Variational Lossy Autoencoder

Chen et al.

Dylan Green
March 17, 2020
Autoencoders
Autoencoders

- Unsupervised deep learning model
- Loss: $\|x - \hat{x}\|^2$
- $\text{dim}(z) \ll \text{dim}(x)$
- Features should extract useful, high-level information from the input data
Applications - Visualization
Applications - Denoising Autoencoders
Applications - Unsupervised Feature Learning

Input data $\mathcal{X}$ → Encoder $\mathcal{Z}$ → Classifier $\hat{y}$ → Loss function (Softmax, etc)

Fine-tune encoder jointly with classifier $y$
Autoencoders as “Generative” Models?

- What if we want to generate new data using this model?
  - Pick a random $z$, use decoder to generate new image

- **Problem:**
  - Model maps each $x$ to a point in $z$-space
  - How to pick a ”good” $z$?
Variational Autoencoders (VAEs)
VAEs

- A probabilistic spin on autoencoders
  - Learn latent variables $z$ from input data
  - Sample from the model to generate new data
- Intuition: $x$ is an image, $z$ encodes high-level information about the image (i.e. attributes, orientation, etc.)
• Assume a generative model with a latent variable $z$ distributed according to some prior distribution $p(z)$
• The observed variable $x$ is then distributed according to a conditional likelihood $p_\theta(x|z)$
• Sample in two steps:
  • $z \sim p(z)$
  • $x \sim p_\theta(x|z)$
• Marginal likelihood of the data under this model is then

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p(z)dz$$
For the standard VAE:

- Choose $p(z) = \mathcal{N}(z|0, I)$
- Represent $p_\theta(x|z)$ with a neural network:
  - Can be thought of as a stochastic decoder network
  - Input: $z$, Outputs: mean $\mu_{x|z}$ and diagonal covariance $\Sigma_{x|z}$
• **Objective**: maximize marginal likelihood of training data:

\[
\max_{\theta} \sum_i \log p_{\theta}(x^{(i)})
\]

where

\[
p_{\theta}(x^{(i)}) = \int p_{\theta}(x^{(i)}|z)p(z)dz
\]

• **Problem**: This integral is intractable
• Potential fix: try Bayes’ rule:

\[ p_\theta(x^{(i)}) = \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \]

• **Another problem:** \( p_\theta(z \mid x^{(i)}) \) is also intractable

• **Solution:** Introduce (stochastic) encoder network \( q_\phi(z \mid x^{(i)}) \)
  - \( q_\phi(z \mid x^{(i)}) \approx p_\theta(z \mid x^{(i)}) \)
  - Input: \( x \), Outputs: mean \( \mu_{z \mid x} \) and diagonal covariance \( \Sigma_{z \mid x} \)

• Jointly train \( q_\phi, p_\theta \)
Plug this in to marginal likelihood

\[ \log p_\theta(x) = \log \int p_\theta(x, z) dz = \log \int q_\phi(z|x) \frac{p_\theta(z, x)}{q_\phi(z|x)} dz \]

\[ \geq \mathbb{E}_{q_\phi(z|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \quad \text{(Jensen’s Inequality)} \]

\[ \triangleq \mathcal{L}(\theta, \phi) \]

\( \mathcal{L}(\theta, \phi) \) is the log Evidence Lower BOund, or ELBO
Rearranging:

$$\log p_{\theta}(x) \geq \left( \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x|z) \right) - KL(q_{\phi}(z|x)||p(z))$$

- **Reconstruction Loss**
- **Regularization**

$$\mathcal{L}(\theta, \phi)$$ — VAE Objective
Another derivation:

\[
D_{KL} [q_x(z) \parallel p(z|x)] = \mathbb{E}_{z \sim q_x(z)} \left[ \log q_x(z) - \log p(z|x) \right]
\]

\[
= \mathbb{E}_{z \sim q_x(z)} \left[ \log q_x(z) - \log \frac{p(z, x)}{p(x)} \right]
\]

\[
= \mathbb{E}_{z \sim q_x(z)} \left[ \log q_x(z) - \log p(z) - \log p(x|z) + \log p(x) \right]
\]

\[
= \mathbb{E}_{z \sim q_x(z)} \left[ \log q_x(z) - \log p(z) - \log p(x|z) \right] + \log p(x)
\]

Only this part depends on \( z \)

Rearranging gives us:

\[
\log p(x) = -\mathbb{E}_{z \sim q_x(z)} \left[ \log q_x(z) - \log p(z) - \log p(x|z) \right] + D_{KL} [q_x(z) \parallel p(z|x)]
\]

\[
= \mathbb{E}_{z \sim q_x(z)} \left[ \log p(z) + \log p(x|z) - \log q_x(z) \right] + D_{KL} [q_x(z) \parallel p(z|x)]
\]

