Stable Diffusion

Parth Pujari

Contents

1 Introduction

Stable Diffusion is a deep learning, text-to-image model released in 2022. It is primarily used to generate detailed images conditioned on text descriptions, though it can also be applied to other tasks such as inpainting, outpainting, and generating image-to-image translations guided by a text prompt.

Stable Diffusion uses a kind of diffusion model (DM), called a latent diffusion model. Diffusion models are trained with the objective of removing successive applications of Gaussian noise on training images, which can be thought of as a sequence of denoising autoencoders.

Stable Diffusion consists of 3 parts: the variational autoencoder (VAE), U-Net, and an optional text encoder. The VAE encoder compresses the image from pixel space to a smaller dimensional latent space, capturing a more fundamental semantic meaning of the image. Gaussian noise is iteratively applied to the compressed latent representation during forward diffusion. The U-Net block, composed of a ResNet backbone, denoises the output from forward diffusion backwards to obtain a latent representation. Finally, the VAE decoder generates the final image by converting the representation back into pixel space.

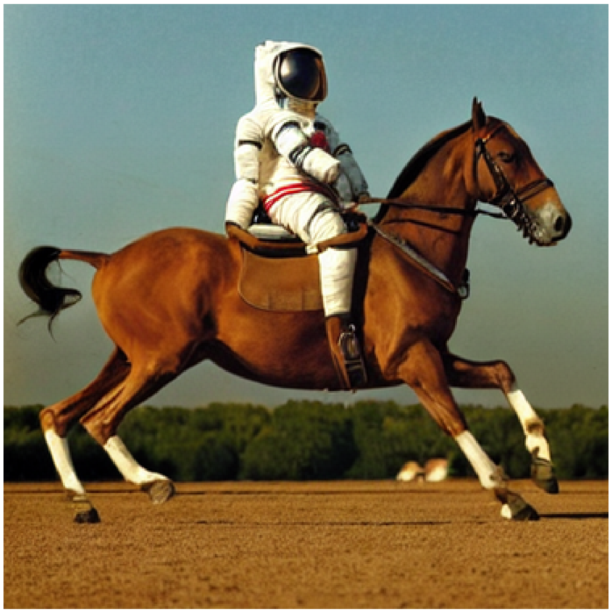

Figure 1: a photograph of an astronaut riding a horse

2 The Autoencoder

2.1 The Variational Autoencoder

A variational autoencoder (VAE), is an artificial neural network architecture belonging to the families of probabilistic graphical models and variational Bayesian methods.These architectures require neural networks as only a part of their overall structure. The neural network components are typically referred to as the encoder and decoder for the first and second component respectively. The first neural network maps the input variable to a latent space that corresponds to the parameters of a variational distribution. In this way, the encoder can produce multiple different samples that all come from the same distribution. The decoder has the opposite function, which is to map from the latent space to the input space, in order to produce or generate data points. Both networks are typically trained together with the usage of the reparameterization trick, although the variance of the noise model can be learned separately.

2.2 Variational Bayes

Bayes Theorem

$$P(Z|X) = \frac{P(X|Z)P(Z)}{P(X)} = \frac{P(X|Z)P(Z)}{\int_Z P(X|Z')\,dZ'} \tag{1}$$

The marginalization over **Z** to calculate **P(X)** is usually intractable. Therefore we seek an approximation $\mathbf{Q}(\mathbf{Z}) \approx \mathbf{P}(\mathbf{Z}|\mathbf{X})$

The distribution Q(Z) is restricted to belong to a family of distributions of simpler form than P(Z|X) (e.g. a family of Gaussian distributions), selected with the intention of making Q(Z) similar to the true posterior.

KL Divergence

The most common type of variational Bayes uses the Kullback–Leibler divergence (KL-divergence) of Q from P as the choice of dissimilarity function. This choice makes this minimization tractable. The KL-divergence is defined as

$$D_{KL}(Q||P) \triangleq \sum_Z Q(Z) log \frac{Q(Z)}{P(Z|X)} \tag{2}$$

Note that Q and P are reversed from what one might expect. This use of reversed KL-divergence is conceptually similar to the expectation-maximization algorithm.

