Variational Autoencoders (VAE) and β-VAE: Theory, Training, and Latent Space Analysis

Prepared for: P4—Training a VAE on Face Attributes

August 28, 2025

Abstract

This note provides a compact yet comprehensive derivation and explanation of Variational Autoencoders (VAE) and the β-VAE variant. We derive the Evidence Lower Bound (ELBO), explain the reparameterization trick, give closed-form losses used in code, describe how the latent space is formed, and how to generate new samples. We also outline practical procedures to identify latent dimensions controlling facial attributes (e.g., smiling, eye openness, hairstyle) and to produce controlled traversals. The content is designed to map directly to typical PyTorch implementations (like the code provided) for 64×64 RGB faces.

Contents

1 The VAE Architecture in Code

The PyTorch class implementing the VAE is:

```
class VAE(nn.Module):
def __init__(self, latent_dim=32):
super(VAE, self).__init__()
self.latent_dim = latent_dim


# Encoder
self.enc = nn.Sequential(
nn.Conv2d(3, 32, 4, 2, 1), # -> 32x32x32
nn.ReLU(),
nn.Conv2d(32, 64, 4, 2, 1), # -> 64x16x16
nn.ReLU(),
nn.Conv2d(64, 128, 4, 2, 1),# -> 128x8x8
nn.ReLU(),
nn.Flatten()
)
self.fc_mu = nn.Linear(128*8*8, latent_dim)
self.fc_logvar = nn.Linear(128*8*8, latent_dim)


# Decoder
self.fc = nn.Linear(latent_dim, 128*8*8)
self.dec = nn.Sequential(
nn.ConvTranspose2d(128, 64, 4, 2, 1),
nn.ReLU(),
nn.ConvTranspose2d(64, 32, 4, 2, 1),
nn.ReLU(),
nn.ConvTranspose2d(32, 3, 4, 2, 1),
nn.Sigmoid()
)
```

1.1 Encoder

The encoder maps an input RGB image $x \in \mathbb{R}^{3\times64\times64}$ through three convolutional layers:

- Conv2d(3,32, kernel=4, stride=2, padding=1) $\Rightarrow$ feature map of size $32 \times 32 \times 32$.
- Conv2d(32,64, kernel=4, stride=2, padding=1) $\Rightarrow$ $64 \times 16 \times 16$.
- Conv2d(64,128, kernel=4, stride=2, padding=1) $\Rightarrow$ $128 \times 8 \times 8$.

Flattening gives a 8192-dimensional vector, passed to two linear layers that output μ and $\log \sigma^2$ of the latent Gaussian distribution.

1.2 Decoder and ConvTranspose2d

The decoder reverses this process. First, a linear layer maps the latent vector $z \in \mathbb{R}^{32}$ to a $128 \times 8 \times 8$ tensor. Then three transposed convolutions gradually upsample:

- ConvTranspose2d(128,64, kernel=4, stride=2, padding=1) $\Rightarrow$ $64 \times 16 \times 16$.

- ConvTranspose2d(64,32, kernel=4, stride=2, padding=1) $\Rightarrow 32 \times 32 \times 32$.
- ConvTranspose2d(32,3, kernel=4, stride=2, padding=1) $\Rightarrow 3 \times 64 \times 64$.

How ConvTranspose2d Works. The transposed convolution (sometimes called "deconvolution") is not the true inverse of a convolution, but it performs a learnable upsampling. For stride $s = 2$, the operator inserts $(s-1)$ zeros between input elements and then applies a normal convolution with the given kernel. With appropriate padding, this doubles spatial resolution. In effect, the decoder learns filters that expand coarse feature maps back into high-resolution RGB images.

Final Activation. The last layer uses a `Sigmoid` to map outputs to $[0, 1]$, representing pixel intensities.

2 The Probabilistic Model

Generative Assumptions. A VAE posits latent variables $\boldsymbol{z} \in \mathbb{R}^d$ and a likelihood $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ such that data $\boldsymbol{x} \in \mathbb{R}^D$ is generated via

$$\boldsymbol{z} \sim p(\boldsymbol{z}) \quad \text{(prior, typically } \mathcal{N}(\mathbf{0}, \boldsymbol{I})\text{)}, \ \boldsymbol{x} \qquad \sim p_\theta(\boldsymbol{x} \mid \boldsymbol{z}). \tag{1}$$

Here $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ is a decoder network that maps $\boldsymbol{z}$ to the parameters of a simple distribution over $\boldsymbol{x}$ (e.g., Bernoulli for normalized pixels, or Gaussian with fixed variance).

