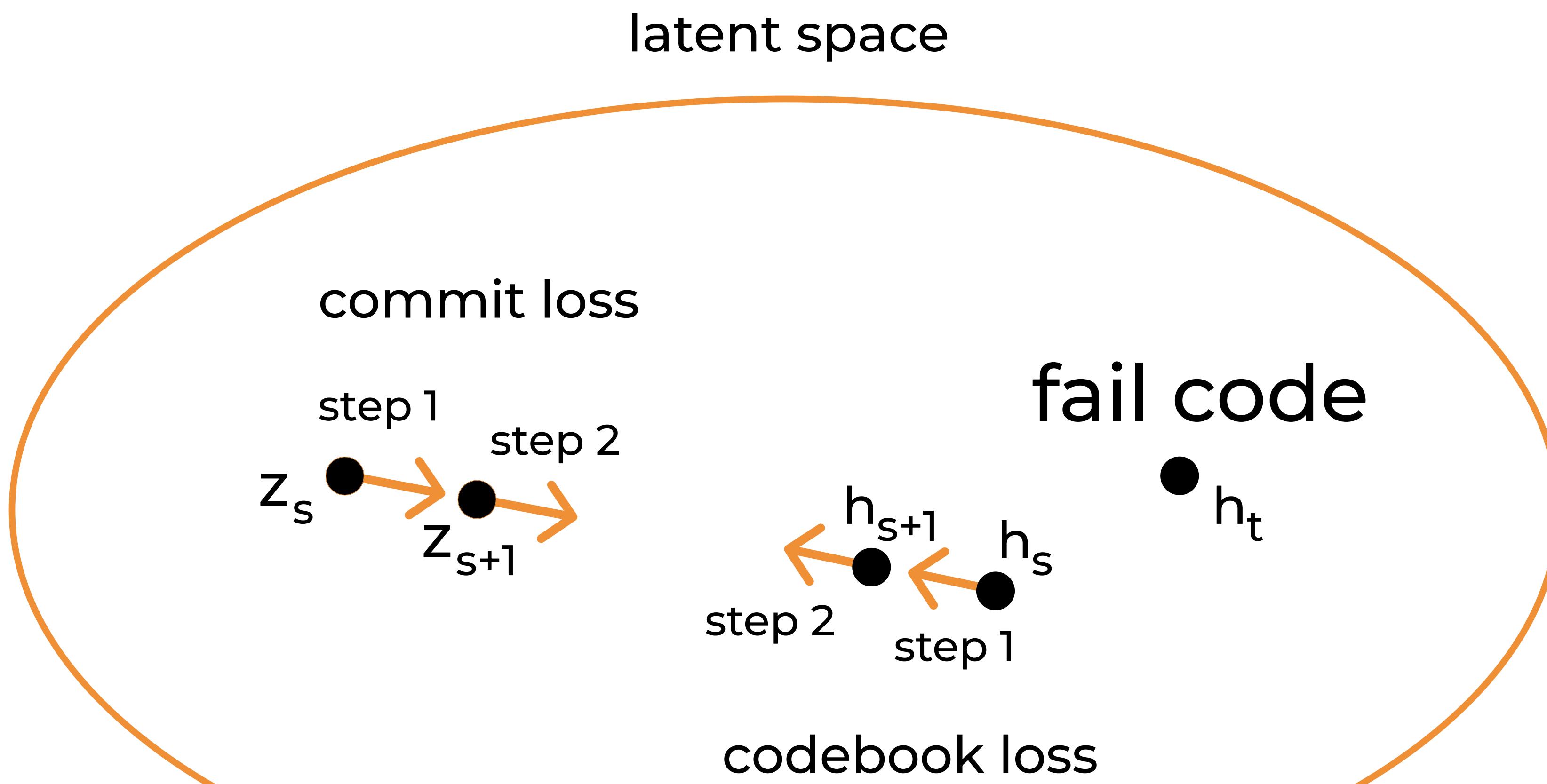
Можем обучать диффузионный кодек с одним лишь *consistency loss*, однако и тут нужно тщательно подбирать гиперы для диффузии. Радует, что здесь нет 10 весов для лоссов... Модель называется CoDiCodec.



