Generative AI

Diffusion Models;Why and How

Diffusion

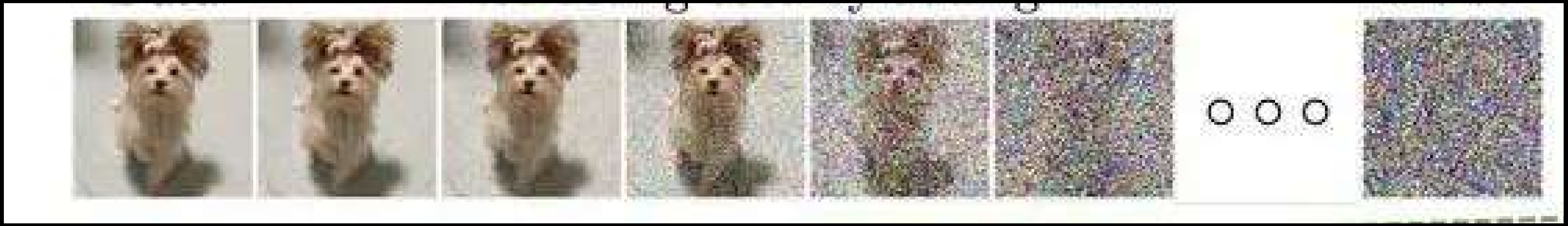
Low → High

Entropy

Can we reverse this?

Yes! Thats how diffusion models work,

We can extend this to images and diffusion models



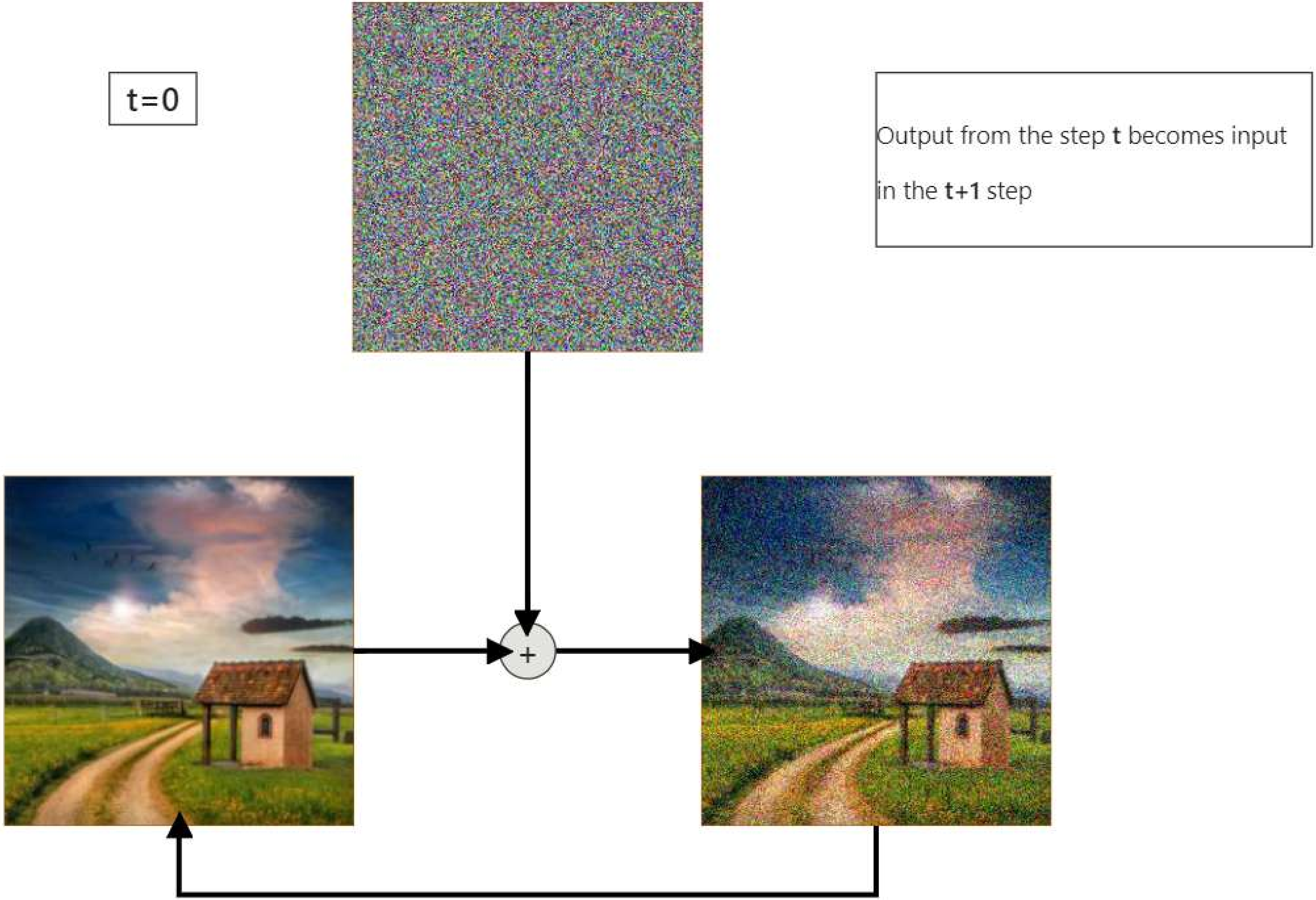
Can we reverse this?