Evidence Lower Bound
Given that,

$$P(Z|X) = \frac{P(Z,X)}{P(X)}$$

We can rewrite the KL Divergence as,

$$D_{KL}(Q||P) \triangleq \sum_Z Q(Z) log \frac{Q(Z)}{P(Z|X)} = \sum_Z Q(Z) \left[log \frac{Q(Z)}{P(Z,X)} + logP(X) \right]$$

$$= \sum_Z Q(Z)[logQ(Z) - logP(Z,X)] + \sum_Z logP(X)$$

Because **P(X)** is constant with respect to **Q(Z)**, $\sum_Z Q(Z) = 1$,we have,

$$D_{KL}(Q||P) = \sum_Z Q(Z)[logQ(Z) - logP(Z,X)] + logP(X)$$

Which on rearranging gives,

$$logP(X) = D_{KL}(Q||P) - \sum_Z Q(Z)[logQ(Z) - logP(Z,X)] = D_{KL}(Q||P) + \mathcal{L}(Q) \tag{3}$$

As the evidence P(X) is fixed, maximizing the the final term $\mathcal{L}(Q)$ minimizes the KL divergence of Q with respect to P. By appropriate choice of Q, this term becomes tractable and we have an analytical approximation for the posterior $P(Z|X)$ and a lower bound for the log evidence since the KL divergence is non negative.

Mean field Approximation
The variational distribution (Q(Z)) is typically assumed to factorize over some partition of the latent variables **Z** into $\mathbf{Z_1}, \mathbf{Z_2} .... \mathbf{Z_M}$

$$Q(Z) = \prod_{i=1}^{M} q_i(Z_i|X) \tag{4}$$

It can be shown using the **calculus of variations** (hence the name "variational Bayes") that the "best" distribution q_j^* for each of the factors q_j (in terms of the distribution minimizing the KL divergence, as described above) satisfies:

$$q_j^*(Z_j|X) = \frac{e^{E_{q_j^*}[ln(p(Z,X))]}}{\int e^{E_{q_j^*}[ln(p(Z,X))]} dZ_j} \tag{5}$$

Where $E_{q_j^*}[ln(p(Z,X))]$ is the expectation of the joint probability of the data and the latent variables taken with respect to q^* over all variables not in the partition.

2.3 Probabilistic Modelling of the Autoencoder

The idea of the variational autoencoder is to maximize the likelihood of the data x by the chosen parameterization $p_\theta(x)$ (typically chosen to be a gaussian). The key point to note is that vanilla autoencoders encode data into a latent space that is generally not regular. It is somewhat overfitted to match the data and this creates a problem while generating new data. Come to think of it, this lack of structure among the encoded data into the latent space is pretty normal. Nothing in the task the autoencoder is trained for enforces it to get such organisation: the autoencoder is solely trained to encode and decode with as few loss as possible, no matter how the latent space is organised.

This is the reason why we encode our data into a variational distribution instead of single data points in a latent space. Analogous to the variational bayes formulation, we have:

$$\begin{gathered} Prior : p_\theta(z) \\ Likelihood : p_\theta(x|z) \\ Posterior : p_\theta(z|x) \end{gathered}$$

As seen in the variational Bayes formulation, $p_\theta(x)$ is difficult to calculate and typically intractable, so $p_\theta(z|x)$ is approximated to some $q_\phi(z|x)$

We parameterize the encoder as $\mathbf{E}_\phi$ and the decoder as $\mathbf{D}_\theta$.

The problem is to now find a good probabilistic autoencoder, where the conditional likelihood $p_\theta(x|z)$ is computed by the decoder and the posterior $q_\phi(z|x)$ is computed by the encoder.

As calculated earlier, the **Evidence Lower Bound** is as follows:

$$\mathcal{L}_{\phi\,\theta}(x) = \mathbb{E}_{z\sim q_\phi(.|x)}\left[ln\frac{p_\phi(x,z)}{q_\theta(z|x)}\right] = ln(p_\theta(x)) - D_{KL}(q_\phi(.|x)||p_\theta(.|x)) \tag{6}$$

and the task at hand is to maximize the lower bound (note that here I've written it as a function of x, however the task is to find the appropriate ϕ and θ to maximize $\mathcal{L}$ for every x.[1]

$$\phi^*, \theta^* = \arg\max_{\phi\,\theta} \mathcal{L}_{\phi\,\theta}(x) \tag{7}$$

The following equivalent form of the loss is easier to optimize:

$$\mathcal{L}_{\phi\,\theta}(x) = \mathbb{E}_{z\sim q_\theta(.|x)}[ln(p_\theta(x|z))] - D_{KL}(q_\phi(.|x)||p_\theta(.)) \tag{8}$$

