Image: Synthetic faces sampled from VQ-VAE-2 model by Razavi et al.

COMP547

DEEP UNSUPERVISED LEARNING

Lecture #10 – Discrete Latent Variable Models

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Spring 2021

Good news, everyone!

- The deadline of the project proposals are postponed by a week
  - The new deadline: April 18
- This change is also reflected to the project progress reports and presentations, too.
- Please refer to the website for the updated dates.

Previously on COMP547

- Motivation & Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Fréchet
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Lecture overview

- Motivation
- Tools: REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Disclaimer: Much of the material and slides for this lecture were borrowed from
—David Duvenaud's STA4273/CSC2547 class
—Stefano Ermon and Aditya Grover's Stanford CS236 class
—Aaron van den Oord's talk on "Neural Discrete Representation Learning"
—Mihaela Rosca's talk title "Training Language GANs from Scratch"
—Graham Neubig CMU CS11-747 class

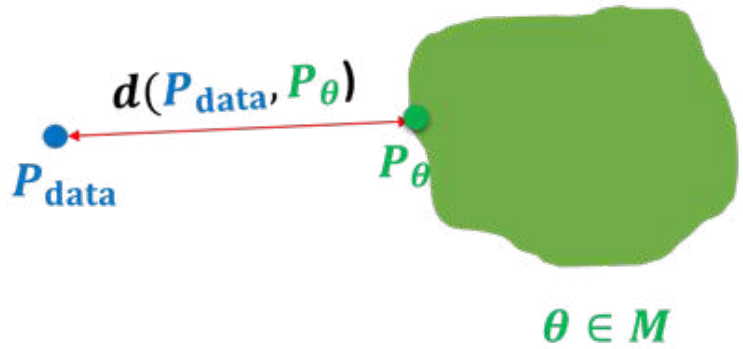
Lecture overview

- **Motivation**
- Tools: REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Generative Modeling

- **Representation:** Latent variable vs. fully observed
- **Objective function and optimization algorithm:** Many divergences and distances optimized via likelihood-free (two sample test) or likelihood-based methods
- **Evaluation** of generative models
- Combining different models and variants

$$\mathbf{x}_i \sim P_{\text{data}}$$
$$i = 1, 2, \ldots, n$$

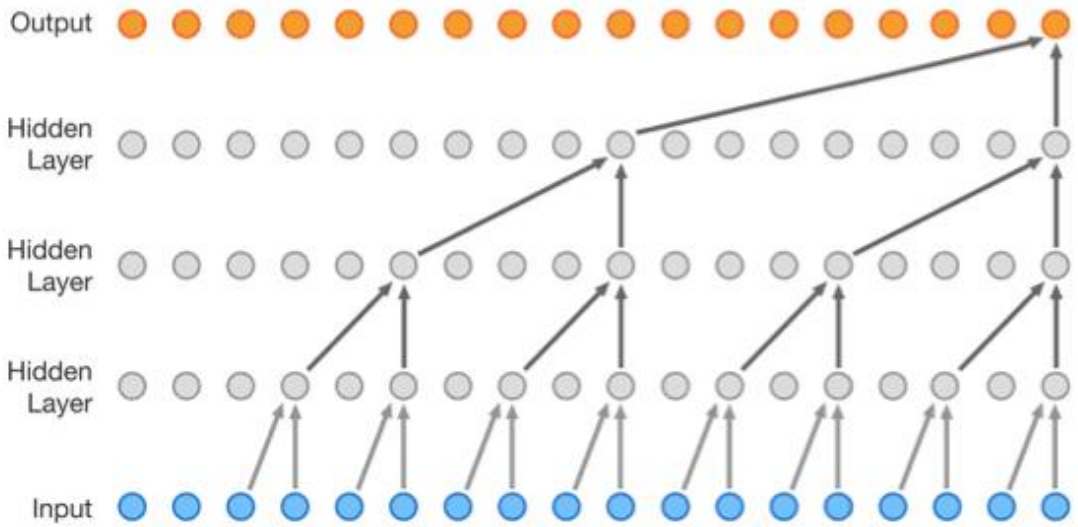


Model Family

What recently became easy in ML?

- Training large supervised models with fixed architectures
- Training continuous latent-variable models (VAEs, GANs) to produce large images
- Building RNNs that can output grid-structured objects (images, waveforms)

horse → zebra



What is still challenging?

- Training GANs to generate text
- Training VAEs with discrete latent variables
- Training agents to communicate with each other using words
- Training agent or programs to decide which discrete action to take.
- Training generative models of structured objects of arbitrary size, like programs, graphs, or large texts.

Adversarial Generation of Natural Language

Word level generations on the Penn Treebank and CMU-SE datasets

Level	Model	PTB	CMU-SE
Word	LSTM	what everything they take everything away from . may tea bill is the best chocolate from emergency . can you show show if any fish left inside . room service , have my dinner please .	<s>will you have two moment ? </s> <s>i need to understand deposit length . </s> <s>how is the another headache ? </s> <s>how there , is the restaurant popular this cheese ? </s>
	CNN	meanwhile henderson said that it has to bounce for. I'm at the missouri burning the indexing manufacturing and through .	<s>i 'd like to fax a newspaper . </s> <s>cruise pay the next in my replacement . </s> <s>what 's in the friday food ? ? </s>

[Subramanian et al. 2017]

Why are the easy things easy?

- Gradients give more information the more parameters you have
- Backprop (reverse-mode AD) only takes about as long as the original function
- Local optima less of a problem than you think

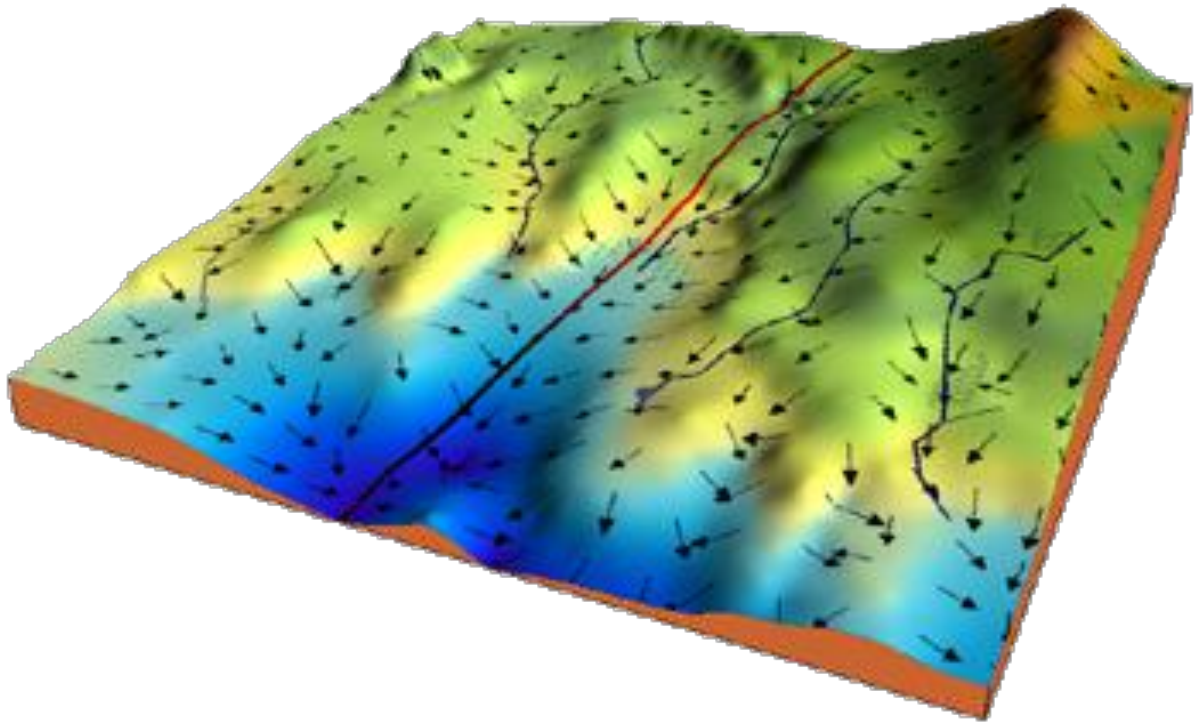


Image credit: ML-Cheatsheet

Why are the hard things hard?

- Discrete structure means we can't use backdrop to get gradients
- No cheap gradients means that we don't know which direction to move to improve
- Not using our knowledge of the structure of the function being optimized
- Becomes as hard as optimizing a black-box function

Image credit: ML-Cheatsheet

Why should we care about discreteness?

- Discreteness is all around us!
- Decision Making: Should I attend COMP 547 lectures at Monday 08:30am or not?
- Structure learning

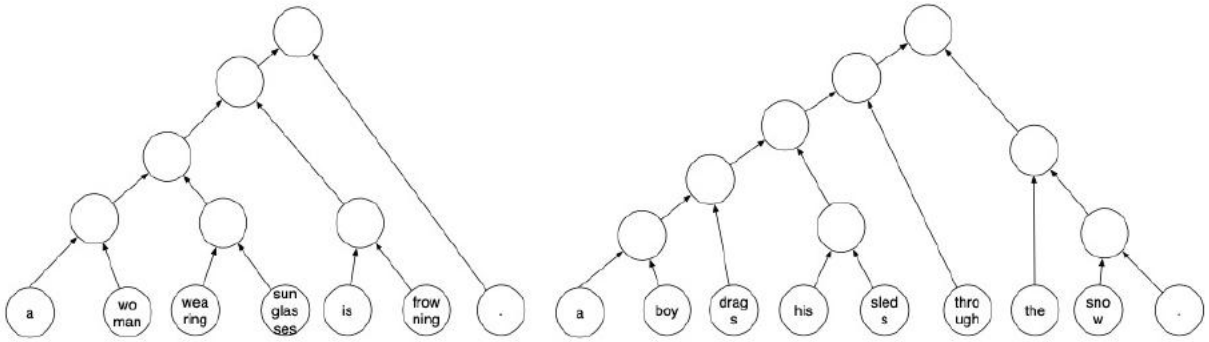


Figure 2: Examples of tree structures learned by our model which show that the model discovers simple concepts such as noun phrases and verb phrases.

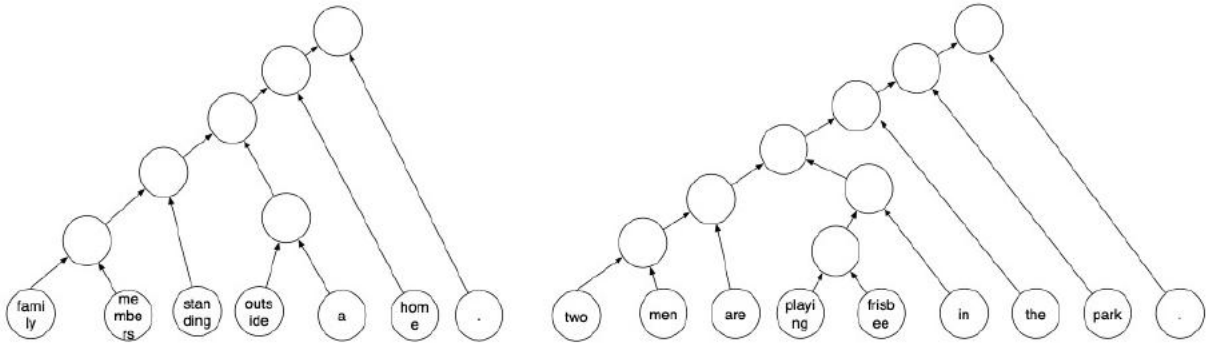
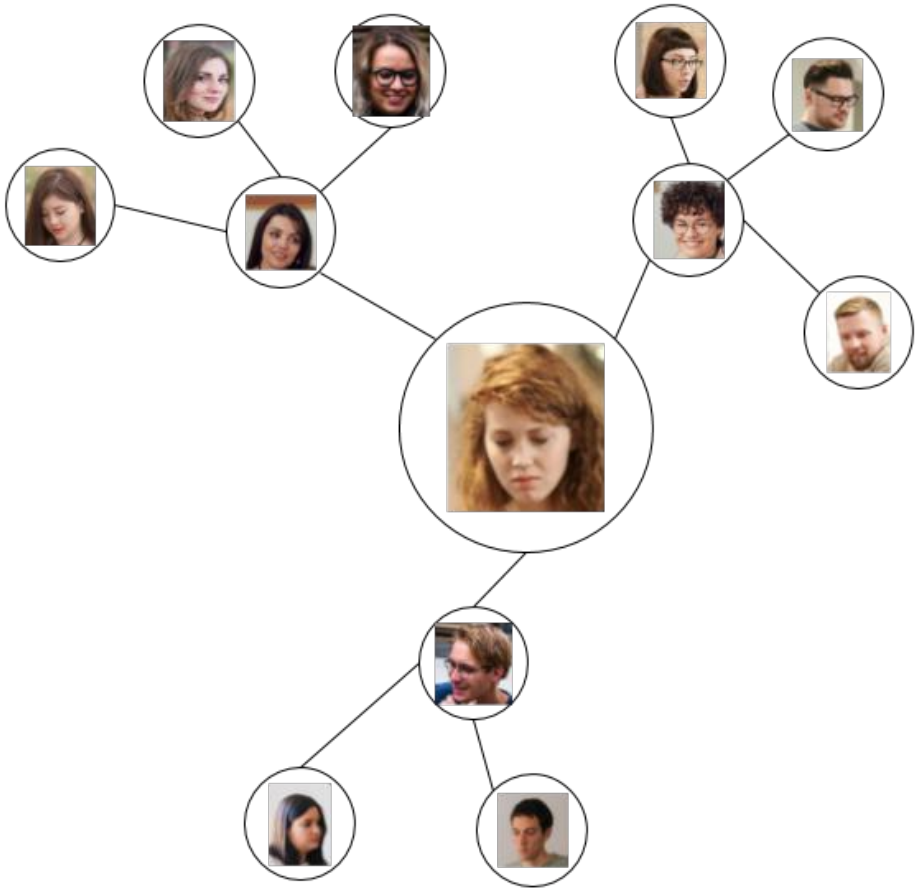


Figure 3: Examples of unconventional tree structures.

[Yogatama et al. 2017]

Why should we care about discreteness?

- Many data modalities are inherently discrete
  - Graphs,
  - Text,
  - DNA Sequences,
  - Program Source Codes,
  - Molecules,
  - and lots more



Lecture overview

- Motivation
- **Tools: REINFORCE**, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Stochastic Optimization

$$\log p_\theta(x) = \log \mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right]$$

$$\geq \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q(z|x)}\right]$$

- Consider the following optimization problem:

$$\max_\phi \mathbb{E}_{q_\phi(z)}[f(z)]$$

- Think of q(·) as the inference distribution for a VAE:

$$\max_{\theta,\phi} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right]$$

- Gradients w.r.t. θ can be derived via linearity of expectation

$$\nabla_\theta \mathbb{E}_{q(\mathbf{z};\phi)}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)] = \mathbb{E}_{q(\mathbf{z};\phi)}\left[\nabla_\theta \log p(\mathbf{z},\mathbf{x};\theta)\right]$$

$$\approx \frac{1}{k}\sum_k \nabla_\theta \log p\left(\mathbf{z}^k,\mathbf{x};\theta\right)$$

- If z is continuous, q(·) is reparameterizable, and f(·) is differentiable in ϕ, then we can use reparameterization to compute gradients w.r.t. ϕ
- What if any of the above assumptions fail?

Stochastic Optimization with REINFORCE

- Consider the following optimization problem:

$$\max_{\phi} \mathbb{E}_{q_\phi(z)}[f(z)]$$

- For many class of problem scenarios, reparameterization trick is inapplicable
- **Scenario 1:** f(·) is non-differentiable in ϕ e.g., optimizing a black box reward function in reinforcement learning
- **Scenario 2:** $q_\phi(z)$ cannot be reparameterized as a differentiable function of ϕ with respect to a fixed base distribution e.g., discrete distributions
- REINFORCE is a general-purpose solution to both these scenarios
- We will first analyze it in the context of reinforcement learning and then extend it to latent variable models with discrete latent variables

REINFORCE for Reinforcement Learning

- **Example:** Pulling arms of slot machines Which arm to pull?
- Set A of possible actions. E.g., pull arm1, arm2, . . . , etc.
- Each action z ∈ A has a reward f(z)
- Randomized policy for choosing actions $q_\phi(z)$ parameterized by ϕ. For example, ϕ could be the parameters of a multinomial distribution
- Goal: Learn the parameters ϕ that maximize our earnings (in expectation): $\max_\phi \mathbb{E}_{q_\phi(z)}[f(z)]$

Policy Gradients

- Want to compute a gradient with respect to $\boldsymbol{\phi}$ of the expected reward

$$\mathbb{E}_{q_\phi(z)}[f(z)] = \sum_z q_\phi(z) f(z)$$

$$\frac{\partial}{\partial \phi_i} \mathbb{E}_{q_\phi(z)}[f(z)] = \sum_z \frac{\partial q_\phi(z)}{\partial \phi_i} f(z) = \sum_z q_\phi(z) \frac{1}{q_\phi(z)} \frac{\partial q_\phi(z)}{\partial \phi_i} f(z)$$

Policy Gradients

- Want to compute a gradient with respect to ϕ of the expected reward

$$\mathbb{E}_{q_\phi(z)}[f(z)] = \sum_z q_\phi(z) f(z)$$

$$\frac{\partial}{\partial \phi_i} \mathbb{E}_{q_\phi(z)}[f(z)] = \sum_z \frac{\partial q_\phi(z)}{\partial \phi_i} f(z) = \sum_z q_\phi(z) \frac{1}{q_\phi(z)} \frac{\partial q_\phi(z)}{\partial \phi_i} f(z)$$

$$= \sum_z q_\phi(z) \frac{\partial \log q_\phi(z)}{\partial \phi_i} f(z) = \mathbb{E}_{q_\phi(z)} \left[\frac{\partial \log q_\phi(z)}{\partial \phi_i} f(z) \right]$$

The REINFORCE rule

REINFORCE Gradient Estimation

- Want to compute a gradient with respect to $\boldsymbol{\phi}$ of

$$\mathbb{E}_{q_\phi(z)}[f(z)] = \sum_z q_\phi(z) f(z)$$

- The REINFORCE rule: $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{q_\phi(z)}\left[f(z) \nabla_\phi \log q_\phi(z)\right]$
- We can now construct a Monte Carlo estimate
  - Sample $z^1,\ldots,z^K$ from $q_{\boldsymbol{\phi}}(z)$ and estimate:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] \approx \frac{1}{K} \sum_k f\left(z^k\right) \nabla_\phi \log q_\phi\left(z^k\right)$$

- **Assumption:** The distribution q(·) is easy to sample from and evaluate probabilities
- Works for both discrete and continuous distributions

Variational Learning of Latent Variable Models

- To learn the variational approximation we need to compute the gradient wrt $\boldsymbol{\phi}$ of

$$\mathcal{L}(x;\theta,\phi) = \sum_z q_\phi(z|x)\log p(z,x;\theta) + H\left(q_\phi(z|x)\right)$$
$$= \mathbb{E}_{q_\phi(z|x)}\left[\log p(z,x;\theta) - \log q_\phi(z|x)\right]$$

- The function inside the brackets also depends on $\boldsymbol{\phi}$ (and θ, x). Want to compute a gradient with respect to $\boldsymbol{\phi}$ of

$$\mathbb{E}_{q_\phi(z|x)}[f(\phi,\theta,z,x)] = \sum_z q_\phi(z|x) f(\phi,\theta,z,x)$$

- The REINFORCE rule is

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(\phi,\theta,z,x)] = \mathbb{E}_{q_\phi(z|x)}\left[f(\phi,\theta,z,x)\nabla_\phi \log q_\phi(z|x) + \nabla_\phi f(\phi,\theta,z,x)\right]$$

- We can now construct a Monte Carlo estimate of $\nabla_\phi \mathcal{L}(x;\theta,\phi)$

REINFORCE Gradient Estimation have High Variance

- Want to compute a gradient with respect to ϕ of

$$\mathbb{E}_{q_\phi(z)}[f(z)] = \sum_z q_\phi(z) f(z)$$

- The REINFORCE rule: $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{q_\phi(z)}\left[f(z) \nabla_\phi \log q_\phi(z)\right]$
- Monte Carlo estimate: Sample $z^1,\ldots,z^K$ from $q_\phi(z)$ and estimate:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] \approx \frac{1}{K} \sum_k f\left(z^k\right) \nabla_\phi \log q_\phi\left(z^k\right) := f_{MC}(z^1, \ldots, z^K)$$

- Monte Carlo estimates of gradients are unbiased

$$\mathbb{E}_{z^1,\ldots,z^K \sim q_\phi(z)}\left[f_{\mathrm{MC}}\left(z^1, \cdots, z^K\right)\right] = \nabla_\phi E_{q_\phi(z)}[f(z)]$$

- Almost never used in practice because of high variance
- Variance can be reduced via carefully designed control variates

Control Variates

- The REINFORCE rule: $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{q_\phi(z)}\left[f(z)\nabla_\phi \log q_\phi(z)\right]$
- Given any constant B (a control variate)

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[(f(z) - B)] = \mathbb{E}_{q_\phi(z)}\left[f(z)\nabla_\phi \log q_\phi(z)\right]$$

- To see why,

$$\mathbb{E}_{q_\phi(z)}\left[B\nabla_\phi \log q_\phi(z)\right] = B\sum_z q_\phi(z)\nabla_\phi \log q_\phi(z) = B\sum_z \nabla_\phi q_\phi(z)$$

$$= B\nabla_\phi \sum_z q_\phi(z) = B\nabla_\phi 1 = 0$$

- Monte Carlo gradient estimates of both f(z) and f(z) - B have same expectation
- These estimates can however have different variances

Control Variates

- Suppose we want to compute

$$\mathbb{E}_{q_\phi(z)}[f(z)] = \sum_z q_\phi(z) f(z)$$

- Define

$$\widehat{f}(z) = f(z) + a\left(h(z) - \mathbb{E}_{q_\phi(z)}[h(z)]\right)$$

  where h(z) is referred to as a control variate

- **Assumption:** $\mathbb{E}_{q_\phi(z)}[h(z)]$ is known
- Monte Carlo gradient estimates of f(z) and $\widehat{f}$(z) have the same expectation

$$\mathbb{E}_{z^1,\cdots,z^K\sim q_\phi(z)}\left[\widehat{f}_{\mathrm{MC}}\left(z^1,\cdots,z^K\right)\right] = \mathbb{E}_{z^1,\cdots,z^K\sim q_\phi(z)}\left[f_{\mathrm{MC}}\left(z^1,\cdots,z^K\right)\right]$$

  but different variances

- Can try to learn and update the control variate during training

Control Variates

- Can derive an alternate Monte Carlo estimate for REINFORCE gradients based on control variates
- Sample $z^1,\ldots,z^K$ from $q_\phi(z)$

$$
\begin{aligned}
&\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] \\
&= \nabla_\phi \mathbb{E}_{q_\phi(z)}\left[f(z) + a\left(h(z) - \mathbb{E}_{q_\phi(z)}[h(z)]\right)\right] \\
&\approx \tfrac{1}{K}\textstyle\sum_k f\left(z^k\right)\nabla_\phi \log q_\phi\left(z^k\right) + a\left(\tfrac{1}{K}\textstyle\sum_{k=1}^K h\left(z^k\right) - \mathbb{E}_{q_\phi(z)}[h(z)]\right) \\
&:= f_{\mathrm{MC}}\left(z^1,\cdots,z^K\right) + a\left(h_{\mathrm{MC}}\left(z^1,\cdots,z^k\right) - \mathbb{E}_{q_\phi(z)}[h(z)]\right) \\
&:= \widehat{f}_{\mathrm{MC}}\left(z^1,\cdots,z^K\right)
\end{aligned}
$$

- What is $\mathrm{Var}(\widehat{f}_{\mathrm{MC}})$ vs. $\mathrm{Var}(f_{\mathrm{MC}})$?