Variational Lower Bound \( \geq 0 \)

Takeaway: \( \mathcal{L}(\theta, \phi) \) becomes exact if \( q_\phi(z|x) = p_\theta(z|x) \)
VAEs - Training

Train by maximizing

\[ \mathcal{L}(\theta, \phi) = \left( \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z) \right) - KL(q_\phi(z|x) \| p(z)) \]

1. Run input through encoder to get \( q_\phi(z|x) \)
2. Sample \( z \) from \( q_\phi(z|x) \) using "reparameterization" trick:
   - \( \epsilon \sim \mathcal{N}(0, I) \)
   - \( z = \mu_{z|x} + \epsilon \odot \Sigma_{z|x} \)
3. Run sampled \( z \) through decoder to get \( p_\theta(x|z) \)
4. Loss can be computed in closed form
To sample from the model:

1. Sample $z \sim p(z)$
2. Run sampled $z$ through decoder to get $p_\theta(x|z)$
3. Sample $x \sim p_\theta(x|z)$ to generate new data
VAEs - Samples

32x32 CIFAR-10

Labeled Faces in the Wild
VAEs - Moving Through Latent Space
VAEs - "Image Editing"
Hold $y$ fixed, vary $z$
Hold $z$ fixed, vary $y$
Variational Lossy Autoencoder (VLAE)
How to improve on VAEs?

- Reconstructed images are often blurry
- Simple decoder distribution $p_\theta(x|z)$ lacks expressivity
  - Due to diagonal covariance $\Sigma_{x|z}$, all pixels are generated independently from one another
  - All entropy in the data must be explained by $z$
  - Not just content and style, but local features like texture
- Idea: use a decoder capable of modelling local correlations
Autoregressive Models

- Define some ordering over pixels
- Chain rule of probability

\[
p(x) = p(x_1, x_2, \ldots, x_d) \\
= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \ldots \\
= \prod_{i=1}^{d} p(x_i|x_{1:i-1})
\]

- Model \( p(x_i|x_{1:i-1}) \) with a neural network \( p_\theta \) and maximize log likelihood

\[
\log p_\theta(x) = \sum_{i=1}^{d} \log p_\theta(x_i|x_{1:i-1})
\]
PixelCNN

- Dependency on previous pixels modelled by a (masked) CNN
- Training for each location can be done in parallel
- Sampling must be done sequentially
- Powerful generative models in their own right
What happens if we use a powerful decoder like this?

- Good news: great for generative modelling
- Bad news: the model completely ignores the latent code
First recall that the goal of designing an efficient coding protocol is to minimize the expected code length of communicating $x$. To explain Bits-Back Coding, let’s first consider a more naive coding scheme. VAE can be seen as a way to encode data in a two-part code: $p(z)$ and $p(x|z)$, where $z$ can be seen as the essence/structure of a datum and is encoded first and then the modeling error (deviation from $z$’s structure) is encoded next. The expected code length under this naive coding scheme for a given data distribution is hence:

$$C_{\text{naive}}(x) = \mathbb{E}_{x \sim \text{data}, z \sim q(z|x)} \left[ -\log p(z) - \log p(x|z) \right]$$ (5)

This coding scheme is, however, inefficient. Bits-Back Coding improves on it by noticing that the encoder distribution $q(z|x)$ can be used to transmit additional information, up to $H(q(z|x))$ expected nats, as long as the receiver also has access to $q(z|x)$. The decoding scheme works as follows: a receiver first decodes $z$ from $p(z)$, then decodes $x$ from $p(x|z)$ and, by running the same approximate posterior that the sender is using, decodes a secondary message from $q(z|x)$. Hence, to properly measure the code length of VAE’s two-part code, we need to subtract the extra information from $q(z|x)$. Using Bit-Back Coding, the expected code length equates to the negative variational lower bound or the so-called Helmholtz variational free energy, which means minimizing code length is equivalent to maximizing the variational lower bound:

$$C_{\text{BitsBack}}(x) = \mathbb{E}_{x \sim \text{data}, z \sim q(z|x)} \left[ \log q(z|x) - \log p(z) - \log p(x|z) \right]$$ (6)

$$= \mathbb{E}_{x \sim \text{data}} [-\mathcal{L}(x)]$$ (7)

