Variational Autoencoders

Foundations, Variations, Applications, Sensations

Jake Levi

OccaMLab Reading Group

13 May 2026

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions are two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we perform efficient approximate inference and learning with directed probabilistic models whose continuous latent variables and/or parameters have intractable posterior distributions? The variational Bayesian (VB) approach involves the optimization of an approximation to the intractable posterior. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a simple differentiable unbiased estimator of the lower bound; this SGVB (Stochastic Gradient Variational Bayes) estimator can be used for efficient approximate posterior inference in almost any model with continuous latent variables and/or parameters, and is straightforward to optimize using standard stochastic gradient ascent techniques.

For the case of an i.i.d. dataset and continuous latent variables per datapoint, we propose the Auto-Encoding VB (AEVB) algorithm. In the AEVB algorithm we make inference and learning especially efficient by using the SGVB estimator to optimize a recognition model that allows us to perform very efficient approximate posterior inference using simple ancestral sampling, which in turn allows us to efficiently learn the model parameters, without the need of expensive iterative inference schemes (such as MCMC) per datapoint. The learned approximate posterior inference model can also be used for a host of tasks such as recognition, denoising, representation and visualization purposes. When a neural network is used for the recognition model, we arrive at the *variational auto-encoder*.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference on the (global) parameters, and variational inference on the latent variables. It is, for example,

1

Historical context

- VAE [Kingma and Welling, 2013] Arxived on Fri, 20 Dec 2013
- Similar [Rezende et al., 2014] published 27 days later
- 55k vs 7k citations

Historical context

- VAE [Kingma and Welling, 2013] Arxived on Fri, 20 Dec 2013
- Similar [Rezende et al., 2014] published 27 days later
- 55k vs 7k citations
- Lesson 1: don't get scooped

Historical context

- VAE [Kingma and Welling, 2013] Arxived on Fri, 20 Dec 2013
- Similar [Rezende et al., 2014] published 27 days later
- 55k vs 7k citations
- Lesson 1: don't get scooped
- 2013: Adam, ResNets, BatchNorm/LayerNorm, etc don't yet exist
- VAEs relative to 2010s DL research:

Historical context

Contribution	Year	Reference	# Citations
"Xavier" / "Glorot" initialisation	2010	[Glorot and Bengio, 2010]	30k
AlexNet	2012	[Krizhevsky et al., 2012]	155k
Dropout	2012	[Hinton et al., 2012]	13k
VAE	**2013**	**[Kingma and Welling, 2013]**	**55k**
VGGNet	2014	[Simonyan and Zisserman, 2014]	161k
Adam	2014	[Kingma and Ba, 2014]	243k
Neural attention	2014	[Bahdanau et al., 2014]	43k
GANs	2014	[Goodfellow et al., 2014]	94k
BatchNorm	2015	[Ioffe and Szegedy, 2015]	68k
ResNet	2015	[He et al., 2016]	318k
"Kaiming" / "He" initialisation	2015	[He et al., 2015]	30k
Diffusion	2015	[Sohl-Dickstein et al., 2015]	12k
LayerNorm	2016	[Ba et al., 2016]	19k
Transformers	2017	[Vaswani et al., 2017]	245k
BERT	2018	[Devlin et al., 2019]	168k
GPT-1	2018	[Radford et al., 2018]	19k
GPT-2	2019	[Radford et al., 2019]	25k

VAE problem setting

- Observe data $x = \{x_1, \ldots, x_N\}$
- Assume $x \sim p(x \mid \theta^*)$
- ★ Learn $\theta = \text{argmax}_\theta \left[\log \left(p(x \mid \theta) \right) \right]$
- ★ Sample $\{x_{N+1}, \ldots, x_{N+M}\} \sim p(x \mid \theta)$
- Model:
  - $\{x_1, \ldots, x_N\}$ IID
  - Sample $z_n \sim p\left(z_n \mid \theta\right)$
  - Sample $x_n \sim p\left(x_n \mid z_n, \theta\right)$
  - Observe x_n (z_n hidden)

VAE objective

- *How* to learn $\theta = \text{argmax}_\theta \Big[\log\Big(p\big(x \,|\, \theta\big)\Big)\Big]$?

$$\log\Big(p\big(x \,|\, \theta\big)\Big) \qquad \text{(Log likelihood)}$$

$$= \sum_{n=1}^{N} \Big[\log\Big(p\big(x_n \,|\, \theta\big)\Big)\Big] \qquad \text{(IID)}$$

$$= \sum_{n=1}^{N} \Big[\log\Big(\int dz_n \Big[p\big(x_n, z_n \,|\, \theta\big)\Big]\Big)\Big] \qquad \text{(Marginalise } z_n\text{)}$$

$$= \sum_{n=1}^{N} \Big[\log\Big(\int dz_n \Big[p\big(z_n \,|\, \theta\big)\; p\big(x_n \,|\, z_n, \theta\big)\Big]\Big)\Big] \qquad \text{(Product rule } x_n|z_n\text{)}$$

$$= \sum_{n=1}^{N} \Big[\log\Big(\mathbb{E}_{p(z_n \,|\, \theta)}\Big[p\big(x_n \,|\, z_n, \theta\big)\Big]\Big)\Big] \qquad \Big(\textstyle\int p = \mathbb{E}\Big)$$

VAE objective

- *How* to learn $\theta = \text{argmax}_\theta \left[\log\Big(p(x \mid \theta)\Big)\right]$?

$$\log\Big(p\big(x \mid \theta\big)\Big) = \sum_{n=1}^{N} \left[\log\left(\mathbb{E}_{p(z_n \mid \theta)}\left[p\big(x_n \mid z_n, \theta\big)\right]\right)\right] \quad \text{(Previous slide)}$$

- Could:
  - Fix $p(z_n \mid \theta) = \mathcal{N}(z_n \mid 0, I)$ and $p\big(x_n \mid z_n, \theta\big) = \text{MLP}$
  - MC with $z_n \sim p(z_n \mid \theta)$
- But:
  - Not unbiased ($\mathbb{E}$ inside log)
  - p(sampled z_n reconstructs x_n) = tiny $\Rightarrow$ need many samples
- Instead choose* some $q(z_n|x_n, \phi) \approx p(z_n|x_n, \theta)$, maximise *ELBO*

ELBO derivation

- Expand log likelihood for each datapoint:

$$
\begin{aligned}
&\log\Big(p(x_n|\theta)\Big) && \text{(Log likelihood)} \\
&= \int dz_n \left[q(z_n|x_n,\phi)\log\Big(p(x_n|\theta)\Big)\right] && \text{(Marginalise } q\text{)} \\
&= \int dz_n \left[q(z_n|x_n,\phi)\log\left(\frac{p(x_n,z_n|\theta)}{p(z_n|x_n,\theta)}\right)\right] && \text{(Product rule } z_n|x_n\text{)} \\
&= \int dz_n \Bigg[q(z_n|x_n,\phi)\log\left(\frac{p(x_n,z_n|\theta)}{q(z_n|x_n,\phi)}\right) \\
&\qquad + q(z_n|x_n,\phi)\log\left(\frac{q(z_n|x_n,\phi)}{p(z_n|x_n,\theta)}\right)\Bigg] && (\times \tfrac{q}{q})
\end{aligned}
$$

ELBO derivation

$$
\begin{aligned}
&= \int dz_n \left[q(z_n|x_n,\phi) \log \left(\frac{p(x_n,z_n|\theta)}{q(z_n|x_n,\phi)} \right) \right. \\
&\qquad \left. + q(z_n|x_n,\phi) \log \left(\frac{q(z_n|x_n,\phi)}{p(z_n|x_n,\theta)} \right) \right] && \text{(From previous slide)} \\
&= \int dz_n \left[q(z_n|x_n,\phi) \log \left(\frac{p(x_n,z_n|\theta)}{q(z_n|x_n,\phi)} \right) \right] \\
&\qquad + \mathrm{KL}\left[q(z_n|x_n,\phi) \,\middle\|\, p(z_n|x_n,\theta) \right] && \text{(KL definition)} \\
&\geq \int dz_n \left[q(z_n|x_n,\phi) \log \left(\frac{p(x_n,z_n|\theta)}{q(z_n|x_n,\phi)} \right) \right] && \text{(KL} \geq 0\text{, ELBO v1)}
\end{aligned}
$$

ELBO derivation

$$
\begin{aligned}
&\geq \int dz_n \left[q(z_n|x_n,\phi) \log \left(\frac{p(x_n,z_n|\theta)}{q(z_n|x_n,\phi)} \right) \right] && \text{(From previous slide)} \\
&= \int dz_n \left[q(z_n|x_n,\phi) \log \left(\frac{p(z_n|\theta)}{q(z_n|x_n,\phi)} \right) \right. \\
&\qquad \left. + q(z_n|x_n,\phi) \log\Big(p(x_n|z_n,\theta) \Big) \right] && \text{(Product rule* } x_n|z_n) \\
&= -\mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta) \Big] \\
&\qquad + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)} \Big[\log\Big(p(x_n|z_n,\theta) \Big) \Big] && \text{(KL and } \mathbb{E} \text{ definitions)} \\
&= \mathcal{L}(x_n,\theta,\phi) && \text{(ELBO, VAE version)}
\end{aligned}
$$