# коллапс кодбука

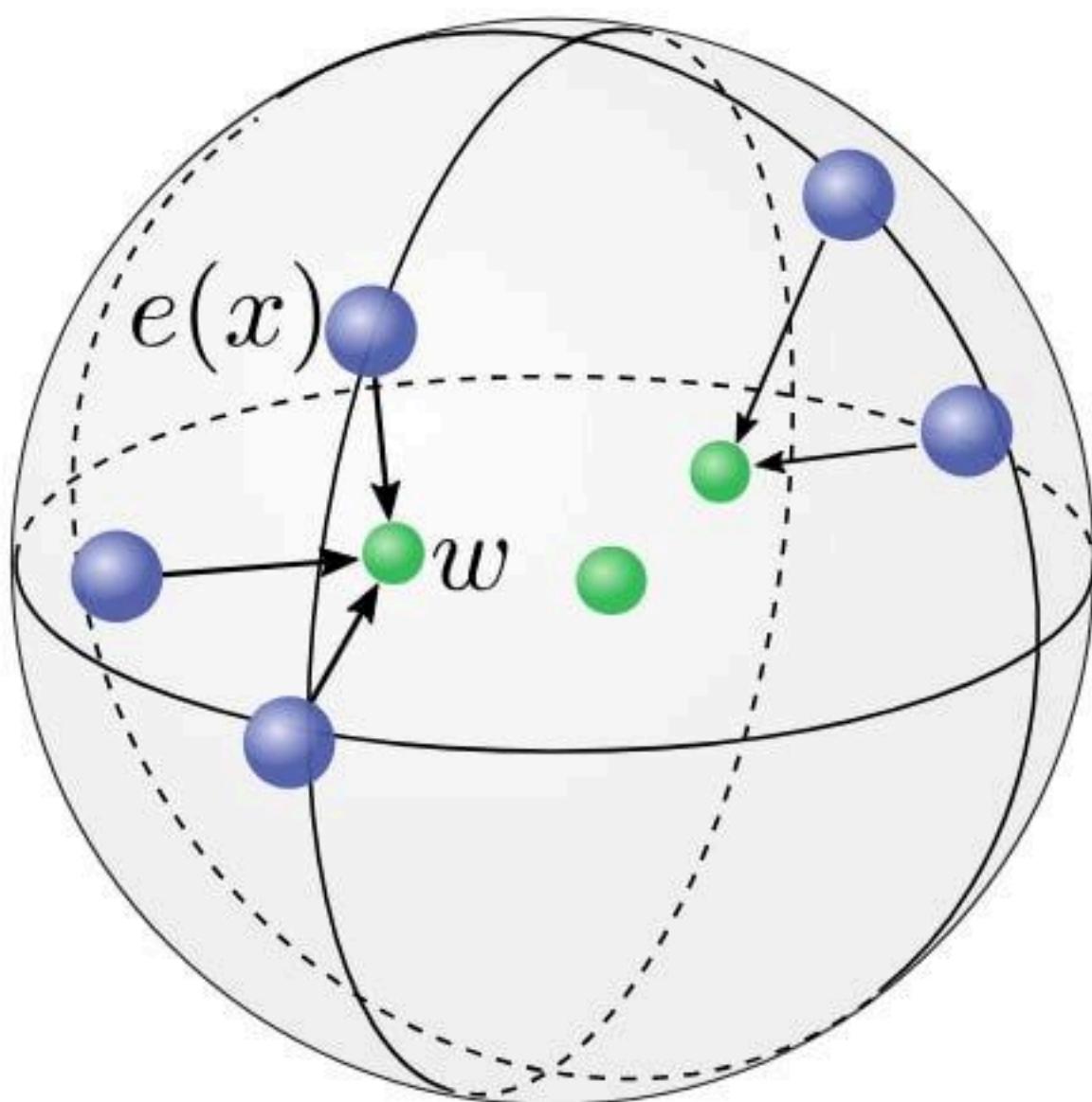
Коллапс кодовой книги - низкая утилизация большинства кодов после VQ. Проблема кроется в том, что мы считаем лоссы по семплам и взаимно приближаем только последние полученные коды с фичермапами. Поэтому некоторые прошлые элементы кодовой книги могут отстать от большинства фичермапов в латентном пространстве и они никогда не будут близко к ним, следовательно использовать эти коды не будут.