Control Variates

- Comparing $\mathrm{Var}(\widehat{f}_{\mathrm{MC}})$ vs. $\mathrm{Var}(f_{\mathrm{MC}})$

$$\begin{aligned}\mathrm{Var}\left(\widehat{f}_{\mathrm{MC}}\right) &= \mathrm{Var}\left(f_{\mathrm{MC}} + a\left(h_{\mathrm{MC}} - \mathbb{E}_{q_\phi(\mathrm{z})}[h(z)]\right)\right) \\ &= \mathrm{Var}\left(f_{\mathrm{MC}} + a h_{\mathrm{MC}}\right) \\ &= \mathrm{Var}\left(f_{\mathrm{MC}}\right) + a^2\,\mathrm{Var}\left(h_{\mathrm{MC}}\right) + 2a\,\mathrm{Cov}\left(f_{\mathrm{MC}}, h_{\mathrm{MC}}\right)\end{aligned}$$

- To get the optimal coefficient a^* that minimizes the variance, take derivatives wrt a and set them to 0

$$a^* = -\frac{\mathrm{Cov}\left(f_{\mathrm{MC}}, h_{\mathrm{MC}}\right)}{\mathrm{Var}\left(h_{\mathrm{MC}}\right)}$$

Control Variates

- Comparing $\mathrm{Var}(\widehat{f}_{\mathrm{MC}})$ vs. $\mathrm{Var}(f_{\mathrm{MC}})$

$$\mathrm{Var}\left(\widehat{f}_{\mathrm{MC}}\right) = \mathrm{Var}\left(f_{\mathrm{MC}}\right) + a^2\,\mathrm{Var}\left(h_{\mathrm{MC}}\right) + 2a\,\mathrm{Cov}\left(f_{\mathrm{MC}}, h_{\mathrm{MC}}\right)$$

- Setting the coefficient $a = a^* = -\dfrac{\mathrm{Cov}\left(f_{\mathrm{MC}}, h_{\mathrm{MC}}\right)}{\mathrm{Var}\left(h_{\mathrm{MC}}\right)}$

$$\begin{aligned}
\mathrm{Var}\left(\widehat{f}_{\mathrm{MC}}\right) &= \mathrm{Var}\left(f_{\mathrm{MC}}\right) - \frac{\mathrm{Cov}\left(f_{\mathrm{MC}}, h_{\mathrm{MC}}\right)^2}{\mathrm{Var}\left(h_{\mathrm{MC}}\right)} \\
&= \mathrm{Var}\left(f_{\mathrm{MC}}\right) - \frac{\mathrm{Cov}\left(f_{\mathrm{MC}}, h_{\mathrm{MC}}\right)^2}{\mathrm{Var}\left(h_{\mathrm{MC}}\right)\mathrm{Var}\left(f_{\mathrm{MC}}\right)}\,\mathrm{Var}\left(f_{\mathrm{MC}}\right) \\
&= \left(1 - \rho\left(f_{\mathrm{MC}}, h_{\mathrm{MC}}\right)^2\right)\mathrm{Var}\left(f_{\mathrm{MC}}\right)
\end{aligned}$$

- Correlation coefficient $\rho(f_{MC}, h_{MC})$ is between -1 and 1. For maximum variance reduction, we want f_{MC} and h_{MC} to be highly correlated

Lecture overview

- Motivation
- **Tools:** REINFORCE, **Gumbel-Softmax**, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Towards reparameterized, continuous relaxations

- Consider the following optimization problem

$$\max_{\phi} \mathbb{E}_{q_\phi(z)}[f(z)]$$

- What if z is a discrete random variable?
  - Categories
  - Permutations
- Reparameterization trick is not directly applicable
- REINFORCE is a general-purpose solution, but needs careful design of control variates
- Relax z to a continuous random variable with a reparameterizable distribution

Gumbel Distribution

- Setting: We are given i.i.d. samples y_1, y_2, ..., y_n from some underlying distribution
  How can we model the distribution of

$$g = \max\{y_1, y_2, \ldots, y_n\}$$

- E.g., predicting maximum water level in a river for a particular river based on historical data to detect flooding
- The **Gumbel distribution** is very useful for modeling extreme, rare events, e.g., natural disasters, finance
- CDF for a Gumbel random variable g is parameterized by a location parameter μ and a scale parameter β

$$F(g; \mu, \beta) = \exp\left(-\exp\left(-\frac{g-\mu}{\beta}\right)\right)$$

- Note: If g is a Gumbel(μ, β) random variable, $-\log g$ is an Exponential(μ, β) random variable. Often, Gumbel r.v. are referred to as doubly exponential random variable

Categorical Distributions

- Let z denote a k-dimensional categorical random variable with distribution q parameterized by class probabilities $\pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$. We will represent z as a one-hot vector
- **Gumbel-Max reparameterization trick** for sampling from categorical random variables

$$z = \text{one-hot}\left(\arg\max_i \left(g_i + \log \pi_i\right)\right)$$

  where $g_1, g_2, \ldots, g_k$ are i.i.d. samples drawn from Gumbel(0,1)
- Reparametrizable since randomness is transferred to a fixed Gumbel(0,1) distribution!
- Problem: arg max is non-differentiable w.r.t. π

Relaxing Categorical Distributions to Gumbel-Softmax

- Gumbel-Max Sampler (non-differentiable w.r.t. π):

$$z = \text{one-hot}\left(\arg\max_i \left(g_i + \log \pi_i\right)\right)$$

- **Key idea:** Replace arg max with soft max to get a Gumbel-Softmax random variable $\hat{z}$
- Output of softmax is differentiable w.r.t. π
- Gumbel-Softmax Sampler (differentiable w.r.t. π):

$$\hat{z} = \text{soft}\max_i \left(\frac{g_i + \log \pi}{\tau}\right)$$

where $\tau > 0$ is a tunable parameter referred to as the temperature

Bias-variance tradeoff via temperature control

- Gumbel-Softmax distribution is parameterized by both class probabilities π and the temperature $\tau>0$

$$\hat{z} = \operatorname{soft}\max_i \left(\frac{g_i + \log \pi}{\tau} \right)$$

- Temperature τ controls the degree of the relaxation via a bias-variance tradeoff
- As $\tau \to 0$, samples from Gumbel-Softmax(π,τ) are similar to samples from Categorical(π)

  Pro: Low bias in approximation
  Con: High variance in gradients

- As $\tau \to \infty$, samples from Gumbel-Softmax(π,τ) are similar to samples from $\mathrm{Categorical}\left([1/k, 1/k, \ldots, 1/k]\right)$ (i.e., uniform over k categories)

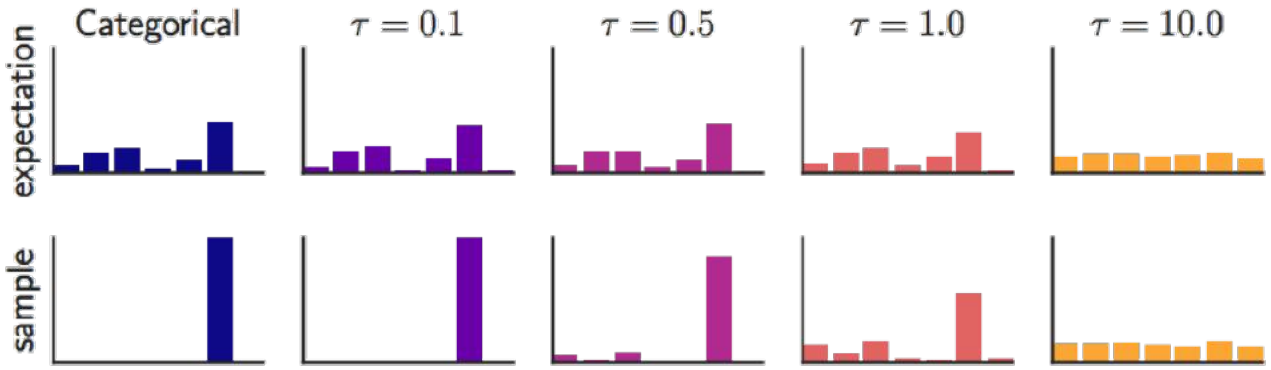
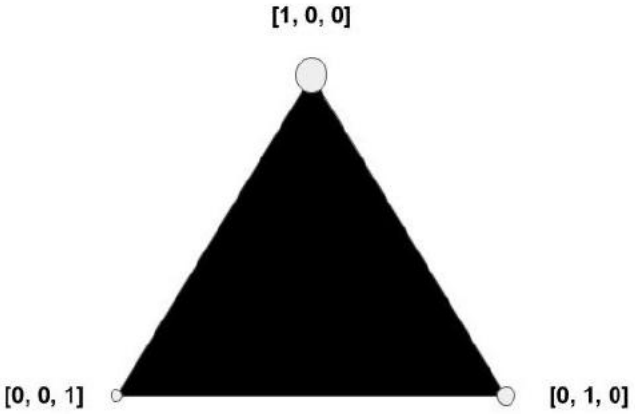


Image credit: Jang et al., 2017

Geometric Interpretation

- Consider a categorical distibution with class probabilities $\pi = [0.60, 0.25, 0.15]$
- Define a probability simplex with the one-hot vectors as vertices
- For a categorical distribution, all probability mass is concentrated at the vertices of the probability simplex
- Gumbel-Softmax samples points within the simplex (lighter color intensity implies higher probability)



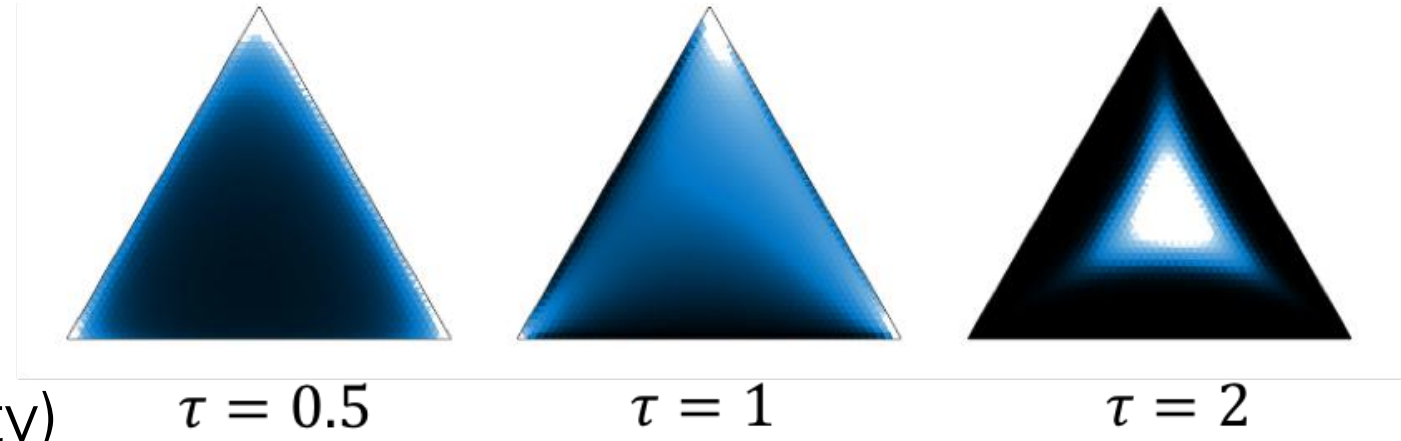


Image credit: Maddison et al., 2018

Lecture overview

- Motivation
- **Tools:** REINFORCE, Gumbel-Softmax, **Straight-through estimator**
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Straight-Through Estimation

- Want to take a gradient of the form $\nabla_\phi \mathop{\mathbb{E}}_{z \sim q_\phi} [f(z)]$.
- The straight-through estimator: $\tilde{\nabla}_\phi \mathop{\mathbb{E}}_{z \sim q_\phi} [f(z)] \equiv \mathop{\mathbb{E}}_{z \sim q_\phi} [(\nabla_\phi f(z)) \nabla_\phi q_\phi(z)]$
- Where does this come from? Ignore sampling and define $\tilde{\nabla}_\phi z \equiv \nabla_\phi q_\phi(z)$
- Use this to compute gradients of expressions with samples z:

$$\tilde{\nabla}_\phi f(z) = (\nabla_\phi f(z)) \tilde{\nabla}_\phi z = (\nabla_\phi f(z)) \nabla_\phi q_\phi(z)$$

Slide credit: John Thickstun

Summary

- Discovering discrete latent structure e.g., categories, rankings, matchings etc. has several applications
- Stochastic Optimization wrt parameterized discrete distributions is challenging
- REINFORCE is the general-purpose technique for gradient estimation, but suffers from high variance
- Control variates can help in controlling the variance
- Continuous relaxations to discrete distributions offer a biased, reparameterizable alternative with the trade-off in significantly lower variance

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- **Neural Variational Inference and Learning (NVIL)**
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Neural Variational Inference and Learning in Belief Networks

Andriy Mnih AMNIH@GOOGLE.COM
Karol Gregor KAROLG@GOOGLE.COM
Google DeepMind

Abstract

Highly expressive directed latent variable models, such as sigmoid belief networks, are difficult to train on large datasets because exact inference in them is intractable and none of the approximate inference methods that have been applied to them scale well. We propose a fast non-iterative approximate inference method that uses a feedforward network to implement efficient exact sampling from the variational posterior. The model and this inference network are trained jointly by maximizing a variational lower bound on the log-likelihood. Although the naive estimator of the inference network gradient is too high-variance to be useful, we make it practical by applying several straightforward model-independent variance reduction techniques. Applying our approach to training sigmoid belief networks and deep autoregressive networks, we show that it outperforms the wake-sleep algorithm on MNIST and achieves state-of-the-art results on the Reuters RCV1 document dataset.

of Markov Chain Monte Carlo (MCMC) methods makes them straightforward to apply to models of this type (Neal, 1992), they tend to suffer from slow mixing and are usually too computationally expensive to be practical in all but the simplest models. Such methods are also difficult to scale to large datasets because they need to store the current state of the latent variables for all the training observations between parameter updates.

Variational methods (Jordan et al., 1999) provide an optimization-based alternative to the sampling-based Monte Carlo methods, and tend to be more efficient. They involve approximating the exact posterior using a distribution from a more tractable family, often a fully factored one, by maximizing a variational lower bound on the log-likelihood w.r.t. the parameters of the distribution. For a small class of models, using such variational posteriors allows the expectations that specify the parameter updates to be computed analytically. However, for highly expressive models such as the ones we are interested in, these expectations are intractable even with the simplest variational posteriors. This difficulty is usually dealt with by lower bounding the intractable expectations with tractable one by introducing more variational parameters, as was done for sig-

Mnih & Gregor, 2014

Neural Variational Inference and Learning

- Latent variable models with discrete latent variables are often referred to as belief networks
- Variational learning objective is same as ELBO

$$\begin{aligned}\mathcal{L}(x;\theta,\phi) &= \sum_z q_\phi(z \mid x) \log p(z,x;\theta) + H\left(q_\phi(z \mid x)\right) \\ &= E_{q_\phi(z|x)}\left[\log p(z,x;\theta) - \log q_\phi(z \mid x))\right] \\ &:= E_{q_\phi(z|x)}[f(\phi,\theta,z,x)]\end{aligned}$$

- Here, z is discrete and hence we cannot use reparameterization

Neural Variational Inference and Learning

- NVIL (Mnih&Gregor, 2014) learns belief networks via REINFORCE + control variates
- Learning objective

$$\mathcal{L}(x;\theta,\phi,\psi,B) = E_{q_\phi(z|x)}\left[f(\phi,\theta,z,x) - h_\psi(x) - B\right]$$

- **Control Variate 1:** Constant baseline B
- **Control Variate 2:** Input dependent baseline $h_\psi(x)$
- Both B and ψ are learned via gradient descent
- Gradient estimates w.r.t. ϕ

$$\nabla_\phi \mathcal{L}(x;\theta,\phi,\psi,B)$$
$$= E_{q_\phi(z|x)}\left[\left(f(\phi,\theta,z,x) - h_\psi(x) - B\right)\nabla_\phi \log q_\phi(z \mid x) + \nabla_\phi f(\phi,\theta,z,x)\right]$$

Neural Variational Inference and Learning

Table 1. Results on the binarized MNIST dataset. "Dim" is the number of latent variables in each layer, starting with the deepest one. NVIL and WS refer to the models trained with NVIL and wake-sleep respectively. NLL is the negative log-likelihood for the tractable models and an estimate of it for the intractable ones.

MODEL	DIM	TEST NLL	
		NVIL	WS
SBN	200	113.1	120.8
SBN	500	112.8	121.4
SBN	200-200	99.8	107.7
SBN	200-200-200	96.7	102.2
SBN	200-200-500	97.0	102.3
FDARN	200	92.5	95.9
FDARN	500	90.7	97.2
FDARN	400	96.3	
DARN	400	93.0	
NADE	500	88.9	
RBM (CD3)	500	105.5	
RBM (CD25)	500	86.3	
MOB	500	137.6	

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- **Vector Quantization VAE (VQ-VAE)**, VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Neural Discrete Representation Learning

Aaron van den Oord
DeepMind
avdnoord@google.com

Oriol Vinyals
DeepMind
vinyals@google.com

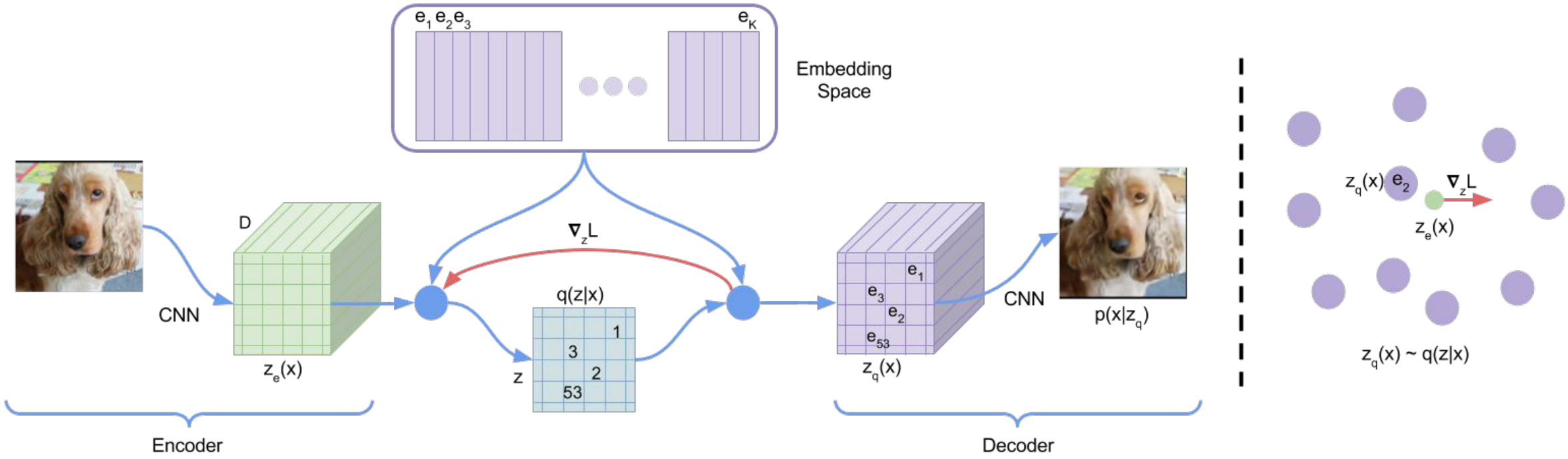
Koray Kavukcuoglu
DeepMind
korayk@google.com

Abstract

Learning useful representations without supervision remains a key challenge in machine learning. In this paper, we propose a simple yet powerful generative model that learns such discrete representations. Our model, the Vector Quantised-Variational AutoEncoder (VQ-VAE), differs from VAEs in two key ways: the encoder network outputs discrete, rather than continuous, codes; and the prior is learnt rather than static. In order to learn a discrete latent representation, we incorporate ideas from vector quantisation (VQ). Using the VQ method allows the model to circumvent issues of "posterior collapse" —- where the latents are ignored when they are paired with a powerful autoregressive decoder —- typically observed in the VAE framework. Pairing these representations with an autoregressive prior, the model can generate high quality images, videos, and speech as well as doing high quality speaker conversion and unsupervised learning of phonemes, providing further evidence of the utility of the learnt representations.

