

TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2023

Variational Auto-Encoders (VAEs)

Generative AI: Autoregression and GANs

A discrete autoregressive model can both generate samples and compute probabilities.

This allows one to train a generative model by cross-entropy loss.

But it is not obvious how to train a generative model of images by cross-entropy loss.

The point of a VAE is to replace the adversarial loss of a GAN with cross-entropy loss.

Generative AI for Continuous Data: VAEs

A variational autoencoder (VAE) is defined by three parts:

- An encoder distribution $P_{\text{enc}}(z|y)$.
- A “prior” distribution $P_{\text{pri}}(z)$
- A generator distribution $P_{\text{gen}}(y|z)$

VAE generation uses $P_{\text{pri}}(z)$ and $P_{\text{gen}}(y|z)$ (like a GAN).

VAE training uses a “GAN inverter” $P_{\text{enc}}(z|y)$.

Fixed Encoder Training

$$\text{pri}^*, \text{gen}^* = \underset{\text{pri}, \text{gen}}{\operatorname{argmin}} E_{y \sim \text{Pop}(y), z \sim \text{enc}(z|y)} \left[-\ln P_{\text{pri}}(z) P_{\text{gen}}(y|z) \right]$$

This is cross-entropy loss from $\text{Pop}(y) P_{\text{enc}}(z|y)$ to $P_{\text{pri}}(z) P_{\text{gen}}(y|z)$

Universality gives

$$P_{\text{pri}^*}(z) P_{\text{gen}^*}(y|z) = \text{Pop}(y) P_{\text{enc}}(y|z)$$

Hence sampling from $P_{\text{pri}^*}(z) P_{\text{gen}^*}(y|z)$ samples y from the population.

Training the Encoder (the GAN Inverter)

Define the ELBO loss as follows (acronym described later).

$$\mathcal{L}(y, z) = -\ln \frac{P_{\text{pri}}(z)P_{\text{gen}}(y|z)}{P_{\text{enc}}(z|y)}$$

$$\text{enc}^*, \text{pri}^*, \text{gen}^* = \underset{\text{enc}, \text{pri}, \text{gen}}{\text{argmin}} \ E_{y \sim \text{Pop}, z \sim P_{\text{enc}}(z|y)} \mathcal{L}(y, z)$$

For any encoder, universality gives

$$P_{\text{pri}^*}(z)P_{\text{gen}^*}(y|z) = \text{Pop}(y)P_{\text{enc}}(y|z)$$

Degrees of Freedom

$$P_{\text{pri}}(z)P_{\text{gen}}(y|z) = P_{\text{op}}(y)P_{\text{enc}}(z|y)$$

Any joint distribution on (y, z) with the desired marginal on y optimizes the bound.

However, if we fix the architecture of $P_{\text{pri}}(z)P_{\text{gen}}(y|z)$ to be StyleGAN we can simultaneously train the parameters of the StyleGAN generator and the StyleGAN inverter by cross-entropy loss.

Bayesian Interpretation

VAEs were originally motivated by a Bayesian interpretation:

- $P_{\text{pri}}(z)$ is the Bayesian prior on hypothesis z .
- $P_{\text{gen}}(y|z)$ is the probability of the “evidence” y given hypothesis z .
- $P_{\text{enc}}(z|y)$ is a model approximating the Bayesian posterior on hypothesis z given evidence y .

The Bayesian motivation is to train $P_{\text{enc}}(z|y)$ to approximate Bayesian inference.

Bayesian Interpretation

$$\begin{aligned}\ln P_{\text{pri,gen}}(y) &= \ln \frac{P_{\text{pri,gen}}(y)P_{\text{pri,gen}}(z|y)}{P_{\text{pri,enc}}(z|y)} \\ &= E_{z \sim P_{\text{enc}}(z|y)} \left[\ln \frac{P_{\text{pri,gen}}(y)P_{\text{pri,gen}}(z|y)}{P_{\text{enc}}(z|y)} \right] + KL(P_{\text{enc}}(z|y), P_{\text{pri,gen}}(z|y)) \\ &\geq E_{z \sim P_{\text{enc}}(z|y)} \left[\ln \frac{P_{\text{pri,gen}}(y)P_{\text{pri,gen}}(z|y)}{P_{\text{enc}}(z|y)} \right]\end{aligned}$$

A Bayesian thinks of y as “evidence” for hypothesis z .

