Ramin Follow

Sep 9 · 6 min read · Listen

What are Diffusion Models and Stable Diffusion?

Image generated using Stable Diffusion with prompt: “a photograph of diffusion model”

If you have heard these names and seen the amazing results of Stable Diffusion model which converts a short textual description to a high-res image and curious to know how these things work, you have come to the right place.

In this short article, I give a high-level description of these concepts which is not too fluffy (as you may find in a pseudo-technical blo[g] 11 [c]etailed to put you to sleep. The goal

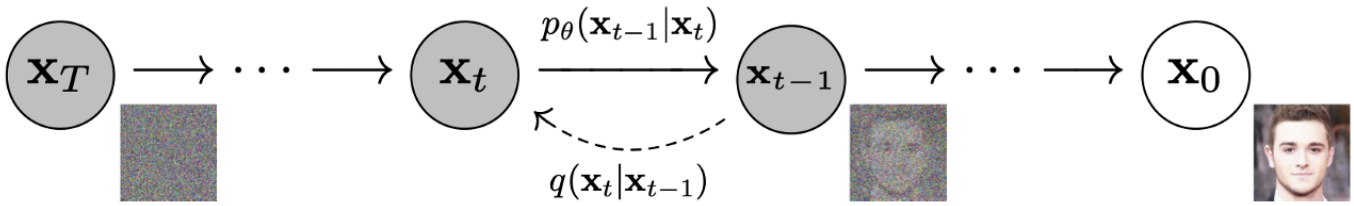
here is to understand the concept of diffusion models and Stable Diffusion so that you can dig deeper. This overview is a summary of 3 papers and two blogs that are listed at the end of this article.

What is a Diffusion Model

The story started with this paper by Ho et. al. in late 2020 (this paper itself is based on work of several other researchers, of course). The goal of such model is to create machine-generated images that resembles the images on which the model is trained. This is similar to what a GAN or a Variational Auto-encoder can do. Specifically, we want to feed noise to our model and get an image that resembles a "real" image from our model.

To do this, we train a network that is learned to remove Gaussian noise from images progressively. It can do this so well that it can start from complete noise and produce a "real" image. So, to generate training data from our model, we start from an image in the training set and add isometric Gaussian noise to it. Isometric here just means that the noise is independent for each pixel (i.e., the covariance matrix is diagonal).

In the following image, as we go from right to left, noise is added at each step such the x_t is noisier than $x_{t\text{-}1}$. This is called the forward process.



Formally, noise is added as follows (mean and covariance are the 2nd and 3rd parameters, respectively):

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}).$$

where, $0<\beta 1<\beta 2<\ldots<\beta T<1$

Using the property of Gaussian noise that sum of Gaussians is also Gaussian, one can easily show that the following holds:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

where,

$$\alpha_t := 1 - \beta_t \text{ and } \bar{\alpha}_t := \Pi_{s=1}^{t} \alpha_s$$

So, given an image from the training set, x_0, and a noise level '*t*', we can produce a noisy image x_t, using the above relationship.

The training steps are as follows:

- pick an image, x_0, from the training set
- pick a noise level *t* between 1 and T at random
- create a noisy image x_t computed as shown below

$$\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon$$

where

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

- Feed scalar *t* and image x_t as inputs to the network
- train a network to predict ε

The following is the pseudo-code for the training algorithm:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Notes about the Neural Network

One of the inputs to the network is *t* which is a scalar and the other one is x_t which is an image. A typical technique for feeding a scalar input in such cases is to use locational embedding which given a scalar, creates a deterministic tensor of size (num_channels, height, width) which can be concatenated with the input image features in early stages of the network.

The network is to produce an "image" of the same size as the input noisy image, so a U-Net is a reasonable choice. In practice, adding an attention block helps.

Image Generation (aka Sampling)

A network trained as described above is able to predict the noise added at a given noise level, *t*. So, we can start from pure noise which represents x_{T} and ask the network to predict the noise level that is needed to construct x_{T-1} and repeat this until we construct x0. In the following pseudo-code, Z_{theta} represents the trained model and:

$$\sigma_t^2 = \beta_t$$

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

That's it! Now you know what the diffusion model is and I hope the intuition behind it is clear.

Guidance

Stable Diffusion has become a household name primarily because its capability of generating high-resolution images based on a short textual description (aka prompt) and that is something that we have not discussed yet.

The "Classifier-free Diffusion Guidance" paper by Ho & Salimans proposes a clever modification to the general diffusion model as follows:

In the training phase, instead of picking images randomly from a dataset (eg, ImageNet) and feeding them to the model as input, also supply the class (or a short textual description) associated with that image to the model. Of course, the text needs to be tokenized and transformed into a feature space (eg, using a pre-trained BERT or CLIP model) to be combined with other features of the U-Net network.

So, the model now has three inputs, the image (or the latent features associated with it), the noise level (denoted by *t* in the previous section), and *c* which represents features associated with the textual description for the image). The following algorithm shows the modified training steps:

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond}: probability of unconditional training

1: **repeat**
2: $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
3: $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
4: $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
5: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6: $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
7: Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model
8: **until** converged

You notice that in step 3 of this algorithm, the class is suppressed with a certain fixed probability. This is one of the major contributions of the "Classifier-free Diffusion Guidance" paper which distinguishes it from other diffusion-based models that use a classifier (ie, the textual description) for generating images (hence, the name "classifier-free"). Eliminating a classifier simplifies training and sampling significantly.