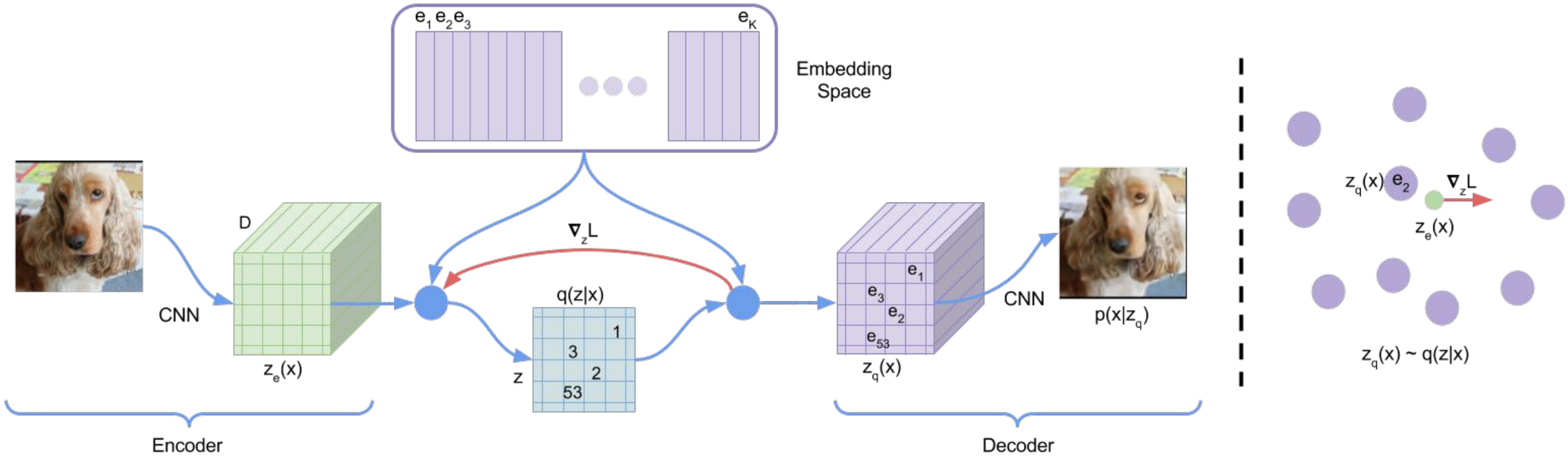
van den Oord et al., 2017

VQ-VAE



- A latent embedding space $e \in \mathbb{R}^{K \times D}$ where K is the size of the discrete latent space (K-way categorical distribution)
- Encoder output $z_e(x)$ is mapped to discrete latent code z by a nearest neighbour look-up using the shared embedding space e

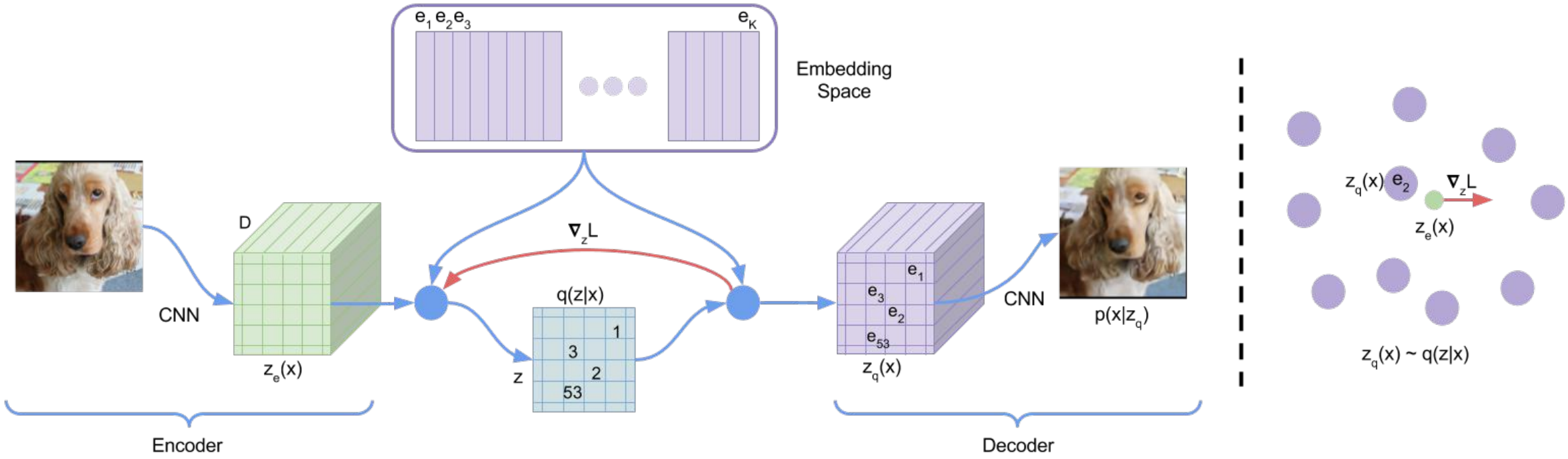
VQ-VAE



- The posterior categorical distribution $q(z|x)$ probabilities are defined as one-hot as:

$$q(z = k \mid x) = \begin{cases} 1 & \text{for k} = \text{argmin}_j \left\| z_e(x) - e_j \right\|_2 \\ 0 & \text{otherwise} \end{cases}$$

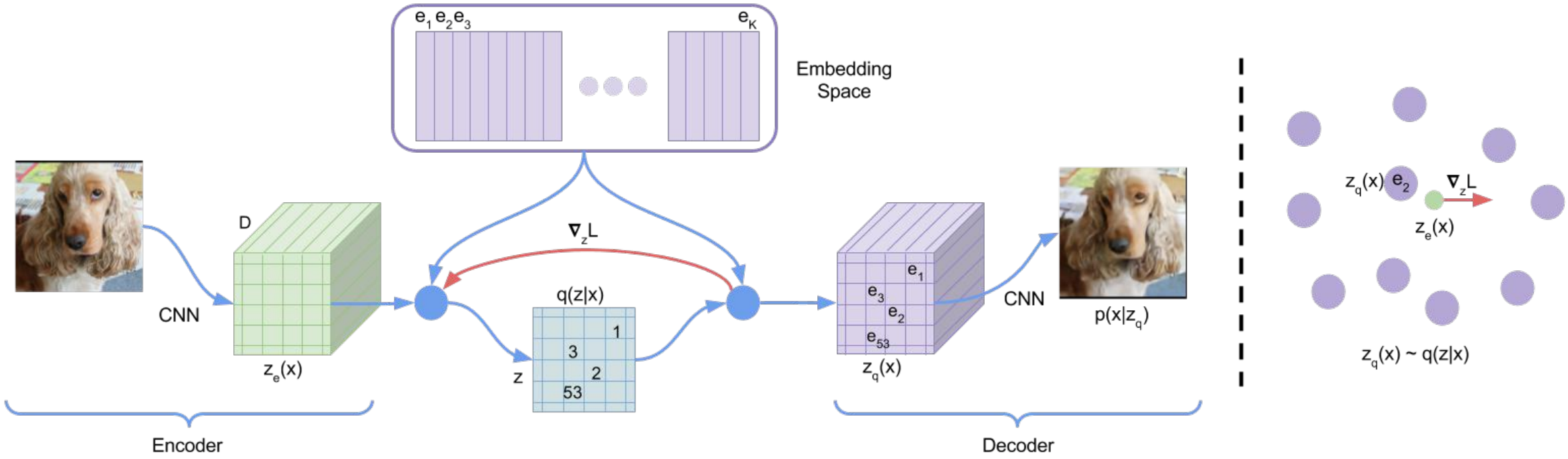
VQ-VAE



- This model can be seen as a VAE in which the proposal distribution $q(z = k|x)$ is deterministic.
- The representation $z_e(x)$ is passed through the discretisation bottleneck followed by mapping onto the nearest element of embedding e

VQ-VAE

sg stands for the stopgradient operator that is defined as identity at forward computation time and has zero partial derivatives, thus effectively constraining its operand to be a non-updated constant.

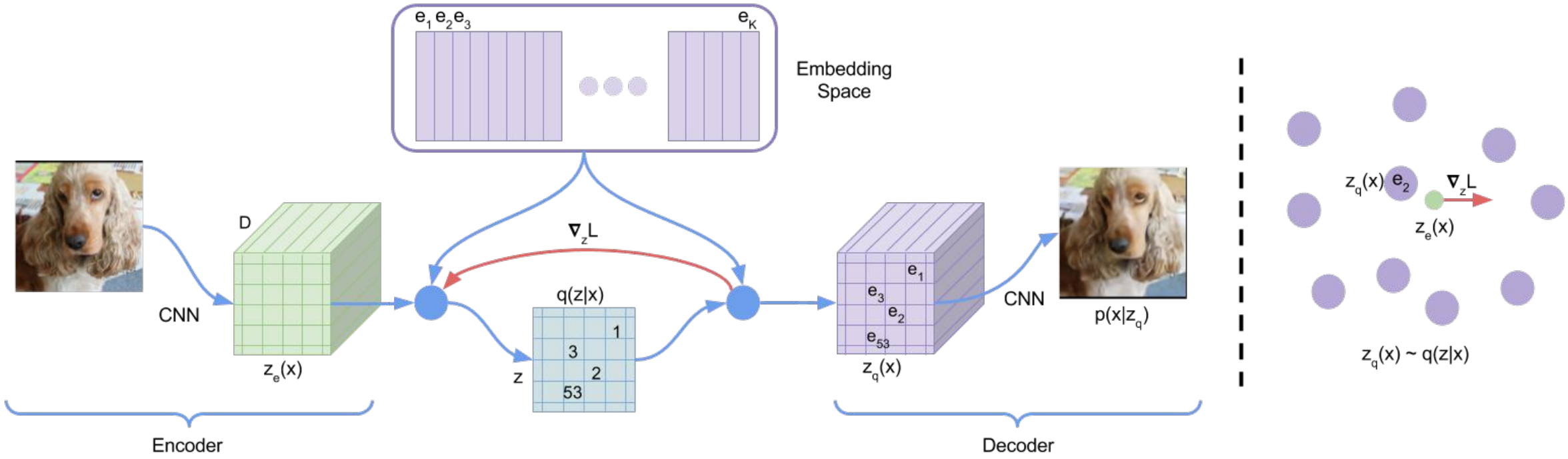


$$z_q(x) = e_k, \quad \text{where} \quad k = \text{argmin}_j \|z_e(x) - e_j\|_2$$

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

VQ-VAE

sg stands for the stopgradient operator that is defined as identity at forward computation time and has zero partial derivatives, thus effectively constraining its operand to be a non-updated constant.



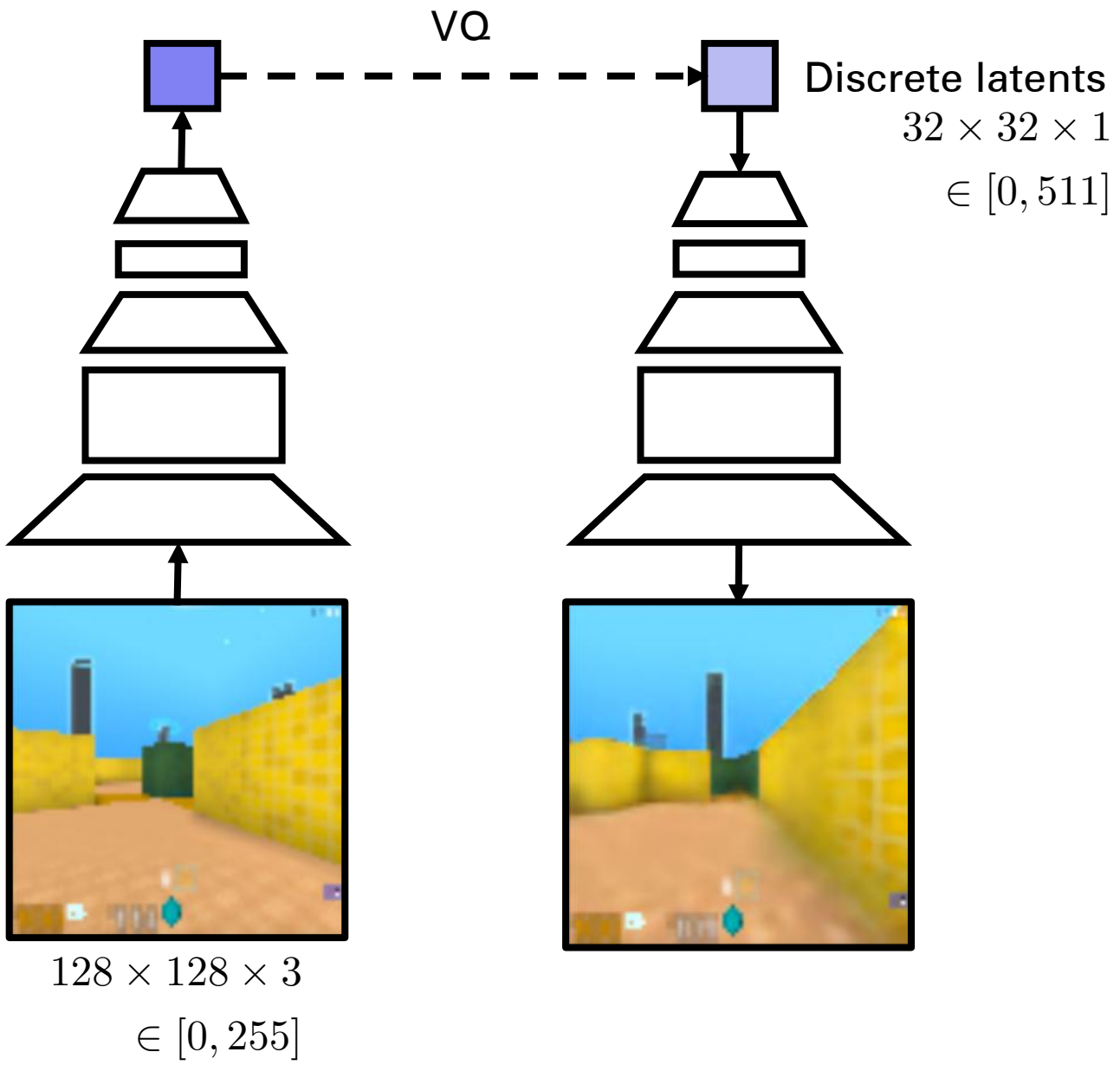
$$L = \log p\left(x \mid z_q(x)\right) + \left\|\mathrm{sg}\left[z_e(x)\right] - e\right\|_2^2 + \beta \left\|z_e(x) - \mathrm{sg}[e]\right\|_2^2$$

The embedding space is learned via Vector Quantisation (VQ), whose objective uses the l_2 error to move the embedding vectors e_i towards the encoder outputs $z_e(x)$

VQ-VAE

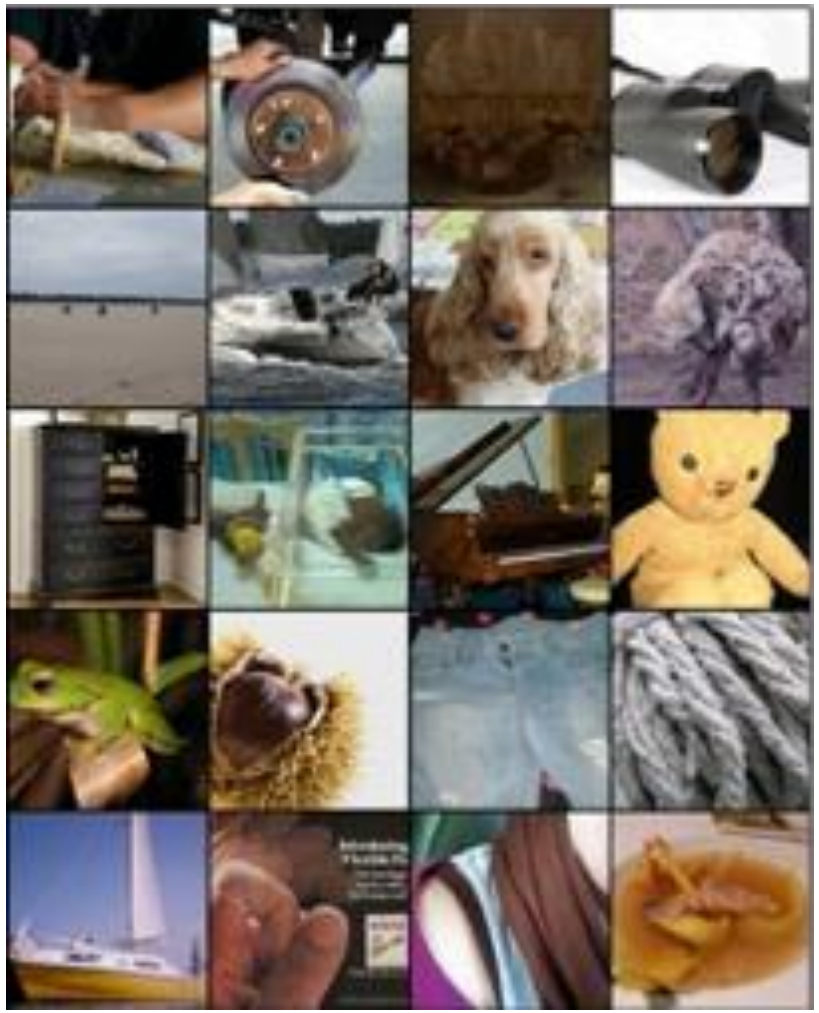
- For speech, image and videos one can respectively extract a 1D, 2D and 3D latent feature spaces

- 32 x 32 latents for ImageNet, or 8 x 8 x 10 for CIFAR10
- 512-dimensional latents for VCTK and LibriSpeech