$E_{z \sim P_{\text{enc}}(z|y)}[-\mathcal{L}(y, z)]$ is called the evidence lower bound (ELBO).

Posterior Collapse

Under the Bayesian interpretation we would like z to provide useful information about (a causal origin of) y .

However the objective function only produces

$$P_{\text{pri}}(z)P_{\text{gen}}(y|z) = \text{Pop}(y)P_{\text{enc}}(z|y)$$

For language models the generator can assign a meaningful probability to a block of text y independent of z .

When we train a sentence encoder (a thought vector) as the latent variable of a language model VAE we can get a constant (zero) thought vector.

This is called “posterior collapse”.

The Reparameterization Trick

$$\text{enc}^* = \underset{\text{enc}}{\operatorname{argmin}} E_{y \sim \text{Pop}(y), z \sim P_{\text{enc}}(z|y)} \left[-\ln \frac{P_{\text{pri}}(z) P_{\text{gen}}(y|z)}{P_{\text{enc}}(z|y)} \right]$$

Gradient descent on the encoder parameters must take into account the fact that we are sampling from the encoder.

To handle this we sample noise ϵ from a fixed noise distribution and replace z with a deterministic function $z_{\text{enc}}(y, \epsilon)$

$$\text{enc}^*, \text{pri}^*, \text{gen}^* = \underset{\text{enc}, \text{pri}, \text{gen}}{\operatorname{argmin}} E_{y, \epsilon, z = \hat{z}_{\text{enc}}(y, \epsilon)} \left[-\ln \frac{P_{\text{pri}}(z) P_{\text{gen}}(y|z)}{P_{\text{enc}}(z|y)} \right]$$

The Reparameterization Trick

$$\text{enc}^*, \text{pri}^*, \text{gen}^* = \underset{\text{enc}, \text{pri}, \text{gen}}{\text{argmin}} \quad E_{y, \epsilon, z = \hat{z}_{\text{enc}}(y, \epsilon)} \left[-\ln \frac{P_{\text{pri}}(z) P_{\text{gen}}(y|z)}{P_{\text{enc}}(z|y)} \right]$$

To get gradients we must have that $\hat{z}_{\text{enc}}(y, \epsilon)$ is a differentiable function of the encoder parameters.

Optimizing the encoder is tricky for discrete z . Discrete z is handled effectively in EM algorithms and general vector quantization (VQ) methods.

The KL-divergence Optimization

$$\begin{aligned}\mathcal{L}(y) &= E_{z \sim P_{\text{enc}}(z|y)} \left[-\ln \frac{P_{\text{pri}}(z) P_{\text{gen}}(y|z)}{P_{\text{enc}}(z|y)} \right] \\ &= \textcolor{red}{KL}(P_{\text{enc}}(z|y), P_{\text{pri}}(z)) + E_{z \sim P_{\text{enc}}(z|y)} [-\ln P_{\text{gen}}(y|z)] \\ &= \frac{\textcolor{red}{||\hat{z}_{\text{enc}}(y) - \hat{z}_{\text{pri}}||^2}}{2\sigma^2} + E_{\epsilon} \frac{||y - \hat{y}_{\text{gen}}(\hat{z}_{\text{enc}}(y, \epsilon))||^2}{2\sigma^2}\end{aligned}$$

A closed-form expression for the KL term avoids sampling noise.