# коллапс кодбука

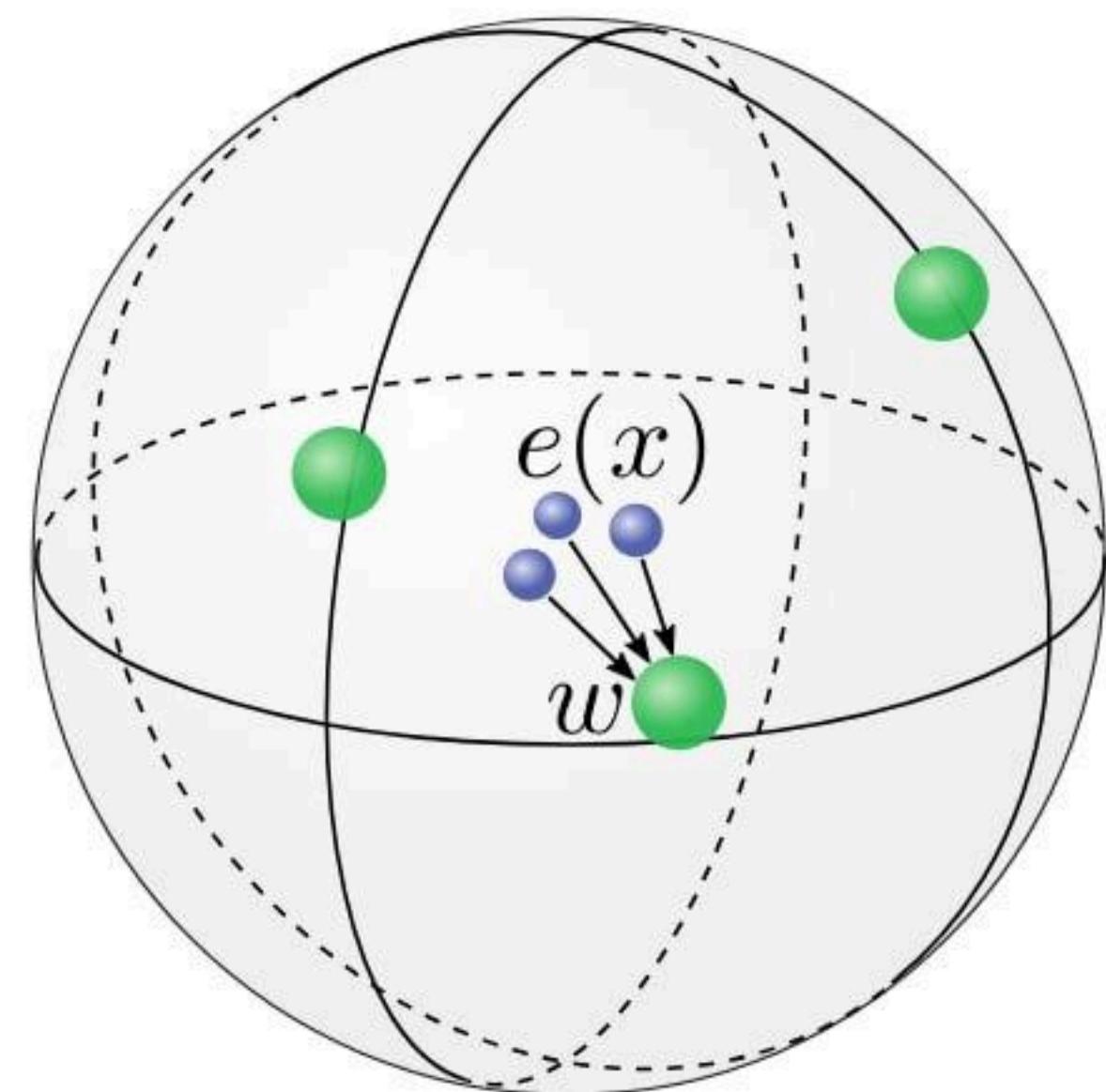


# ПОЛЕЗНАЯ ИНТУИЦИЯ

- K-Means это алгоритм рассчитанный на фиксированные данные
- Норма элементов кодбука должна быть меньше нормы feature maps из энкодера