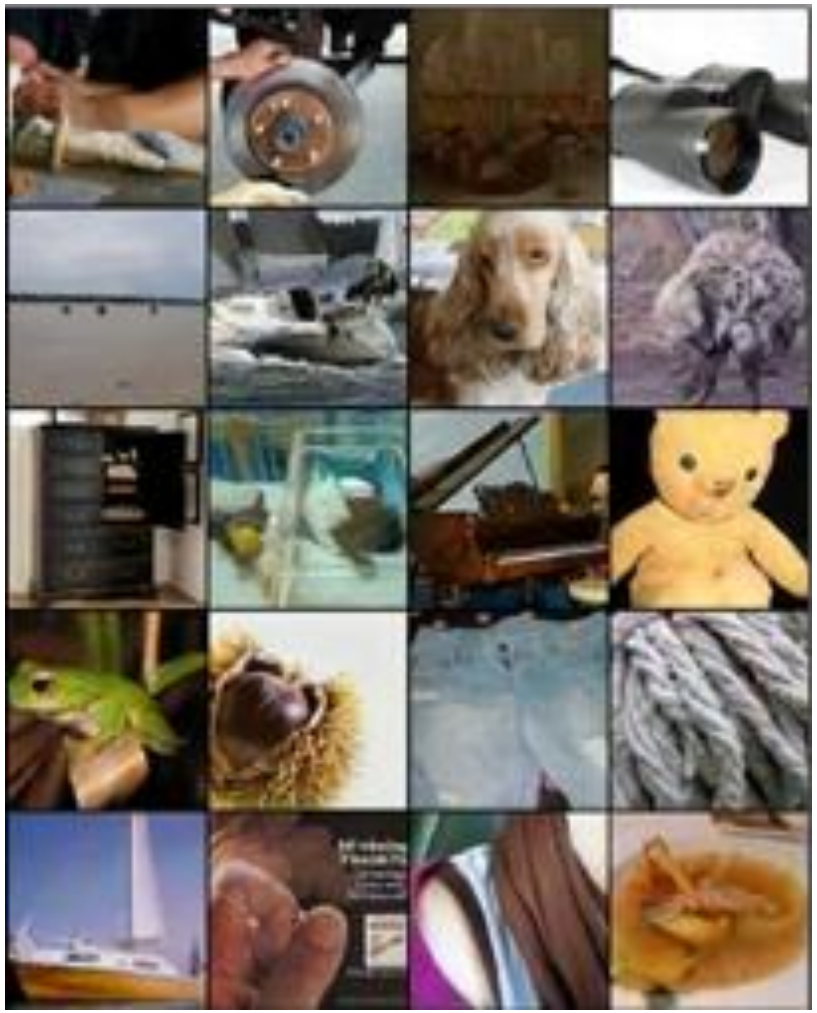


VQ-VAE ImageNet Reconstructions

Original 128 x 128 images

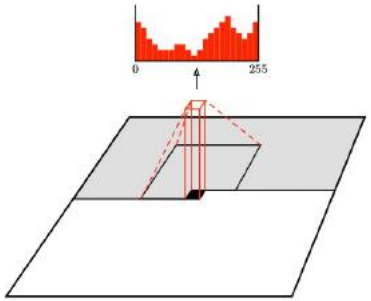


Reconstructions



Slide credit: Aäron van den Oord

VQ-VAE Sampling from prior



PixelCNN Prior

Sampled Discrete latents

$32 \times 32 \times 1$

$\in [0, 511]$

Decoder

Slide credit: Aäron van den Oord

VQ-VAE ImageNet Samples

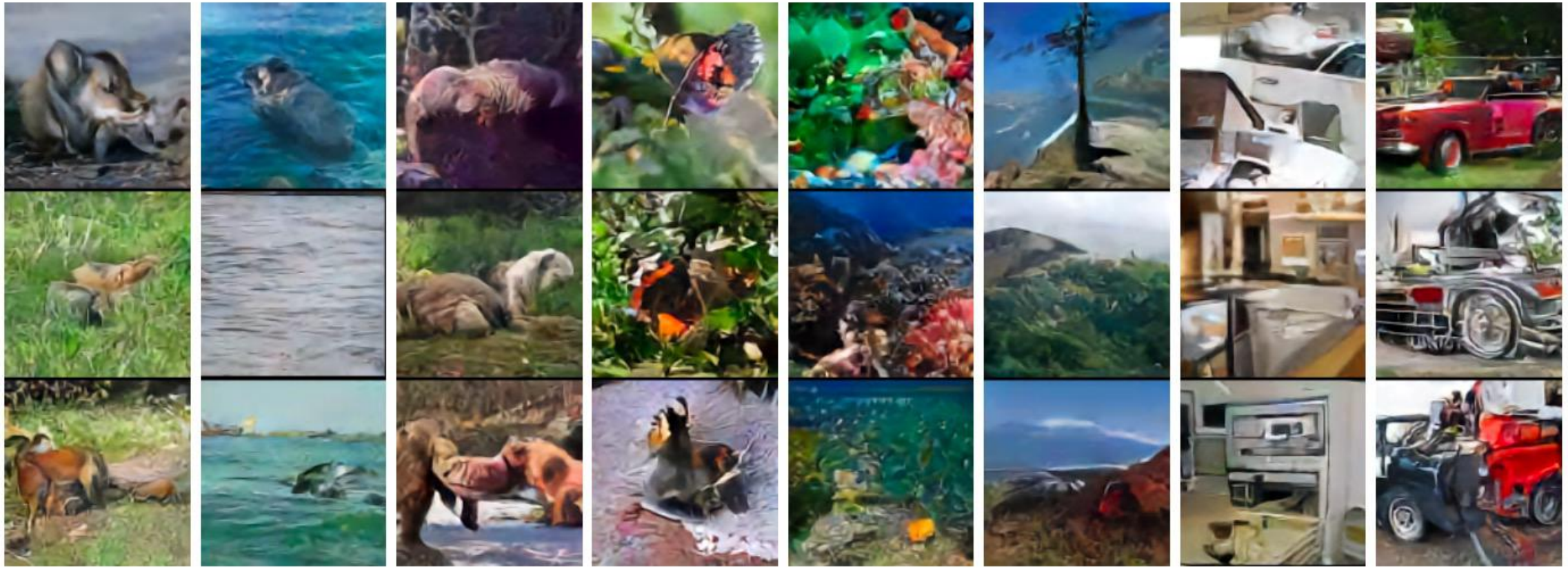
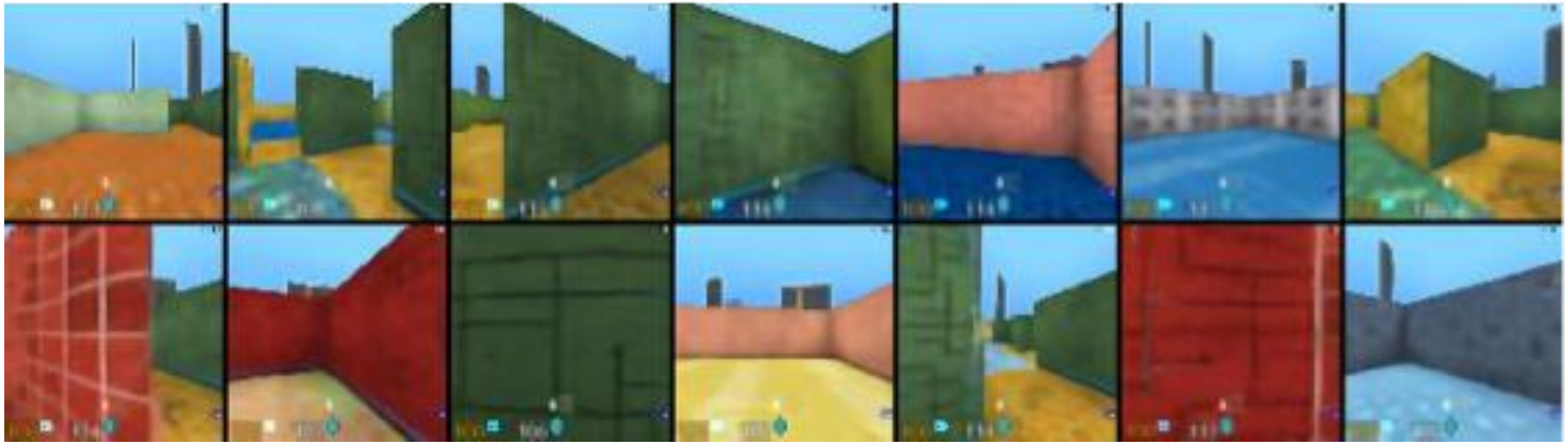
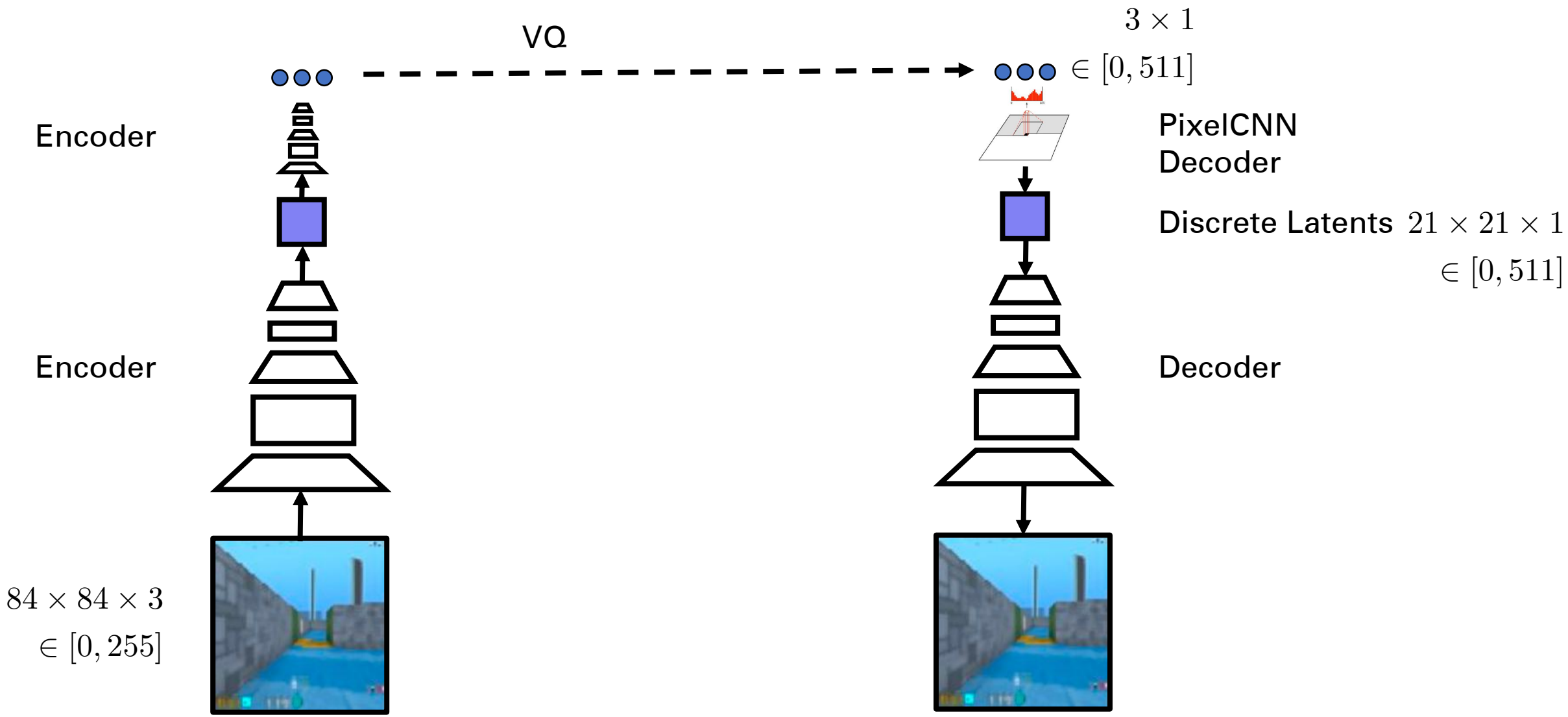


Figure 3: Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images. From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave, pickup.

VQ-VAE DM-Lab Samples



VQ-VAE 3 Global Latents Reconstruction

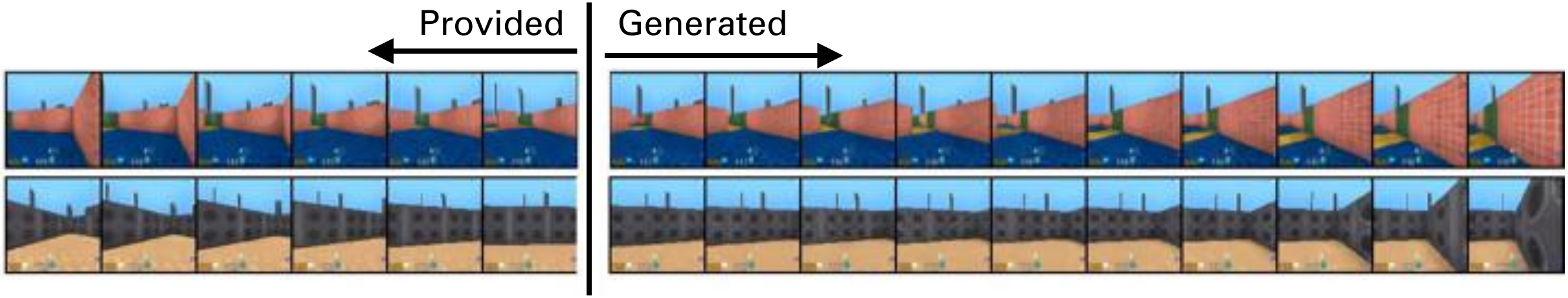


VQ-VAE 3 Global Latents Reconstruction

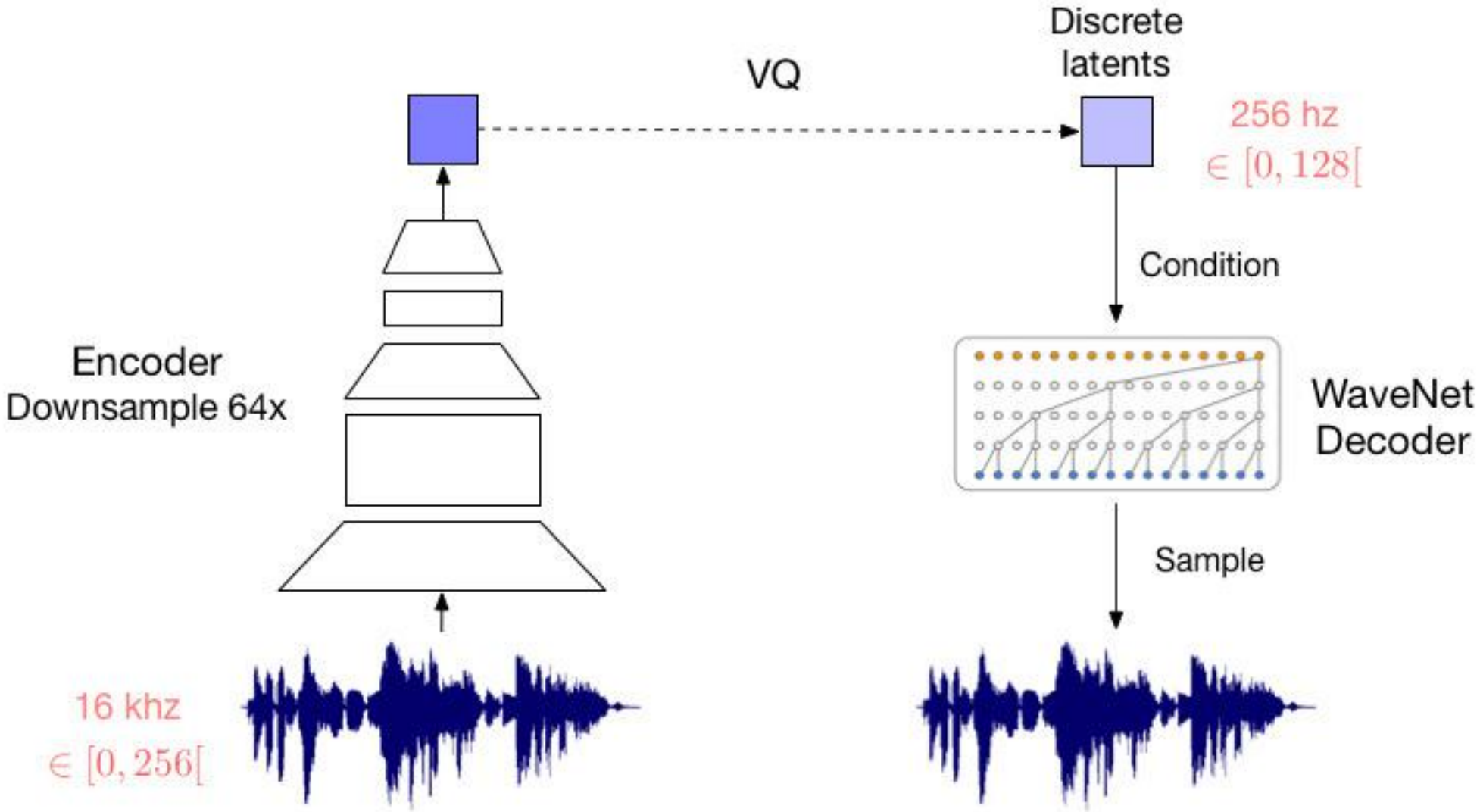
Originals

Reconstructions from compressed representations (27 bits per image)

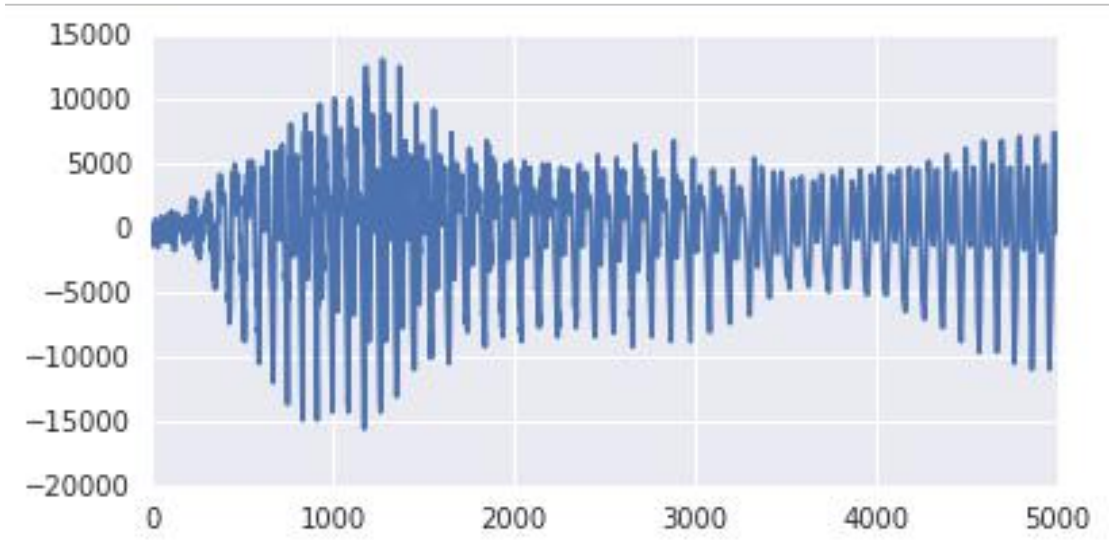
VQ-VAE Video Generation in the Latent Space



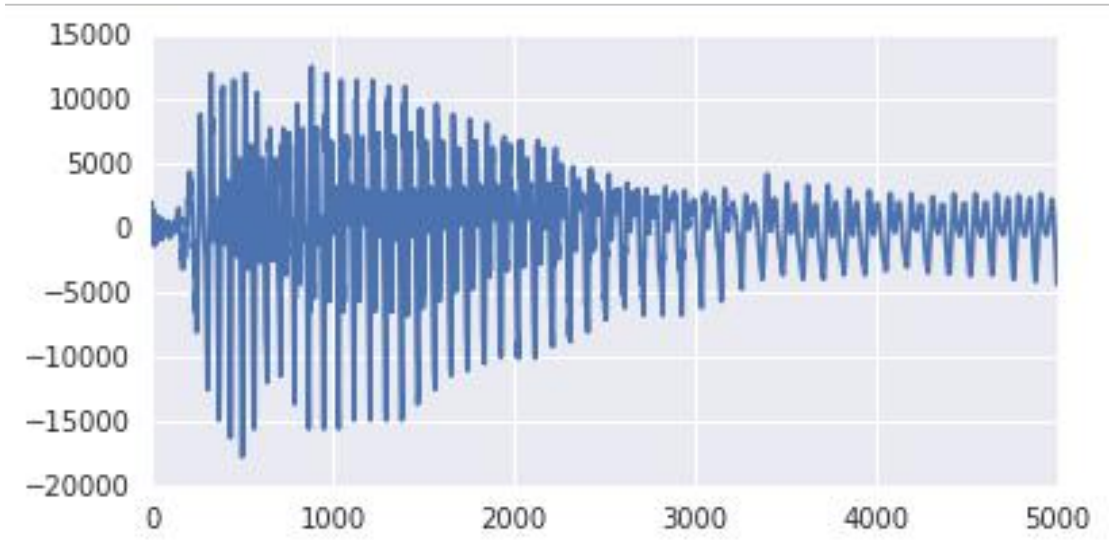
VQ-VQA Speech Modeling



VQ-VQA VCTK Reconstructions

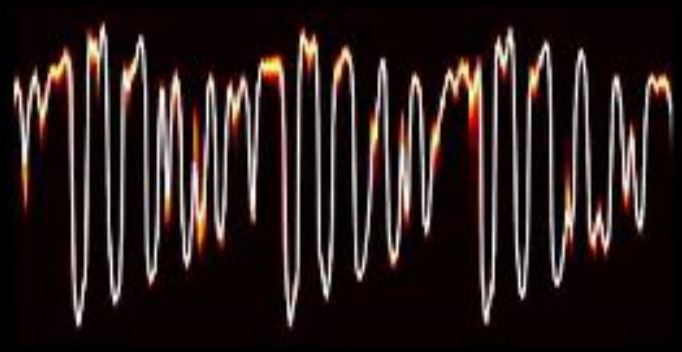


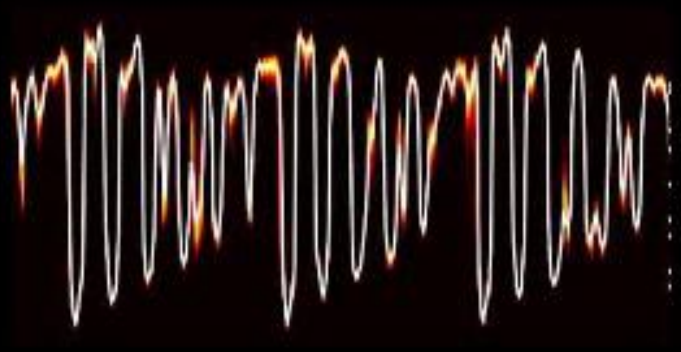
Original



Reconstruction

VQ-VAE Reconstructions





VQ-VAE Sampling from prior

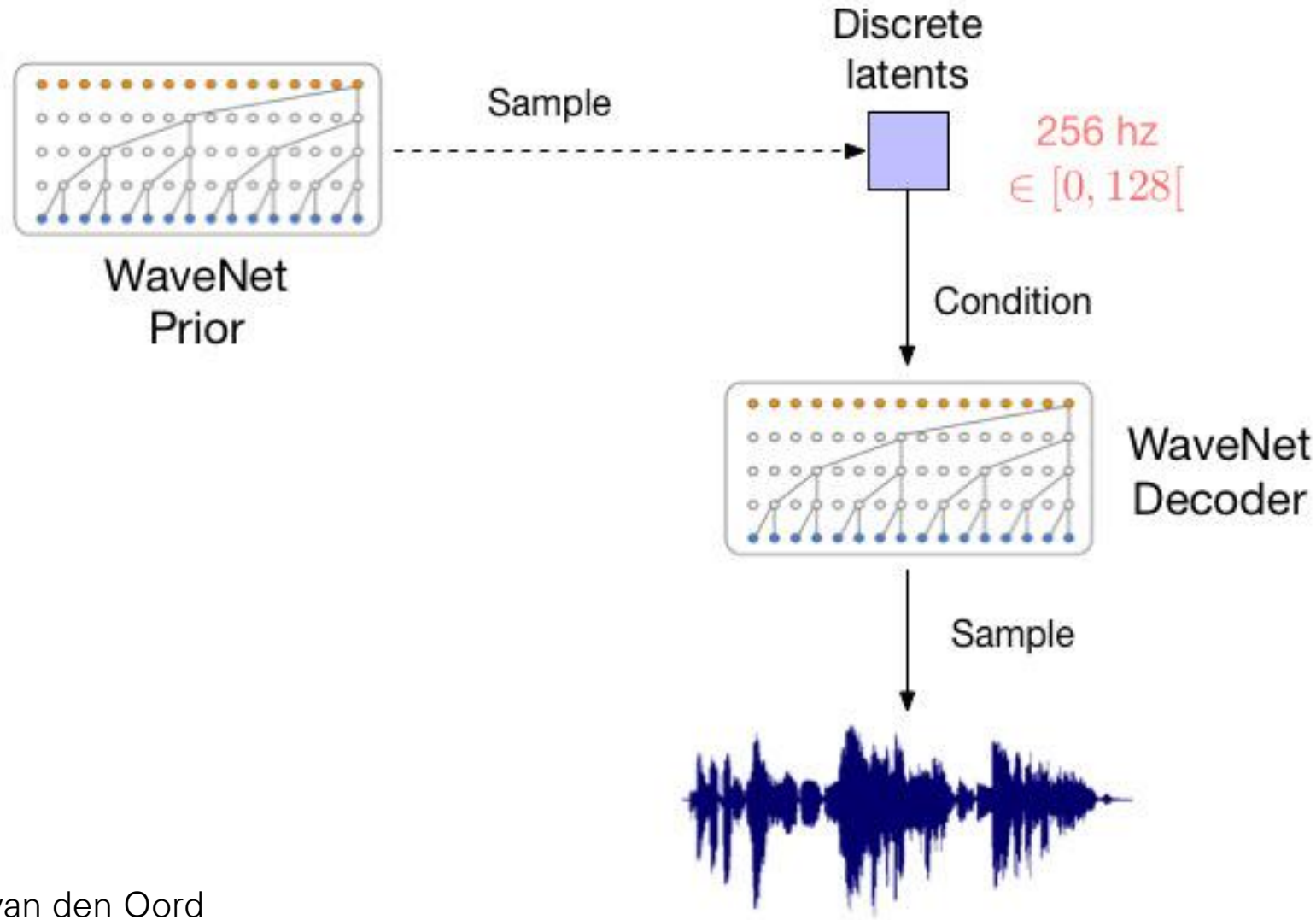
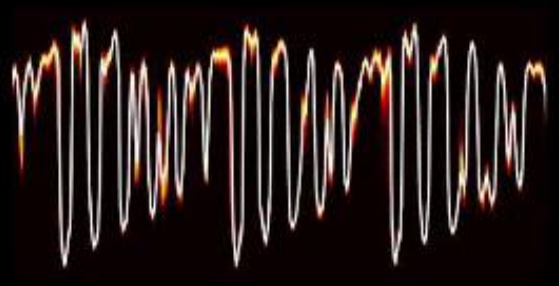
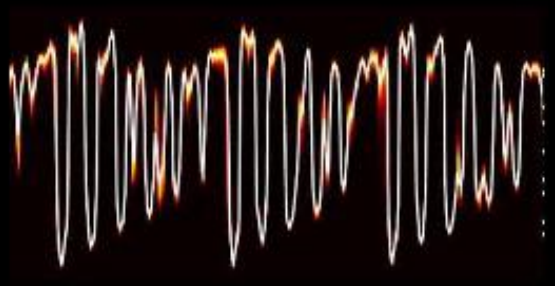
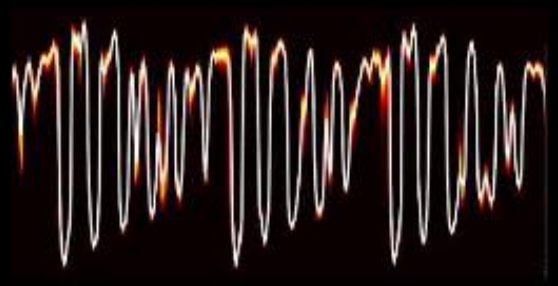