Yes! Thats how diffusion models work,

Math behind Diffusion models

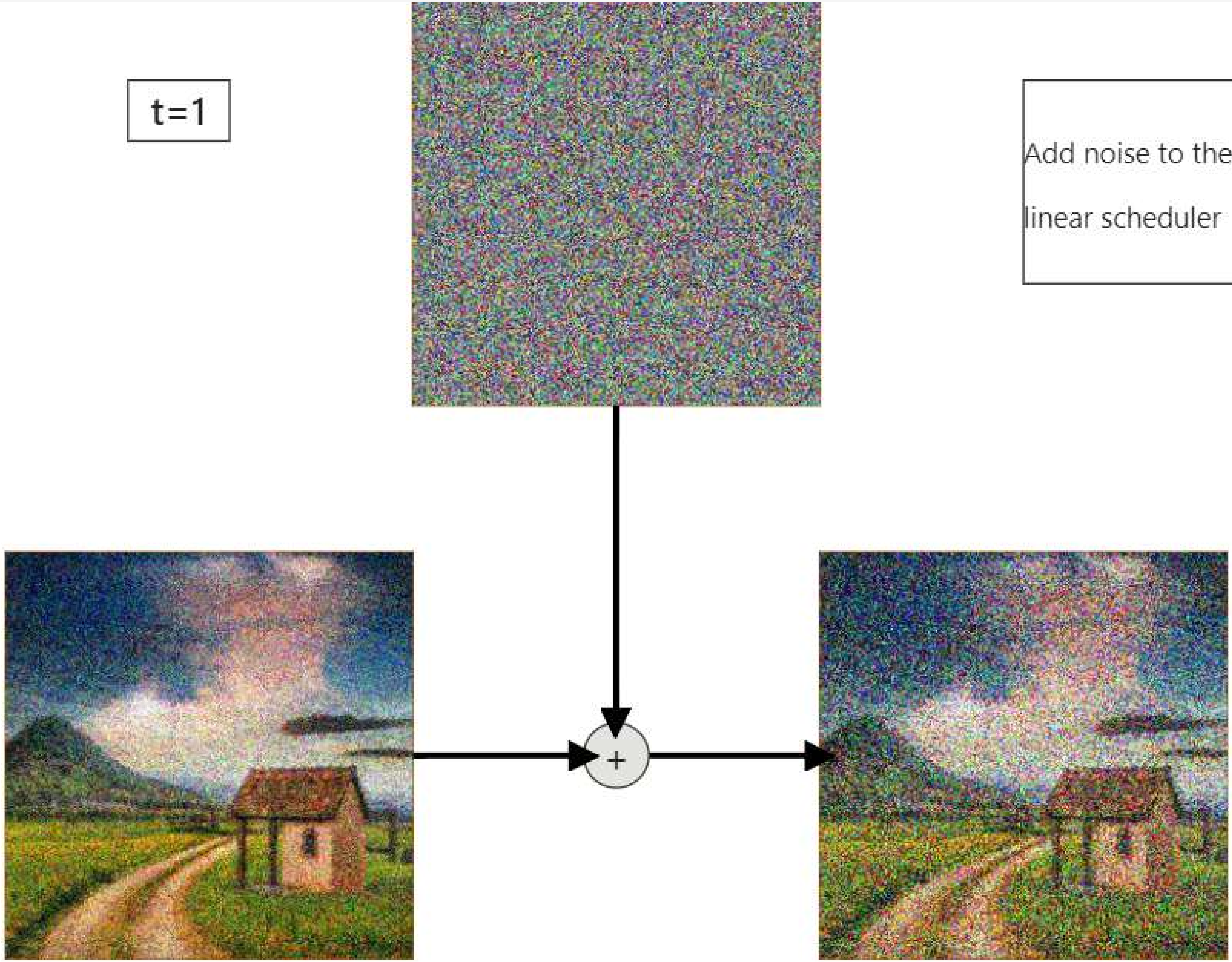
Forward Diffusion

