Stable Diffusion

Gaussian Noise

Color Distribution

Stable Diffusion is an artificial intelligence model that can generate images from descriptive
text inputs.

It consists of :

- Text Encoder : It takes the descriptive text input and creates a numerical representation of it.
- Image Generator : Here the Diffusion process happens.

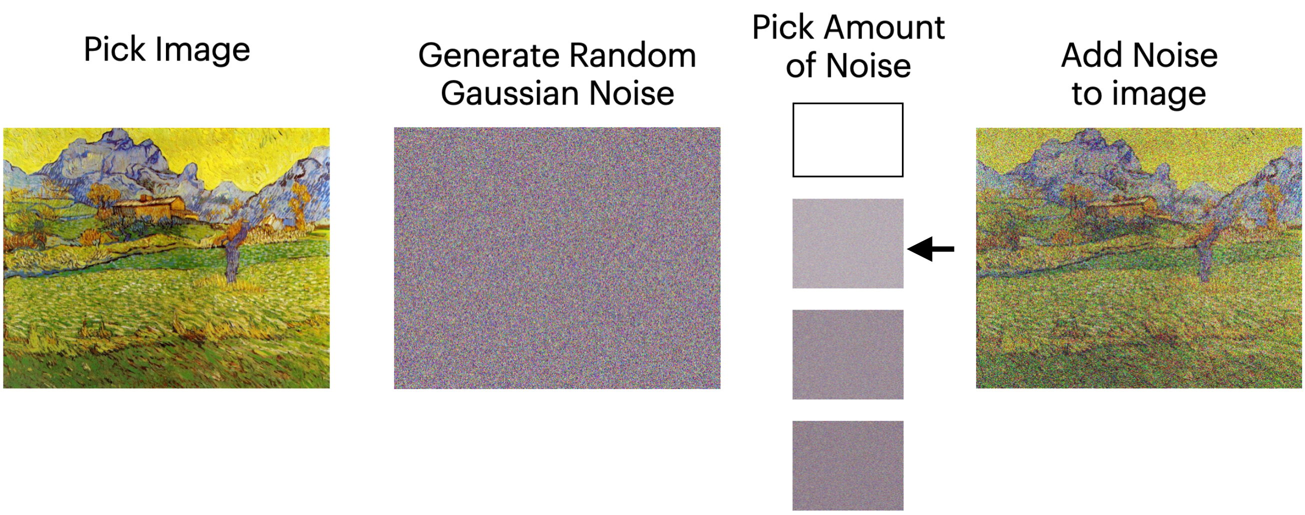
The Diffusion Process

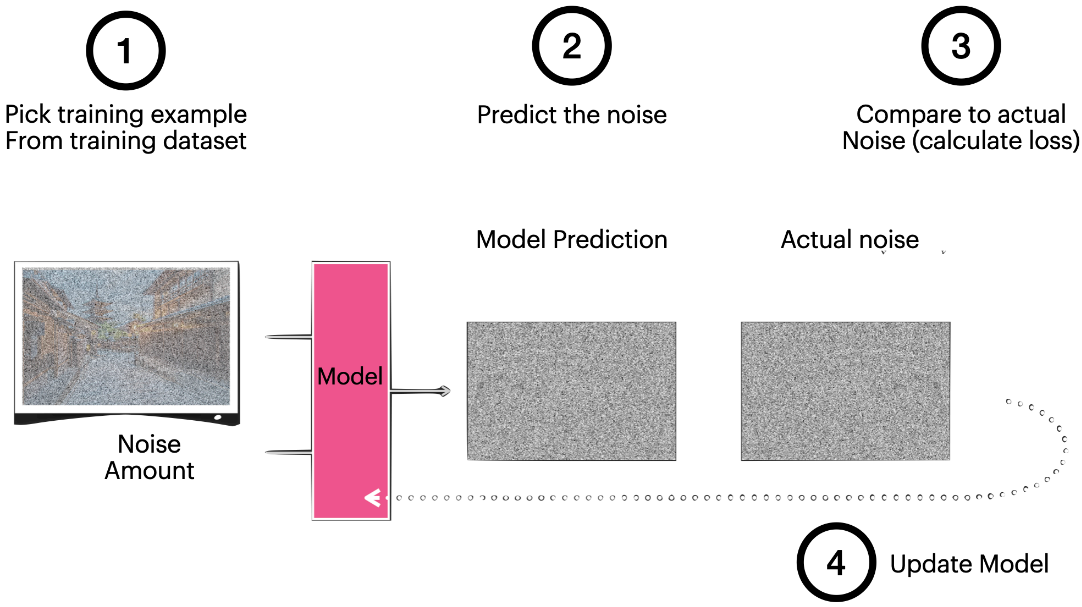
To generate a unique desired image, the model starts with a random Gaussian Noise image information, and from here undergoes multiple steps to reach our desired image. Each step the model changes the information in a way that more resembles the input text but also in a way that resembles the visual information the model picked up from all images it has been trained on.

