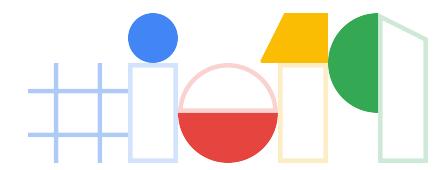
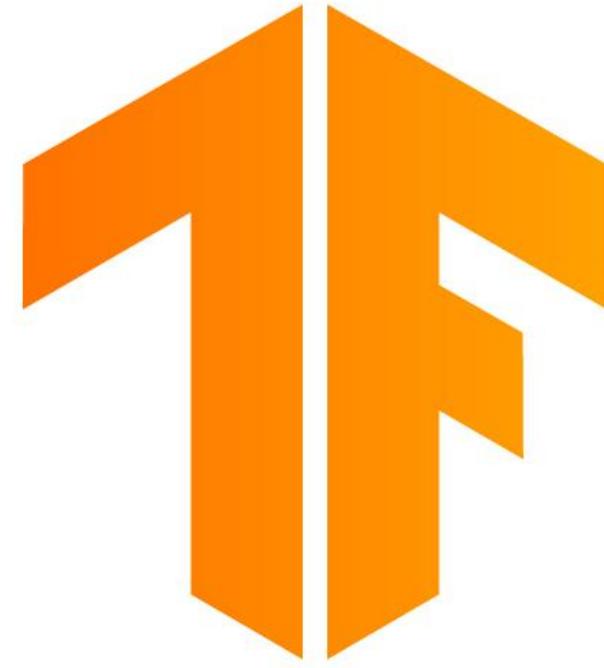


Jukebox: A Generative Model for Music

郭成凯

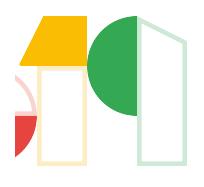
引子

用Transformer-XL训练音乐生成，效果有局限
openai jukebox根据歌词生成歌曲 awesome !