Image credit: Aäron van den Oord

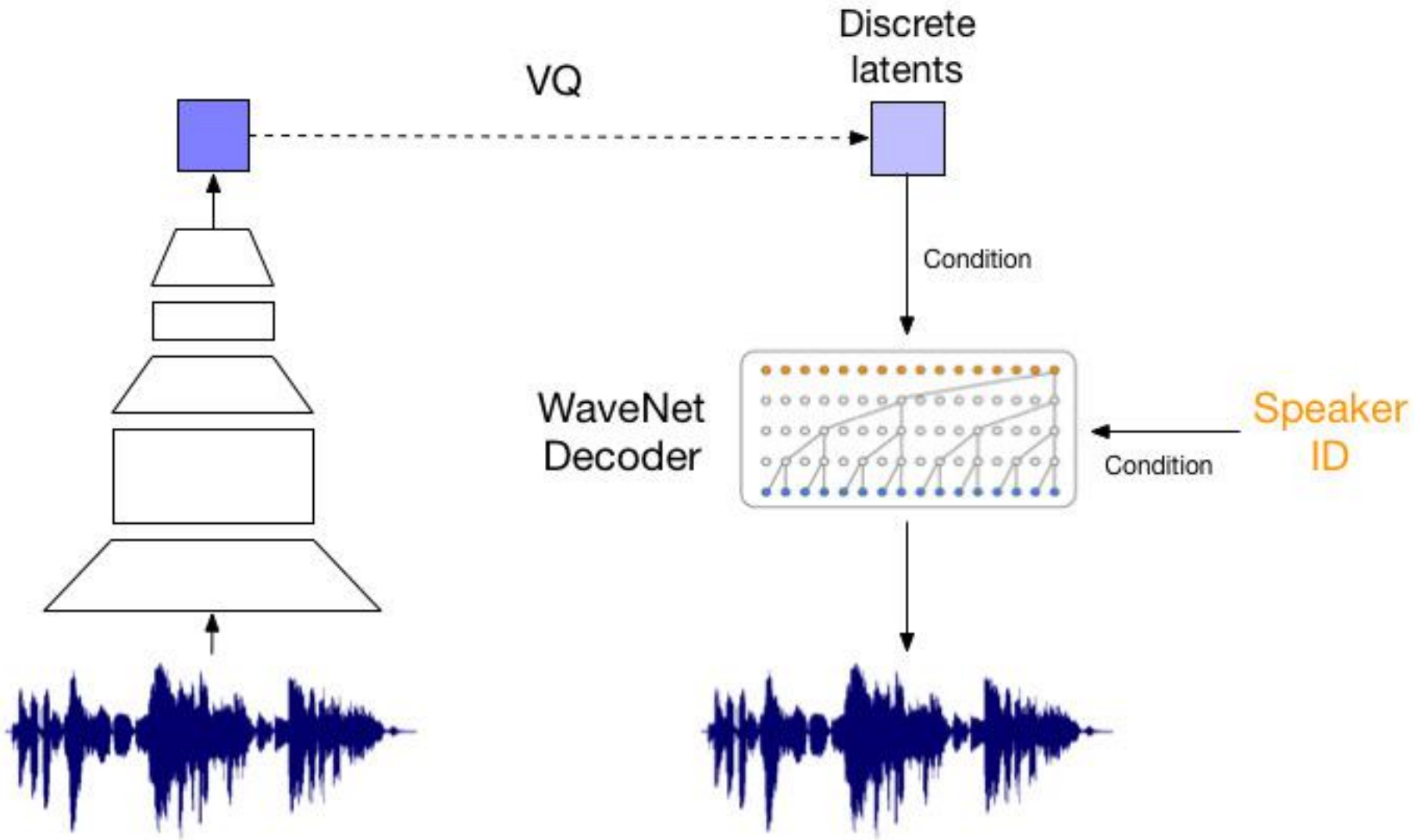
VQ-VAE Samples



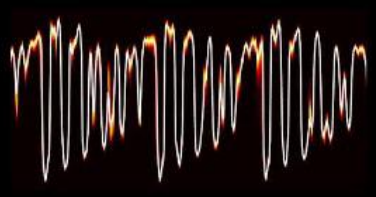


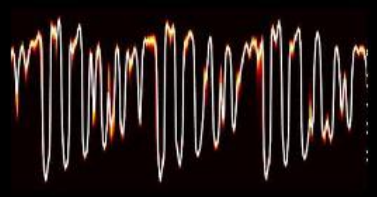


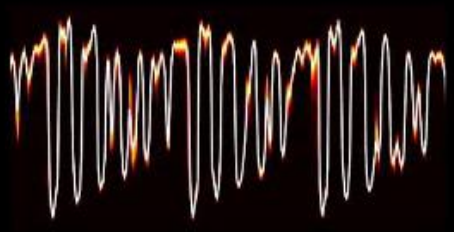
VQ-VAE Voice Style-Transfer

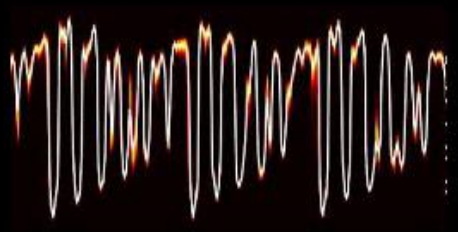


VQ-VAE Voice Style-Transfer Results

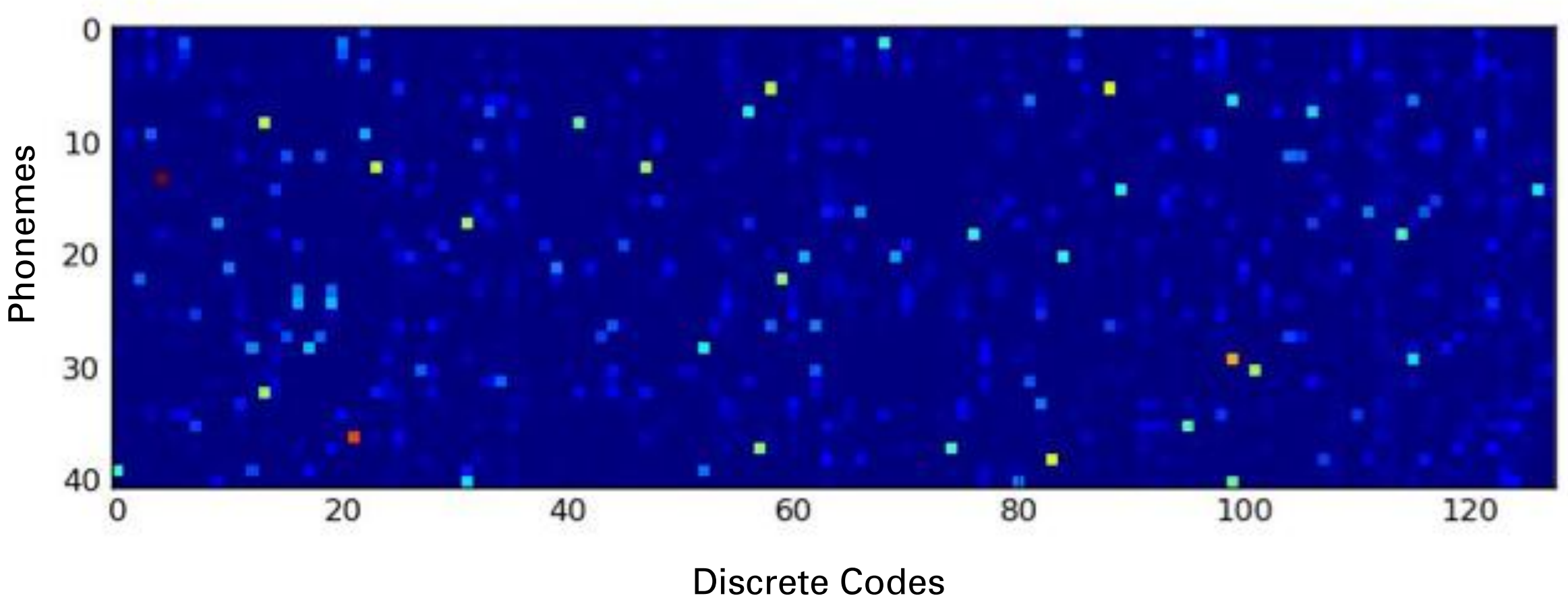








Unsupervised Learning of Phonemes



- 41-way classification
- 49.3% accuracy **fully unsupervised**

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), **VQ-VAE-2**, VQGAN
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Generating Diverse High-Fidelity Images with VQ-VAE-2

Ali Razavi* DeepMind alirazavi@google.com

Aäron van den Oord* DeepMind avdnoord@google.com

Oriol Vinyals DeepMind vinyals@google.com

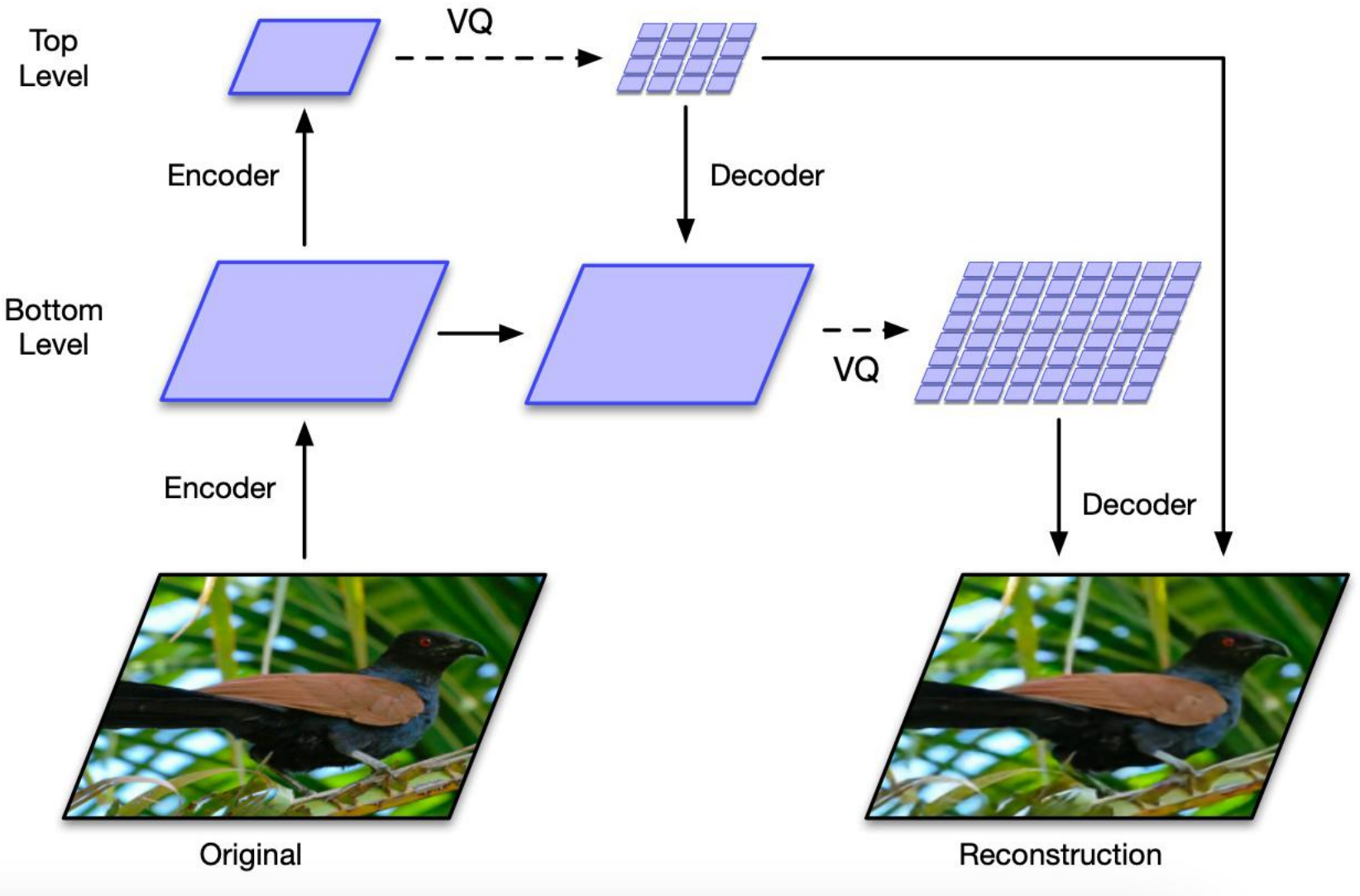
Abstract

We explore the use of Vector Quantized Variational AutoEncoder (VQ-VAE) models for large scale image generation. To this end, we scale and enhance the autoregressive priors used in VQ-VAE to generate synthetic samples of much higher coherence and fidelity than possible before. We use simple feed-forward encoder and decoder networks, making our model an attractive candidate for applications where the encoding and/or decoding speed is critical. Additionally, VQ-VAE requires sampling an autoregressive model only in the compressed latent space, which is an order of magnitude faster than sampling in the pixel space, especially for large images. We demonstrate that a multi-scale hierarchical organization of VQ-VAE, augmented with powerful priors over the latent codes, is able to generate samples with quality that rivals that of state of the art Generative Adversarial Networks on multifaceted datasets such as ImageNet, while not suffering from GAN's known shortcomings such as mode collapse and lack of diversity.

Razavi et al., 2019

VQ-VAE-2

- The input 256×256 image is compressed to quantized latent maps of size 64×64 and 32×32
- The decoder reconstructs the image from the two latent maps.



VQ-VAE-2 Reconstructions

h_{top}

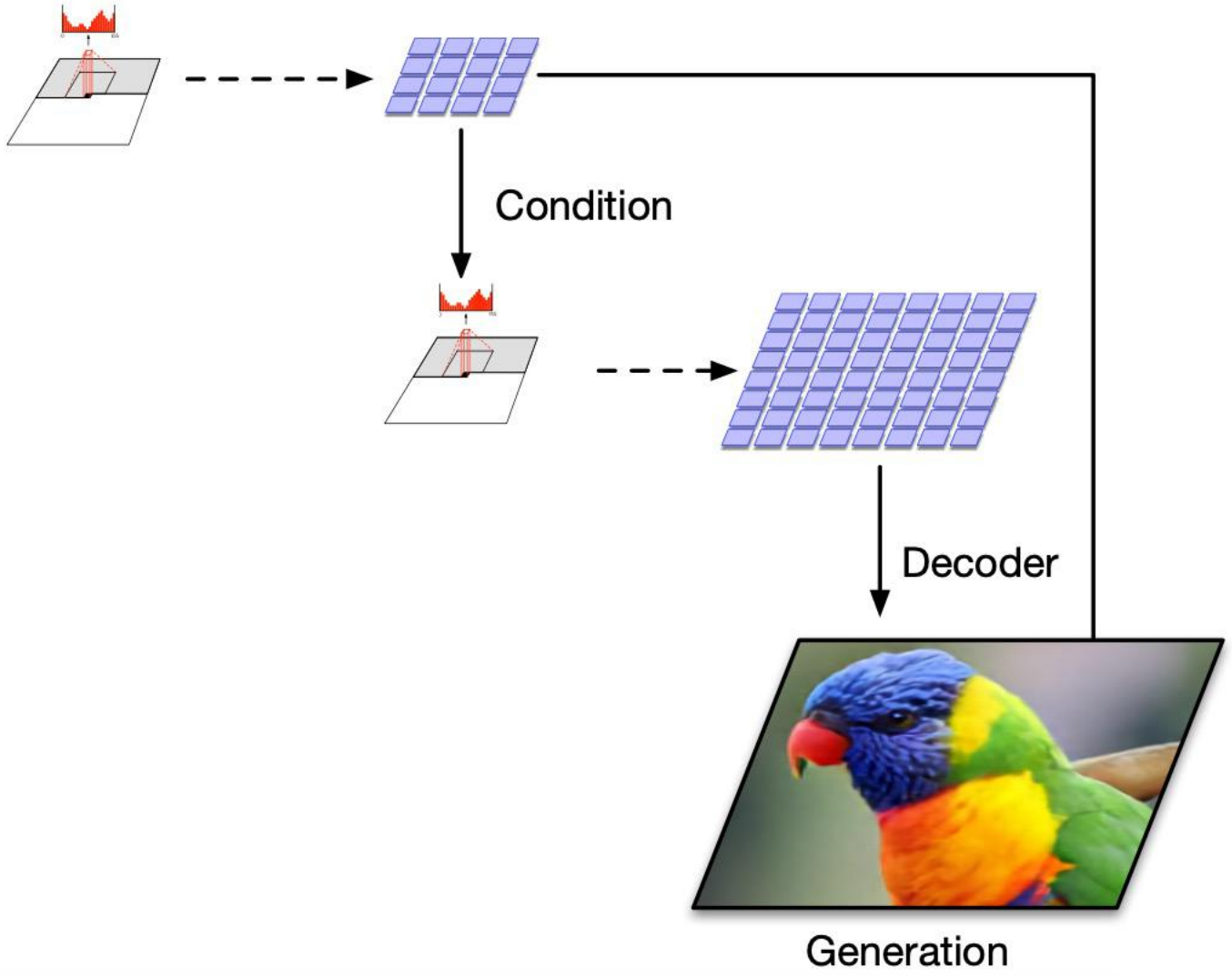
$h_{\text{top}}, h_{\text{middle}}$

$h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$

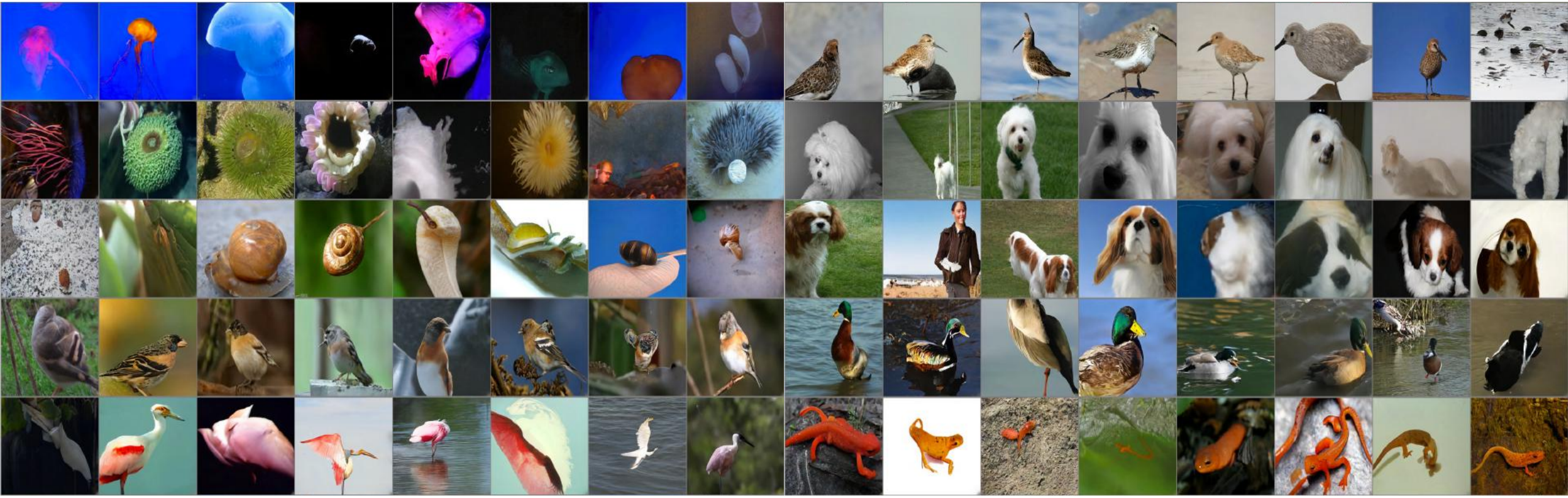
Original

- Each latent map adds extra detail to the reconstruction.
- These latent maps are approximately 3072x, 768x, 192x times smaller than the original image, respectively.

