Variational Autoencoders

Advanced Topics in Machine Learning

Andrea Cini, Cesare Alippi

September 2021

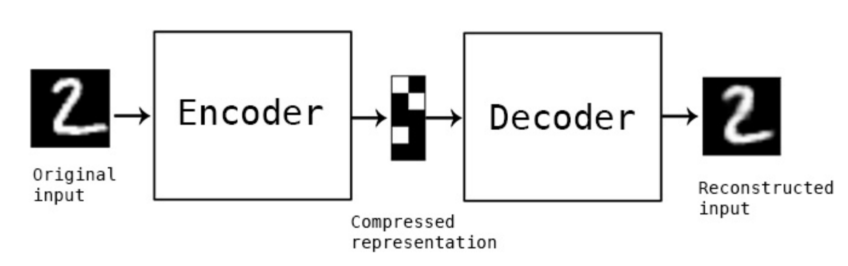
Università della Svizzera italiana

Table of contents

Preliminaries

Autoencoders (I)

General idea behind autoencoders:



- We want to learn a low-dimensional representation of our data.
- We want to be able go back and forth from the encoding space and the original space.

Autoencoders (II)

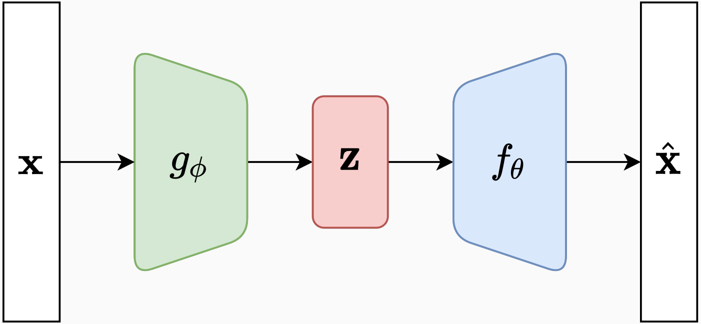
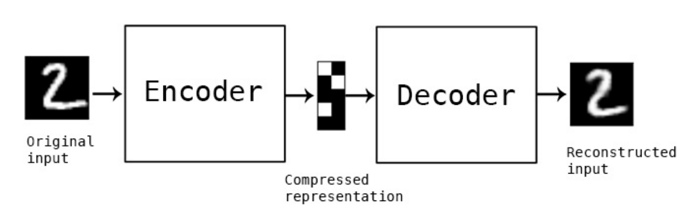


Figure 1: A vanilla Autoencoder (AE)

- Use a bottleneck Encoder-Decoder architecture to force a compact representation.
- Train a neural network to learn the identity function, i.e., to reconstruct the input.

Autoencoders (III)



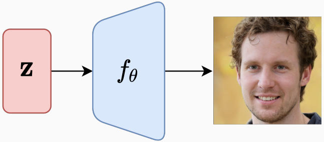
What are Autoencoders good for?

- Dimensionality reduction
- Representation learning
- Removing noise from the input data
- Generating new samples, but...

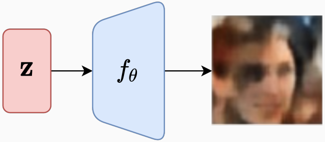
Autoencoders (IV)

We can try to generate new samples sampling at random the encoding space:

What we want



What we get



- Why?

Discriminative vs Generative models (I)

Standard AEs are discriminative models.

- Discriminative models learn to make predictions, i.e., learn a mapping from input to an output.
- AEs are trained to reconstruct the input, not to generate new data.
- It is hard to say something about the structure of the encoding space of a standard AE ...
- ... it is not clear how to sample it to generate new datapoints: a point in the latent space does not necessary correspond to a plausible observation.

Discriminative vs Generative models (II)

Variational Autoencoders (VAEs)[1] are generative models.

- Generative approaches model the stochastic process generating the data and predictions are done using Bayes rule.
- Forcing the representations to be useful for generation often results in *better generalization*.
- VAEs learn to (1) *infer* the latent representation behind each observation and (2) *generate* realistic synthetic observations from points in the latent space.