- NB $\mathcal{L}$ shorthand for "ELBO", not "loss" (we will maximise $\mathcal{L}$)

ELBO interpretation

$$\underbrace{\mathcal{L}(x_n, \theta, \phi)}_{\text{Objective function}} = \underbrace{-\mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big]}_{\text{Regularise } z_n}$$

$$+ \underbrace{\mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big]}_{\text{Reconstruct } x_n} \qquad \text{(ELBO)}$$

$$\leq \log\Big(p(x_n|\theta)\Big) \qquad \text{(Log likelihood)}$$

- Regularise $\Rightarrow$ don't overfit latent for every x_n
- Reconstruct $\Rightarrow$ learn accurate generative model
- Together $\Rightarrow$ new samples $z_n \sim p(z_n|\theta)$ decode to realistic x_n

VAE challenges

$1mQs:

1. How to define $q(z_n|x_n, \phi)$?
2. How to estimate $\frac{\partial \mathcal{L}}{\partial(\theta,\phi)}$ using BP?

VAE challenges

$1mQs:

1. How to define $q(z_n|x_n, \phi)$?
2. How to estimate $\frac{\partial \mathcal{L}}{\partial(\theta,\phi)}$ using BP?

Latent posterior

- How to choose $q(z_n|x_n, \phi)$?
- *Could* learn some $q(z_n|x_n, \phi)$ separately for every x_n
  - N large $\Rightarrow$ learn with batches
  - Store all $q(z_n|x_n, \phi)$ in memory? $\Rightarrow$ Memory $O(N)$ ☹
  - Learn $q(z_n|x_n, \phi)$ from scratch every iteration? Hard+wasteful ☹

Latent posterior

- How to choose $q(z_n|x_n, \phi)$?
- ~~*Could* learn some $q(z_n|x_n, \phi)$ separately for every x_n~~
- "Amortised inference" ☺
- EG:

$$(\mu, \sigma) = \text{MLP}(x_n, \phi) \qquad \text{(MLP forward pass)}$$

$$q(z_n \mid x_n, \phi) = \mathcal{N}\left(z_n \,\middle|\, \mu, \text{diag}(\sigma)^2\right) \qquad \text{(Latent posterior)}$$

- Cost "amortised" into ϕ
- Learn good $\phi \Rightarrow$ have $q(z_n|x_n, \phi)$ for *all* x_n from MLP
- Assumptions: smooth $x_n \to p(z_n|x_n, \theta)$, MLPs generalise smoothly

Recognition and generative models

- Recall ELBO:

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n, \phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n, \theta)\Big)\Big] \qquad \text{(ELBO)}$$

- 3 distributions: $p(z_n|\theta)$, $q(z_n|x_n, \phi)$, $p(x_n|z_n, \theta)$
- VAE:
  - $p(z_n|\theta)$ fixed, EG $\mathcal{N}(z_n|0, I)$
  - $q(z_n|x_n, \phi) =$ "Encoder" / "Recognition" model
  - $p(x_n|z_n, \theta) =$ "Decoder" / "Generative" model
  - Encoder, decoder = neural networks
- NB recognition/generative terminology $\sim$ 18yo [Hinton et al., 1995]

VAE challenges

$1mQs:

1. ~~How to define $q(z_n|x_n, \phi)$?~~
2. How to estimate $\frac{\partial \mathcal{L}}{\partial(\theta,\phi)}$ using BP?
   - 2.1 KL term
   - 2.2 Reconstruction term

Differentiating KL

- Want to differentiate KL term WRT ϕ:

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big] \qquad \text{(ELBO)}$$

- Define encoder as diagonal Gaussian:

$$(\mu, \sigma) = \mathrm{MLP}_{\mathrm{enc}}(x_n, \phi) \qquad \text{(Encoder forward)}$$

$$q(z_n \,|\, x_n, \phi) = \mathcal{N}\Big(z_n \,\Big|\, \mu, \mathrm{diag}(\sigma)^2\Big) \qquad \text{(Encoder distribution)}$$

- Assume:

$$p(z_n|\theta) = \mathcal{N}(z_n|0, I) \qquad \text{(Prior distribution)}$$

Differentiating KL

- Want to differentiate KL term WRT ϕ:

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big] \qquad \text{(ELBO)}$$

$$q(z_n \,|\, x_n, \phi) = \mathcal{N}\Big(z_n \,\Big|\, \mu, \mathrm{diag}(\sigma)^2\Big) \qquad \text{(Encoder distribution)}$$

$$p(z_n|\theta) = \mathcal{N}(z_n|0, I) \qquad \text{(Prior distribution)}$$

$$\Rightarrow \quad \mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big] = \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)} \underbrace{\left[\log\left(\frac{\mathcal{N}\Big(z_n \,\Big|\, \mu, \mathrm{diag}(\sigma)^2\Big)}{\mathcal{N}(z_n|0, I)}\right)\right]}_{\text{Quadratic in } z_n}$$

Differentiating KL

- Want to differentiate KL term WRT ϕ:

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big] \qquad \text{(ELBO)}$$

$$q(z_n \,|\, x_n, \phi) = \mathcal{N}\Big(z_n \,\Big|\, \mu, \mathrm{diag}(\sigma)^2\Big) \qquad \text{(Encoder distribution)}$$

$$p(z_n|\theta) = \mathcal{N}(z_n|0, I) \qquad \text{(Prior distribution)}$$

$$\Rightarrow \quad \mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big] = -\frac{1}{2}\sum_i \Big[1 + 2\log(\sigma_i) - \mu_j^2 - \sigma_j^2\Big] \qquad \text{(Closed-form KL ☺)}$$

VAE challenges

$1mQs:

1. ~~How to define $q(z_n|x_n, \phi)$?~~
2. How to estimate $\frac{\partial \mathcal{L}}{\partial(\theta,\phi)}$ using BP?
   - 2.1 ~~KL term~~
   - 2.2 Reconstruction term

Score function estimator?

- Want to differentiate reconstruction term WRT ϕ:

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big] \qquad \text{(ELBO)}$$

- *Can* differentiate using SFE:

$$\frac{\partial}{\partial\phi}\Big[\mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big]\Big]$$

$$= \frac{\partial}{\partial\phi}\Big[\int dz_n \Big[q(z_n|x_n,\phi)\log\Big(p(x_n|z_n,\theta)\Big)\Big]\Big] \qquad (\mathbb{E} = \textstyle\int q)$$

$$= \int dz_n \Big[\frac{\partial}{\partial\phi}\Big[q(z_n|x_n,\phi)\Big]\log\Big(p(x_n|z_n,\theta)\Big)\Big] \qquad (\text{Swap } \partial \text{ and } \textstyle\int)$$

Score function estimator?

$$= \int dz_n \left[\frac{\partial}{\partial \phi} \Big[q(z_n|x_n, \phi) \Big] \log\Big(p(x_n|z_n, \theta) \Big) \right] \qquad \text{(From previous slide)}$$

$$= \int dz_n \left[q(z_n|x_n, \phi) \frac{\frac{\partial}{\partial \phi} \Big[q(z_n|x_n, \phi) \Big]}{q(z_n|x_n, \phi)} \log\Big(p(x_n|z_n, \theta) \Big) \right] \qquad (\times \tfrac{q}{q})$$

$$= \int dz_n \left[q(z_n|x_n, \phi) \frac{\partial}{\partial \phi} \left[\log\Big(q(z_n|x_n, \phi) \Big) \right] \log\Big(p(x_n|z_n, \theta) \Big) \right]$$

$$(\tfrac{\nabla[q]}{q} = \nabla \big[\log(q) \big])$$

$$= \mathbb{E}_{z_n \sim q(z_n|x_n, \phi)} \left[\frac{\partial}{\partial \phi} \left[\log\Big(q(z_n|x_n, \phi) \Big) \right] \log\Big(p(x_n|z_n, \theta) \Big) \right] \qquad (\textstyle\int q = \mathbb{E})$$

Score function estimator?

$$\Rightarrow \quad \frac{\partial}{\partial \phi}\left[\mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\left[\log\Big(p(x_n|z_n,\theta)\Big)\right]\right]$$

$$= \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\left[\frac{\partial}{\partial \phi}\left[\log\Big(q(z_n|x_n,\phi)\Big)\right]\log\Big(p(x_n|z_n,\theta)\Big)\right] \qquad \text{(SFE)}$$

- *Can* MC and evaluate (SFE) with BP
- *But* (SFE) "exhibits very high variance" [Kingma and Welling, 2013]
  - (Empirical comparison?)

~~Score function estimator?~~ Reparameterisation trick!