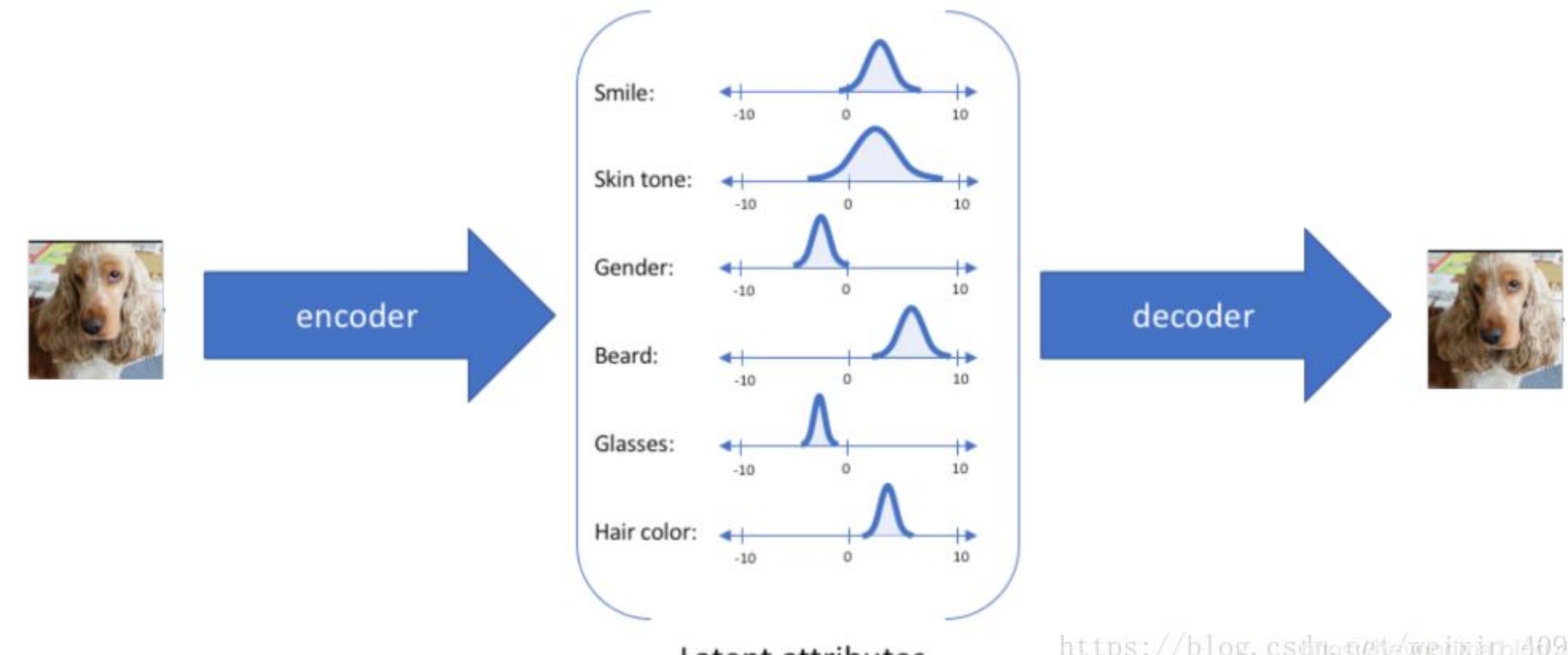
VQ-VAE-2 DeepMind

- deepmind的模型VQ-VAE-2的图像生成效果甚至超过了BigGAN ,
2019/06

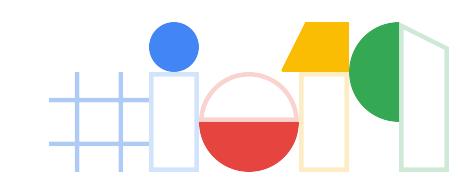


VAE

- VAE的主要思想是他认为图像、声音等信息是由多个隐变量 (latent attribute)，比如对于人的面部图像来说就由笑容，肤色、发色、发型等变量决定

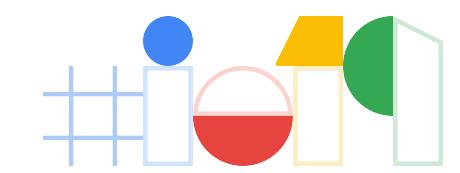


<https://blog.csdn.net/weiniarid40955254>



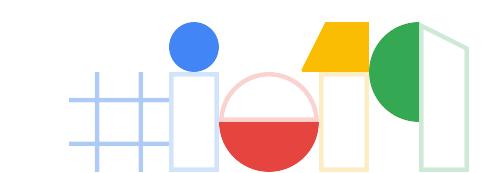
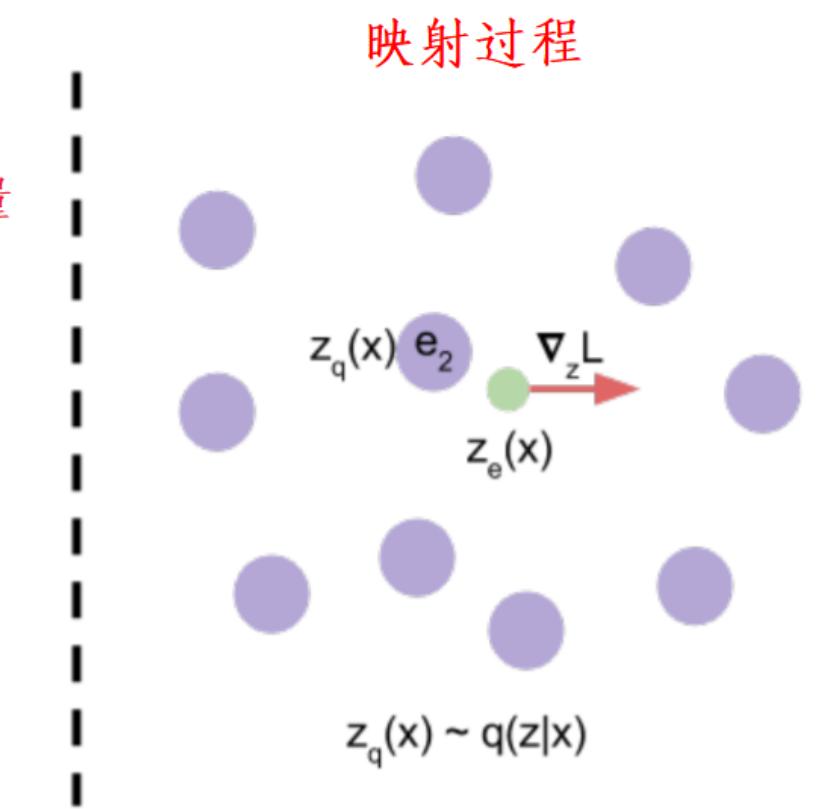
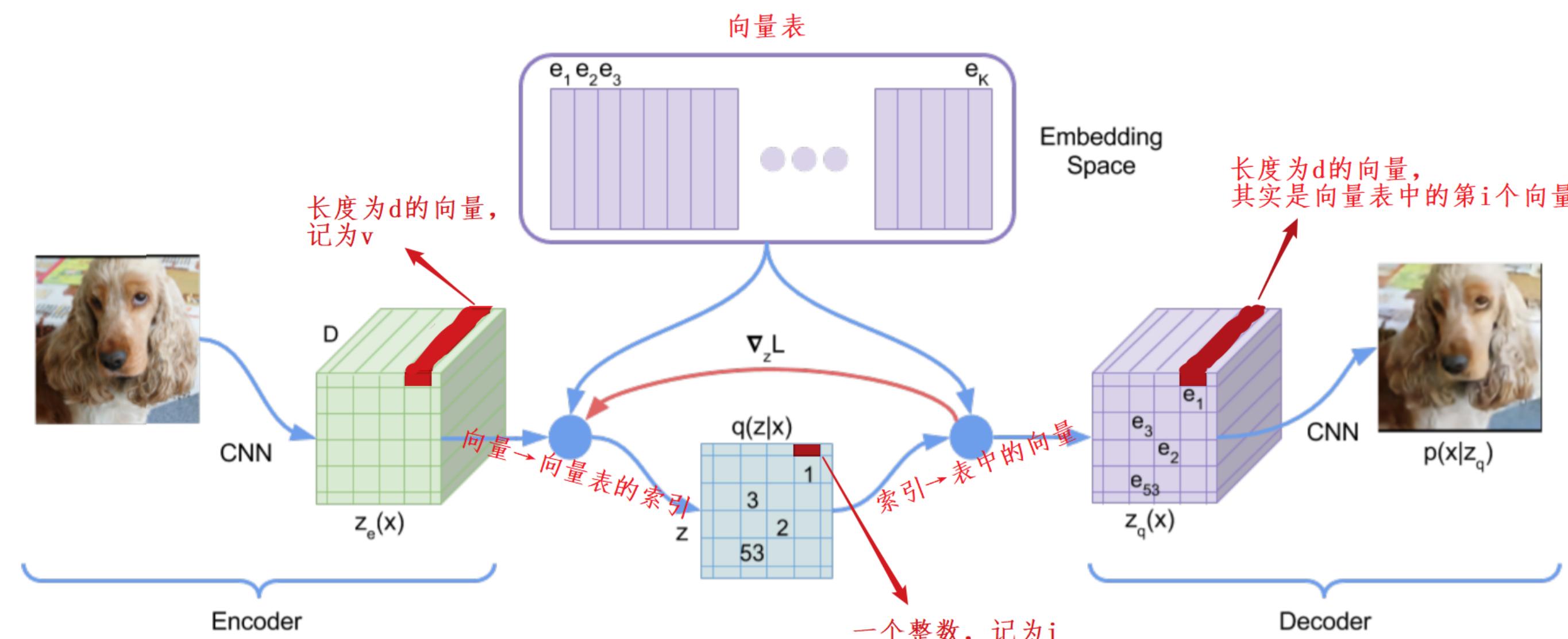
VQ