The distributions of $p_\theta(z)$ and $q_\phi(z|x)$ are modelled as Gaussians as $z \sim \mathcal{N}(0, I)$ and $z|x \sim \mathcal{N}(E_\phi(x), \sigma_\phi(x)^2 I)$. $ln(p_\theta(x|z))$ is implemented as $-\mathbf{\frac{1}{2}}||\mathbf{x} - \mathbf{D}_\theta(\mathbf{z})||^{\mathbf{2}}_{\mathbf{2}}$, since that is up to an additive constant what $x \sim \mathcal{N}(D_\theta(z), I)$ yields. We now use the KL Divergence for Gaussians:

$$\mathcal{L}_{\phi\,\theta}(x) = -\frac{1}{2}\mathbb{E}_{z\sim q_\theta(.|x)}[||x - D_\theta(z)||^2_2] - \frac{1}{2}(N\sigma_\phi(x)^2 + ||\mathbb{E}_\phi(x)||^2_2 - 2Nln(\sigma_\phi(x)) + Const \tag{9}$$

Here N is the dimension of z

[1] We simultaneously maximize the log likelihood and minimize the KL divergence by maximizing the evidence loss

2.4 Reparameterization

To find

$$\phi^*, \theta^* = \arg\max_{\phi\,\theta} \mathcal{L}_{\phi\,\theta}(x)$$

the usual method is gradient descent.
The computation of the following is straightforward,

$$\nabla_\theta \, \mathbb{E}_{z \sim q_\phi(.|x)} \left[ln \frac{p_\phi(x, z)}{q_\theta(z|x)} \right] = \mathbb{E}_{z \sim q_\phi(.|x)} \left[\nabla_\theta \, ln \frac{p_\phi(x, z)}{q_\theta(z|x)} \right]$$

However,

$$\nabla_\phi \, \mathbb{E}_{z \sim q_\phi(.|x)} \left[ln \frac{p_\phi(x, z)}{q_\theta(z|x)} \right]$$

is not as we cannot put the ∇_ϕ inside the expectation.
The **reparameterization trick** (also known as stochastic backpropagation) bypasses this difficulty.

When $z \sim q_\phi(.|x)$ is a Gaussian, $\mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, it can be parameterized using $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ as a "standard random number generator" and construct $z = \mu_\phi(x) + L_\phi(x)\epsilon$. Here $L_\phi(x)$ is found using the Cholesky Decomposition. The Cholesky decomposition is the decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose.

$$\Sigma_\phi(x) = L_\phi(x) L_\phi(x)^T \tag{10}$$

Then we get,

$$\nabla_\phi \, \mathbb{E}_{z \sim q_\phi(.|x)} \left[ln \frac{p_\phi(x, z)}{q_\theta(z|x)} \right] = \nabla_\phi \, \mathbb{E}_\epsilon \left[ln \frac{p_\phi(x, \mu_\phi(x) + L_\phi(x)\epsilon)}{q_\theta(\mu_\phi(x) + L_\phi(x)\epsilon|x)} \right] \tag{11}$$

and so we obtained an unbiased estimator of the gradient, allowing **stochastic gradient descent**.

Since we parameterized z, we want to find $q_\phi(z|x)$. Let q_0 be the probability density function for ϵ, then

$$ln\, q_\phi(z|x) = ln\, q_0(\epsilon) - ln|det(\partial_\epsilon(z))|$$

Where $\partial_\epsilon(z)$ is the **Jacobian matrix** of ϵ with respect to z. Since $z = \mu_\phi(x) + L_\phi(x)\epsilon$, this becomes,

$$ln\, q_\phi(z|x) = -\frac{1}{2}||\epsilon||^2 - ln|det(L_\phi(x))| - \frac{n}{2} ln(2\pi)$$

2.5 Neural Network model

The idea is that now our approximation to the posterior $q_\phi(z|x)$ is going to be a neural network and the parameters θ and ϕ are jointly optimized using stochastic gradient descent.

I modelled the distribution (of which the parameters in the latent space are to be found) as a Gaussian. The network outputs the mean and log-variance parameters of a factorized Gaussian. I output log-variance instead of the variance directly for numerical stability.

The network layers are fully connected layers (or convolutions and maxpools) for the encoder and fully connected (or convolutions and upsampling) for the decoder. The reparameterization is implemented as mentioned in the section above.