- Want to differentiate reconstruction term WRT ϕ:

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n, \phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n, \theta)\Big)\Big] \qquad \text{(ELBO)}$$

- Recall encoder = diagonal Gaussian:

$$(\mu, \sigma) = \mathrm{MLP}_{\mathrm{enc}}(x_n, \phi) \qquad \text{(Encoder forward)}$$

$$q(z_n \,|\, x_n, \phi) = \mathcal{N}\Big(z_n \,\Big|\, \mu, \mathrm{diag}(\sigma)^2\Big) \qquad \text{(Encoder distribution)}$$

- "Reparameterise" z_n as transform of auxiliary noise variable ε:

$$\varepsilon \sim \mathcal{N}(\varepsilon|0, I) \qquad \text{(Auxiliary noise sample)}$$

$$z_n = \mu + \varepsilon \odot \sigma \qquad \text{(Reparameterisation trick)}$$

$$\Rightarrow \quad z_n \sim q(z_n|x_n, \phi) \qquad \text{(Desired latent sample)}$$

~~Score function estimator?~~ Reparameterisation trick!

- Want to differentiate reconstruction term WRT ϕ:

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n,\phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big] \qquad \text{(ELBO)}$$

$$\Rightarrow \quad \frac{\partial}{\partial\phi}\Big[\mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n,\theta)\Big)\Big]\Big]$$

$$= \frac{\partial}{\partial\phi}\Big[\mathbb{E}_{\varepsilon\sim\mathcal{N}(\varepsilon|0,I)}\Big[\log\Big(p\Big(x_n \,\Big|\, \mu + \varepsilon \odot \sigma, \theta\Big)\Big)\Big]\Big] \qquad \text{(Reparameterise)}$$

$$= \mathbb{E}_{\varepsilon\sim\mathcal{N}(\varepsilon|0,I)}\Big[\frac{\partial}{\partial\phi}\Big[\log\Big(p\Big(x_n \,\Big|\, \mu + \varepsilon \odot \sigma, \theta\Big)\Big)\Big]\Big] \qquad \text{(Swap } \partial \text{ and } \mathbb{E}\text{)}$$

$$\approx \frac{1}{L}\sum_{l=1}^{L}\Big[\frac{\partial}{\partial\phi}\Big[\log\Big(p\Big(x_n \,\Big|\, \mu + \varepsilon^{(l)} \odot \sigma, \theta\Big)\Big)\Big]\Big] \qquad \text{(MC)}$$

Evaluating reconstruction term

$$\mathbb{E}_{z_n \sim q(z_n|x_n,\phi)} \left[\log \Big(p(x_n|z_n,\theta) \Big) \right]$$

$$\approx \frac{1}{L} \sum_{l=1}^{L} \left[\log \left(p\left(x_n \,\middle|\, \mu + \varepsilon^{(l)} \odot \sigma, \theta \right) \right) \right] \qquad \text{(Reconstruction term)}$$

- Large batch size $\Rightarrow$ can set $L = 1$ [Kingma and Welling, 2013]
- Freedom to choose $p(x_n|z_n,\theta)$
- EG x_n is continuous $\Rightarrow$ diagonal Gaussian:

$$\left(\mu^{(x)}, \sigma^{(x)} \right) = \text{MLP}_{\text{dec}}(z_n, \theta) \qquad \text{(Decoder forward)}$$

$$p(x_n \,|\, z_n, \theta) = \mathcal{N}\left(x_n \,\middle|\, \mu^{(x)}, \text{diag}\left(\sigma^{(x)} \right)^2 \right) \qquad \text{(Decoder distribution)}$$

Evaluating reconstruction term

$$
\begin{aligned}
&\log\Big(p(x_n \mid z_n, \theta)\Big) \\
&= \log\left(\mathcal{N}\left(x_n \,\middle|\, \mu^{(x)}, \operatorname{diag}\left(\sigma^{(x)}\right)^2\right)\right) && \text{(Decoder log likelihood)} \\
&= \log\left(\prod_i \left[\frac{1}{\sqrt{2\pi}\sigma_i^{(x)}} \exp\left(-\frac{1}{2}\left(\frac{x_{ni} - \mu_i^{(x)}}{\sigma_i^{(x)}}\right)^2\right)\right]\right) && (\mathcal{N} \text{ definition}) \\
&= \frac{1}{2}\sum_i \left[-\log(2\pi) - 2\log\left(\sigma_i^{(x)}\right) - \left(\frac{x_{ni} - \mu_i^{(x)}}{\sigma_i^{(x)}}\right)^2\right] && \text{(Expand logs)}
\end{aligned}
$$

Evaluating reconstruction term

$$\Rightarrow \quad \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)} \left[\log \Big(p(x_n|z_n,\theta) \Big) \right]$$

$$\approx \frac{1}{2} \sum_i \left[-\log(2\pi) - 2\log\left(\sigma_i^{(x)}\right) - \left(\frac{x_{ni} - \mu_i^{(x)}}{\sigma_i^{(x)}} \right)^2 \right] \quad (\mathcal{N}, L = 1)$$

- Putting this together...

ELBO estimator

- Now we have a differentiable + unbiased ELBO estimator ☺

$$\mathcal{L}(x_n, \theta, \phi) = -\mathrm{KL}\Big[q(z_n|x_n, \phi) \,\Big\|\, p(z_n|\theta)\Big] + \mathbb{E}_{z_n \sim q(z_n|x_n,\phi)}\Big[\log\Big(p(x_n|z_n, \theta)\Big)\Big] \tag{ELBO}$$

$$\widehat{\mathcal{L}}(x_n, \theta, \phi) = \frac{1}{2}\sum_i \Big[1 + 2\log(\sigma_i) - \mu_j^2 - \sigma_j^2\Big] + \frac{1}{2}\sum_i \left[-\log(2\pi) - 2\log\left(\sigma_i^{(x)}\right) - \left(\frac{x_{ni} - \mu_i^{(x)}}{\sigma_i^{(x)}}\right)^2\right] \tag{Estimator}$$

VAE training algorithm

- For each batch of data $\{x_1, x_2, \dots\}$:
  - For each element x_n in the batch:

$$(\mu, \sigma) = \text{MLP}_{\text{enc}}(x_n, \phi) \qquad \text{(Encoder forward)}$$

$$\varepsilon \sim \mathcal{N}(\varepsilon | 0, I) \qquad \text{(Auxiliary noise sample)}$$

$$z_n = \mu + \varepsilon \odot \sigma \qquad \text{(Reparameterise)}$$

$$\left(\mu^{(x)}, \sigma^{(x)}\right) = \text{MLP}_{\text{dec}}(z_n, \theta) \qquad \text{(Decoder forward)}$$

$$\widehat{\mathcal{L}}(x_n, \theta, \phi) = \frac{1}{2} \sum_i \left[1 + 2\log(\sigma_i) - \mu_j^2 - \sigma_j^2\right] + \frac{1}{2} \sum_i \left[-\log(2\pi) - 2\log\left(\sigma_i^{(x)}\right) - \left(\frac{x_{ni} - \mu_i^{(x)}}{\sigma_i^{(x)}}\right)^2 \right]$$

$$\widehat{(\text{ELBO})}$$

VAE training algorithm

- For each batch of data $\{x_1, x_2, \dots\}$:
  - For each element x_n in the batch:

$$\widehat{\mathcal{L}}(x_n, \theta, \phi) = \frac{1}{2} \sum_i \left[1 + 2 \log(\sigma_i) - \mu_j^2 - \sigma_j^2 \right] + \frac{1}{2} \sum_i \overbrace{\left[-\log(2\pi) - 2 \log\left(\sigma_i^{(x)}\right) - \left(\frac{x_{ni} - \mu_i^{(x)}}{\sigma_i^{(x)}} \right)^2 \right]}^{\text{(ELBO)}}$$

$$(\theta, \phi) \leftarrow (\theta, \phi) - \frac{\partial}{\partial(\theta, \phi)} \left[\sum_n \left[-\widehat{\mathcal{L}}(x_n, \theta, \phi) \right] \right] \qquad \text{(SGD)}$$

VAE sampling algorithm

$$z \sim \mathcal{N}(z|0, I) \quad \text{(Sample latent from prior)}$$

$$\left(\mu^{(x)}, \sigma^{(x)}\right) = \text{MLP}_{\text{dec}}(z, \theta) \quad \text{(Decoder forward)}$$

$$\varepsilon \sim \mathcal{N}(\varepsilon|0, I) \quad \text{(Auxiliary noise samples)}$$