- 自回归模型在图像比较大的情况下因计算需求暴增而失效。一个自然而然的想法是，如果能将照片压缩到低维空间，在低维空间训练自回归神经网络，再解码到高维就好了。这也是 VQ-VAE 以及 VQ-VAE-2 算法的精髓。即在压缩空间上使用自回归神经网络。
- VQ 实际就是先把输入的图像进行 k-means 聚类，完成后只保留最终留下的 K 个簇质心，簇上的其它点全部近似化为质心来进行存储



VQ-VAE-1

- VQ-VAE-1算法，在Encoder层计算latent attribute(隐向量)的向量族 z , 然后传递给隐层，在隐层按照刚刚所述的vQ算法进行压缩，然后输出给Decoder进行生成



更新字典向量

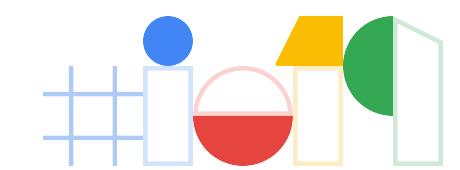
\mathbf{x} 是输入图像， D 是解码器， E 是编码器， **sg** 是**stop gradient** 的缩写， 表示不对 **sg** 方括号里的变量计算梯度， 误差不向此变量传递。 β 是一个不重要的超参数。