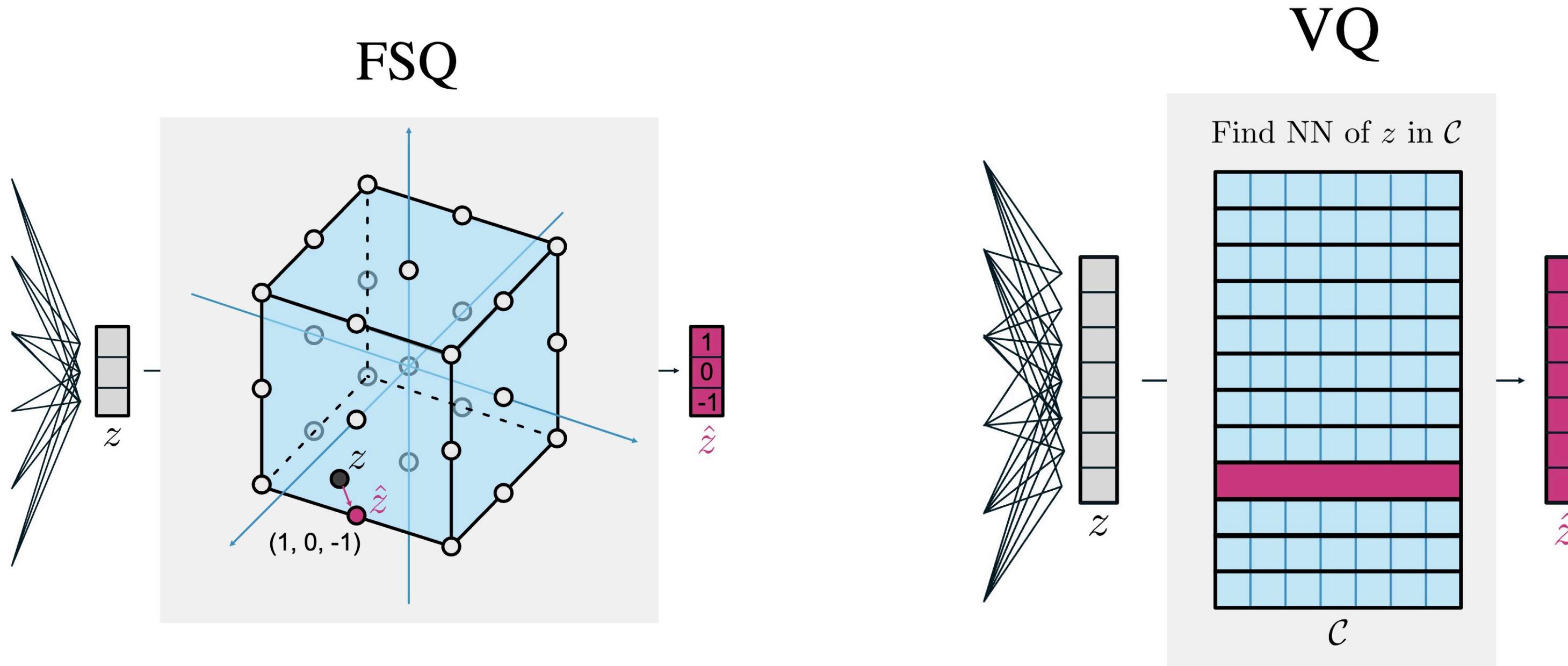
(b)  $|e(x_i)| \gg |w_j|$



(c)  $|e(x_i)| \ll |w_j|$

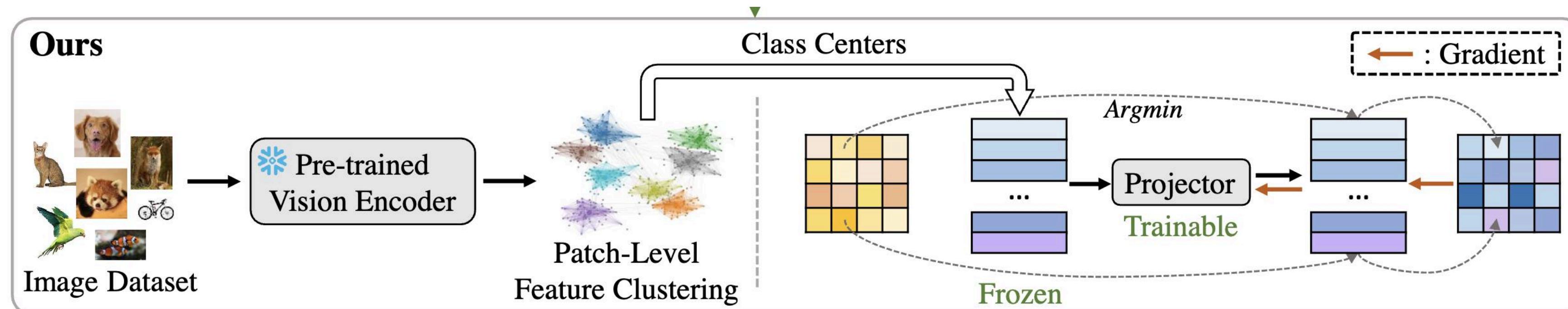
# решений много

Но достаточно использовать Finite Scalar Quantization. Берем элементы фичермапов и значение каждой размерности ограничивается пространством меньшей размерности по этой формуле:  $Z_i \rightarrow \lfloor L/2 \rfloor \tanh(Z_i)$



# решений много

Или VQGAN-LC loss. Кодовая книга будет фиксированной, инициализирована с помощью предобученного энкодера. Далее делаем проекцию кодовой книги через линейный слой и считаем L2 между проекцией и элементами фичермапов, получаем квантованные фичермапы. Затем считаем стандартный VQ лосс, но кодовая книга остается фиксированной, а линейный слой для проекции кодовой книги и наш энкодер обучаются.