VQ-VAE-2



VQ-VAE-2 ImageNet Samples



VQ-VAE-2 FFHQ-2014 Samples

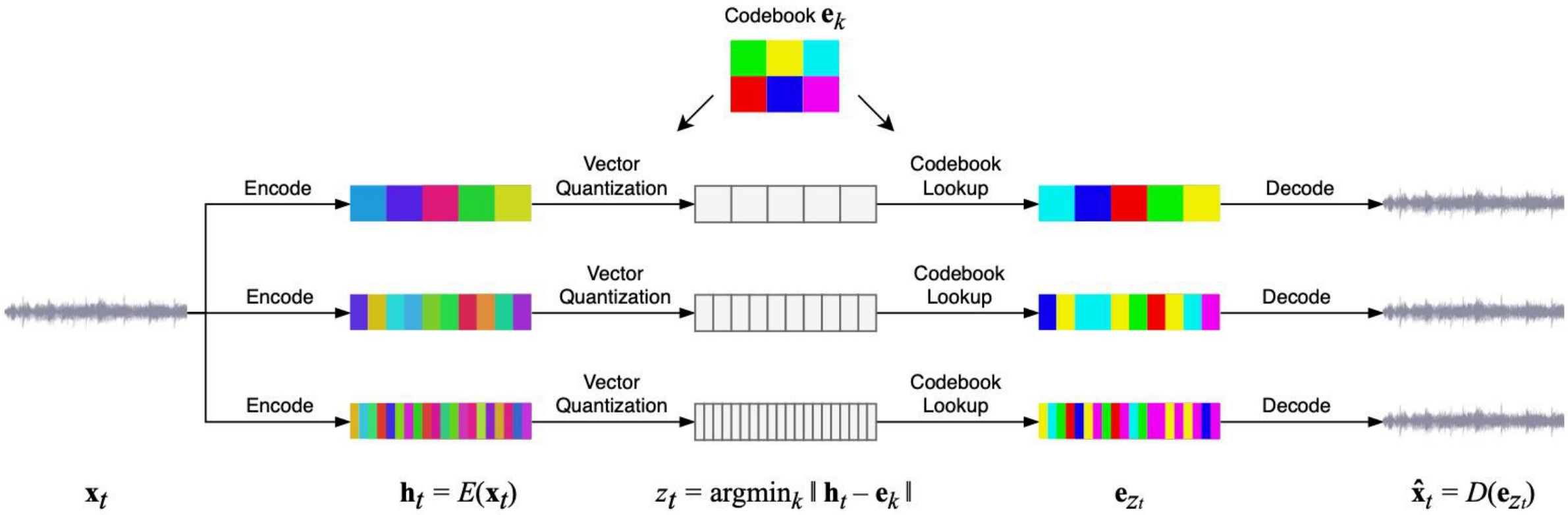
VQ-VAE-2 Quantitative Evaluation

- The VQ-VAE-2 paper reports a classification accuracy score (CAS) for class-conditional image generation.
- Generate image-class pairs from the generative model trained on the ImageNet training data.
- Train an image classifier from the generated pairs and measure its accuracy on the ImageNet test set.

	Top-1 Accuracy	Top-5 Accuracy
BigGAN deep	42.65	65.92
VQ-VAE	54.83	77.59
VQ-VAE after reconstructing	58.74	80.98
Real data	73.09	91.47

Slide credit: David McAllester

VQ-VAE-2 for Music Generation



- Three separate VQ-VAE models with different temporal resolutions

Jukebox

We're introducing Jukebox, a neural net that generates music, including rudimentary singing, as raw audio in a variety of genres and artist styles. We're releasing the model weights and code, along with a tool to explore the generated samples.

READ PAPER VIEW CODE

https://openai.com/blog/jukebox/

April 30, 2020
12 minute read, 10 day listen

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, **VQGAN**
- Discrete Flows, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

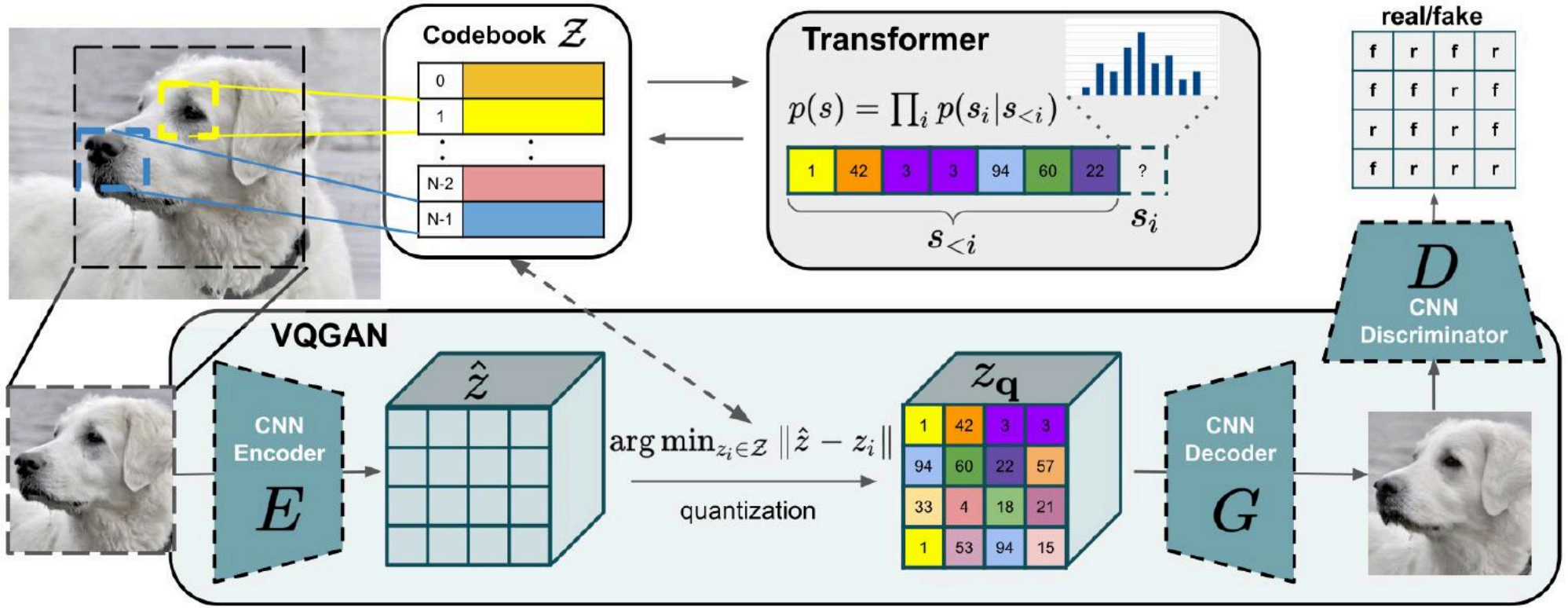
Taming Transformers for High-Resolution Image Synthesis

Patrick Esser* Robin Rombach* Björn Ommer
Heidelberg Collaboratory for Image Processing, IWR, Heidelberg University, Germany
*Both authors contributed equally to this work

Designed to learn long-range interactions on sequential data, transformers continue to show state-of-the-art results on a wide variety of tasks. In contrast to CNNs, they contain no inductive bias that prioritizes local interactions. This makes them expressive, but also computationally infeasible for long sequences, such as high-resolution images. We demonstrate how combining the effectiveness of the inductive bias of CNNs with the expressivity of transformers enables them to model and thereby synthesize high-resolution images. We show how to (i) use CNNs to learn a context-rich vocabulary of image constituents, and in turn (ii) utilize transformers to efficiently model their composition within high-resolution images. Our approach is readily applied to conditional synthesis tasks, where both non-spatial information, such as object classes, and spatial information, such as segmentations, can control the generated image. In particular, we present the first results on semantically-guided synthesis of megapixel images with transformers. Project page at https://git.io/JLlvY.

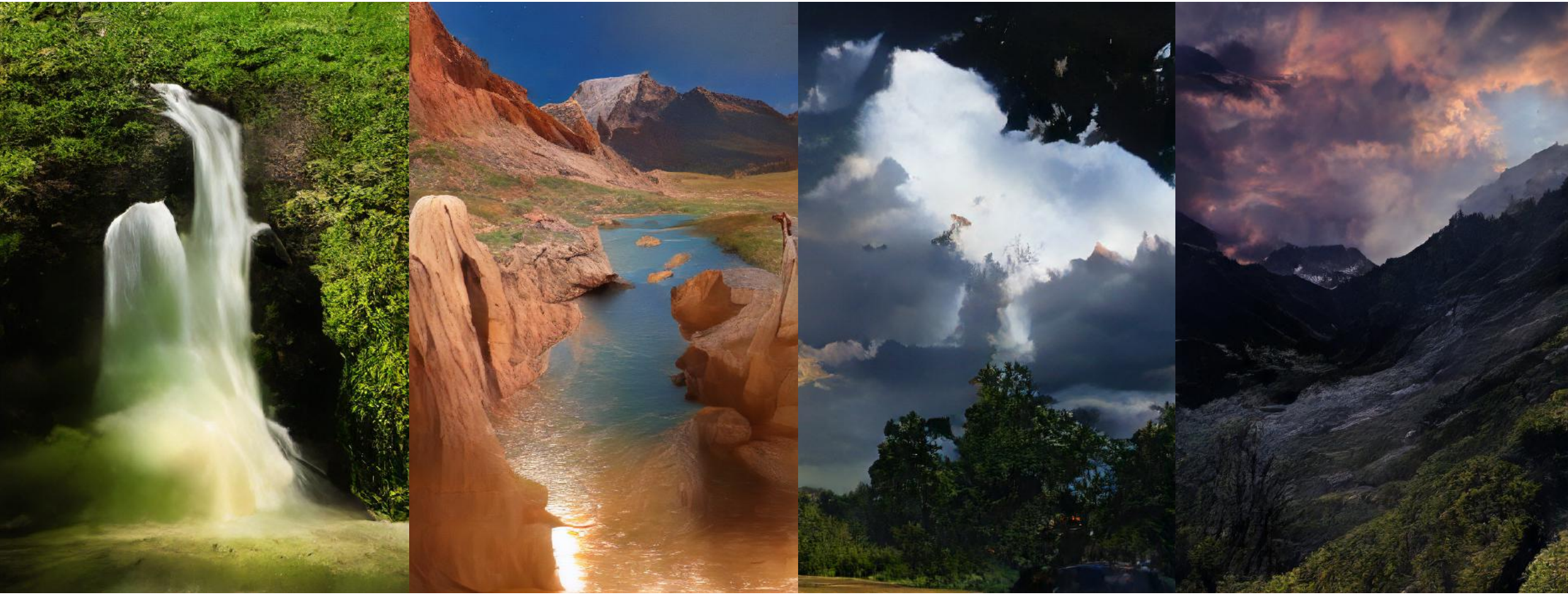
Esser et al. 2021

VQGAN



- A convolutional VQGAN to learn a codebook of context-rich visual parts
- An autoregressive Transformer to generate novel samples

S-FLCKR Samples from Semantic Layouts



ImageNet Samples

Quantitative Evaluation

CelebA-HQ 256 × 256		FFHQ 256 × 256	
Method	FID ↓	Method	FID ↓
GLOW [33]	69.0	VDVAE ($t = 0.7$) [11]	38.8
NVAE [59]	40.3	VDVAE ($t = 1.0$)	33.5
PIONEER (B.) [21]	39.2 (25.3)	VDVAE ($t = 0.8$)	29.8
NCPVAE [1]	24.8	VDVAE ($t = 0.9$)	28.5
VAEBM [66]	20.4	*VQGAN*+P.SNAIL	21.9
Style ALAE [49]	19.2	BigGAN	12.4
DC-VAE [47]	15.8	**ours**	11.4
ours	10.7	U-Net GAN (+aug) [57]	10.9 (7.6)
PGGAN [27]	8.0	StyleGAN2 (+aug) [30]	3.8 (3.6)

Table 3. FID score comparison for face image synthesis. CelebA-HQ results reproduced from [1, 47, 66, 22], FFHQ from [57, 28].

Summary: VQ-VAE/VQ-VAE-2/VQGAN

- VQ-VAE/VQ-VAE-2 uses discrete latents, thus does not suffer from "posterior collapse" and has no variance issues
- VQ-VAE/VQ-VAE-2 perform as well as its continuous model counterparts in terms of log-likelihood scores
- When paired with a powerful prior (decoder), generated samples are high quality

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- **Discrete Flows**, Discrete Integer Flows
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Discrete Flows: Invertible Generative Models of Discrete Data

Dustin Tran[1] **Keyon Vafa**[12*] **Kumar Krishna Agrawal**[1†] **Laurent Dinh**[1] **Ben Poole**[1]
[1]Google Brain [2]Columbia University

Abstract

While normalizing flows have led to significant advances in modeling high-dimensional continuous distributions, their applicability to discrete distributions remains unknown. In this paper, we show that flows can in fact be extended to discrete events—and under a simple change-of-variables formula not requiring log-determinant-Jacobian computations. Discrete flows have numerous applications. We consider two flow architectures: discrete autoregressive flows that enable bidirectionality, allowing, for example, tokens in text to depend on both left-to-right and right-to-left contexts in an exact language model; and discrete bipartite flows that enable efficient non-autoregressive generation as in RealNVP. Empirically, we find that discrete autoregressive flows outperform autoregressive baselines on synthetic discrete distributions, an addition task, and Potts models; and bipartite flows can obtain competitive performance with autoregressive baselines on character-level language modeling for Penn Tree Bank and text8.

Tran et al. 2019

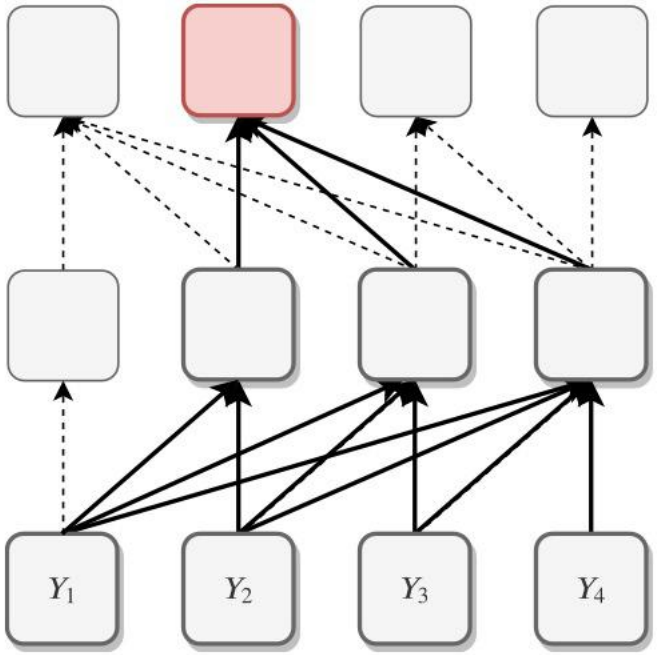
Discrete Flows

1. **Discrete autoregressive flows** enable multiple levels of autoregressivity. For example, one can design a bidirectional language model of text where each token depends on both left- to-right and right-to-left contexts while maintaining an exact likelihood and sampling.

2. **Discrete bipartite flows** enable flexible models with parallel generation by using coupling layers similar to RealNVP. For example, one can design nonautoregressive text models which maintain an exact likelihood for training and evaluation.

Discrete Flows

Autoregressive flows

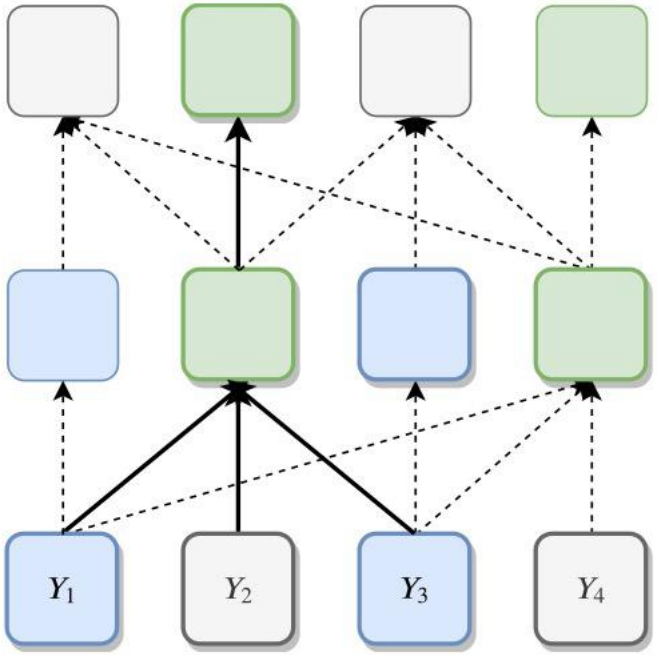


$$\mathbf{y}_d = \boldsymbol{\mu}_d + \boldsymbol{\sigma}_d \cdot \mathbf{x}_d$$

$$\boldsymbol{\mu}_d, \boldsymbol{\sigma}_d = f\left(\mathbf{y}_1, \ldots, \mathbf{y}_{d-1}\right)$$

$$\mathbf{x}_d = \boldsymbol{\sigma}_d^{-1}\left(\mathbf{y}_d - \boldsymbol{\mu}_d\right)$$

Bipartite flows



$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{y}_{d+1:D} = \boldsymbol{\mu}_{(d+1):D} + \boldsymbol{\sigma}_{(d+1):D} \cdot \mathbf{x}_{(d+1):D}$$

Discrete Change of Variables

Let $\mathbf{x}$ be a discrete random variable and $\mathbf{y} = f(\mathbf{x})$ where f is some function of $\mathbf{x}$. The induced probability mass function of $\mathbf{y}$ is the sum over the pre-image of f:

$$p(\mathbf{y} = y) = \sum_{x \in f^{-1}(y)} p(\mathbf{x} = x),$$

where $f^{-1}(y)$ is the set of all elements such that $f(x) = y$. For an invertible function f, this simplifies to

$$p(\mathbf{y} = y) = p(\mathbf{x} = f^{-1}(y)). \tag{4}$$

This change of variables formula for discrete variables is similar to the continuous change of variables formula (Equation 1), but without the log-determinant-Jacobian. Intuitively, the log-determinant-Jacobian corrects for changes to the volume of a continuous space; volume does not exist for discrete distributions so there is no need to account for it. Computationally, Equation 4 is appealing as there are no restrictions on f such as fast Jacobian computations in the continuous case, or tradeoffs in how the log-determinant-Jacobian influences the output density compared to the base density.

Discrete Flows

- **Motivation:** Given a D-dimensional binary vector x, one natural function applies the XOR bitwise operator,

$$\mathbf{y}_d = \boldsymbol{\mu}_d \oplus \mathbf{x}_d, \qquad \text{for } d \text{ in } 1, \ldots, D,$$

  where where μ_d is a function of previous outputs, $y_1, \ldots, y_{d-1}$; $\oplus$ is the XOR function (0 if μ_d and x_d are equal and 1 otherwise).

- The inverse is $\mathbf{x}_d = \boldsymbol{\mu}_d \oplus \mathbf{y}_d$.

Discrete Flows

- Use location-scale transformation on the modulo integer space:

$$\mathbf{y}_d = (\boldsymbol{\mu}_d + \boldsymbol{\sigma}_d \cdot \mathbf{x}_d) \bmod K$$

where $\boldsymbol{\mu}_d$ and $\boldsymbol{\sigma}_d$ are autoregressive or bipartite functions of y in 0,1, …, K-1 and 1, …, K-1

- For the flow to be invertible, $\boldsymbol{\sigma}_d$ and K must be coprime (inverse uses Euclid's algorithm). For example: mask noninvertible values for a given K; or make K prime.