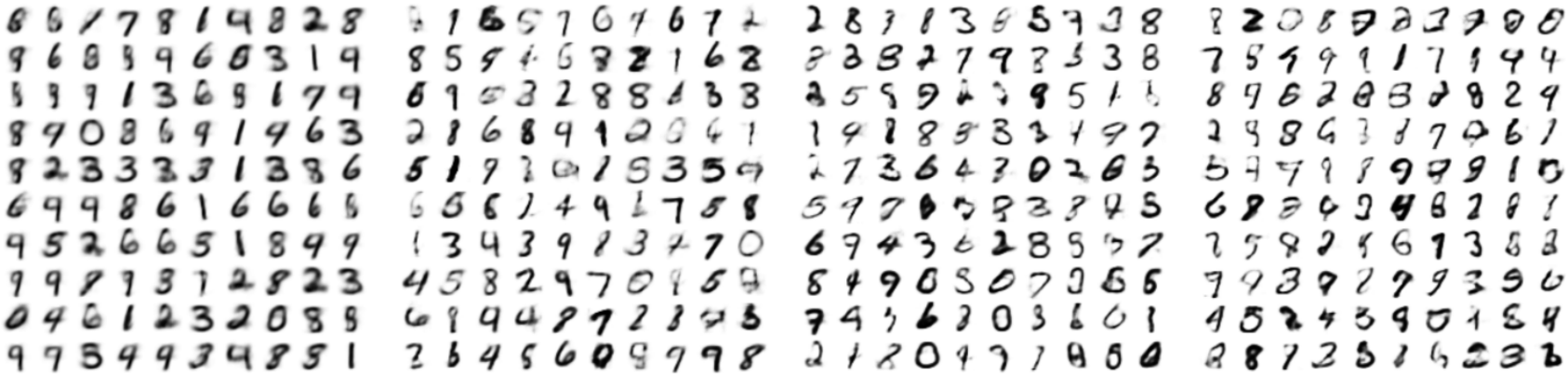
$$x = \mu^{(x)} + \varepsilon \odot \sigma^{(x)} \quad \text{(Reparameterise)}$$

- NB don't use q or ϕ during sampling
- Simple, non-iterative, no extra HPs

VAE summary

- 3 tricks (2 established, 1 innovation*):
  - $\text{argmax}_\theta \left[p(x|\theta) \right] \rightarrow \text{argmax}_\theta \left[\text{ELBO} \right]$
  - Amortised inference $q(z_n|x_n, \theta)$
  - Reparameterisation trick* $z_n = \mu + \varepsilon \odot \sigma$
- Fits into *now-standard* (then-visionary) DL framework:
  - Define model(s) + $\mathcal{L}$
  - $\partial \mathcal{L} / \partial x$ (BP)
  - SGD

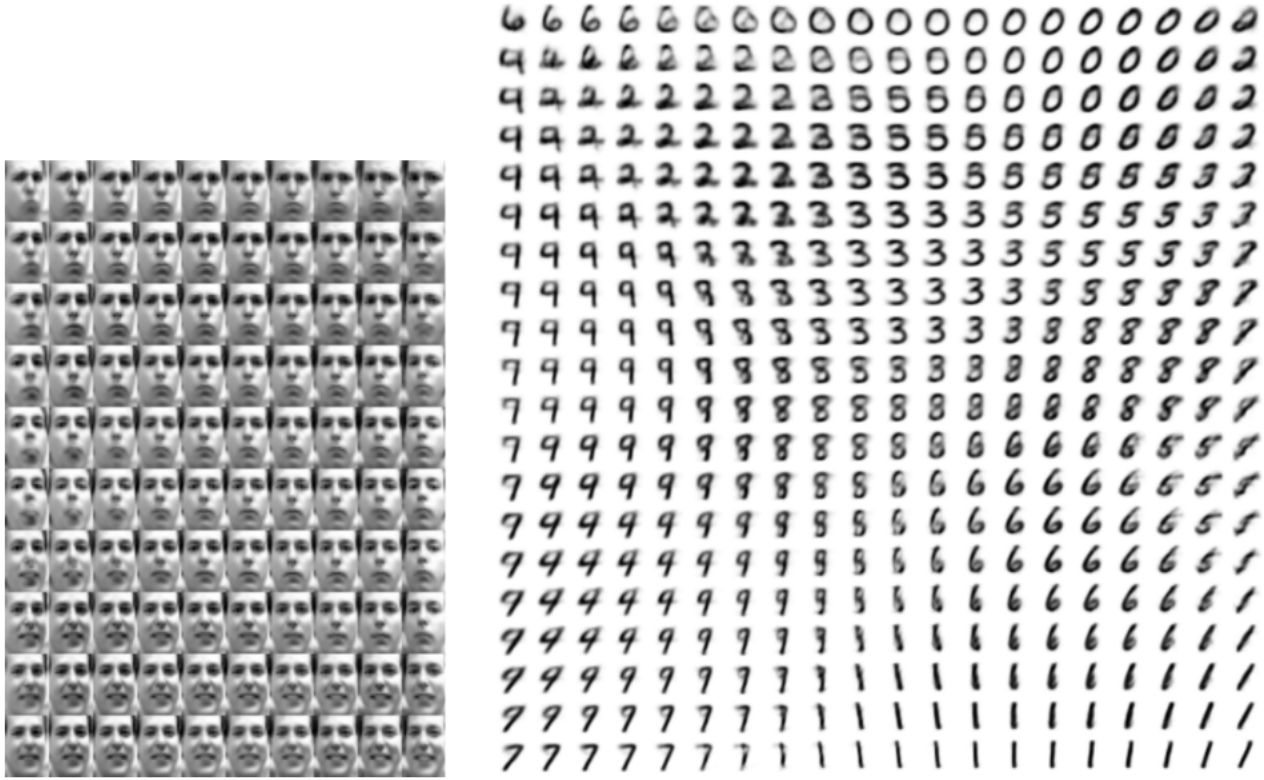
Results [Kingma and Welling, 2013]



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

Results [Kingma and Welling, 2013]



(a) Learned Frey Face manifold

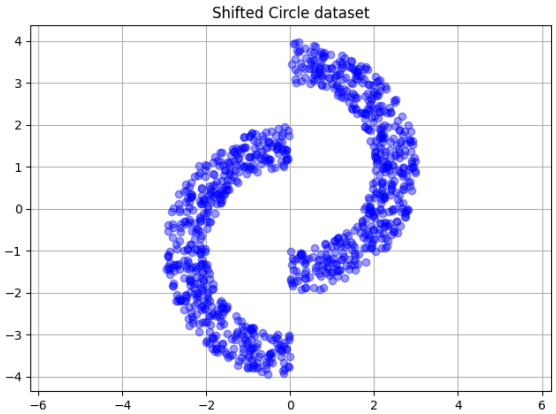
(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

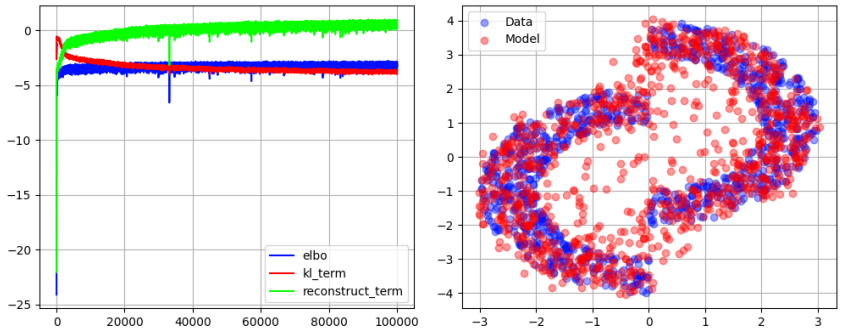
VAE demo

- VAE implementation: `https://github.com/jakelevi1996/vae/blob/main/src/vae/models/vae.py`