In order to understand VAEs we need to introduce the concept of Latent Variable Models.

[1] Kingma and Welling 2013; Kingma and Welling 2019.

Latent Variable Models (I)

- Given a dataset $\mathcal{D}$, consider each sample $\boldsymbol{x}$ as a realization of a random vector, i.e., $\boldsymbol{x} \sim p^*(\boldsymbol{x})$.
- $p^*(\boldsymbol{x})$ is the unknown probability distribution underlying the process generating the data.
- We want to learn an approximating distribution, with a model parametrized by $\boldsymbol{\theta}$, such that:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx p^*(\boldsymbol{x});$$

  $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is known as model evidence. The notation $p_{\boldsymbol{\theta}}(\,\cdot\,)$ indicates that probability distribution $p_{\boldsymbol{\theta}}(\,\cdot\,)$ is a function of model parameters $\boldsymbol{\theta}$.

Latent Variable Models (II)

A *Latent Variable Model (LVM)* uses an approximating distribution of the form:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\, dz \tag{1}$$

$$= \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z})\, dz, \qquad \textit{(product rule)} \tag{2}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is the likelihood of observation $\boldsymbol{x}$ given the latent variables $\boldsymbol{z}$ with prior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$. In the following we refer to the case where $\boldsymbol{z}$ is a finite dimensional random vector and the likelihood is a known distribution, e.g., Gaussian.

Latent Variable Models (III)

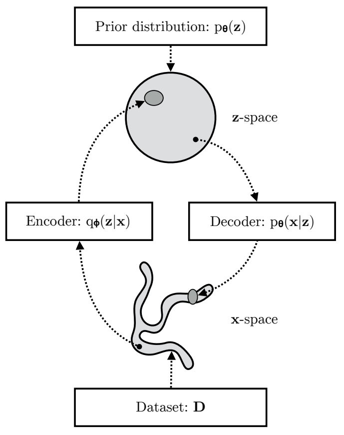
At this point you might have an intuition about what is going on...

- Sampling from $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ we could generate new observations.
- Symmetrically, modeling $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, known as the posterior distribution, we could make inference about the latent variables given an observation...
- ... unfortunately computing the posterior is intractable for most models[2]!
- Solution: we use an approximate posterior $q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) \approx p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ *(we'll come back on this later)*

[2] To see why that is the case note that $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}$ and that computing $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ involves computing the integral in Eq. 1. See (Blei, Kucukelbir, and McAuliffe 2017) for a concrete example.

Variational Autoencoders

Towards VAEs: the big picture



- The input space is a generic manifold.
- For the prior we usually use a Gaussian $\boldsymbol{z} \sim \mathcal{N}(0, I)$
- Similarly for the approximate posterior: $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}; \mu_\phi(\boldsymbol{x}), I\sigma^2_\phi(\boldsymbol{x})\right)$
- The output of the probabilistic decoder depends on the original space (e.g., Bernoulli for binary outputs)

[3] Kingma and Welling 2019.

A possible implementation

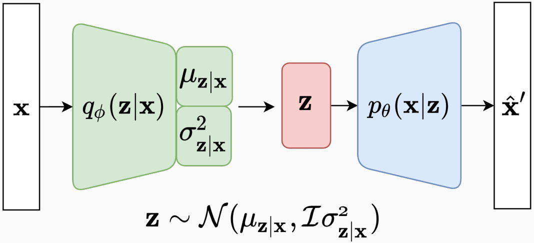


Figure 2: A Variational Autoencoder

Let's take a step back and consider the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

Variational Inference (I)

Variational Inference[4] tackles the problem of approximating an intractable posterior from an optimization perspective.

$$q_{\phi^*}(\boldsymbol{z}|\boldsymbol{x}) = \text{argmin}_{\phi} \mathcal{D}_{KL}(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