Training Discrete Flows

- The maximum likelihood objective per datapoint is

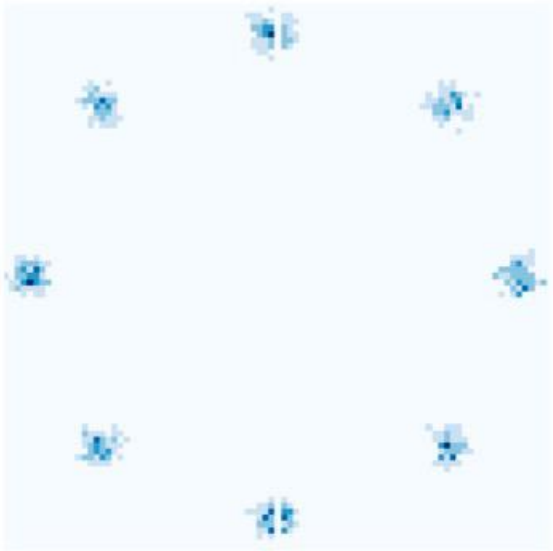
$$\log p(\mathbf{y}) = \log p(f^{-1}(\mathbf{y})),$$

$$\boldsymbol{\mu}_d = \text{one\_hot}(\text{argmax}(\theta_d))$$

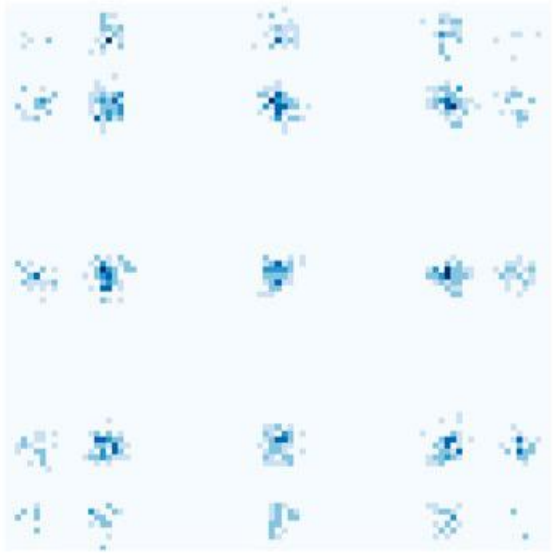
$$\frac{d\boldsymbol{\mu}_d}{d\theta_d} \approx \frac{d}{d\theta_d}\text{softmax}\left(\frac{\theta_d}{\tau}\right)$$

- Gradient descent with respect to base distribution parameters is straightforward.
- Gradient descent with respect to flow parameters requires backprop through the discrete-output function, computed by the straight-through gradient estimator

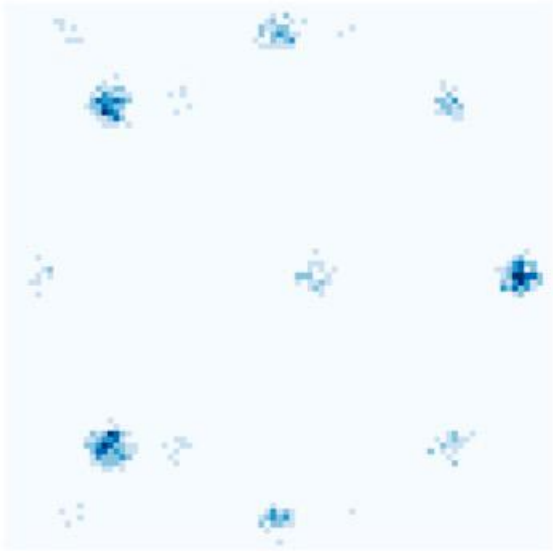
Learning a discretized mixture of Gaussians



(a) Data

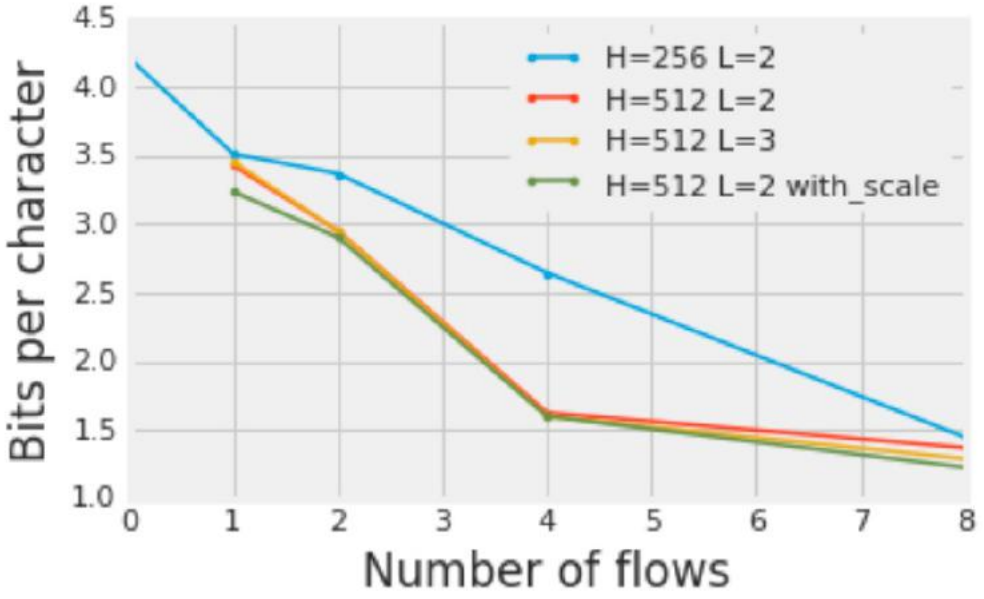


(b) Factorized Base



(c) 1 Flow

Character-Level Language Modeling



	bpc	Gen.
LSTM (Cooijmans+2016)	1.43	19.8s
64-layer Transformer (Al-Rfou+2018)	**1.13**	35.5s
Bipartite flow (4 flows, w/ σ)	1.60	**0.15s**
Bipartite flow (8 flows, w/o σ)	1.29	**0.16s**
Bipartite flow (8 flows, w/ σ)	1.23	**0.16s**

Figure 3: Character-level language modeling results on text8. The test bits per character decreases as the number of flows increases. More hidden units H and layers L in the Transformer per flow, and applying a scale transformation instead of only location, also improves performance.

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, **Discrete Integer Flows**
- GANs for Text: SeqGAN, MaskGAN, ScratchGAN

Integer Discrete Flows and Lossless Compression

Emiel Hoogeboom*
UvA-Bosch Delta Lab
University of Amsterdam
Netherlands
e.hoogeboom@uva.nl

Jorn W.T. Peters*
UvA-Bosch Delta Lab
University of Amsterdam
Netherlands
j.w.t.peters@uva.nl

Rianne van den Berg†
University of Amsterdam
Netherlands
riannevdberg@gmail.com

Max Welling
UvA-Bosch Delta Lab
University of Amsterdam
Netherlands
m.welling@uva.nl

Abstract

Lossless compression methods shorten the expected representation size of data without loss of information, using a statistical model. Flow-based models are attractive in this setting because they admit exact likelihood optimization, which is equivalent to minimizing the expected number of bits per message. However, conventional flows assume continuous data, which may lead to reconstruction errors when quantized for compression. For that reason, we introduce a flow-based generative model for ordinal discrete data called Integer Discrete Flow (IDF): a bijective integer map that can learn rich transformations on high-dimensional data. As building blocks for IDFs, we introduce a flexible transformation layer called integer discrete coupling. Our experiments show that IDFs are competitive with other flow-based generative models. Furthermore, we demonstrate that IDF based compression achieves state-of-the-art lossless compression rates on CIFAR10, ImageNet32, and ImageNet64. To the best of our knowledge, this is the first lossless compression method that uses invertible neural networks.

Hoogeboom et al. 2019

Integer Discrete Coupling

- Let $[\mathbf{x}_a, \mathbf{x}_b] = \mathbf{x} \in \mathbb{Z}^d$ be an input of the layer.
- The output $\mathbf{z} = [\mathbf{z}_a, \mathbf{z}_b]$ is defined as a copy $\mathbf{z}_a = \mathbf{x}_a$, and a transformation $\mathbf{z}_b = \mathbf{x}_b + \lfloor \mathbf{t}(\mathbf{x}_a) \rceil$, where $\lfloor \cdot \rceil$ denotes a nearest rounding operation and $\mathbf{t}$ is a neural network
- Backpropagation through the rounding operation is done using the Straight Through Estimator.

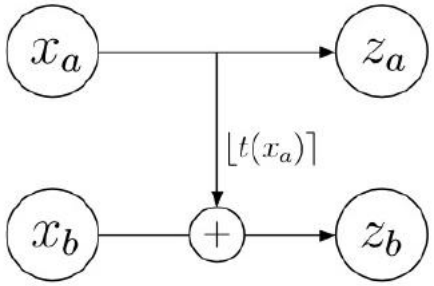


Figure 2: Forward computation of an integer discrete coupling layer. The input is split in two parts. The output consists of a copy of the first part, and a conditional transformation of the second part. The inverse of the coupling layer is computed by inverting the conditional transformation.

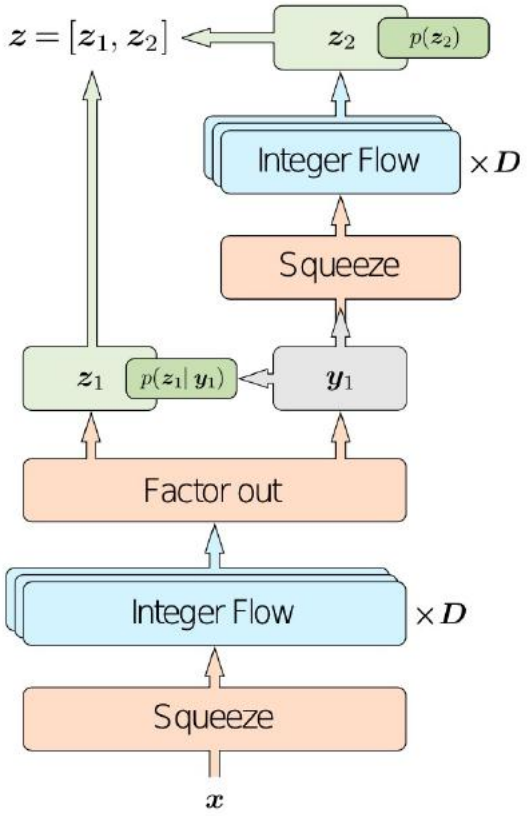
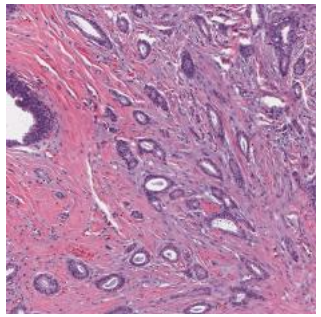
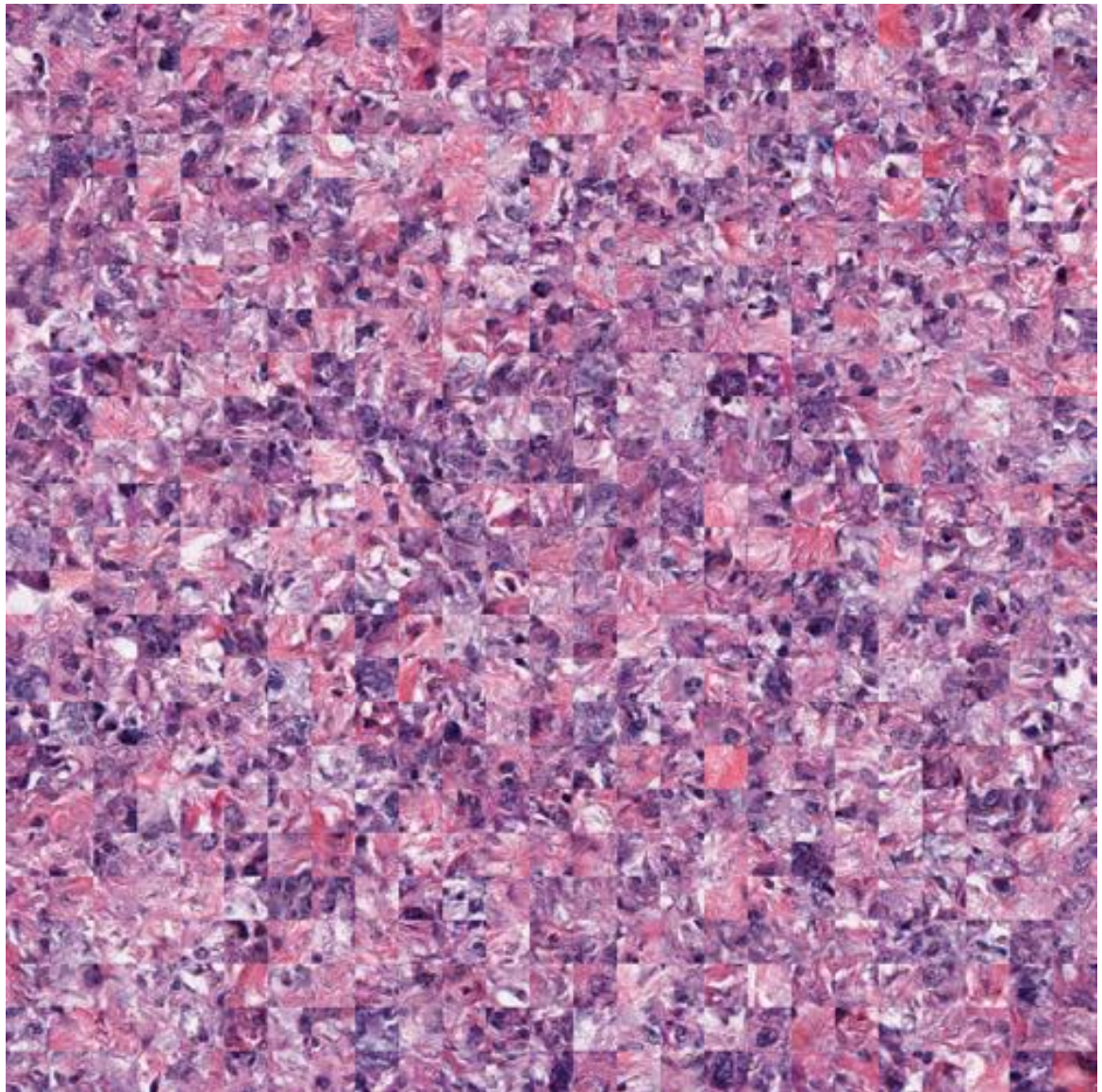


Figure 4: Example of a 2-level flow architecture. The squeeze layer reduces the spatial dimensions by two, and increases the number of channels by four. A single integer flow layer consists of a channel permutation and an integer discrete coupling layer. Each level consists of D flow layers.

IDF ER + BCa and ImageNet Samples



An example from the ER + BCa histology dataset



625 IDF samples of size 80×80px

ImageNet 64×64 IDF Samples

IDF Quantitative Results

Table 1: Compression performance of IDFs on CIFAR10, ImageNet32 and ImageNet64 in bits per dimension, and compression rate (shown in parentheses). The Bit-Swap results are retrieved from [23]. The column marked IDF† denotes an IDF trained on ImageNet32 and evaluated on the other datasets.

Dataset	IDF	IDF†	Bit-Swap	FLIF [35]	PNG	JPEG2000
CIFAR10	**3.34 (2.40×)**	3.60 (2.22×)	3.82 (2.09×)	4.37 (1.83×)	5.89 (1.36×)	5.20 (1.54×)
ImageNet32	**4.18 (1.91×)**	**4.18 (1.91×)**	4.50 (1.78×)	5.09 (1.57×)	6.42 (1.25×)	6.48 (1.23×)
ImageNet64	**3.90 (2.05×)**	3.94 (2.03 ×)	–	4.55 (1.76×)	5.74 (1.39×)	5.10 (1.56×)

IDF Compression Results

Table 2: Compression performance on the ER + BCa histology dataset in bits per dimension and compression rate. JP2-WSI is a specialized format optimized for virtual microscopy.

Dataset	IDF	JP2-WSI	FLIF [35]	JPEG2000
Histology	**2.42 (3.19×)**	3.04 (2.63×)	4.00 (2.00×)	4.26 (1.88×)

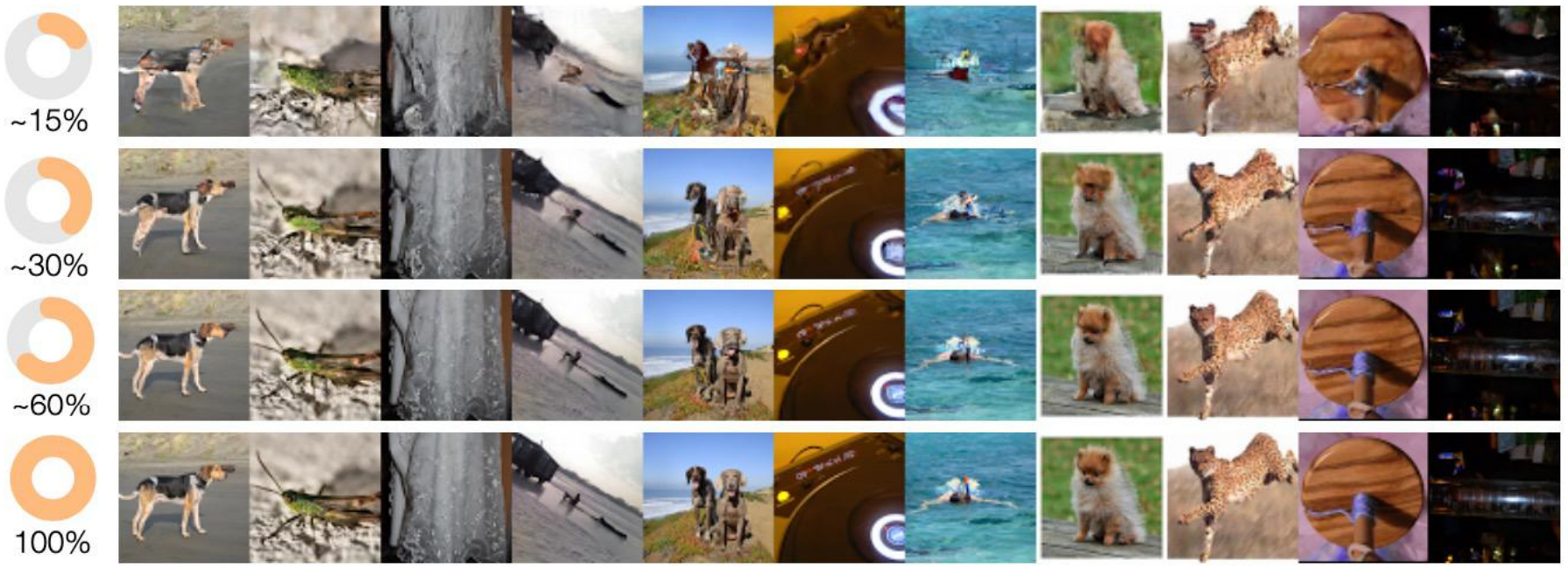


Figure 8: Progressive display of the data stream for images taken from the test set of ImageNet64. From top to bottom row, each image uses approximately 15%, 30%, 60% and 100% of the stream, where the remaining dimensions are sampled. Best viewed electronically.

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- **GANs for Text**: SeqGAN, MaskGAN, ScratchGAN

Why are text GANs hard?

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\mathbf{x})} \left[\log \mathcal{D}_{\boldsymbol{\phi}}(\mathbf{x})\right] + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \left[\log(1 - \mathcal{D}_{\boldsymbol{\phi}}(\mathbf{x})\right]$$

- Gradient estimation
  - high variance (REINFORCE)
  - biased gradients (Gumbel Softmax/Concrete)
- Discriminator/reward structure
- Easy task for the discriminator early in training

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- **GANs for Text: SeqGAN**, MaskGAN, ScratchGAN

SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

Lantao Yu[†], **Weinan Zhang**[†*], **Jun Wang**[‡], **Yong Yu**[†]
[†]Shanghai Jiao Tong University, [‡]University College London
{yulantao,wnzhang,yyu}@apex.sjtu.edu.cn, j.wang@cs.ucl.ac.uk

Abstract

As a new way of training generative models, Generative Adversarial Net (GAN) that uses a discriminative model to guide the training of the generative model has enjoyed considerable success in generating real-valued data. However, it has limitations when the goal is for generating sequences of discrete tokens. A major reason lies in that the discrete outputs from the generative model make it difficult to pass the gradient update from the discriminative model to the generative model. Also, the discriminative model can only assess a complete sequence, while for a partially generated sequence, it is nontrivial to balance its current score and the future one once the entire sequence has been generated. In this paper, we propose a sequence generation framework, called SeqGAN, to solve the problems. Modeling the data generator as a stochastic policy in reinforcement learning (RL), SeqGAN bypasses the generator differentiation problem by directly performing gradient policy update. The RL reward signal comes from the GAN discriminator judged on a complete sequence, and is passed back to the intermediate state-action steps using Monte Carlo search. Extensive experiments on synthetic data and real-world tasks demonstrate significant improvements over strong baselines.