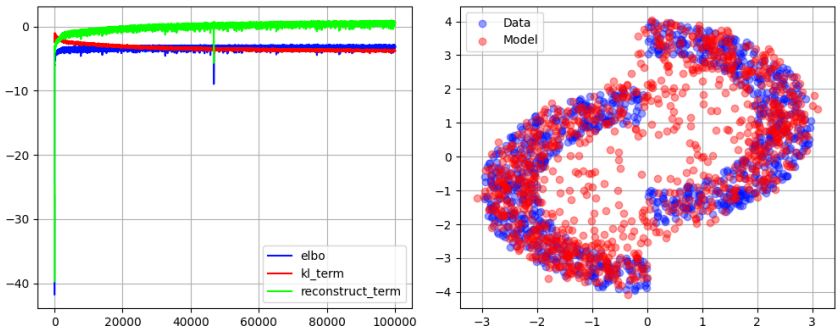
VAE demo 2D dataset results



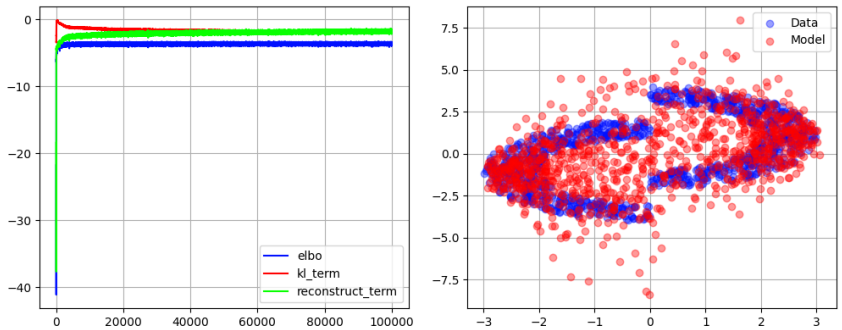
(a) Dataset



(b) Seed = 0



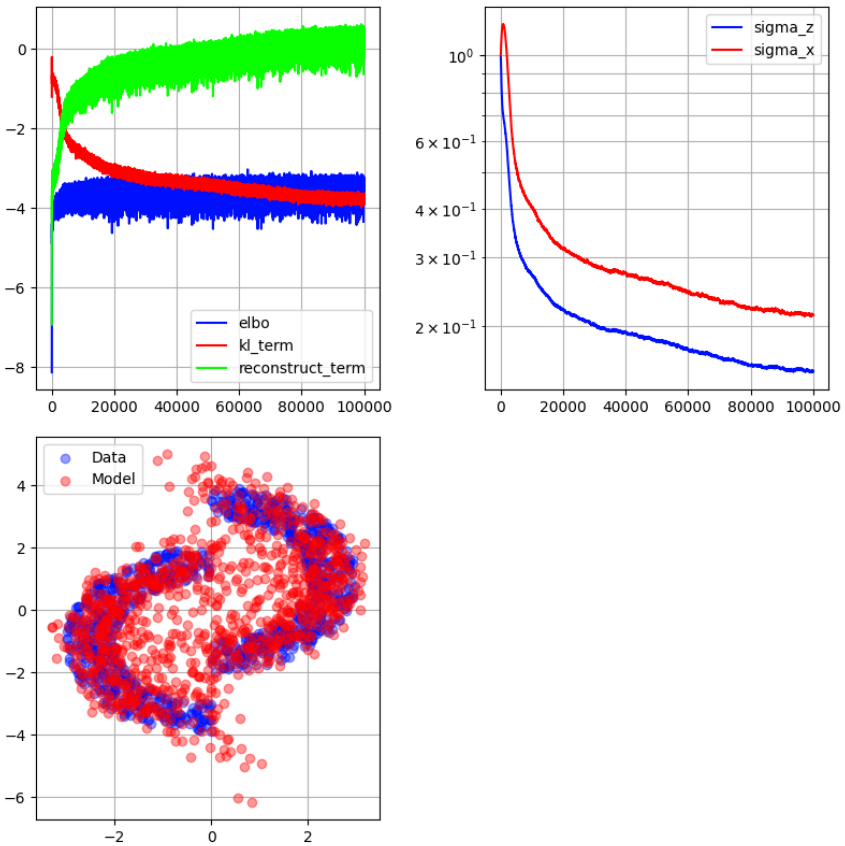
(c) Seed = 1



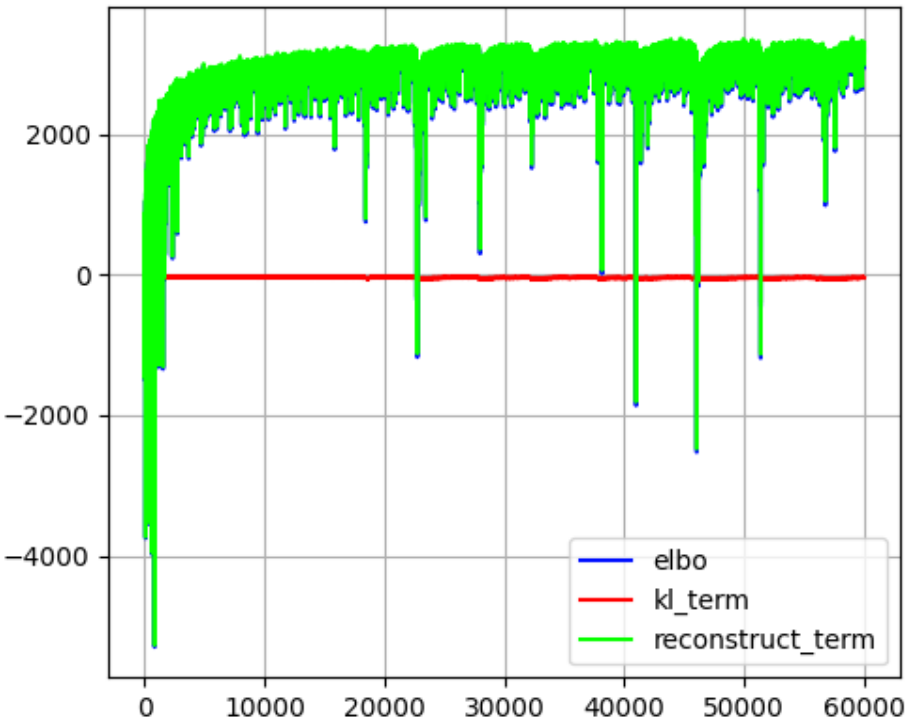
(d) Seed = 2

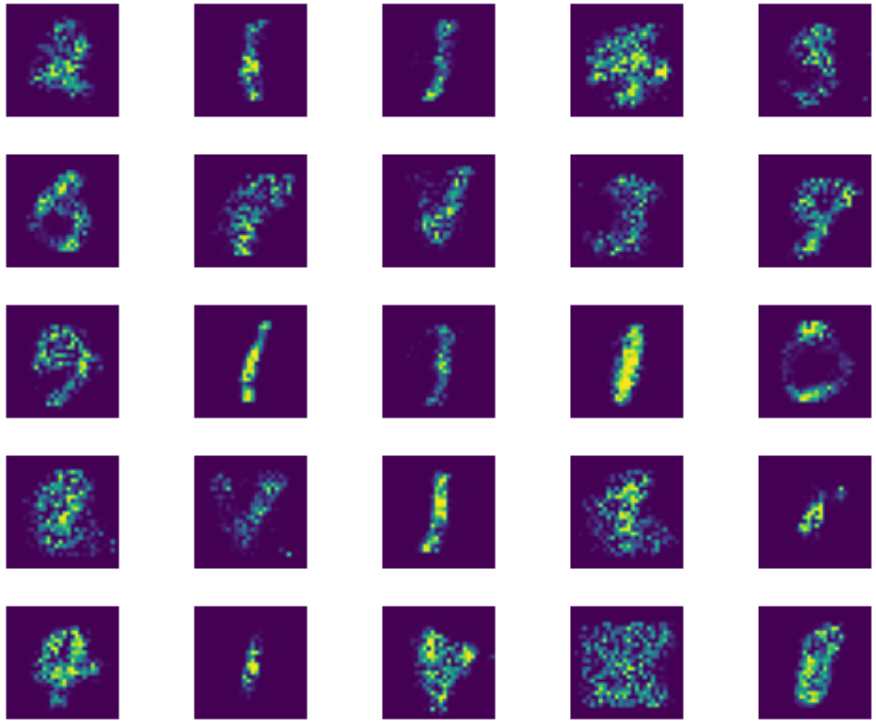
VAE demo 2D dataset results

- Example with learned shared σ (not from MLP):



VAE MNIST demo





VAE MNIST demo

- NB for MNIST:
  - Above demo = 100 epochs $\approx$ 30 minutes

VAE MNIST demo

- NB for MNIST:
  - Above demo = 100 epochs $\approx$ 30 minutes
  - [Kingma and Welling, 2013] train for >1000 epochs/33 hours !!!
  - Classification: 5 epochs $\Rightarrow$ <30 seconds $\Rightarrow$ >98% test acc
  - Discriminative VS generative?

What's next?

1. Conditional generation [**Kingma** et al., 2014, Sohn et al., 2015, Yan et al., 2016]

2. Molecule generation [**Gómez-Bombarelli** et al., 2018, Jin et al., 2018, Griffiths and Hernández-Lobato, 2020, Grosnit et al., 2021, Maus et al., 2022, Sevgen et al., 2025]

3. Object-centric learning [Watters et al., 2019, Burgess et al., 2019, Locatello et al., 2020]

4. Discrete latent variables [Jang et al., 2016, Maddison et al., 2016, Van Den Oord et al., 2017]

5. ...

Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma*, **Danilo J. Rezende**†**, Shakir Mohamed**†**, Max Welling***
*Machine Learning Group, Univ. of Amsterdam, {D.P.Kingma, M.Welling}@uva.nl
†Google Deepmind, {danilor, shakir}@google.com

Abstract

The ever-increasing size of modern data sets combined with the difficulty of obtaining label information has made semi-supervised learning one of the problems of significant practical importance in modern data analysis. We revisit the approach to semi-supervised learning with generative models and develop new models that allow for effective generalisation from small labelled data sets to large unlabelled ones. Generative approaches have thus far been either inflexible, inefficient or non-scalable. We show that deep generative models and approximate Bayesian inference exploiting recent advances in variational methods can be used to provide significant improvements, making generative approaches highly competitive for semi-supervised learning.