# Современные тенденции

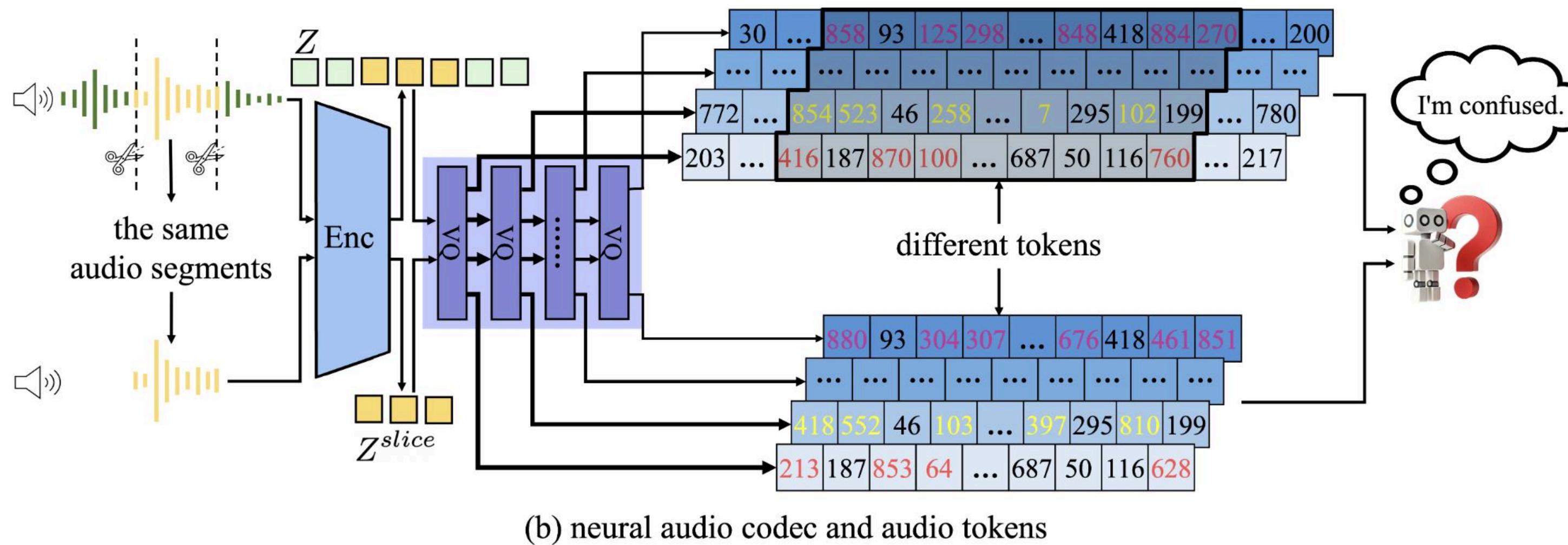
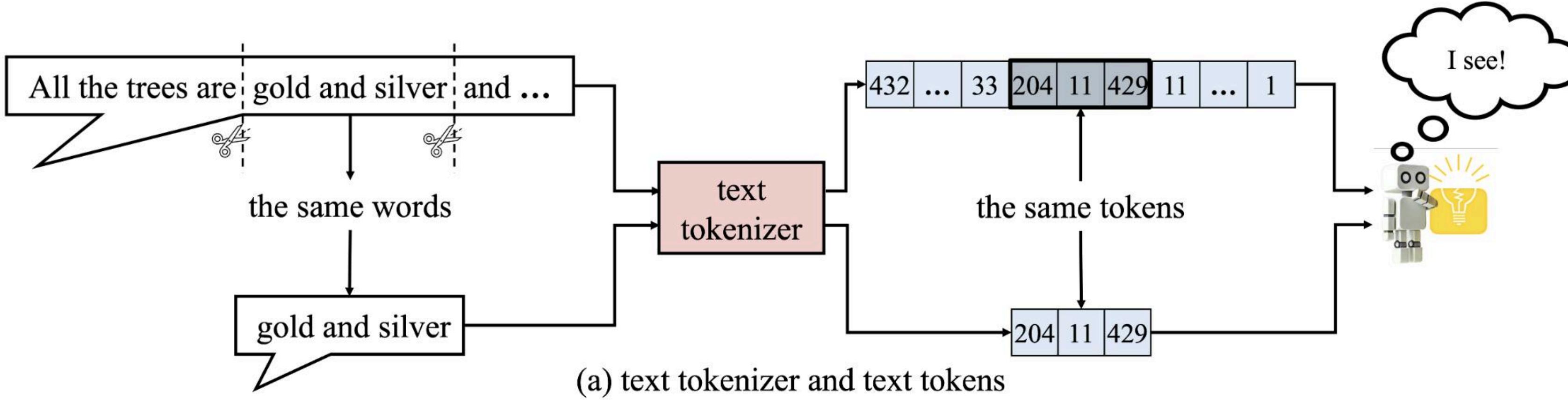
- Снижение плотности токенов — переход от сотен токенов в секунду (например, 900 TPS у DAC) к двухзначным значениям (40–75 TPS у WavTOKENIZER, 75 TPS у UNI\_CODEC), что облегчает обучение LLM-ов над аудио.
- Минимизация битрейта — реальные рабочие битрейты опустились до 1 кбит/с (BigCODEC) без серьёзных потерь качества.
- Отказ от иерархических RVQ — возрастающая популярность кодеков с одной кодовой книгой (WavTOKENIZER, X-CODEC) вместо каскадов RVQ.
- Увеличение кодовой книги — проблема коллапса кодовой книги решена, что позволяет масштабировать размер кодовой книги вплоть до 1 миллиона токенов (VQGAN-LC), улучшая качество генерации.

# Современные тенденции

- Семантическая насыщенность — интеграция МАЕ маскирования + дистилляция из SELF-SUPERVISED предобученного энкодера.
- Доменно-разделённые кодбуки — единый кодбук, «нарезанный» по доменам (UniCODEC, SQ\_CODEC), даёт возможность обучать один универсальный кодек для речи, музыки и общих звуков.