损失函数分三部分。第一部分是重构误差， 可以看到这里重构误差与普通的AutoEncoder重构误差 不一样， 因为使用了量子化， 解码器的输入变为量子化后的字典向量 \mathbf{e} 。这一项同时更新编码器和解码器。

第二部分计算编码器得到的潜在向量与字典向量的距离，并将其作为辅助误差项。此误差项只向字典向量 \mathbf{e} 传递， 通过对误差惩罚来学习 \mathbf{e} 向量。不更新编码器和解码器。

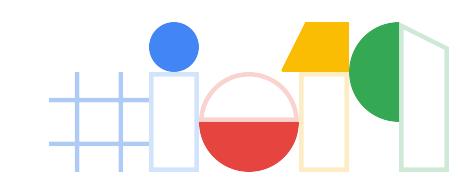
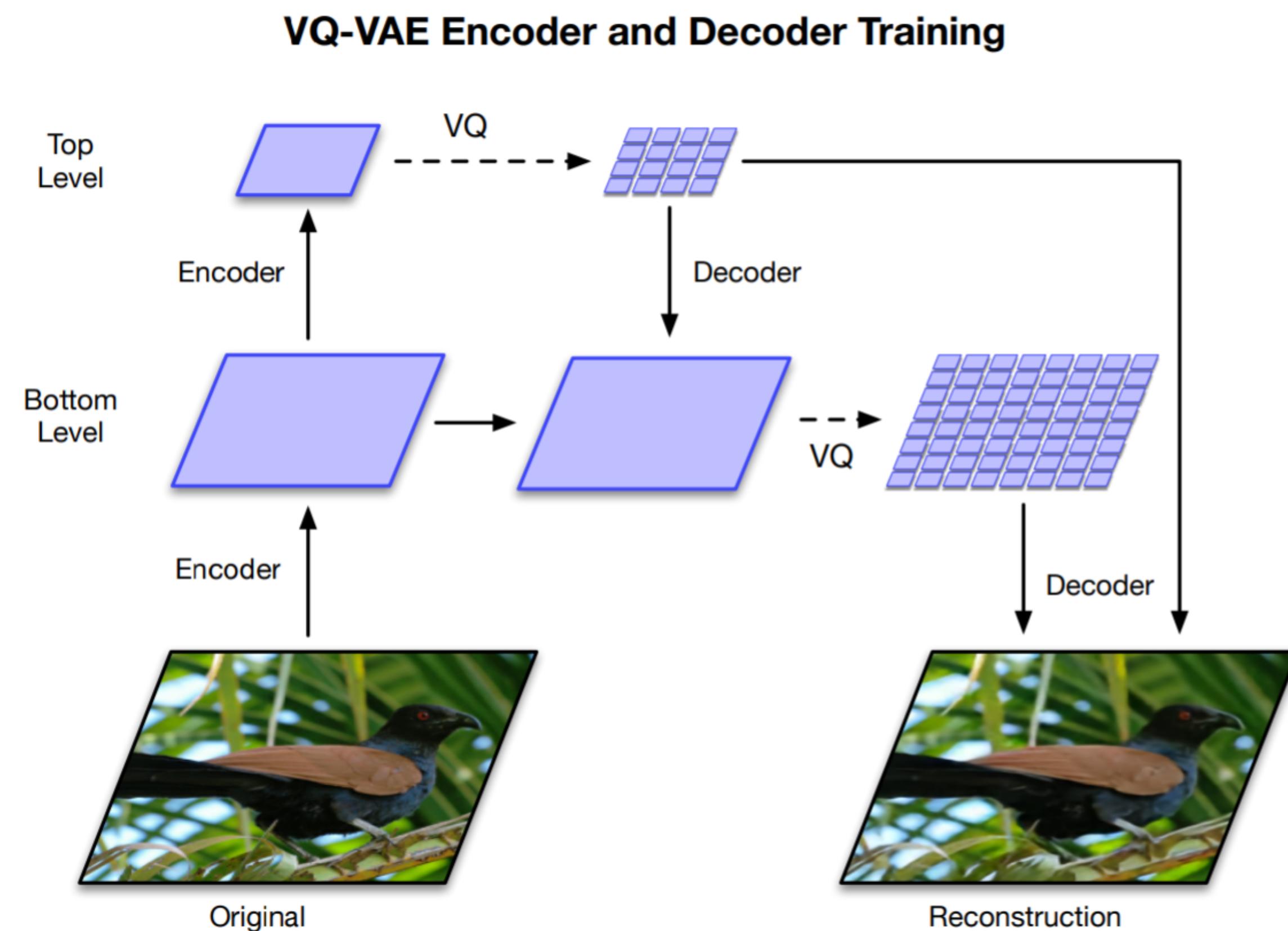
第三部分与第二部分一致， 也是计算潜在向量与字典向量的距离。不过这里对字典向量 \mathbf{e} 使用了**stop gradient** 约束， 使得此误差项只向编码器反向传递