Intractable Posterior. Exact Bayesian inference requires $p_\theta(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p_\theta(\boldsymbol{x})}$, but the marginal likelihood $p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$ is intractable for neural decoders.

3 Variational Inference and the ELBO

Amortized Variational Family. Introduce an encoder $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ (a recognition network) as a variational approximation to the true posterior. For VAEs, we commonly use a diagonal Gaussian

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}\,;\, \boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\boldsymbol{x}))). \tag{2}$$

Deriving the ELBO. Starting from the log-evidence:

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \log \int p_\theta(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z} && (3)\\
&= \log \int q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\, \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\, d\boldsymbol{z} && (4)\\
&\geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right] \quad \text{(Jensen)} && (5)\\
&= \mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})] - \mathrm{KL}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p(\boldsymbol{z})\big) \ \triangleq \ \mathcal{L}_{\mathrm{ELBO}}(\theta, \phi; \boldsymbol{x}). && (6)
\end{aligned}$$

Maximizing the ELBO w.r.t. (θ, ϕ) is equivalent to minimizing $\mathrm{KL}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z} \mid \boldsymbol{x})\big)$ and provides a lower bound to the log-evidence.

Reconstruction and Regularization. The ELBO separates into a data term (expected reconstruction log-likelihood) and a regularizer forcing the approximate posterior towards the prior.

4 The Reparameterization Trick

The expectation over $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ in Eq. (6) depends on ϕ. To enable low-variance gradients, we rewrite sampling as a deterministic function of a parameter-free noise variable:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{7}$$

$$\boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}. \tag{8}$$

Now $\nabla_\phi \mathbb{E}_{\boldsymbol{\epsilon}}[f\big(g_\phi(\boldsymbol{\epsilon})\big)] = \mathbb{E}_{\boldsymbol{\epsilon}}[\nabla_\phi f\big(g_\phi(\boldsymbol{\epsilon})\big)]$, enabling backpropagation through stochastic nodes.

5 Closed-Form Terms Used in Practice

5.1 Gaussian Prior and Diagonal-Gaussian Posterior

With $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$, the KL term has a closed form:

$$\mathrm{KL}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p(\boldsymbol{z})\big) = \tfrac{1}{2} \sum_{j=1}^{d} \Big(\mu_j^2 + \sigma_j^2 - 1 - \log \sigma_j^2\Big). \tag{9}$$

In code, we often predict $\log \sigma^2$ (`logvar`) and compute Eq. (9) stably.

5.2 Likelihood Choices for Images

For images scaled to $[0, 1]$ per pixel, two common choices are:

- **Bernoulli (per pixel):** $p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \mathrm{Bernoulli}(\boldsymbol{x}\,;\, \boldsymbol{\pi}_\theta(\boldsymbol{z}))$ with logits from the decoder. The reconstruction term is the negative binary cross-entropy (BCE).
- **Gaussian (per pixel):** $p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}\,;\, \boldsymbol{\mu}_\theta(\boldsymbol{z}), \sigma_x^2 \boldsymbol{I})$. With fixed σ_x^2, maximizing the log-likelihood is (up to constants) minimizing MSE.

The provided code uses a `Sigmoid` output and an MSE loss (`F.mse_loss`), corresponding to a Gaussian likelihood with fixed variance.

6 Training Objective (Minimization Form)

Given a minibatch $\{\boldsymbol{x}^{(i)}\}_{i=1}^{B}$, a standard loss minimized in code is

$$\mathcal{J}(\theta, \phi) = -\frac{1}{B} \sum_{i=1}^{B} \mathcal{L}_{\mathrm{ELBO}}(\theta, \phi; \boldsymbol{x}^{(i)}) \tag{10}$$

$$= \underbrace{\frac{1}{B} \sum_{i} \ell_{\mathrm{recon}}\big(\boldsymbol{x}^{(i)},\, \hat{\boldsymbol{x}}^{(i)}\big)}_{\text{reconstruction term}} + \underbrace{\frac{1}{B} \sum_{i} \mathrm{KL}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}^{(i)}) \| p(\boldsymbol{z})\big)}_{\text{regularizer}}, \tag{11}$$

where $\hat{\boldsymbol{x}} =_\theta (\boldsymbol{z})$ with $\boldsymbol{z}$ obtained via Eq. (8). In the code, losses are accumulated as sum over pixels and then normalized by batch size.