# discrete representation inconsistency

Аудио кодеки очень чувствительны к петрубациям (неуловимым на слух изменениям в аудио) и контексту



# аудио кодеки

WavTokenizer

Обучение мощного декодера и дискриминатора с 1 кодбуком

Использовать RVQ неудобно из за иерархических архитектур для декодирования

- Размер кодовой книги  $2^{12}$ , утилизация 100%, если кодбук больше то % падает
- Низкий фрейм рейт: 45/75 токенов в секунду!
- В декодер был добавлен механизм внимания, расширенное контекстное окно, STFT и ConvNeXt блоки. Также итоговый лосс состоит из 4 лоссов: потеря квантования (евклидово расстояние), L1 между мел-спектrogramами, состязательный хиндж лосс, L1 фичермапов между субдискриминаторами.

# аудио кодеки

x-Codec

## Добавление семантических фичей перед RVQ

Добавление семантики помогает лучше решать задачи генерации

- Делаем дистилляцию в первые N слоев RVQ из семантически богатых энкодеров - это всякие HuBERT и Wav2Vec
- Добавляем вспомогательный семантический модуль, будет брать фичи HuBERT, подгонять под RVQ, затем из выходов RVQ будем пытаться восстановить исходные семантические фичи

# аудио кодеки

UniCodec

## Доменно разделенная кодовая книга (MoE на кодбук)

Чтобы сделать 1 кодбук, который хорошо перекроет general audio домен, давайте разделим латентное пространство кодов на регионы для речи, музыки и звуков.

- Для маршрутизации между доменами авторы вдохновились DeepSeekMoE, а сами доменные регионы составляются и обучаются изолированно друг от друга. Деление кодов по доменам: речь - первые 4095, музыка - от 4096 до 8191, звуки - от 8191 до 16383 кодов.
- Модель пытается предсказать скрытые представления для случайно замаскированных фрагментов и оптимизируется контрастивной функцией; это обогащает высокоуровневые признаки, не ухудшая реконструкцию.