Casting the problem of optimizing VAE into designing an efficient coding scheme easily allows us to reason when the latent code $z$ will be used: the latent code $z$ will be used when the two-part code is an efficient code. Recalling that the lower-bound of expected code length for data is given by the Shannon entropy of data generation distribution: $\mathcal{H}(\text{data}) = \mathbb{E}_{x \sim \text{data}} \left[ -\log p_{\text{data}}(x) \right]$, we can analyze VAE’s coding efficiency:

$$C_{\text{BitsBack}}(x) = \mathbb{E}_{x \sim \text{data}, z \sim q(z|x)} \left[ \log q(z|x) - \log p(z) - \log p(x|z) \right]$$ (8)

$$= \mathbb{E}_{x \sim \text{data}} \left[ -\log p(x) + D_{KL}(q(z|x)||p(z|x)) \right]$$ (9)

$$\geq \mathbb{E}_{x \sim \text{data}} \left[ -\log p_{\text{data}}(x) + D_{KL}(q(z|x)||p(z|x)) \right]$$ (10)

$$= \mathcal{H}(\text{data}) + \mathbb{E}_{x \sim \text{data}} \left[ D_{KL}(q(z|x)||p(z|x)) \right]$$ (11)
Another argument...

- What’s the maximum ELBO?

$$E_{x \sim p_{\text{data}}(x)}[ELBO] \leq E_{x \sim p_{\text{data}}(x)}[\log p_\theta(x)] \leq E_{x \sim p_{\text{data}}(x)}[\log p_{\text{data}}(x)]$$

- What if $p(x|z) = p_{\text{data}}(x)$?

$$E_{x \sim p_{\text{data}}} [ELBO] = E_{x \sim p_{\text{data}}, z \sim q}[\log p(x|z) + \log p(z) - \log q(z|x)]$$
$$= E_x [\log p_{\text{data}}(x) + E_z[\log p(z) - \log q(z|x)]]$$
$$= E_x [\log p_{\text{data}}(x) - KL(q(z|x)||p(z))]$$

- $q(z|x)$ will be set to $p(z)$; $z$ contains no information
Here’s a seemingly silly idea: let’s try to encode a single bit of information with a variational autoencoder (VAE). Our data set thus consists of two i.i.d. samples. In fact, here’s what it looks like:

```python
data = np.array([[0.],
                 [1.]])
```

We will attempt to autoencoder this data using a variational autoencoder with a single-dimensional $z$ (after all, one dimension should be sufficient), where $p(z)$ is unit Gaussian, $p(x \mid z)$ is Bernoulli, and $q(z \mid x)$ is a conditional Gaussian—a standard formulation of the VAE.
Hence there exists an information preference when a VAE is optimized:

- Information that can be modelled locally by $p(x|z)$ without access to $z$ will be encoded locally and only the remainder will be encoded in $z$

This property can be exploited to give us fine-grained control over the kind of information included in the learned representation:

- Construct a decoder which is capable of modelling the part of the information we don’t want the latent code to capture
• Example: want a global representation for images that doesn’t encode local information like textures

• Use a PixelCNN with limited receptive field, i.e.

\[ p_{\text{local}} (x|z) = \prod_i p \left( x_i|z, x_{\text{WindowAround} (i)} \right) \]

• As long as \( x_{\text{WindowAround} (i)} \) is smaller than \( x_{<i} \), \( p_{\text{local}} (x|z) \) won’t be able to model \( p_{\text{data}}(x) \) without dependence on \( z \)
Also: Learned Prior

- Additionally, the paper introduces learned priors using autoregressive flows
- Repeatedly transform spherical Gaussian noise source with invertible parameterized functions
- Show equivalence to a more expressive approximate posterior
\[ \mathbb{E} \left[ D_{KL}(q(z|x)\|p(z)) \right] \] (number of bits used to encode an image on average): 19.2 bits for VLAЕ, 37.3 bits for VAE
(a) Original test-set images (left) and “decompressed” versions from VLAE’s lossy code (right)
### Table 1: Statically Binarized MNIST

<table>
<thead>
<tr>
<th>Model</th>
<th>NLL Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalizing flows (Rezende &amp; Mohamed, 2015)</td>
<td>85.10</td>
</tr>
<tr>
<td>DRAW (Gregor et al., 2015)</td>
<td>&lt; 80.97</td>
</tr>
<tr>
<td>Discrete VAE (Rolfe, 2016)</td>
<td>81.01</td>
</tr>
<tr>
<td>PixelRNN (van den Oord et al., 2016a)</td>
<td>79.20</td>
</tr>
<tr>
<td>IAF VAE (Kingma et al., 2016)</td>
<td>79.88</td>
</tr>
<tr>
<td>AF VAE</td>
<td>79.30</td>
</tr>
<tr>
<td>VLAE</td>
<td><strong>79.03</strong></td>
</tr>
</tbody>
</table>
### Table 2: Dynamically binarized MNIST