7 β-VAE

7.1 Objective

β-VAE [3] modifies the ELBO by up-weighting the KL term:

$$\mathcal{L}_{\beta\text{-VAE}}(\theta, \phi; \boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})] - \beta\, \mathrm{KL}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p(\boldsymbol{z})\big), \quad \beta \geq 1. \tag{12}$$

Equivalently, minimize reconstruction loss + $\beta\times$ KL. Larger β encourages disentanglement by pushing $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ closer to an axis-aligned prior, often at the cost of reconstruction fidelity.

7.2 Capacity Control (Optional Variant)

A useful training schedule replaces $\beta\,\mathrm{KL}$ with $\beta\,|\mathrm{KL} - C|$, gradually increasing the target capacity C from 0 to a desired value. This allows the model to first learn simple factors before modeling fine detail.

8 How the Latent Space is Formed

Per-Example Posteriors. For each input $\boldsymbol{x}$, the encoder outputs $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\boldsymbol{x})))$. These Gaussians populate latent space near their means, with spread set by $\boldsymbol{\sigma}_\phi(\boldsymbol{x})$.

Aggregate Posterior. Over the dataset, the *aggregate* posterior is $q(\boldsymbol{z}) = \int q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\, p_{\mathrm{data}}(\boldsymbol{x})\, d\boldsymbol{x}$. The KL term regularizes each $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ towards the prior $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, encouraging a globally smooth, roughly isotropic latent space with local linearity.

Disentanglement Pressure. With $\beta > 1$ or capacity control, the model is penalized for using correlated or overly informative latents. This often yields axes that align with independent generative factors (pose, expression, hairstyle), though perfect identifiability requires additional assumptions and is not guaranteed.

9 Generating New Samples

To sample a novel image:

1. Draw $\tilde{\boldsymbol{z}} \sim p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
2. Compute $\hat{\boldsymbol{x}} =_\theta (\tilde{\boldsymbol{z}})$.
3. If the decoder outputs parameters (e.g., Bernoulli probabilities), optionally sample a pixel-level draw. In practice, we often use the mean $\hat{\boldsymbol{x}}$ as the generated image.

This matches the top-down generative story in Eq. (1).

10 Latent Traversals and Attribute Control

10.1 Single-Dimension Traversal

Fix a base latent $\boldsymbol{z}_0$ (sampled from $\mathcal{N}(0, I)$ or encoded from an image). For a chosen index k, create a sequence $\boldsymbol{z}(t) = \boldsymbol{z}_0 + t\,\boldsymbol{e}_k$ with $t \in [-\tau, \tau]$. Decode each $\boldsymbol{z}(t)$ to visualize how attribute(s) change along axis k.

10.2 Discovering Attribute Axes

To identify dimensions controlling smiling, eye openness, and hairstyle:

1. **Supervised linear probes (optional):** If you have attribute labels (CelebA has 40), fit a ridge/logistic regression from $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ to each attribute; inspect the largest-weight dimensions.
2. **Unsupervised correlation:** Compute the correlation between each latent dimension z_j (posterior means) and simple image-derived statistics (e.g., mouth aspect ratio for smiling, eye-opening detectors). Rank by absolute correlation.
3. **Interventional check:** For top candidates, perform 1D traversals while freezing others and visually verify monotonic attribute change.

4. **Multi-dim subspace (if needed):** If an attribute loads on multiple z_j, traverse along the learned probe vector $\boldsymbol{w}$ (normalize $\boldsymbol{w}$) via $\boldsymbol{z}(t) = \boldsymbol{z}_0 + t\,\boldsymbol{w}/\|\boldsymbol{w}\|$.

10.3 Practical Traversal Settings

Use evenly spaced values $t \in \{-\tau, \ldots, \tau\}$ with $\tau \in [2, 3]$ in standard deviations (as in the provided code using $[-3, 3]$). Decode and tile results to create before/after grids.