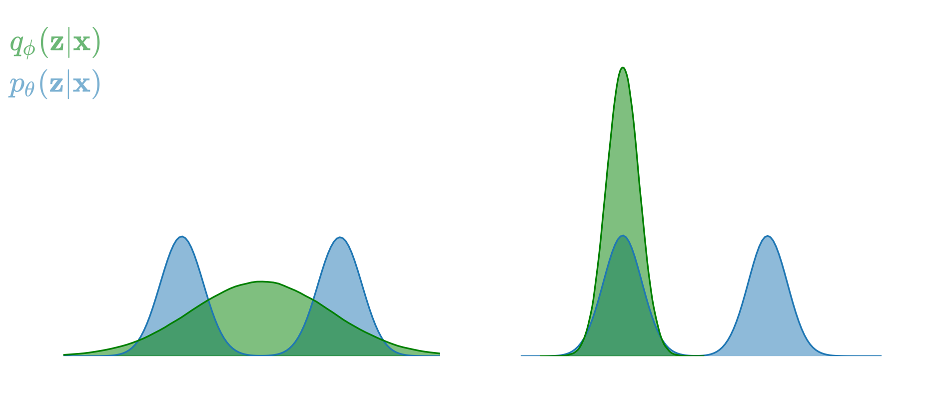
The KL-divergence $\mathcal{D}_{KL}(\cdot \parallel \cdot)$ is intuitively a measure of *dissimilarity* between probability distributions:

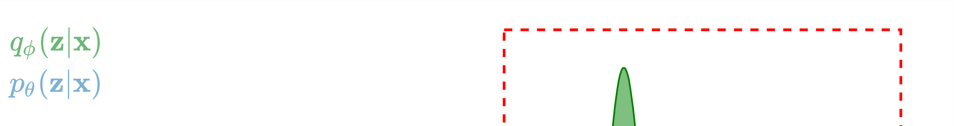
$$\mathcal{D}_{KL}(p(x) \parallel g(x)) = \int p(x) \log \frac{p(x)}{g(x)} \, dx$$

[4]Blei, Kucukelbir, and McAuliffe 2017.

Remark

Which approximation would you prefer? In which case $\mathcal{D}_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x}))$ is lower?





Variational Inference (II)

What optimization objective should we use?

$$\begin{aligned}
&\mathcal{D}_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})) \\
&= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\, d\boldsymbol{z} && \textit{(def.)} \\
&= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x}) p_\theta(\boldsymbol{x})}{p_\theta(\boldsymbol{x}, \boldsymbol{z})}\, d\boldsymbol{z} && \textit{(product rule)} \\
&= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \left(\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{x}, \boldsymbol{z})} + \log p_\theta(\boldsymbol{x})\right) d\boldsymbol{z} \\
&= \log p_\theta(\boldsymbol{x}) + \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{x}, \boldsymbol{z})}\, d\boldsymbol{z} && \left(\textstyle\int q_\phi(\boldsymbol{z}|\boldsymbol{x})\, d\boldsymbol{z} = 1\right) \\
&= \log p_\theta(\boldsymbol{x}) + \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \left(\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z})} - \log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right) d\boldsymbol{z} && \textit{(product rule)} \\
&= \log p_\theta(\boldsymbol{x}) + \mathcal{D}_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right]
\end{aligned}$$

Variational Inference (III)

Rearranging the terms we get:

$$\begin{aligned}\underbrace{\text{ELBO}}_{\text{Evidence Lower BOund}} &= \underbrace{\log p_{\boldsymbol{\theta}}(x)}_{\text{evidence}} - \underbrace{\mathcal{D}_{KL}\left(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right)}_{\text{KL-divergence}} \\ &= \mathbb{E}_{\mathbf{z}\sim q_{\phi}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \mathcal{D}_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{z})) \leq \log p_{\boldsymbol{\theta}}(x)\end{aligned}$$

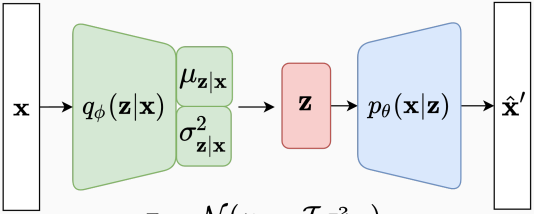
Intuitively, by maximizing the ELBO we maximize the probability of generating plausible samples while pushing the approximated posterior closer to the actual one.