EM is Alternating Optimization of the ELBO Loss

Expectation Maximimization (EM) applies in the (highly special) case where the exact posterior $P_{\text{pri,gen}}(z|y)$ is samplable and computable. EM alternates exact optimization of enc and the pair (pri, gen) in:

$$\text{VAE: } \text{pri}^*, \text{gen}^* = \underset{\text{pri,gen}}{\operatorname{argmin}} \min_{\text{enc}} E_{y, z \sim P_{\text{enc}}(z|y)} - \ln \frac{P_{\text{pri,gen}}(z, y)}{P_{\text{enc}}(z|y)}$$

$$\text{EM: } \text{pri}^{t+1}, \text{gen}^{t+1} = \underset{\text{pri,gen}}{\operatorname{argmin}} E_{y, z \sim P_{\text{pri}^t, \text{gen}^t}(z|y)} - \ln P_{\text{pri,gen}}(z, y)$$

Inference
(E Step)

$$P_{\text{enc}}(z|y) = P_{\text{pri}^t, \text{gen}^t}(z|y)$$

Update
(M Step)

Hold $P_{\text{enc}}(z|y)$ fixed

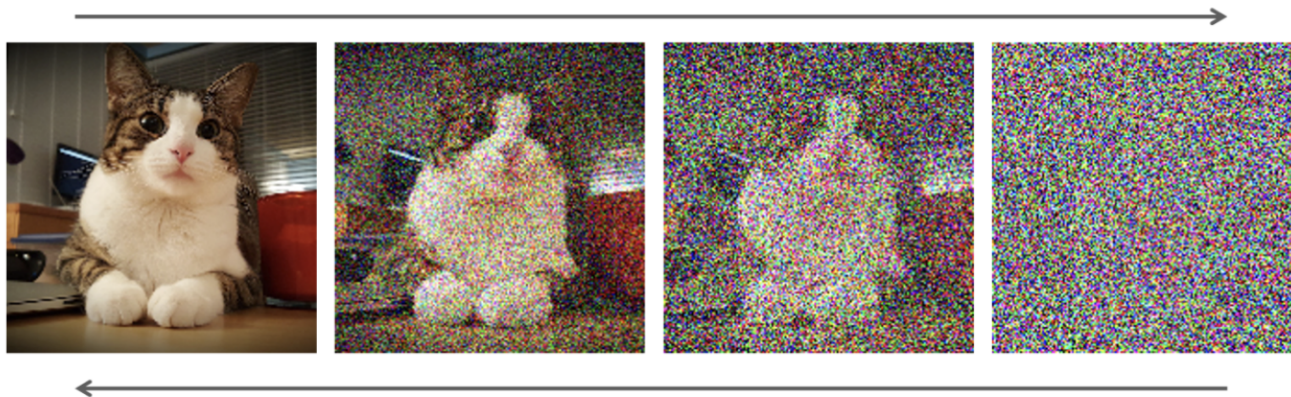
Generative AI for Continuous Data: Flow Models

$$\text{gen}^* = \underset{\text{gen}}{\operatorname{argmin}} E_{y \sim \text{pop}(y)} - \ln p_{\text{gen}}(y)$$

Flow-based generative models work with Jacobians over continuous transformations (no ReLUs) and can be directly trained with cross-entropy loss.

But flow models have not caught on and we will not cover them.

Markovian VAEs



[Sally talked to John] $\overset{\rightarrow}{\leftarrow}$ [Sally talked to] $\overset{\rightarrow}{\leftarrow}$ [Sally talked] $\overset{\rightarrow}{\leftarrow}$ [Sally] $\overset{\rightarrow}{\leftarrow}$ []

$$y \overset{\rightarrow}{\leftarrow} z_1 \overset{\rightarrow}{\leftarrow} \dots \overset{\rightarrow}{\leftarrow} z_N$$

Markovian VAEs

$$y \overset{\rightarrow}{\leftarrow} z_1 \overset{\rightarrow}{\leftarrow} \dots \overset{\rightarrow}{\leftarrow} z_N$$

Encoder: $\text{Pop}(y)$, $P_{\text{enc}}(z_1|y)$, and $P_{\text{enc}}(z_{\ell+1}|z_\ell)$.