1 Introduction

Semi-supervised learning considers the problem of classification when only a small subset of the observations have corresponding class labels. Such problems are of immense practical interest in a wide range of applications, including image search (Fergus et al., 2009), genomics (Shi and Zhang, 2011), natural language parsing (Liang, 2005), and speech analysis (Liu and Kirchhoff, 2013), where unlabelled data is abundant, but obtaining class labels is expensive or impossible to obtain for the entire data set. The question that is then asked is: how can properties of the data be used to improve decision boundaries and to allow for classification that is more accurate than that based on classifiers constructed using the labelled data alone. In this paper we answer this question by developing *probabilistic models for inductive and transductive semi-supervised learning* by utilising an explicit model of the data density, building upon recent advances in deep generative models and scalable variational inference (Kingma and Welling, 2014; Rezende et al., 2014).

Amongst existing approaches, the simplest algorithm for semi-supervised learning is based on a *self-training* scheme (Rosenberg et al., 2005) where the the model is bootstrapped with additional labelled data obtained from its own highly confident predictions; this process being repeated until some termination condition is reached. These methods are heuristic and prone to error since they can reinforce poor predictions. *Transductive SVMs* (TSVM) (Joachims, 1999) extend SVMs with the aim of max-margin classification while ensuring that there are as few unlabelled observations near the margin as possible. These approaches have difficulty extending to large amounts of unlabelled data, and efficient optimisation in this setting is still an open problem. *Graph-based methods* are amongst the most popular and aim to construct a graph connecting similar observations; label information propagates through the graph from labelled to unlabelled nodes by finding the minimum energy (MAP) configuration (Blum et al., 2004; Zhu et al., 2003). Graph-based approaches are sensitive to the graph structure and require eigen-analysis of the graph Laplacian, which limits the scale to which these methods can be applied – though efficient spectral methods are now available (Fergus et al., 2009). *Neural network*-based approaches combine unsupervised and supervised learning

For an updated version of this paper, please see http://arxiv.org/abs/1406.5298

1

M2

- [**Kingma** et al., 2014]
- Semi-supervised learning:
  - LARGE unlabelled dataset $\{x_i, \dots\} \sim \tilde{p}_u$
  - small labelled dataset $\{(x_j, y_j), \dots\} \sim \tilde{p}_l$
  - Train classifier
  - Outperform ignoring $\{x_i, \dots\}$
- 3 models proposed:
  - M1 = train VAE on x, train classifier on y from z
  - M2 (next slide)
  - M1+M2 = use z from M1 as x for M2

M2

- Model:

$$p_\theta(x, y, z) = p_\theta(x|y, z)\ p(y)\ p(z) \qquad \text{(Generative model)}$$
$$q_\phi(z, y|x) = q_\phi(z|x)\ q_\phi(y|x) \qquad \text{(Factorised posterior)}$$

- Different loss functions for with/without labels:
- With labels:

$$\begin{aligned}
\log\Big(p(x, y)\Big) &\geq \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p(x, y, z)}{q_\phi(z|x)}\right)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\Big[\log\Big(p_\theta(x|y, z)\Big) + \log\Big(p(y)\Big) \\
&\qquad + \log\Big(p(z)\Big) - \log\Big(q_\phi(z|x)\Big)\Big] \\
&= -\mathcal{L}(x, y)
\end{aligned}$$

M2

- With label:

$$-\mathcal{L}(x,y) = \mathbb{E}_{q_\phi(z|x)}\Big[\log\Big(p_\theta(x|y,z)\Big) + \log\Big(p(y)\Big) + \log\Big(p(z)\Big) - \log\Big(q_\phi(z|x)\Big)\Big]$$

- Without label:

$$\begin{aligned}
\log\Big(p(x)\Big) &\geq \mathbb{E}_{q_\phi(y,z|x)}\left[\log\left(\frac{p(x,y,z)}{q_\phi(y,z|x)}\right)\right] \\
&= \mathbb{E}_{q_\phi(y|x)}\Big[\mathbb{E}_{q_\phi(z|x)}\Big[\log\Big(p_\theta(x|y,z)\Big) + \log\Big(p(y)\Big) \\
&\qquad + \log\Big(p(z)\Big) - \log\Big(q_\phi(z|x)\Big) - \log\Big(q_\phi(y|x)\Big)\Big]\Big] \\
&= \mathbb{E}_{q_\phi(y|x)}\Big[-\mathcal{L}(x,y)\Big] + \mathcal{H}\Big[q_\phi(y|x)\Big]
\end{aligned}$$

M2

- With label:

$$-\mathcal{L}(x,y) = \mathbb{E}_{q_\phi(z|x)}\Big[\log\Big(p_\theta(x|y,z)\Big) + \log\Big(p(y)\Big) + \log\Big(p(z)\Big) - \log\Big(q_\phi(z|x)\Big)\Big]$$

- Without label:

$$-\mathcal{U}(x) = \mathbb{E}_{q_\phi(y|x)}\Big[-\mathcal{L}(x,y)\Big] + \mathcal{H}\Big[q_\phi(y|x)\Big]$$

- Bound on ML for full dataset:

$$\mathcal{J} = \sum_{(x,y)\sim\tilde{p}_l}\Big[\mathcal{L}(x,y)\Big] + \sum_{x\sim\tilde{p}_u}\Big[\mathcal{U}(x)\Big]$$

- Total loss: $+$ term to train classifier $q_\phi(y|x)$ on $(x,y) \sim \tilde{p}_l$

M2

- Semi-supervised classification results:

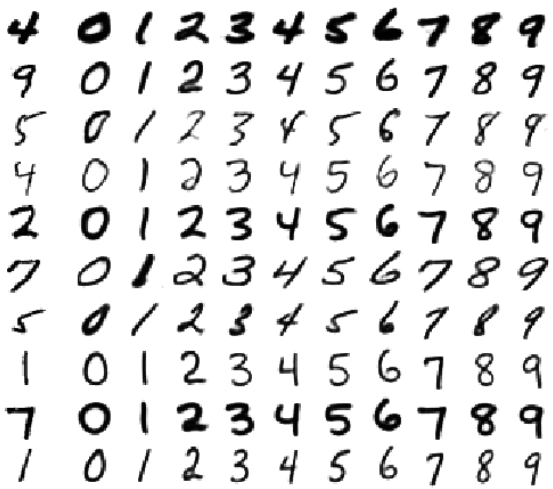
Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 ($\pm$ 0.95)	11.82 ($\pm$ 0.25)	11.97 ($\pm$ 1.71)	**3.33** ($\pm$ 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 ($\pm$ 0.049)	4.94 ($\pm$ 0.13)	**2.59** ($\pm$ 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 ($\pm$ 0.12)	4.24 ($\pm$ 0.07)	3.60 ($\pm$ 0.56)	**2.40** ($\pm$ 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 ($\pm$ 0.04)	3.92 ($\pm$ 0.63)	**2.18** ($\pm$ 0.04)

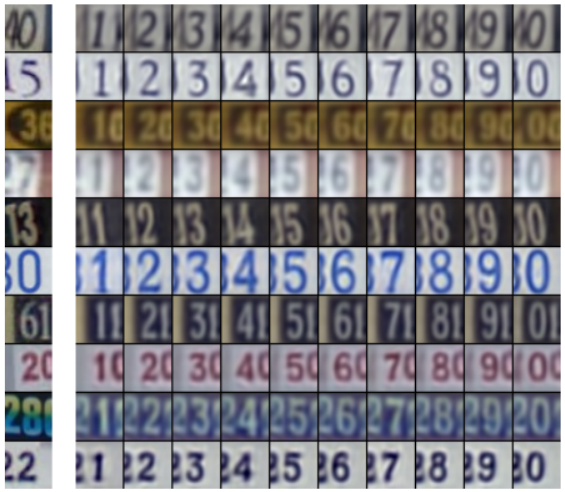
M2

- Using $p_\theta(x|y,z)$, fixing y/z respectively:

(a) Handwriting styles for MNIST obtained by fixing the class label and varying the 2D latent variable **z**



(b) MNIST analogies



(c) SVHN analogies

[Gómez-Bombarelli et al., 2018]

ACS central science

Research Article

Cite This: ACS Cent. Sci. 2018, 4, 268−276

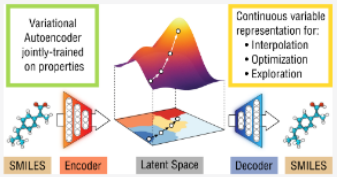
Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules

Rafael Gómez-Bombarelli,[†,#] Jennifer N. Wei,[‡,#] David Duvenaud,[¶,#] José Miguel Hernández-Lobato,[§,#] Benjamín Sánchez-Lengeling,[‡] Dennis Sheberla,[‡] Jorge Aguilera-Iparraguirre,[†] Timothy D. Hirzel,[†] Ryan P. Adams,[∇,∥] and Alán Aspuru-Guzik*[,‡,⊥]