Training Diffusion Models

Noise Predictor:

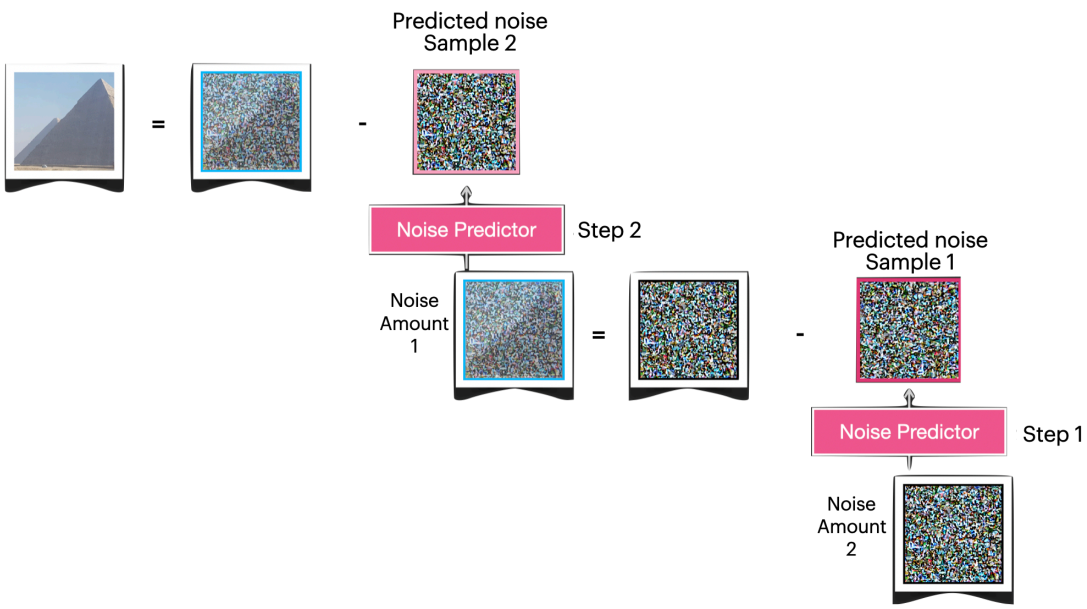
We get an image, generate some random gaussian noise, and then let the model predict this noise. After that the model compares its prediction to the actual noise (calculate loss), then updates itself to be better next time.





Reverse Diffusion (Denoising)

The trained noise predictor takes a noisy image, the amount of noise, and is able to predict some of the noise by each step. The noise is predicted so that if we subtract it from the noisy image we get an image close to the images the model has been trained on, not the exact images but fits the distribution of those images. Though until now we have no control over the generated images, here is where we incorporate the text input and generate what we want.



Learning about Everything

It learns about objects, places, living things, nature, the world, literally everything with its details by feeding it large datasets of images attached with captions describing them. How it's trained is we pass the image along with a text and let it predict if the text describes the image or not, then it updates itself according to the level of its prediction. So with that model added to the Diffusion model a layer of conditionality of the similarity between the image produced in this step the text information is added.

Generating an image that fits the data it trained on

This data has certain distributions for everything : colors, objects, mood, style, details, and much more. So what the model does is generate an image that belongs to these distributions in the way needed. We will now consider images of Van Gogh as our data, we will study their color distributions namely Histograms. By doing that we will know how this artist uses color and what are the most likely colors used. A model like Stable Diffusion uses that information to generate new images that look like it's drawn by this artist, Van Gogh's rise from the dead.

Gaussian Noise

Probability Density Function

X ~ N(μ , σ)

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-\left(x^2-\mu\right)2}{2\sigma^2}}$$

σ : changes noise value spread - noise increases when standard deviation inceases

μ : changes intensity shift - image gets brighter when the mean increases

Cumulative Distribution Function

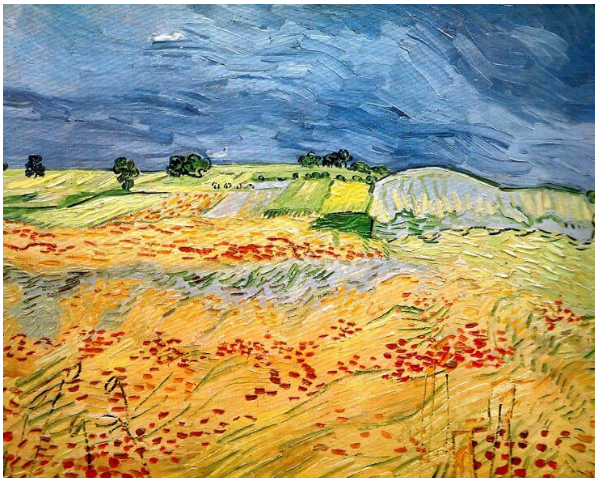
$$\phi(x) = P(X \leq x) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} e^{\frac{-(x-\mu)2}{2\sigma^2}} dx$$