Generator: $P_{\text{pri}}(z_N)$, $P_{\text{gen}}(z_{\ell-1}|z_\ell)$, $P_{\text{gen}}(y|z_1)$.

The encoder and the decoder define distributions $P_{\text{enc}}(y, \dots, z_N)$ and $P_{\text{gen}}(y, \dots, z_N)$ respectively.

Markovian VAEs

$$y \overset{\rightarrow}{\leftarrow} z_1 \overset{\rightarrow}{\leftarrow} \dots \overset{\rightarrow}{\leftarrow} z_N$$

- autoregressive models
- diffusion models
- Byte VAEs (original [here](#))

Diffusion ELBO

$$\begin{aligned}
H(y) &= E_{\text{enc}} \left[-\ln \frac{P_{\text{enc}}(y)P_{\text{enc}}(z_1, \dots, z_N|y)}{P_{\text{enc}}(z_1, \dots, z_N|y)} \right] \\
&= E_{\text{enc}} \left[-\ln \frac{P_{\text{enc}}(y|z_1)P_{\text{enc}}(z_1|z_2) \cdots P_{\text{enc}}(z_{N-1}|z_N)P_{\text{enc}}(z_N)}{P_{\text{enc}}(z_1|z_2, y) \cdots P_{\text{enc}}(z_{N-1}|z_N, y)P_{\text{enc}}(z_N|y)} \right] \\
&\leq E_{\text{enc}} \left[-\ln \frac{P_{\text{gen}}(y|z_1)P_{\text{gen}}(z_1|z_2) \cdots P_{\text{gen}}(z_{N-1}|z_N)P_{\text{gen}}(z_N)}{P_{\text{enc}}(z_1|z_2, y) \cdots P_{\text{enc}}(z_{N-1}|z_N, y)P_{\text{enc}}(z_N|y)} \right] \\
&= \begin{cases} E_{\text{enc}} [-\ln P_{\text{gen}}(y|z_1)] \\ + \sum_{i=2}^N E_{\text{enc}} KL(P_{\text{enc}}(z_{i-1}|z_i, y), P_{\text{gen}}(z_{i-1}|z_i)) \\ + E_{\text{enc}} KL(P_{\text{enc}}(Z_N|y), p_{\text{gen}}(Z_N)) \end{cases}
\end{aligned}$$

Byte VAEs

A byte VAE is a numerical Markovian VAE with a deterministic encoder and where each z_i is a vector of bytes.

To computing $z_{i+1}(z_i)$ we first compute a floating point vector $\hat{z}_i^{\text{enc}}(z_{i-1})$ and then round each coordinate to the nearest byte.

We model $-\ln P_{\text{gen}}(z_i|z_{i+1})$ by $\|\hat{z}_i^{\text{enc}}(z_{i-1}) - \hat{z}_i^{\text{gen}}(z_{i+1})\|^2$.

Although the model is discrete, this gives gradients on both the encoder and the decoder. This can be viewed as a straight-through gradient on the encoder.

The Byte VAE ELBO

$$\begin{aligned} H(y) &\leq E_{y \sim \text{Pop}} \left[-\ln \frac{P_{\text{gen}}(z_N, z_{N-1}, \dots, z_1, y)}{P_{\text{enc}}(z_1, \dots, z_N | y)} \right] \\ &= E_{y \sim \text{Pop}} [-\ln P_{\text{gen}}(z_N, z_{N-1}, \dots, z_1, y)] \\ &\approx E_{y \sim \text{Pop}} \left[\|y - \hat{y}^{\text{gen}}(z_1)\|^2 + \sum_{i=1}^{N-1} \|\hat{z}_i^{\text{enc}} - \hat{z}_i^{\text{gen}}(z_{i+1})\|^2 \right] \end{aligned}$$

END