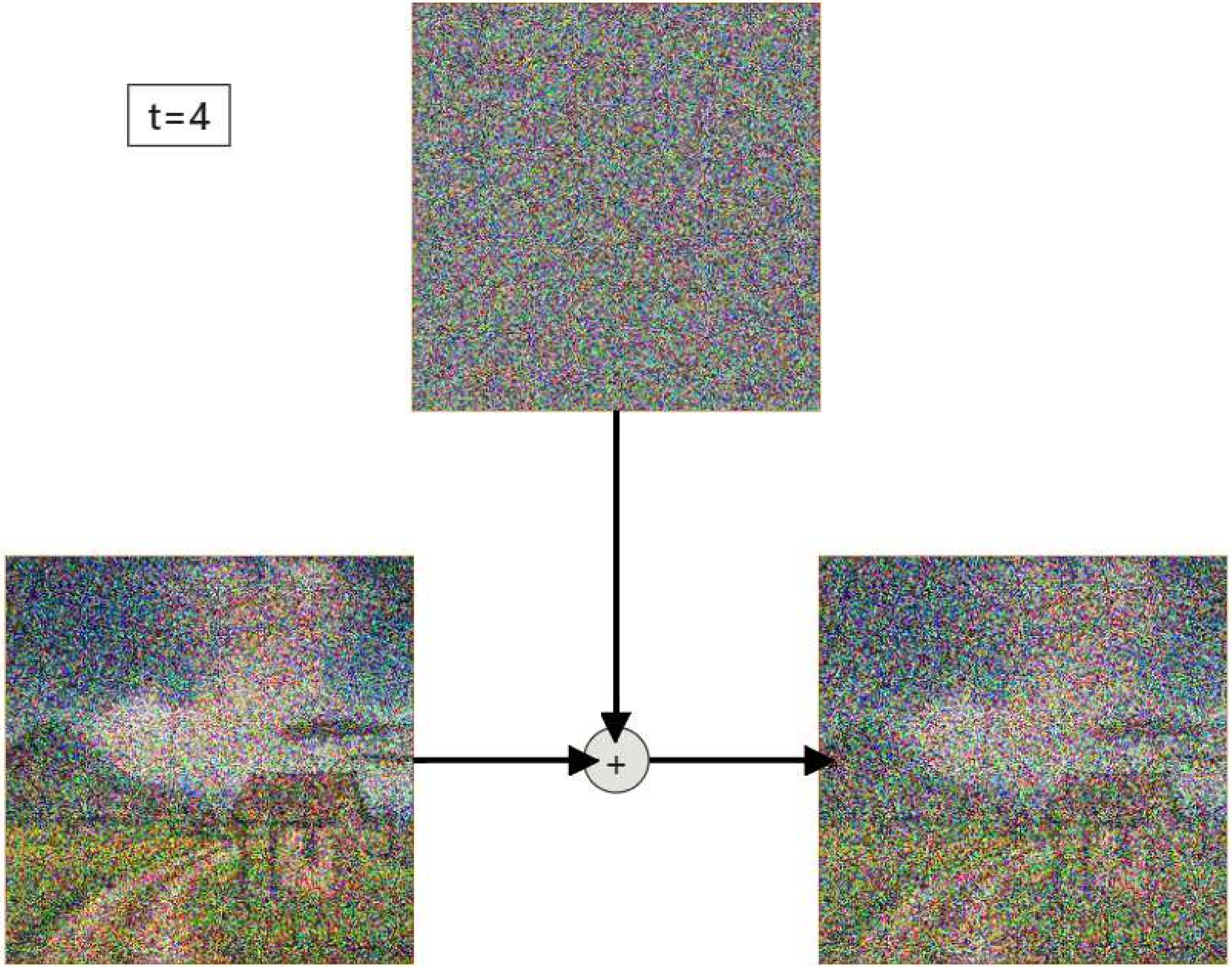
t=0


t=0
+
Output from the step **t** becomes input
in the **t+1** step