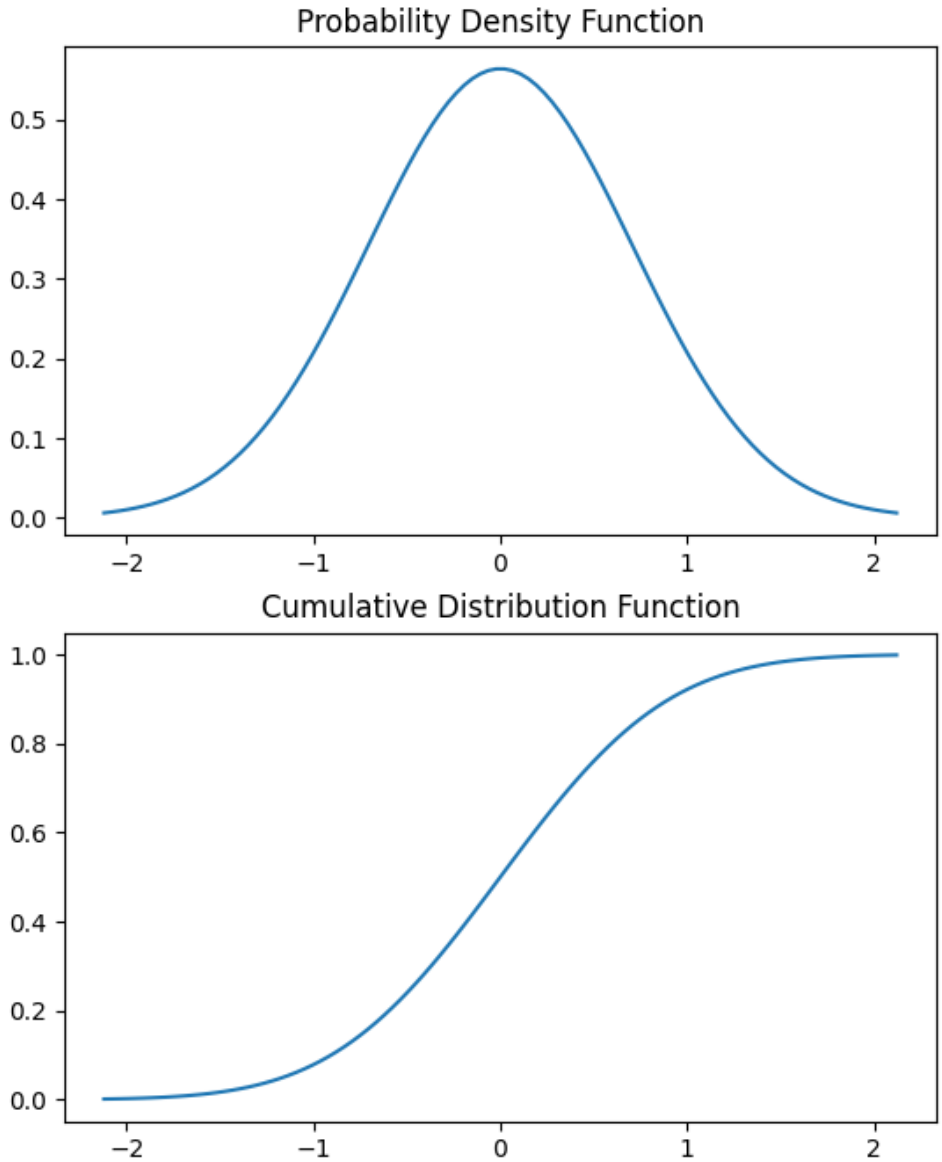
Expectation E[x] = μ

Varriance Var(x) = σ^2

Python program that generates random Gaussian noise and add it on an image also outputs PDF and CDF

Setting μ = 0 , σ = 0.5


Probability Density Function
0.5
0.4
0.3
0.2
0.1
0.0
−2
−1
0
1
2
Cumulative Distribution Function
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2