Yu et al. 2017

SeqGAN

Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

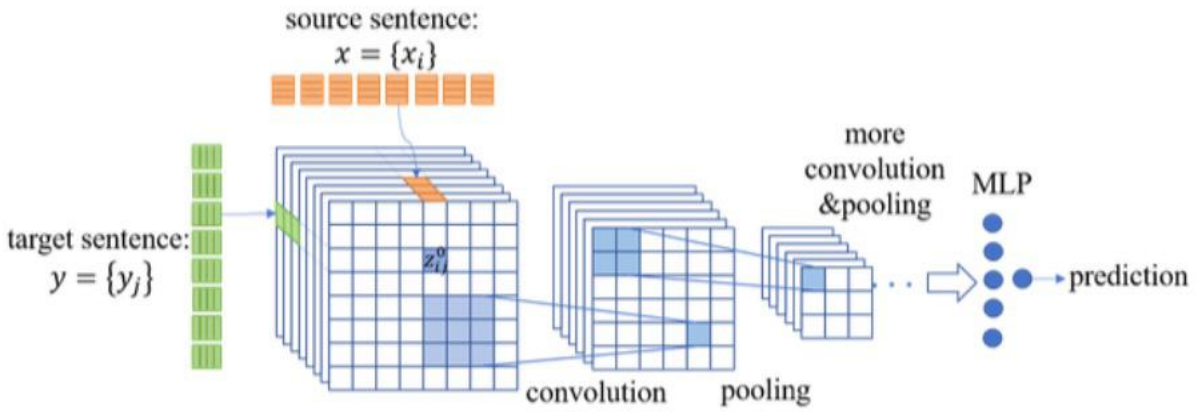
Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
  for g-steps **do**
    Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
    for t in $1 : T$ **do**
      Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
    end for
    Update generator parameters via policy gradient Eq. (8)
  end for
  for d-steps **do**
    Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
    Train discriminator D_ϕ for k epochs by Eq. (5)
  end for
  $\beta \leftarrow \theta$
until SeqGAN converges

- At the beginning of the training, it uses the maximum likelihood estimation (MLE) to pre-train G_θ on training set
- CNN as the discriminator
- The objective of the generator G_θ is to generate a sequence from the start state s_0 to maximize its expected end reward:

$$J(\theta) = \mathbb{E}\left[R_T \mid s_0, \theta\right] = \sum_{y_1 \in \mathcal{Y}} G_\theta\left(y_1 \mid s_0\right) \cdot Q_{D_\phi}^{G_\theta}\left(s_0, y_1\right)$$

Discriminators for Sequences

- Decide whether a particular generated output is true or not
- Commonly use CNNs as discriminators, either on sentences or pairs of sentences



Stabilization Tricks: Assigning Reward to Specific Actions

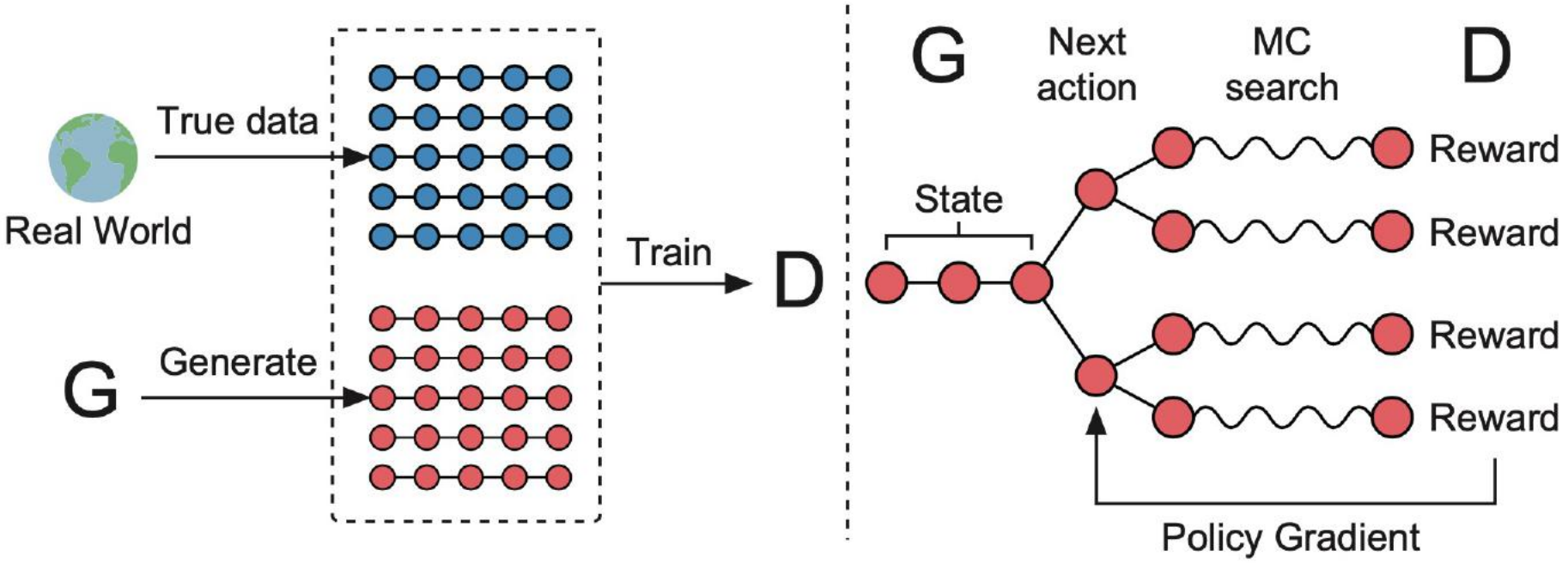
- Getting a reward at the end of the sentence gives a credit assignment problem

- Solution: assign reward for partial sequences

D(this)
D(this is)
D(this is a)
D(this is a fake)
D(this is a fake sentence)

Stabilization Tricks: Performing Multiple Rollouts

- Like other methods using discrete samples, instability is a problem
- This can be helped somewhat by doing multiple rollouts



Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- **GANs for Text:** SeqGAN, **MaskGAN,** ScratchGAN

MASKGAN: BETTER TEXT GENERATION VIA FILLING IN THE _____

William Fedus, Ian Goodfellow and Andrew M. Dai
Google Brain
liam.fedus@gmail.com, {goodfellow, adai}@google.com

ABSTRACT

Neural text generation models are often autoregressive language models or seq2seq models. These models generate text by sampling words sequentially, with each word conditioned on the previous word, and are state-of-the-art for several machine translation and summarization benchmarks. These benchmarks are often defined by validation perplexity even though this is not a direct measure of the quality of the generated text. Additionally, these models are typically trained via maximum likelihood and teacher forcing. These methods are well-suited to optimizing perplexity but can result in poor sample quality since generating text requires conditioning on sequences of words that may have never been observed at training time. We propose to improve sample quality using Generative Adversarial Networks (GANs), which explicitly train the generator to produce high quality samples and have shown a lot of success in image generation. GANs were originally designed to output differentiable values, so discrete language generation is challenging for them. We claim that validation perplexity alone is not indicative of the quality of text generated by a model. We introduce an actor-critic conditional GAN that fills in missing text conditioned on the surrounding context. We show qualitatively and quantitatively, evidence that this produces more realistic conditional and unconditional text samples compared to a maximum likelihood trained model.

[Fedus et al. 2018]

MaskGAN Generator

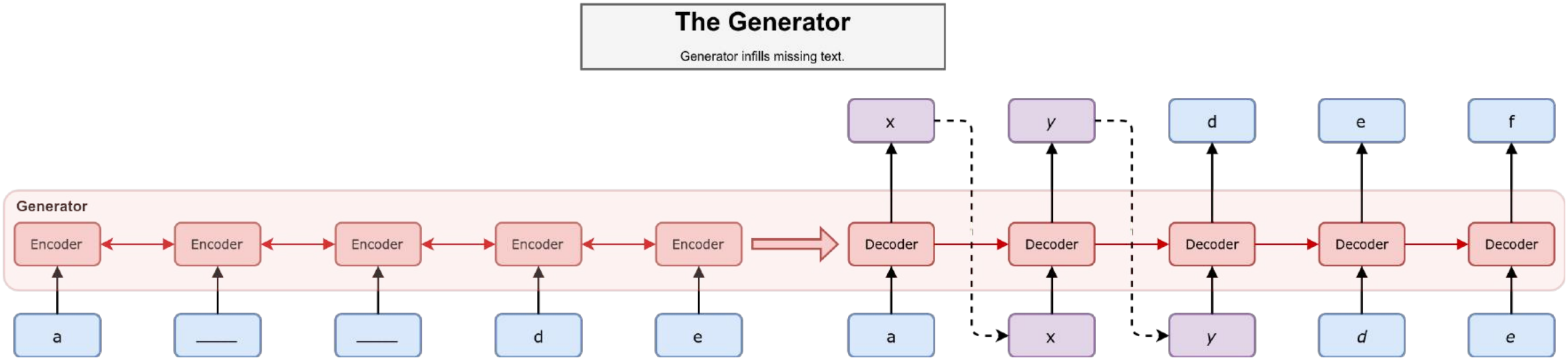


Figure 1: seq2seq generator architecture. Blue boxes represent known tokens and purple boxes are imputed tokens. We demonstrate a sampling operation via the dotted line. The encoder reads in a masked sequence, where masked tokens are denoted by an underscore, and then the decoder imputes the missing tokens by using the encoder hidden states. In this example, the generator should fill in the alphabetical ordering, (a,b,c,d,e).

$$P\left(\hat{x}_1, \cdots, \hat{x}_T \mid \boldsymbol{m}(\boldsymbol{x})\right) = \prod_{t=1}^{T} P\left(\hat{x}_t \mid \hat{x}_1, \cdots, \hat{x}_{t-1}, \boldsymbol{m}(\boldsymbol{x})\right)$$

$$G\left(x_t\right) \equiv P\left(\hat{x}_t \mid \hat{x}_1, \cdots, \hat{x}_{t-1}, \boldsymbol{m}(\boldsymbol{x})\right)$$

MaskGAN Discriminator

- The discriminator is also a Seq2Seq network and receives the filled in sequence and $\boldsymbol{m}(\boldsymbol{x})$

  G: `the director director guided the series`
  Possible Label 1: `The *associate* director guided the series`
  Possible Label 2: `The director *expertly* guided the series`

  Without the context of which words are real, the discriminator was found to assign equal probability to both words.

- To prevent this, the discriminator D_φ computes the probability of each token $\tilde{x}_t$ being real given the true context of the masked sequence $\boldsymbol{m}(\boldsymbol{x})$

$$D_\phi\left(\tilde{x}_t \mid \tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x})\right) = P\left(\tilde{x}_t = x_t^{\text{real}} \mid \tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x})\right)$$

MaskGAN Training using RL

- The logarithm of the discriminator estimates are regarded as the reward

$$r_t \equiv \log D_\phi \left(\tilde{x}_t \mid \tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x}) \right)$$

- The generator maximizes expected reward $\mathbb{E}_G \left[R \right]$ using the REINFORCE gradient estimator

- Two pretraining stages:
  - Train a language model using standard maximum likelihood objective.
  - Use the pretrained language model weights, pretrain the seq2seq model on the in-filling task using maximum likelihood.

MaskGAN Conditional Samples

Ground Truth	**the next day 's show <eos> interactive telephone technology has taken a new leap in <unk> and television programmers are**
MaskGAN	the next day 's show <eos> interactive telephone technology has taken a new leap in its retail business <eos> a
MaskMLE	the next day 's show <eos> interactive telephone technology has taken a new leap in the complicate case of the

Table 1: Conditional samples from PTB for both MaskGAN and MaskMLE models.

Ground Truth	**Pitch Black was a complete shock to me when I first saw it back in 2000 In the previous years I**
MaskGAN	Pitch Black was a complete shock to me when I first saw it back in 1979 I was really looking forward
MaskMLE	Black was a complete shock to me when I first saw it back in 1969 I live in New Zealand

Table 3: Conditional samples from IMDB for both MaskGAN and MaskMLE models.

MaskGAN Unconditional Samples

MaskGAN	oct. N as the end of the year the resignations were approved <eos> the march N N <unk> was down

Table 2: Language model (unconditional) sample from PTB for MaskGAN.

MaskGAN	**Positive**: Follow the Good Earth movie linked Vacation is a comedy that credited against the modern day era yarns which has helpful something to the modern day s best It is an interesting drama based on a story of the famed

Table 4: Language model (unconditional) sample from IMDB for MaskGAN.

MaskGAN Failure Cases

- Mode Collapse

  It is a very funny film that is very funny It s a very funny movie and it s charming It

- Matching Syntax at Boundaries

  Cartoon is one of those films me when I first saw it back in 2000

- Loss of Global Context

  This movie is terrible The plot is ludicrous The title is not more interesting and original This is a great movie
  Lord of the Rings was a great movie John Travolta is brilliant

Lecture overview

- Motivation
- REINFORCE, Gumbel-Softmax, Straight-through estimator
- Neural Variational Inference and Learning (NVIL)
- Vector Quantization VAE (VQ-VAE), VQ-VAE-2, VQGAN
- Discrete Flows, Discrete Integer Flows
- **GANs for Text:** SeqGAN, MaskGAN, **ScratchGAN**

Training Language GANs from Scratch

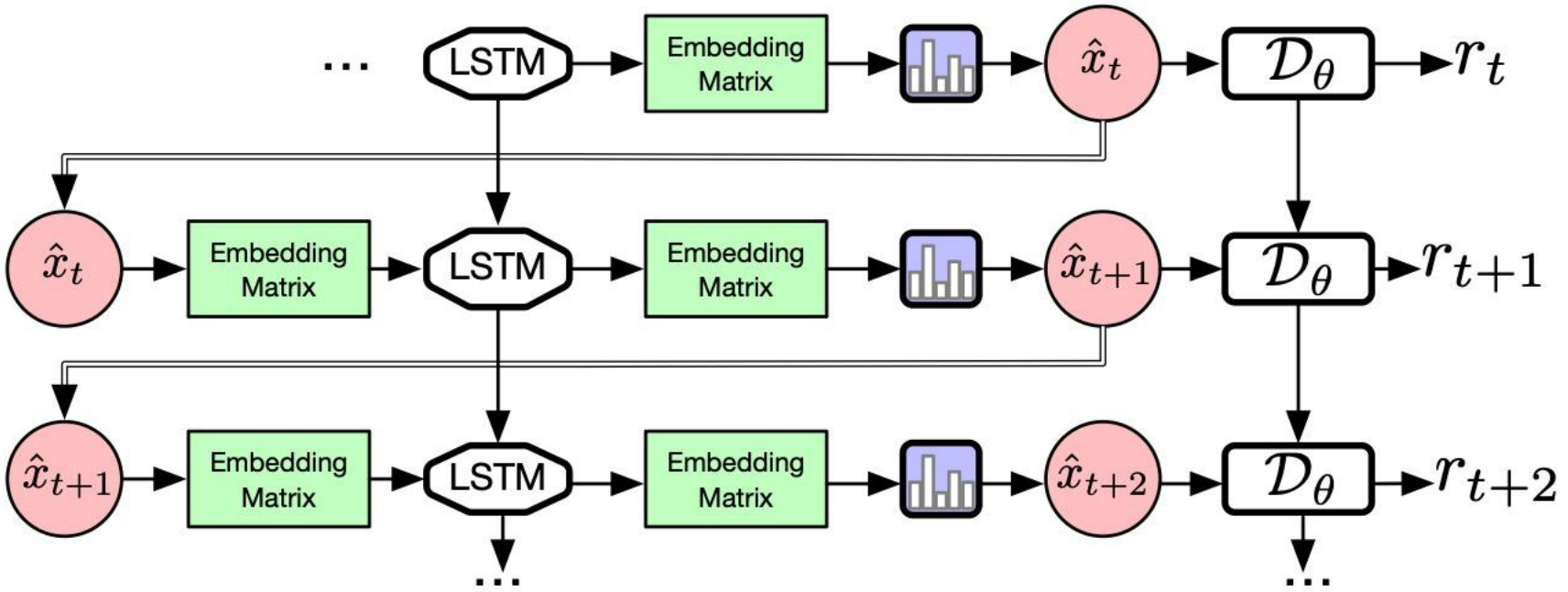
Cyprien de Masson d'Autume* **Mihaela Rosca*** **Jack Rae** **Shakir Mohamed**
DeepMind
{cyprien,mihaelacr,jwrae,shakir}@google.com

Abstract

Generative Adversarial Networks (GANs) enjoy great success at image generation, but have proven difficult to train in the domain of natural language. Challenges with gradient estimation, optimization instability, and mode collapse have lead practitioners to resort to maximum likelihood pre-training, followed by small amounts of adversarial fine-tuning. The benefits of GAN fine-tuning for language generation are unclear, as the resulting models produce comparable or worse samples than traditional language models. We show it is in fact possible to train a language GAN from scratch — without maximum likelihood pre-training. We combine existing techniques such as large batch sizes, dense rewards and discriminator regularization to stabilize and improve language GANs. The resulting model, ScratchGAN, performs comparably to maximum likelihood training on EMNLP2017 News and WikiText-103 corpora according to quality and diversity metrics.

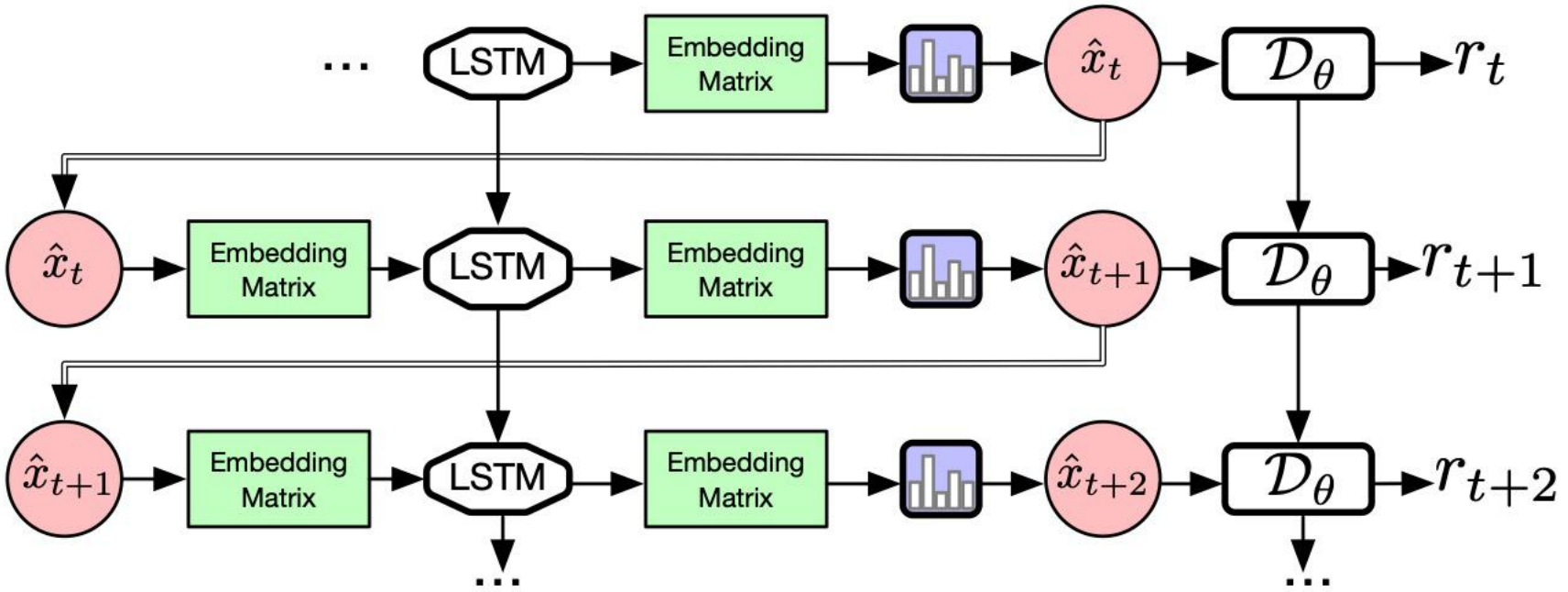
de Masson d'Autume et al. 2019

ScratchGAN



- Gradient variance
  - large batch sizes for REINFORCE
  - moving average baselines
- Dense rewards
- Discriminator regularization
  - layer normalization
  - Dropout
  - l2 norm

ScratchGAN



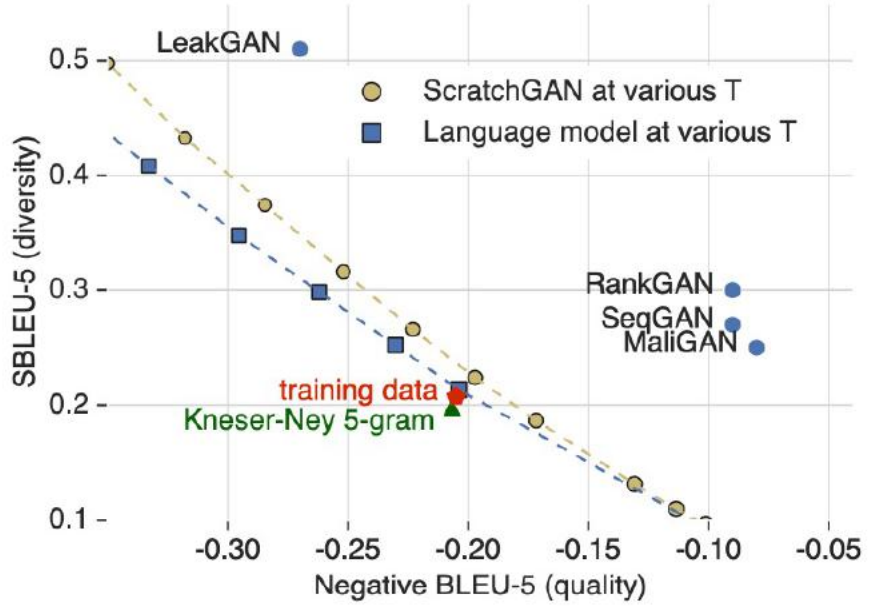
$$r_t = 2\mathcal{D}_\phi\left(\hat{x}_t \mid x_{t-1} \dots x_1\right) - 1$$

- The goal of the generator at timestep t is to maximize the sum of discounted future rewards using a discount factor γ: $R_t = \sum_{s=t}^{T} \gamma^{s-t} r_s$

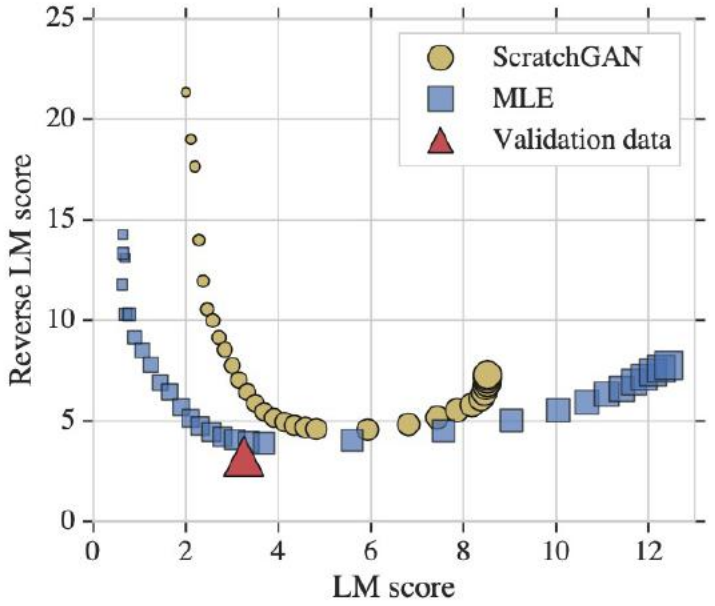
ScratchGAN

- Using a Wasserstein Loss on generator logits, with a straight-through gradient.
- Using ensembles of discriminators and generators.
- Training against past versions of generators/discriminators.
- Using a hand-designed curriculum.
- Other losses (Hinge loss) for the discriminator.
- Small datasets like Penn Tree Bank - overfitting.
- More in the paper.

ScratchGAN: Quantitative Results



(a) Negative BLEU-5 versus Self-BLEU-5.



(b) Language- and reverse language-model scores.

Figure 2: BLEU scores on EMNLP2017 News (left) and language model scores on Wikitext-103 (right). For BLEU scores, left is better and down is better. LeakGAN, MaliGAN, RankGAN and SeqGAN results from Caccia et al. [20].

ScratchGAN: Quantitative Results

Model	World level perplexity
Random	5725
ScratchGAN	154
MLE	**42**

Table 2: EMNLP2017 News perplexity.

Next lecture:
Strengths and Weaknesses of Current Models