# аудио кодеки

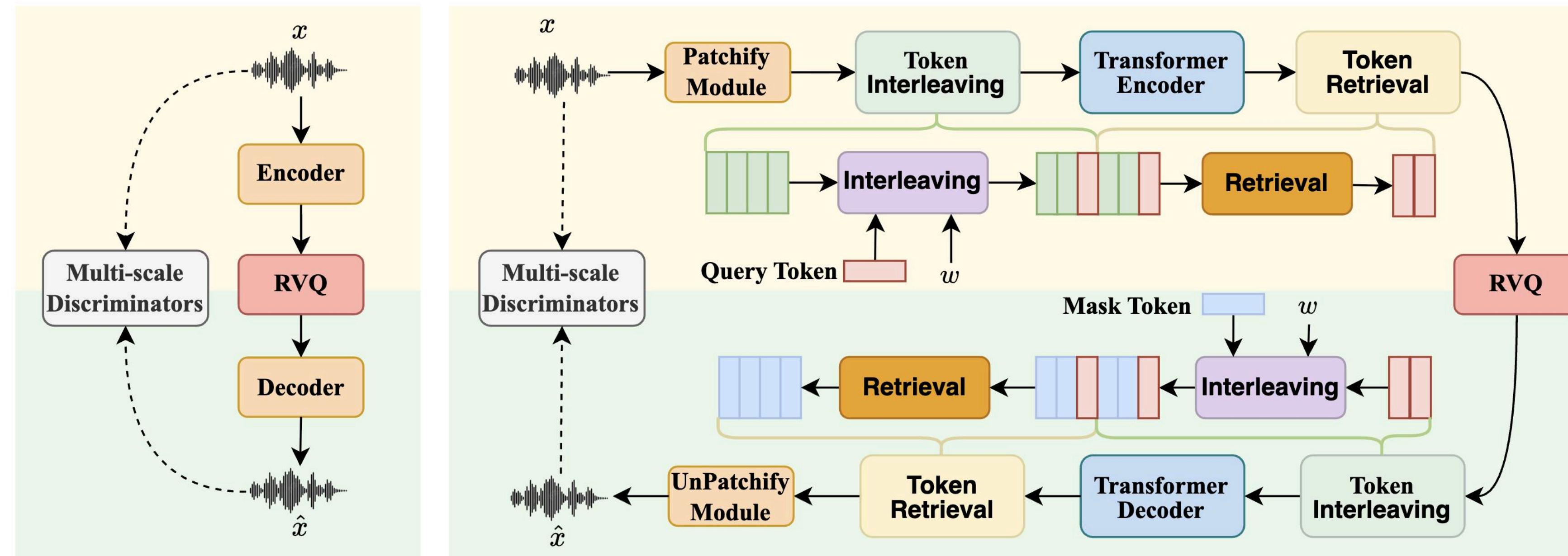
ALMTokenizer

Динамический битрейт за счет разного фрейм рейтa

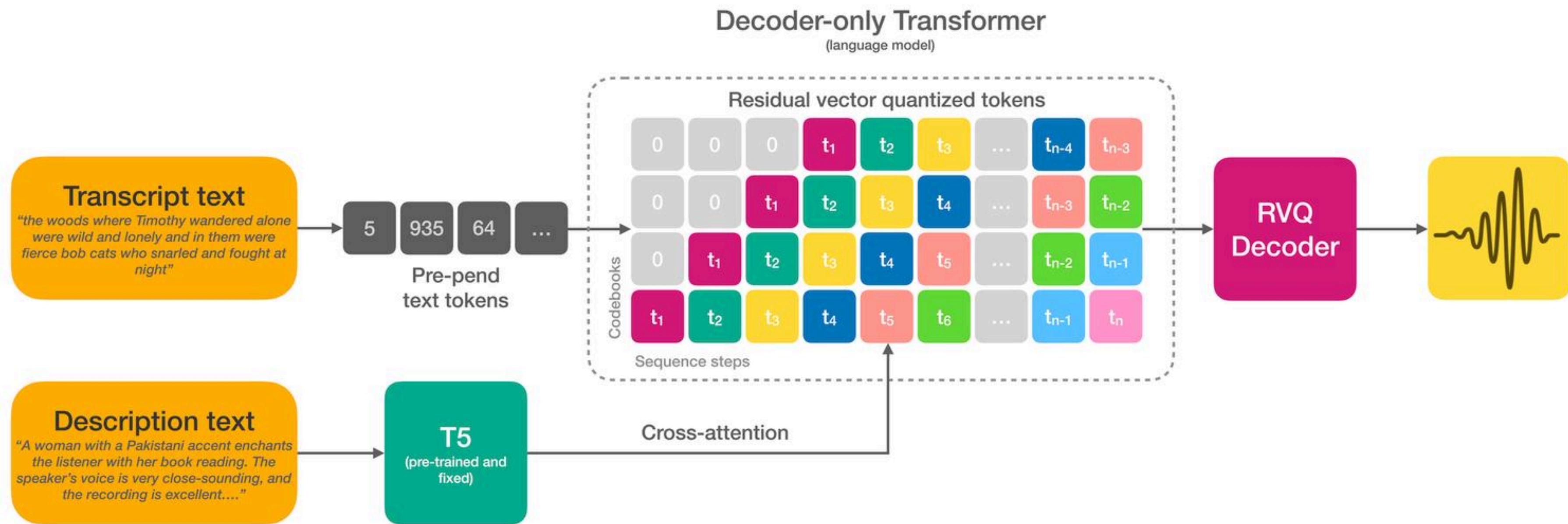
В разных фреймах содержится разный объем информации, где то стоит моделировать точнее, где то менее.

- Обучаемые query-based токены
- Masked AE loss
- Семантический prior для всех VQ слоев, используя WavLM
- AR лосс для резидуалов из RVQ

# ALMTokenizer



# Parler TTS



# доп. литература

- Recent Advances in Discrete Speech Tokens: A Review
- Discrete Audio Tokens: More Than a Survey!
- GIVT: Generative Infinite-Vocabulary Transformers
- Autoregressive Image Generation without Vector Quantization
- Autoregressive Speech Synthesis without Vector Quantization
- Balance of Number of Embedding and their Dimensions in Vector Quantization
- A Variational EM Acceleration for Efficient Clustering at Very Large Scales
- Restructuring Vector Quantization with the Rotation Trick
- StableToken: A Noise-Robust Semantic Speech Tokenizer for Resilient SpeechLLMs

# TELEGRAM



@vf\_science