$$\mathcal{L}(\mathbf{x}, D(\mathbf{e})) = \|\mathbf{x} - D(\mathbf{e})\|_2^2 + \|sg[E(\mathbf{x})] - \mathbf{e}\|_2^2 + \beta \|sg[\mathbf{e}] - E(\mathbf{x})\|_2^2$$



VQ-VAE-2

- 分层量子化
- 将顶层全局与底层局部信息分离开来，生成全局自治，局部高清的图像



Jukebox

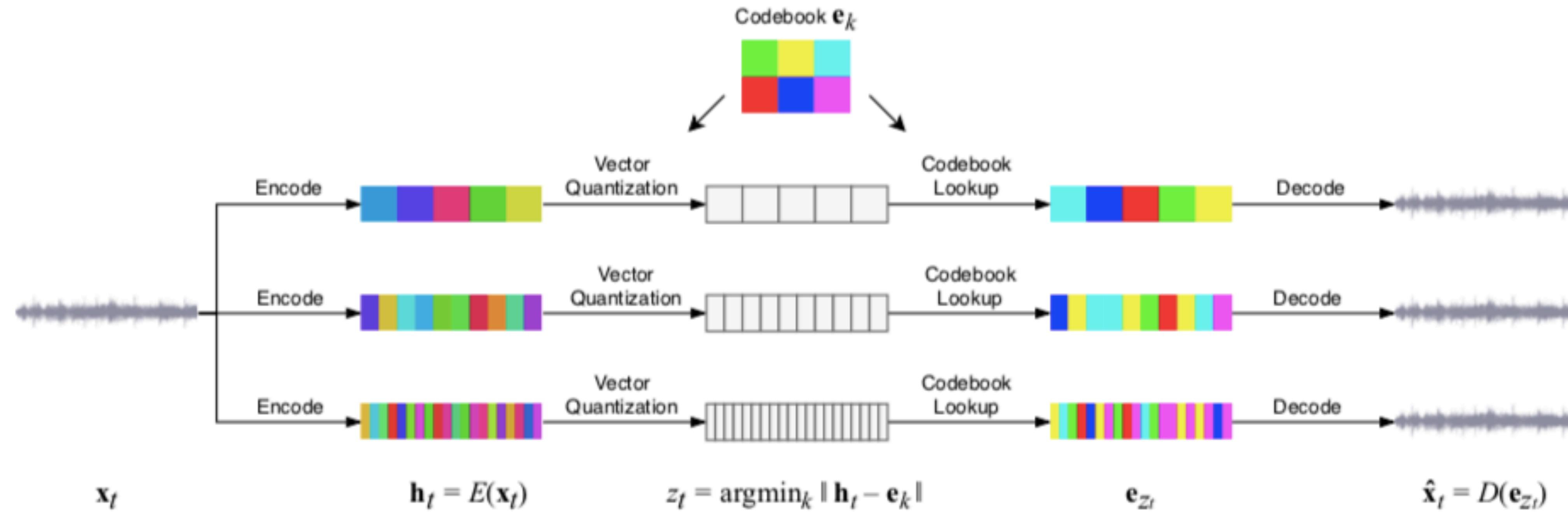
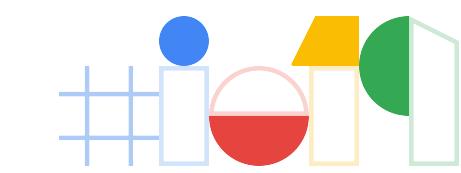
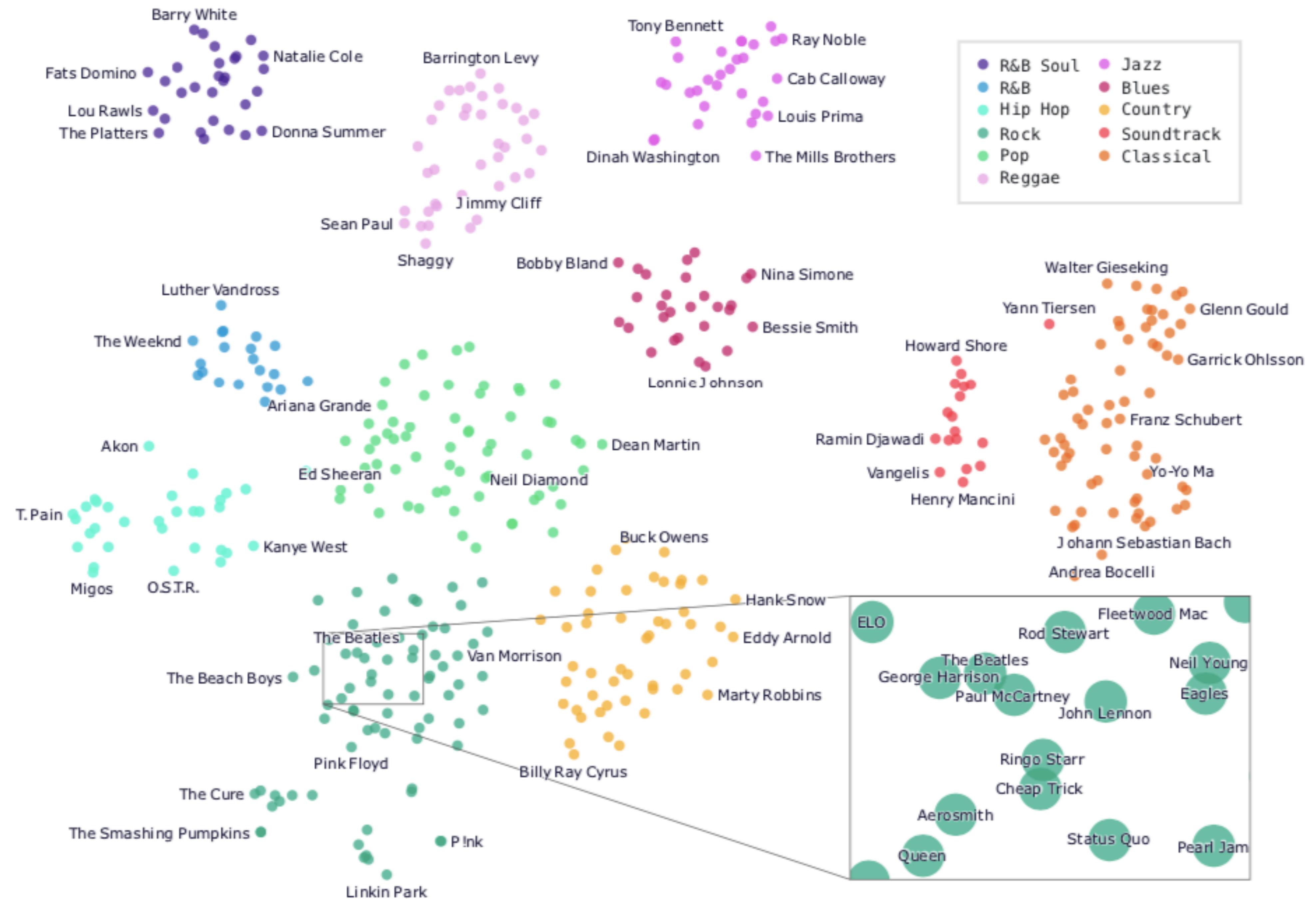


Figure 1. We first train three separate VQ-VAE models with different temporal resolutions. At each level, the input audio is segmented and encoded into latent vectors \mathbf{h}_t , which are then quantized to the closest codebook vectors \mathbf{e}_{z_t} . The code z_t is a discrete representation of the audio that we later train our prior on. The decoder takes the sequence of codebook vectors and reconstructs the audio. The top level learns the highest degree of abstraction, since it is encoding longer audio per token while keeping the codebook size the same. Audio can be reconstructed using the codes at any one of the abstraction levels, where the least abstract bottom-level codes result in the highest-quality audio, as shown in Figure 4. For the detailed structure of each component, see Figure 7.



Artists



enjoy!



Thanks! Any Questions?