Variational Inference (IV)

$$\text{ELBO} = \underbrace{\mathbb{E}_{\boldsymbol{z} \sim q_\phi} [\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \mathcal{D}_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}))}_{\mathcal{L}(\mathrm{x};\boldsymbol{\theta},\phi)}$$

This expression is pretty easy to evaluate empirically by sampling the available data and the approximated posterior, so we are done... are we?

Reparameterization trick (I)



$$\mathbf{z} \sim \mathcal{N}(\mu_{\mathbf{z}|\mathbf{x}}, \mathcal{I}\sigma^2_{\mathbf{z}|\mathbf{x}})$$

Problem: Sampling from a distribution is not a differentiable operation!

Solution: The reparameterization trick

Reparameterization trick (II)

In general the problem requires to estimate the gradient of an expectation:

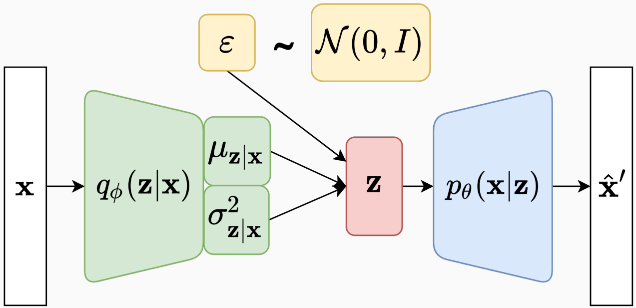
$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{z \sim q_{\boldsymbol{\theta}}} \left[f(z) \right]$$

With the reparameterization trick *we change the sampling distribution* so that it becomes independent from $\boldsymbol{\theta}$:

$$\begin{aligned}
\varepsilon &\sim p(\varepsilon) \\
z &= g(x, \varepsilon) \\
\nabla_{\boldsymbol{\theta}} \mathbb{E}_{z \sim q_{\boldsymbol{\theta}}} \left[f(z) \right] &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\varepsilon \sim p(\varepsilon)} \left[f(g(x, \varepsilon)) \right] \\
&= \mathbb{E}_{\varepsilon \sim p(\varepsilon)} \left[\nabla_{\boldsymbol{\theta}} f(g(x, \varepsilon)) \right]
\end{aligned}$$

Other alternatives exist (e.g., see the REINFORCE estimator from the RL literature).

Reparameterization trick (III)



$$\varepsilon \sim \mathcal{N}(0, I)$$
$$\boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}}, \boldsymbol{\sigma}^2_{\boldsymbol{z}|\boldsymbol{x}} = \text{Encoder}(\boldsymbol{x})$$
$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}} + \varepsilon \odot \boldsymbol{\sigma}_{\boldsymbol{z}|\boldsymbol{x}}$$

Reparameterization trick (IV)

In the Gaussian case, assuming a J-dimensional latent space:

$$-\mathcal{D}_{KL}(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z})) = \int q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) \left(\log p_{\boldsymbol{\theta}}(\boldsymbol{z}) - \log q_{\phi}(\boldsymbol{z}|\boldsymbol{x})\right) \, d\boldsymbol{z}$$
$$= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_{\boldsymbol{z}|\boldsymbol{x}}^{(j)})^2) - (\mu_{\boldsymbol{z}|\boldsymbol{x}}^{(j)})^2 - (\sigma_{\boldsymbol{z}|\boldsymbol{x}}^{(j)})^2\right).$$