[†]Kyulux North America Inc., 10 Post Office Square, Suite 800, Boston, Massachusetts 02109, United States
[‡]Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States
[¶]Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Ontario M5S 3H5, Canada
[§]Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, U.K.
[∇]Google Brain, Mountain View, California, United States
[∥]Princeton University, Princeton, New Jersey, United States
[⊥]Biologically-Inspired Solar Energy Program, Canadian Institute for Advanced Research (CIFAR), Toronto, Ontario M5S 1M1, Canada

Supporting Information

ABSTRACT: We report a method to convert discrete representations of molecules to and from a multidimensional continuous representation. This model allows us to generate new molecules for efficient exploration and optimization through open-ended spaces of chemical compounds. A deep neural network was trained on hundreds of thousands of existing chemical structures to construct three coupled functions: an encoder, a decoder, and a predictor. The encoder converts the discrete representation of a molecule into a real-valued continuous vector, and the decoder converts these continuous vectors back to discrete molecular representations. The predictor estimates chemical properties from the latent continuous vector representation of the molecule. Continuous representations of molecules allow us to automatically generate novel chemical structures by performing simple operations in the latent space, such as decoding random vectors, perturbing known chemical structures, or interpolating between molecules. Continuous representations also allow the use of powerful gradient-based optimization to efficiently guide the search for optimized functional compounds. We demonstrate our method in the domain of drug-like molecules and also in a set of molecules with fewer that nine heavy atoms.



INTRODUCTION

The goal of drug and material design is to identify novel molecules that have certain desirable properties. We view this as an optimization problem, in which we are searching for the molecules that maximize our quantitative desiderata. However, optimization in molecular space is extremely challenging, because the search space is large, discrete, and unstructured. Making and testing new compounds are costly and time-consuming, and the number of potential candidates is overwhelming. Only about 10^8 substances have ever been synthesized,[1] whereas the range of potential drug-like molecules is estimated to be between 10^{23} and 10^{60}.[2]

Virtual screening can be used to speed up this search.[3−6] Virtual libraries containing thousands to hundreds of millions of candidates can be assayed with first-principles simulations or statistical predictions based on learned proxy models, and only the most promising leads are selected and tested experimentally.

However, even when accurate simulations are available,[7] computational molecular design is limited by the search strategy used to explore chemical space. Current methods either exhaustively search through a fixed library,[8,9] or use discrete local search methods such as genetic algorithms[10−15] or similar discrete interpolation techniques.[16−18] Although these techniques have led to useful new molecules, these approaches still face large challenges. Fixed libraries are monolithic, costly to fully explore, and require hand-crafted rules to avoid impractical chemistries. The genetic generation of compounds requires manual specification of heuristics for mutation and crossover rules. Discrete optimization methods have difficulty

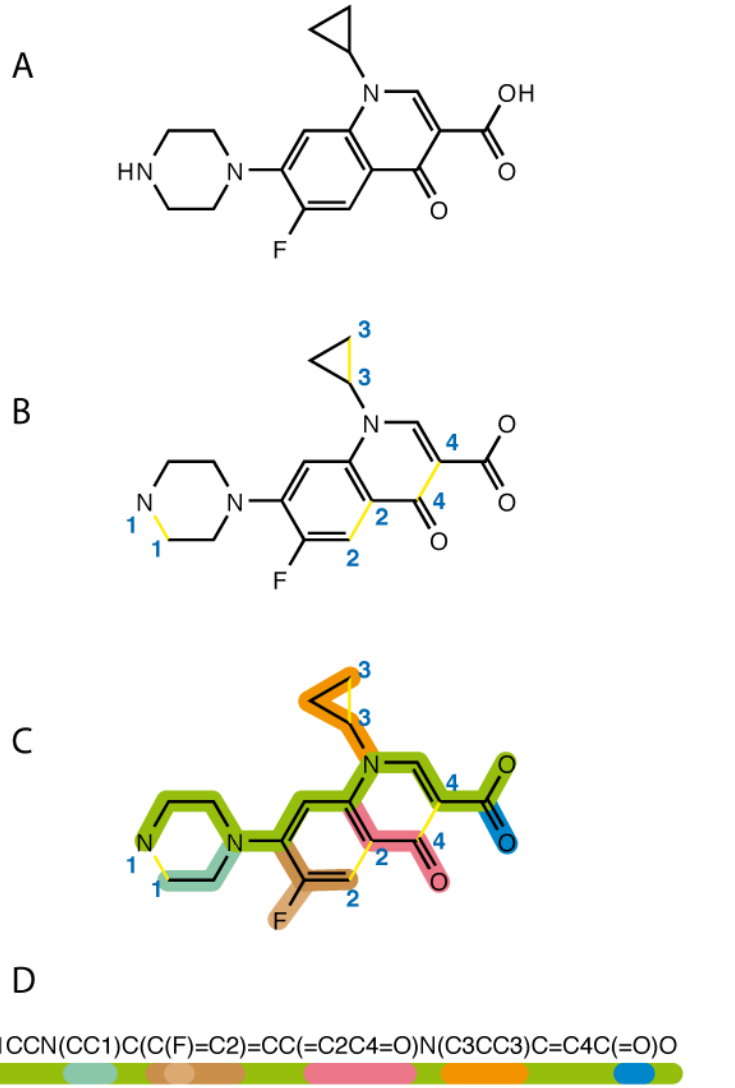
Received: December 2, 2017
Published: January 12, 2018

DOI: 10.1021/acscentsci.7b00572
ACS Cent. Sci. 2018, 4, 268−276

[**Gómez-Bombarelli** et al., 2018]

★ Generate new molecules with desirable properties

- Train VAE on SMILES strings
- SMILES = "Simplified Molecular Input Line Entry System"
- Chemical structure → ASCII string



https://en.wikipedia.org/wiki/Simplified_Molecular_Input_Line_Entry_System

[Gómez-Bombarelli et al., 2018]

- Train $f_{\text{enc}}(x_{\text{SMILES}}) = z_n$
- Train $f_{\text{dec}}(z_n) = x_{\text{SMILES}}$
- $f_{\text{enc}}, f_{\text{dec}}$ = CNNs or RNNs
- Train property prediction $f_{\text{prop}}(z_n) = \pi$, f_{prop} = MLP or GP, $\pi = \ldots$
  - "logP" = water-octanol partition coefficient
  - "SAS" = synthetic accessibility score
  - "QED" = Quantitative Estimation of Drug-likeness
  - ...
- To sample molecules with properties π^*:
  - $z_n = \text{argmin}_{z_n} \left[\mathcal{L}\left(f_{\text{prop}}(z_n), \pi^* \right) \right]$ (EG BP through f_{prop} + SGD)
  - $f_{\text{dec}}(z_n) = x_{\text{SMILES}}$

[Gómez-Bombarelli et al., 2018]