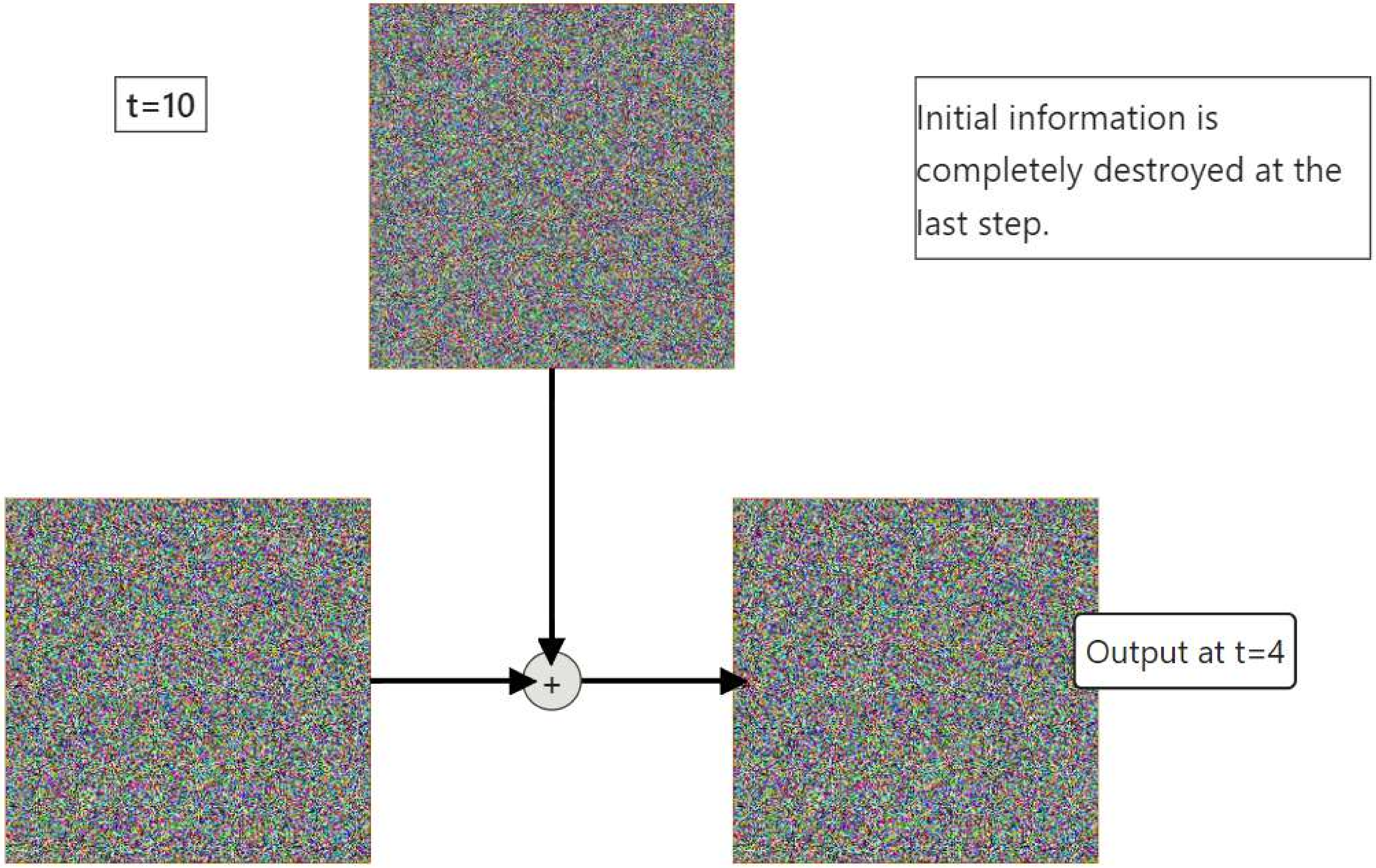
t=1

Add noise to the new input image using linear scheduler