<table>
<thead>
<tr>
<th>Model</th>
<th>NLL Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional VAE + HVI (Salimans et al., 2014)</td>
<td>81.94</td>
</tr>
<tr>
<td>DLGM 2hl + IWAE (Burda et al., 2015a)</td>
<td>82.90</td>
</tr>
<tr>
<td>Discrete VAE (Rolfe, 2016)</td>
<td>80.04</td>
</tr>
<tr>
<td>LVAE (Kaae Sønderby et al., 2016)</td>
<td>81.74</td>
</tr>
<tr>
<td>DRAW + VGP (Tran et al., 2015)</td>
<td>&lt; 79.88</td>
</tr>
<tr>
<td>IAF VAE (Kingma et al., 2016)</td>
<td>79.10</td>
</tr>
<tr>
<td>Unconditional Decoder</td>
<td>87.55</td>
</tr>
<tr>
<td>VLAE</td>
<td>78.53</td>
</tr>
</tbody>
</table>

### Table 3: OMNIGLOT. [1] (Burda et al., 2015a), [2] (Burda et al., 2015b), [3] (Gregor et al., 2015), [4] (Gregor et al., 2016),

<table>
<thead>
<tr>
<th>Model</th>
<th>NLL Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE [1]</td>
<td>106.31</td>
</tr>
<tr>
<td>IWAE [1]</td>
<td>103.38</td>
</tr>
<tr>
<td>RBM (500 hidden) [2]</td>
<td>100.46</td>
</tr>
<tr>
<td>DRAW [3]</td>
<td>&lt; 96.50</td>
</tr>
<tr>
<td>Conv DRAW [4]</td>
<td>&lt; 91.00</td>
</tr>
<tr>
<td>Unconditional Decoder</td>
<td>95.02</td>
</tr>
<tr>
<td>VLAE</td>
<td>90.98</td>
</tr>
<tr>
<td>VLAE (fine-tuned)</td>
<td><strong>89.83</strong></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Model</th>
<th>NLL Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWS SBN [1]</td>
<td>113.3</td>
</tr>
<tr>
<td>RBM [2]</td>
<td>107.8</td>
</tr>
<tr>
<td>NAIS NADE [3]</td>
<td>100.0</td>
</tr>
<tr>
<td>Discrete VAE [4]</td>
<td>97.6</td>
</tr>
<tr>
<td>SpARN [5]</td>
<td>88.48</td>
</tr>
<tr>
<td>Unconditional Decoder</td>
<td>89.26</td>
</tr>
<tr>
<td>VLAE</td>
<td><strong>77.36</strong></td>
</tr>
</tbody>
</table>
Results: CIFAR10

(a) 4x2

(b) 5x3

(c) 7x4

(d) 7x4 Grayscale
## Results: CIFAR10

<table>
<thead>
<tr>
<th>Method</th>
<th>bits/dim ≤</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Results with tractable likelihood models:</strong></td>
<td></td>
</tr>
<tr>
<td>Uniform distribution [1]</td>
<td>8.00</td>
</tr>
<tr>
<td>Multivariate Gaussian [1]</td>
<td>4.70</td>
</tr>
<tr>
<td>NICE [2]</td>
<td>4.48</td>
</tr>
<tr>
<td>Deep GMMs [3]</td>
<td>4.00</td>
</tr>
<tr>
<td>Real NVP [4]</td>
<td>3.49</td>
</tr>
<tr>
<td>PixelCNN [1]</td>
<td>3.14</td>
</tr>
<tr>
<td>Gated PixelCNN [5]</td>
<td>3.03</td>
</tr>
<tr>
<td>PixelRNN [1]</td>
<td>3.00</td>
</tr>
<tr>
<td>PixelCNN++ [6]</td>
<td><strong>2.92</strong></td>
</tr>
<tr>
<td><strong>Results with variationally trained latent-variable models:</strong></td>
<td></td>
</tr>
<tr>
<td>Deep Diffusion [7]</td>
<td>5.40</td>
</tr>
<tr>
<td>Convolutional DRAW [8]</td>
<td>3.58</td>
</tr>
<tr>
<td>ResNet VAE with IAF [9]</td>
<td>3.11</td>
</tr>
<tr>
<td>ResNet VLAE</td>
<td>3.04</td>
</tr>
<tr>
<td>DenseNet VLAE</td>
<td><strong>2.95</strong></td>
</tr>
</tbody>
</table>
Conclusion

- Analyzed the condition under which the latent code in VAEs is used
- Through carefully designing decoder network, able to control what sort of information is stored in latent representations
- Proposed two complementary improvements to VAE architecture shown to have strong performance empirically
Thanks 😊
Learning the prior/posterior