The sampling process is very similar to the sampling process described in the previous section with the difference that it takes another input, *c* , which is the textual description of the image that we want the sampling process to generate. This process is shown below:

Algorithm 2 Conditional sampling with classifier-free guidance

Require: w: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}, \lambda_T = \lambda_{\max}$

1: $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = 1, \ldots, T$ **do**
▷ Form the classifier-free guided score at log SNR λ_t
3: $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
4: $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t} \tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
5: $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v} (\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
6: **end for**
7: **return** $\mathbf{z}_{T+1}$

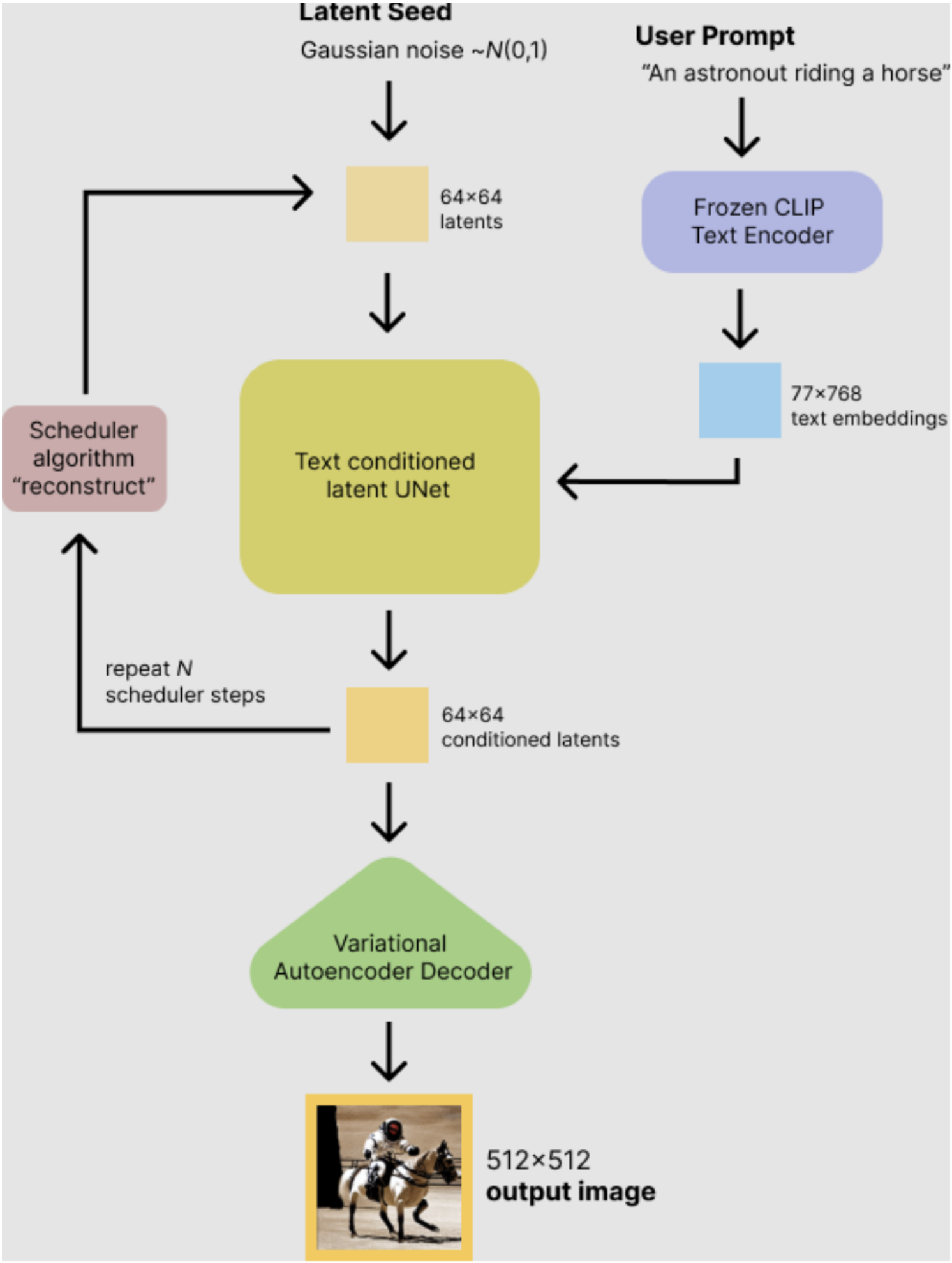
Note that in step 3, we perform the inference twice: once with the input *c* (aka "guidance") and once without it. In practice, for the inference without guidance, we use the exact same model but provide an empty string for the input *c*. The predicted noise model is then linearly combined using a "guidance weight" parameter *w*. This parameter allows a trade-off between diversity of generated images and fidelity to the guidance phrase (*c*).

Stable Diffusion

In the previous sections we notice that to generate a “real” image from pure noise we need to run the inference T times (a typical value for T is ~200) and each time the input to the model has the same resolution as the final image. This is relatively slow and memory consuming.

The “High-Resolution Image Synthesis with Latent Diffusion Models” paper addresses this problem by noting that instead of processing everything in the pixel space, we can operate at a much lower resolution by executing all the steps in training and sampling on “compressed” feature space produced by a variational auto-encoder. At the end of the T steps of de-noising (aka Sampling), the de-noised latent features are given to the decoder portion of the auto-encoder to produce a high resolution “real” image.

The following shows the flow diagram of the sampling process:



source: this

Now you also know how Stable Diffusion works and some theory behind it.

I hope you find this short article useful and it motivated you to dig deeper.

References

- "High-Resolution Image Synthesis with Latent Diffusion Models
- CLASSIFIER-FREE DIFFUSION GUIDANCE
- "Denoising Diffusion Probabilistic Models"
- "The Annotated Diffusion Model"
- "Stable Diffusion with Diffusers"