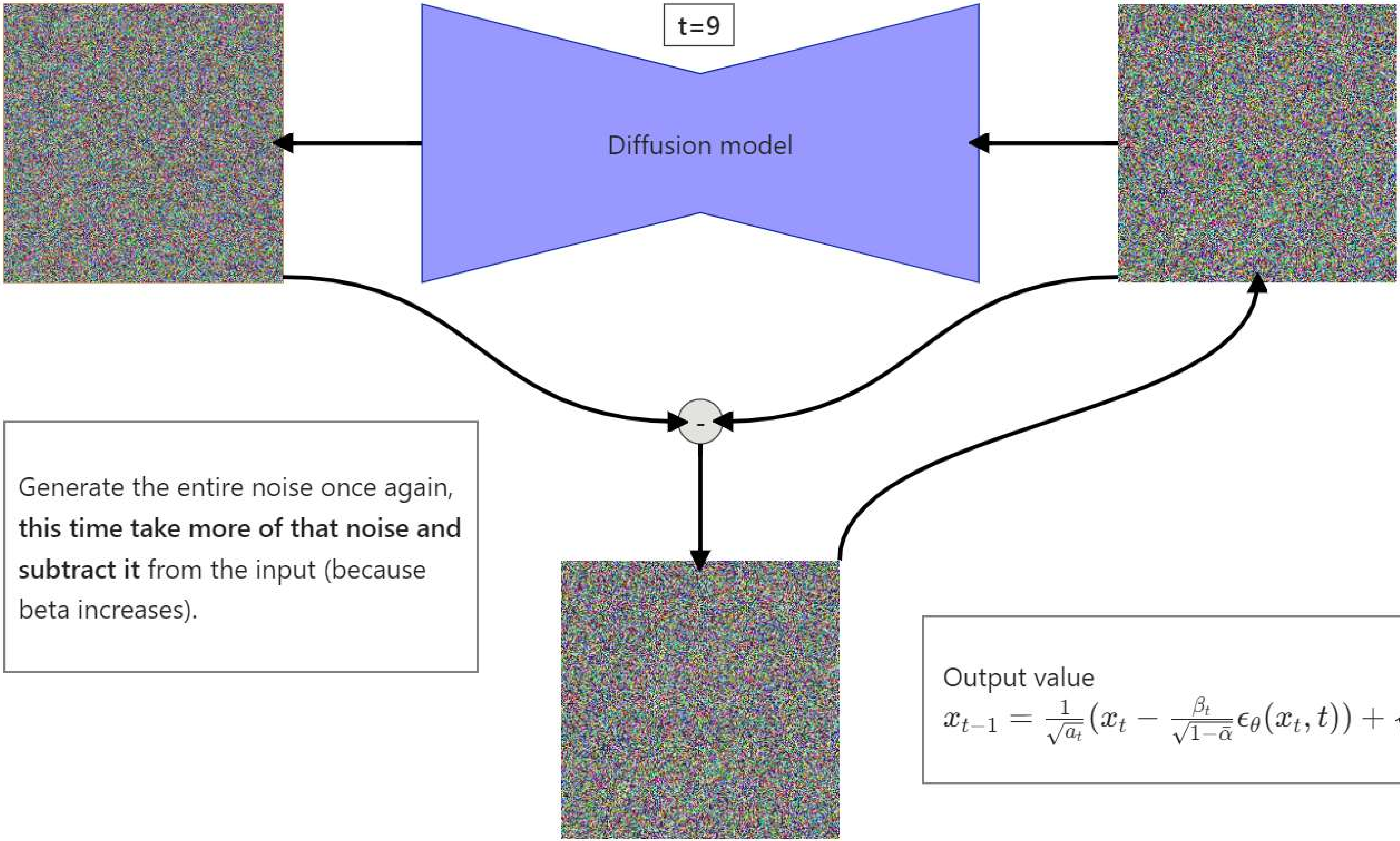




t=4
+


t=10
Initial information is completely destroyed at the last step.
+
Output at t=4