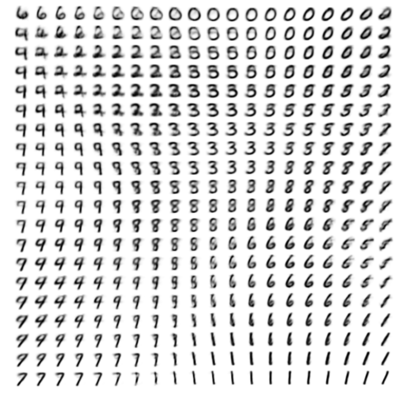
Considering the i-th available sample $\boldsymbol{x}^{(i)}$ and L samples from the approximate posterior:

$$\tilde{\mathcal{L}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}, \phi) =$$
$$\frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_{\boldsymbol{z}|\boldsymbol{x}^{(i)}}^{(j)})^2) - (\mu_{\boldsymbol{z}|\boldsymbol{x}^{(i)}}^{(j)})^2 - (\sigma_{\boldsymbol{z}|\boldsymbol{x}^{(i)}}^{(j)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}),$$

where $\boldsymbol{z}^{(i,l)} = \boldsymbol{\mu}_{\boldsymbol{z}|\boldsymbol{x}^{(i)}} + \boldsymbol{\varepsilon}^{(l)} \odot \boldsymbol{\sigma}_{\boldsymbol{z}|\boldsymbol{x}^{(i)}}$. The final optimization objective is:

$$\tilde{\mathcal{L}}_N(\boldsymbol{\theta}, \phi) = \sum_{i=1}^{N} \tilde{\mathcal{L}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}, \phi)$$

Examples of learned manifolds



[5] Kingma and Welling 2013.

State of the art

[6]Razavi, Oord, and Vinyals 2019.

Demo

Demo

https://cutt.ly/pfS0TKv

Questions?

Likelihood for a Bernoulli decoder

In the case where we model the output of the probabilistic decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ as multivariate Bernoulli the log-likelihood can be compute as:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}) = \underbrace{\sum_{i=1}^{N} \boldsymbol{x}^{(i)} \log \hat{\boldsymbol{x}}^{(i)} + \left(1 - \boldsymbol{x}^{(i)}\right) \log \left(1 - \hat{\boldsymbol{x}}^{(i)}\right)}_{\text{binary cross-entropy}}$$

where $\hat{\boldsymbol{x}}^{(i)} = Decoder(\boldsymbol{z}^{(i,l)})$.

Likelihood for a Gaussian decoder

In the case of a multivariate diagonal Gaussian decoder:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}) = \log \mathcal{N}(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}^{(i,l)}}, \boldsymbol{\sigma}^2_{\boldsymbol{x}|\boldsymbol{z}^{(i,l)}})$$

where $\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}^{(i,l)}}, \boldsymbol{\sigma}_{\boldsymbol{x}|\boldsymbol{z}^{(i,l)}} = Decoder(\boldsymbol{z}^{(i,l)})$.

Note that:

- The output of the Decoder network is the mean and the variance of the Gaussian.
- No need of tricks to estimate the gradient since the expectation is over $\boldsymbol{z}$ and not $\boldsymbol{x}$.

Credits

Most of the material was inspired and adapted from:

- The tutorial on VAEs from the authors of the original paper: `https://arxiv.org/abs/1906.02691`
- This nice blogpost by Lilian Weng: `https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html`

References i

- Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe (Apr. 2017). "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518, pp. 859–877. ISSN: 1537-274X. DOI: 10.1080/01621459.2017.1285773. URL: http://dx.doi.org/10.1080/01621459.2017.1285773.
- Kingma, Diederik P. and Max Welling (2013). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114 [stat.ML].
- – (2019). "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392. ISSN: 1935-8245. DOI: 10.1561/2200000056. URL: http://dx.doi.org/10.1561/2200000056.
- Razavi, Ali, Aaron van den Oord, and Oriol Vinyals (2019). *Generating Diverse High-Fidelity Images with VQ-VAE-2*. arXiv: 1906.00446 [cs.LG].