11 Implementation Notes Mapping to Code

- **Encoder/Decoder:** The code uses strided convs down to a 8×8 feature map with 128 channels, flattened into a vector for two linear heads $\boldsymbol{\mu}$ and `logvar`. The decoder mirrors this with transposed convs.
- **Parameterization:** Predicting `logvar` ensures positivity of variance via $\sigma = \exp\big(\frac{1}{2}\,\texttt{logvar}\big)$; sampling uses Eq. (8).
- **Loss:** The function `vae_loss` computes MSE recon (Gaussian with fixed variance) plus the Gaussian KL in Eq. (9). Losses are summed over pixels then divided by batch size (equivalent to per-sample average).
- **Checkpoints & Grids:** After each epoch, a small batch is reconstructed and saved in a grid; this empirically tracks training progress.
- **Batch Size/Optimizer:** Adam with $\eta = 10^{-3}$ is a standard baseline. Typical latent sizes for 64×64 faces are $d \in \{16, 32, 64\}$.

12 Why β-VAE Encourages Disentanglement (Intuition)

Increasing β penalizes mutual information between $\boldsymbol{x}$ and $\boldsymbol{z}$, pushing the model to represent only coarse, high-variance factors that are easiest to reconstruct. This often yields axes aligned with independent, semantically meaningful factors. However, disentanglement is not guaranteed in full generality without additional inductive biases.

13 Evaluation and Diagnostics

- **Reconstruction vs. KL curve:** Track both. Excessively low KL ("posterior collapse") indicates the decoder ignores $\boldsymbol{z}$; excessively high KL with poor recon means underfitting.
- **FID/LPIPS (optional):** For sample quality and diversity.
- **Disentanglement metrics (optional):** MIG/DCI/FactorVAE score if ground-truth factors or surrogates are available.

14 Common Variants (Brief)

- **KL annealing:** Start with small β (or small target C) and ramp up.
- **Learned variance decoders:** Predict per-pixel σ_x^2 for heteroscedastic likelihoods.
- **Normalizing-flow posteriors:** Replace diagonal Gaussian q_ϕ with flows for richer inference while keeping reparameterization.

15 End-to-End Algorithm (Pseudocode)

For a minibatch $\{\boldsymbol{x}^{(i)}\}_{i=1}^{B}$:

1. Encode: $(\boldsymbol{\mu}^{(i)}, \log \boldsymbol{\sigma}^{2\,(i)}) =_{\phi} (\boldsymbol{x}^{(i)})$.
2. Sample: draw $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and set $\boldsymbol{z}^{(i)} = \boldsymbol{\mu}^{(i)} + \exp\big(\frac{1}{2} \log \boldsymbol{\sigma}^{2\,(i)}\big) \odot \boldsymbol{\epsilon}^{(i)}$.
3. Decode: $\hat{\boldsymbol{x}}^{(i)} =_{\theta} (\boldsymbol{z}^{(i)})$.
4. Compute losses: $\ell_{\text{recon}}(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{x}}^{(i)})$ (BCE or MSE) and $\mathrm{KL}(q\|p)$ via Eq. (9).
5. Update θ, ϕ by minimizing $\frac{1}{B} \sum_i \ell_{\text{recon}} + \beta\, \mathrm{KL}$ (with $\beta = 1$ for standard VAE).

16 From Latent Space to New Images: A Worked Example

Assume $d = 32$ and a trained model. To generate a series illustrating "smiling":

1. Pick an anchor face $\boldsymbol{x}_0$ and compute $\boldsymbol{\mu}_0 = \boldsymbol{\mu}_{\phi}(\boldsymbol{x}_0)$.
2. Choose a dimension k correlated with the "Smiling" attribute (via probes/correlation).
3. Evaluate $\boldsymbol{z}(t) = \boldsymbol{\mu}_0 + t\, \boldsymbol{e}_k$ for $t \in \{-3, -2, \ldots, 3\}$.
4. Decode $\hat{\boldsymbol{x}}(t) =_{\theta} (\boldsymbol{z}(t))$ and tile the images left-to-right to visualize the change.

Repeat for eye openness and hairstyle; if multi-dimensional, replace $\boldsymbol{e}_k$ with a normalized direction vector $\boldsymbol{w}$ from the linear probe.

Acknowledgments and References

References

[1] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes*. arXiv:1312.6114, 2013.

[2] Danilo Rezende, Shakir Mohamed, and Daan Wierstra. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models*. ICML, 2014.

[3] Irina Higgins et al. *beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework*. ICLR, 2017.