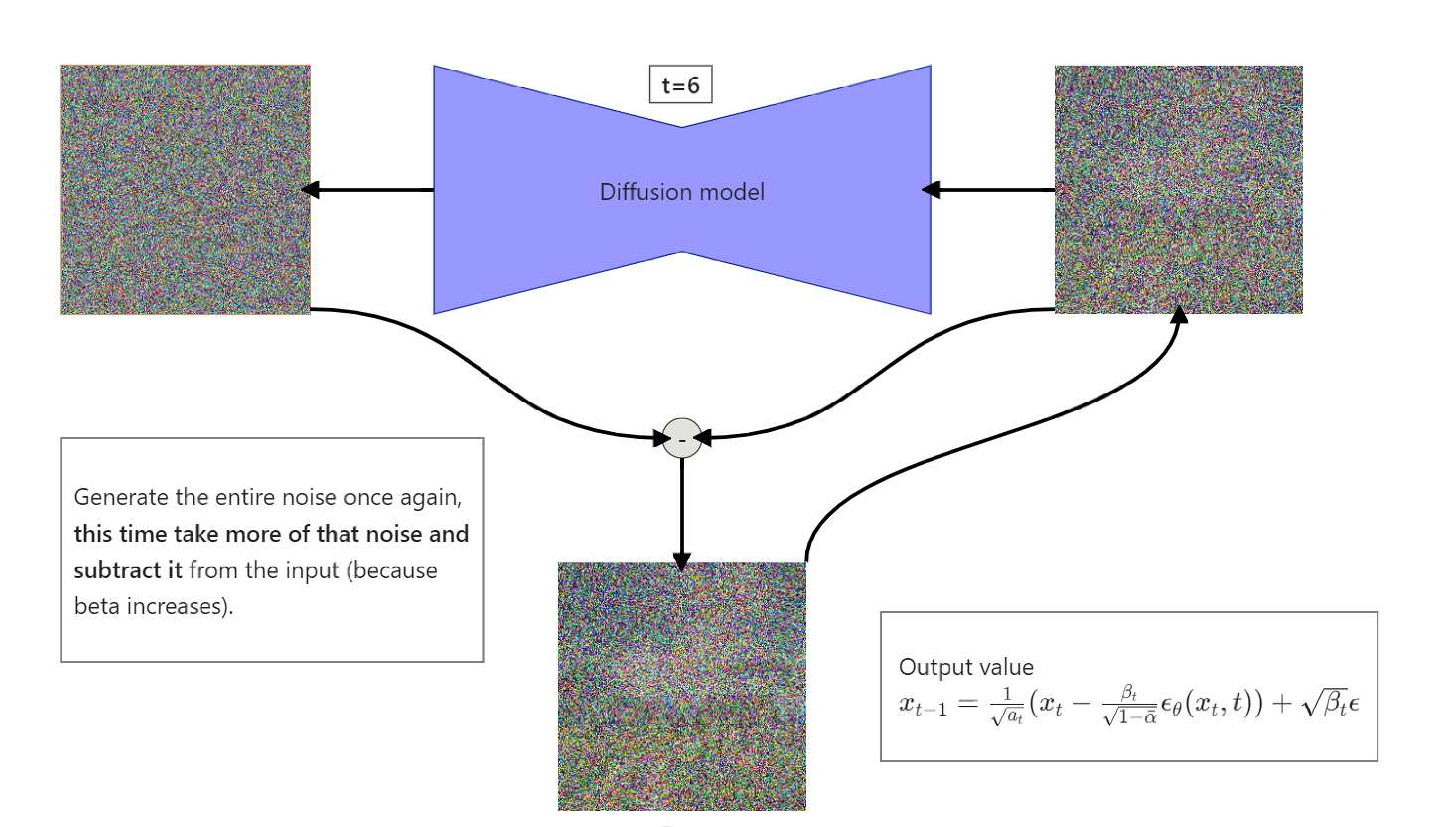
Reverse Diffusion

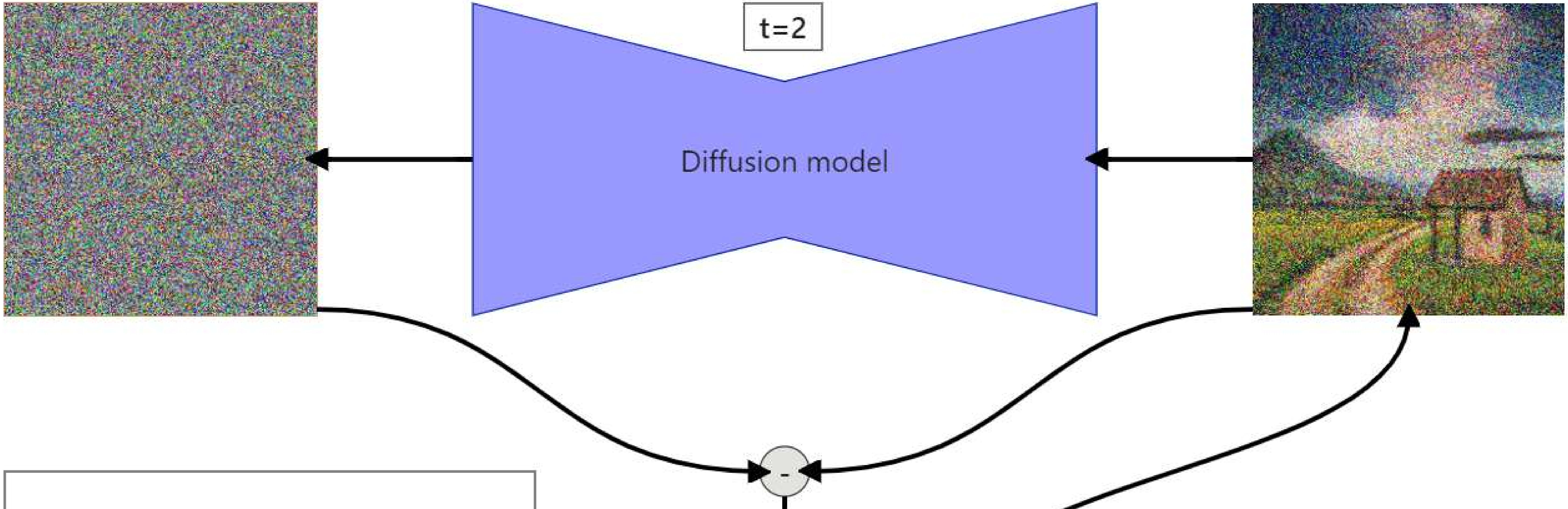


Generate the entire noise once again, **this time take more of that noise and subtract it** from