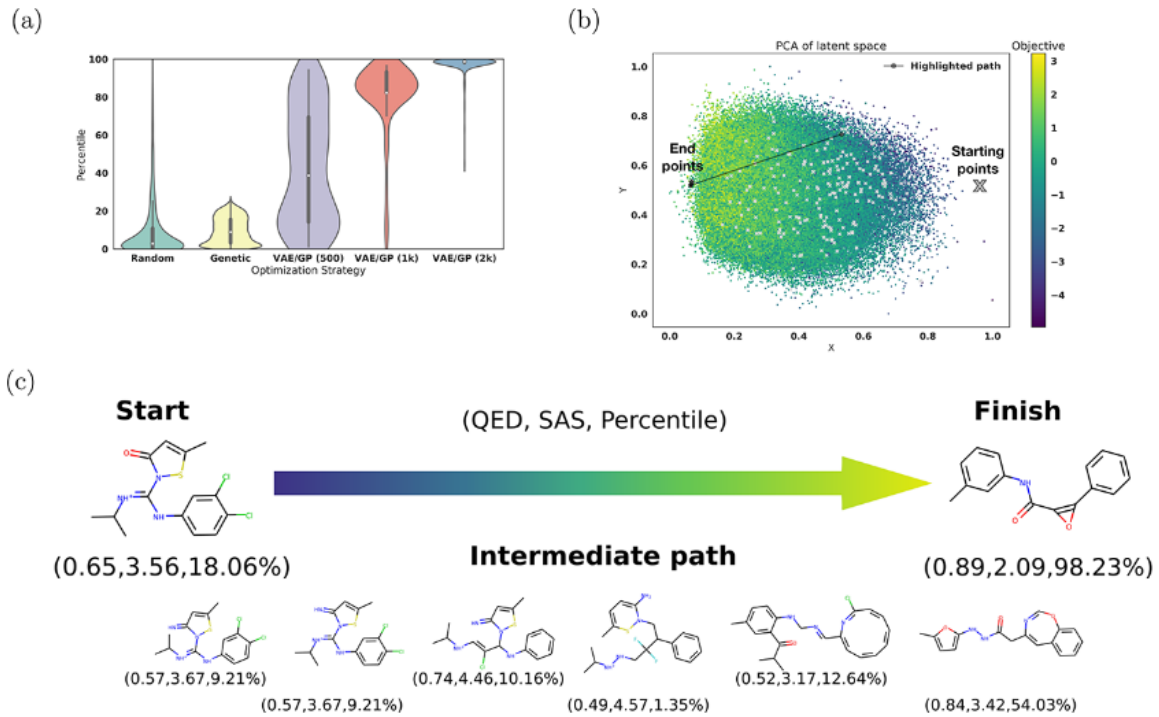
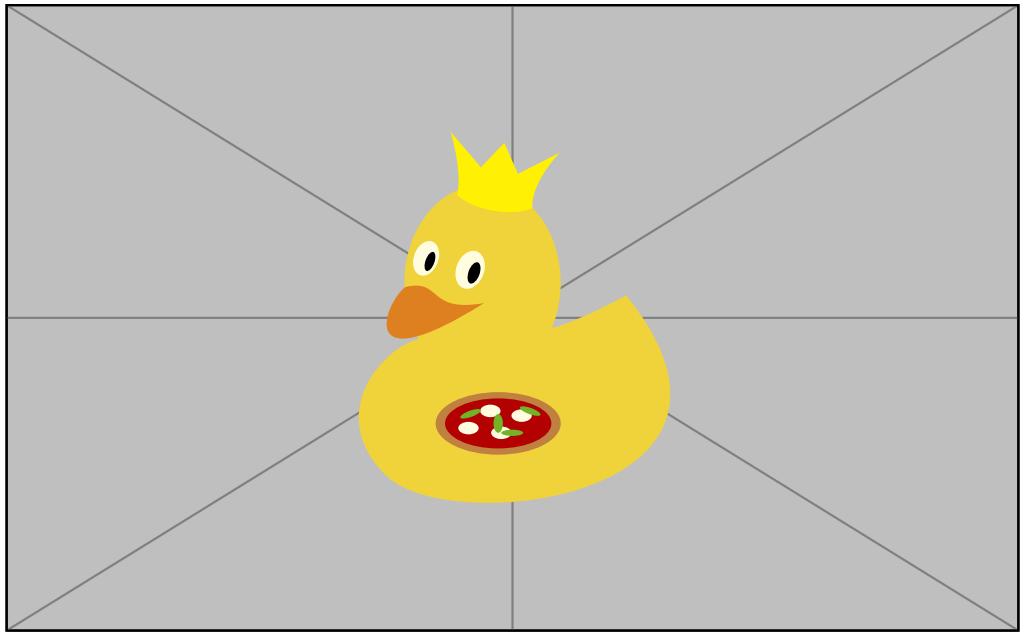


Figure 4. Optimization results for the jointly trained autoencoder using $5 \times \text{QED} - \text{SAS}$ as the objective function. (a) shows a violin plot which compares the distribution of sampled molecules from normal random sampling, SMILES optimization via a common chemical transformation with a genetic algorithm, and from optimization on the trained Gaussian process model with varying amounts of training points. To offset differences in computational cost between the random search and the optimization on the Gaussian process model, the results of 400 iterations of random search were compared against the results of 200 iterations of optimization. This graph shows the combined results of four sets of trials. (b) shows the starting and ending points of several optimization runs on a PCA plot of latent space colored by the objective function. Highlighted in black is the path illustrated in part (c). (c) shows a spherical interpolation between the actual start and finish molecules using a constant step size. The QED, SAS, and percentile score are reported for each molecule.

VAE molecule generation developments

- ~~SMILES~~ Encode/decode molecular graphs directly [Jin et al., 2018]
- *Constrained* BO $\Rightarrow$ avoid generating invalid molecules [Griffiths and Hernández-Lobato, 2020]
- "Deep metric learning" EG $\mathcal{L}_{\text{triplet}}(x^b, x^+, x^-) \Rightarrow$ improve VAE latent space structure $\Rightarrow$ improve BO performance [Grosnit et al., 2021]
- Adapt trust regions to account for input space structure [Maus et al., 2022]
- Using "pretrained ProtT5nv transformer-based T5 encoder and decode" in the VAE pipeline [Sevgen et al., 2025]
- ...

The end.



Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016).
Layer normalization.
arXiv preprint arXiv:1607.06450.

Bahdanau, D., Cho, K., and Bengio, Y. (2014).
Neural machine translation by jointly learning to align and translate.
arXiv preprint arXiv:1409.0473.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019).
Monet: Unsupervised scene decomposition and representation.
arXiv preprint arXiv:1901.11390.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019).
Bert: Pre-training of deep bidirectional transformers for language understanding.
In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Glorot, X. and Bengio, Y. (2010).

Understanding the difficulty of training deep feedforward neural networks.
In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.


Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014).
Generative adversarial nets.
Advances in neural information processing systems, 27.


Griffiths, R.-R. and Hernández-Lobato, J. M. (2020).
Constrained bayesian optimization for automatic chemical design using variational autoencoders.
Chemical science, 11(2):577–586.


Grosnit, A., Tutunov, R., Maraval, A. M., Griffiths, R.-R., Cowen-Rivers, A. I., Yang, L., Zhu, L., Lyu, W., Chen, Z., Wang, J., et al. (2021).

High-dimensional bayesian optimisation with variational autoencoders and deep metric learning.
arXiv preprint arXiv:2106.03609.


He, K., Zhang, X., Ren, S., and Sun, J. (2015).
Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.
In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.


He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.


Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995).
The" wake-sleep" algorithm for unsupervised neural networks.
Science, 268(5214):1158–1161.


Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012).

Improving neural networks by preventing co-adaptation of feature detectors.
arXiv preprint arXiv:1207.0580.


Ioffe, S. and Szegedy, C. (2015).
Batch normalization: Accelerating deep network training by reducing internal covariate shift.
In *International conference on machine learning*, pages 448–456. pmlr.


Jang, E., Gu, S., and Poole, B. (2016).
Categorical reparameterization with gumbel-softmax.
arXiv preprint arXiv:1611.01144.


Jin, W., Barzilay, R., and Jaakkola, T. (2018).
Junction tree variational autoencoder for molecular graph generation.
In *International conference on machine learning*, pages 2323–2332. PMLR.


Kingma, D. P. and Ba, J. (2014).
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980.

Kingma, D. P. and Welling, M. (2013).
Auto-encoding variational bayes.
arXiv preprint arXiv:1312.6114.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).
Imagenet classification with deep convolutional neural networks.
Advances in neural information processing systems, 25.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020).
Object-centric learning with slot attention.
Advances in neural information processing systems, 33:11525–11538.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016).
The concrete distribution: A continuous relaxation of discrete random variables.
arXiv preprint arXiv:1611.00712.

Maus, N., Jones, H., Moore, J., Kusner, M. J., Bradshaw, J., and Gardner, J. (2022).
Local latent space bayesian optimization over structured inputs.

Advances in neural information processing systems, 35:34505–34518.


Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018).
Improving language understanding by generative pre-training.
OpenAI blog.


Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019).
Language models are unsupervised multitask learners.
OpenAI blog, 1(8):9.


Rezende, D. J., Mohamed, S., and Wierstra, D. (2014).
Stochastic backpropagation and approximate inference in deep generative models.
In *International conference on machine learning*, pages 1278–1286. PMLR.


Sevgen, E., Moller, J., Lange, A., Parker, J., Quigley, S., Mayer, J., Srivastava, P., Gayatri, S., Hosfield, D., Dilks, C., et al. (2025).

Prot-vae: protein transformer variational autoencoder for functional protein design.
Proceedings of the National Academy of Sciences, 122(41):e2408737122.


Simonyan, K. and Zisserman, A. (2014).
Very deep convolutional networks for large-scale image recognition.
arXiv preprint arXiv:1409.1556.


Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015).
Deep unsupervised learning using nonequilibrium thermodynamics.
In *International conference on machine learning*, pages 2256–2265. pmlr.


Sohn, K., Lee, H., and Yan, X. (2015).
Learning structured output representation using deep conditional generative models.
Advances in neural information processing systems, 28.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018).
Automatic chemical design using a data-driven continuous representation of molecules.
ACS central science, 4(2):268–276.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014).
Semi-supervised learning with deep generative models.
Advances in neural information processing systems, 27.

Van Den Oord, A., Vinyals, O., et al. (2017).
Neural discrete representation learning.
Advances in neural information processing systems, 30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017).
Attention is all you need.
Advances in neural information processing systems, 30.


Watters, N., Matthey, L., Burgess, C. P., and Lerchner, A. (2019).
Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes.
arXiv preprint arXiv:1901.07017.


Yan, X., Yang, J., Sohn, K., and Lee, H. (2016).
Attribute2image: Conditional image generation from visual attributes.
In *European conference on computer vision*, pages 776–791. Springer.