Color Distribution

Probability Mass Function

i : color intensity value
i = {0,..256}

$$PDF(i) = H(i) / N$$

H(i) : Frequency (count) of intensity value i in histogram

N : total number of pixels in image

Cumulative Distribution Function

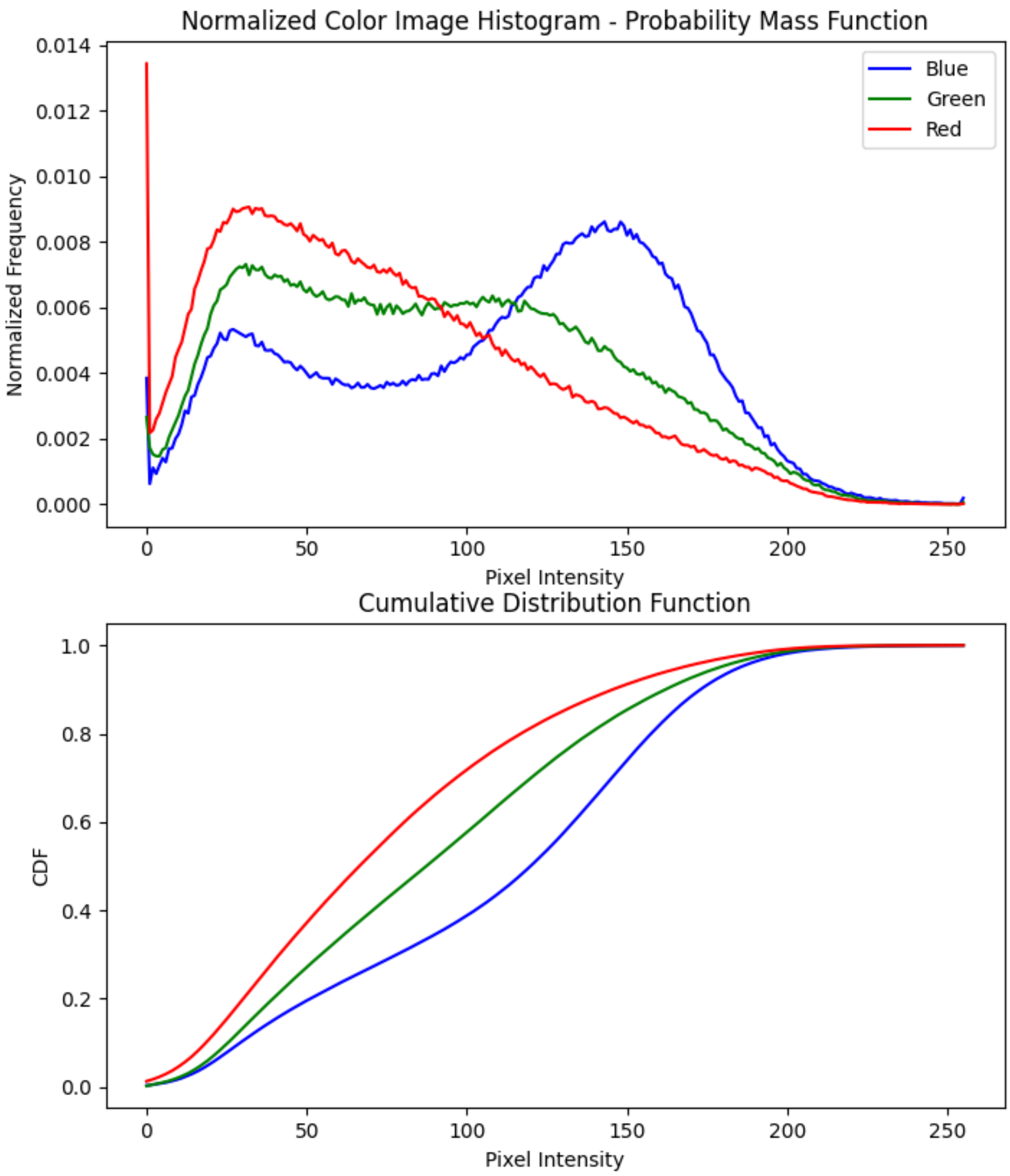
$$CDF(i) = [\sum H(i)] / N$$

Expectation E[x] = μ = $\sum$[i * PDF(i)]

Varriance Var(x) = E(x^2) - $E(x)^2$

Python program that takes an image and generates its PDF and CDF

image:


Normalized Color Image Histogram - Probability Mass Function
Blue
Green
Red
Normalized Frequency
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250
Pixel Intensity
Cumulative Distribution Function
CDF
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
Pixel Intensity

Conditional Probability

To compute a dependant probability of a color intensity on another color, a joint histogram representing the two colors can be created with normalizing the graph. The Denisty stripe on the right represents the confitional probability of the values. For example taking a vertical line on the blue intensity 150, we can observe the probabilities of the green intensities given that the blue is 150.