the input (because beta increases).

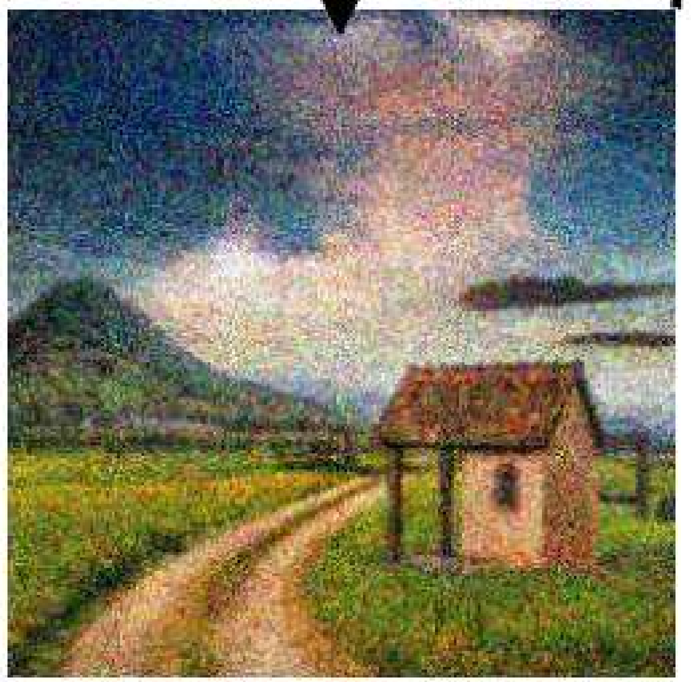
Output value

$$x_{t-1} = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(x_t, t)\right) + \sqrt{\beta_t}\epsilon$$

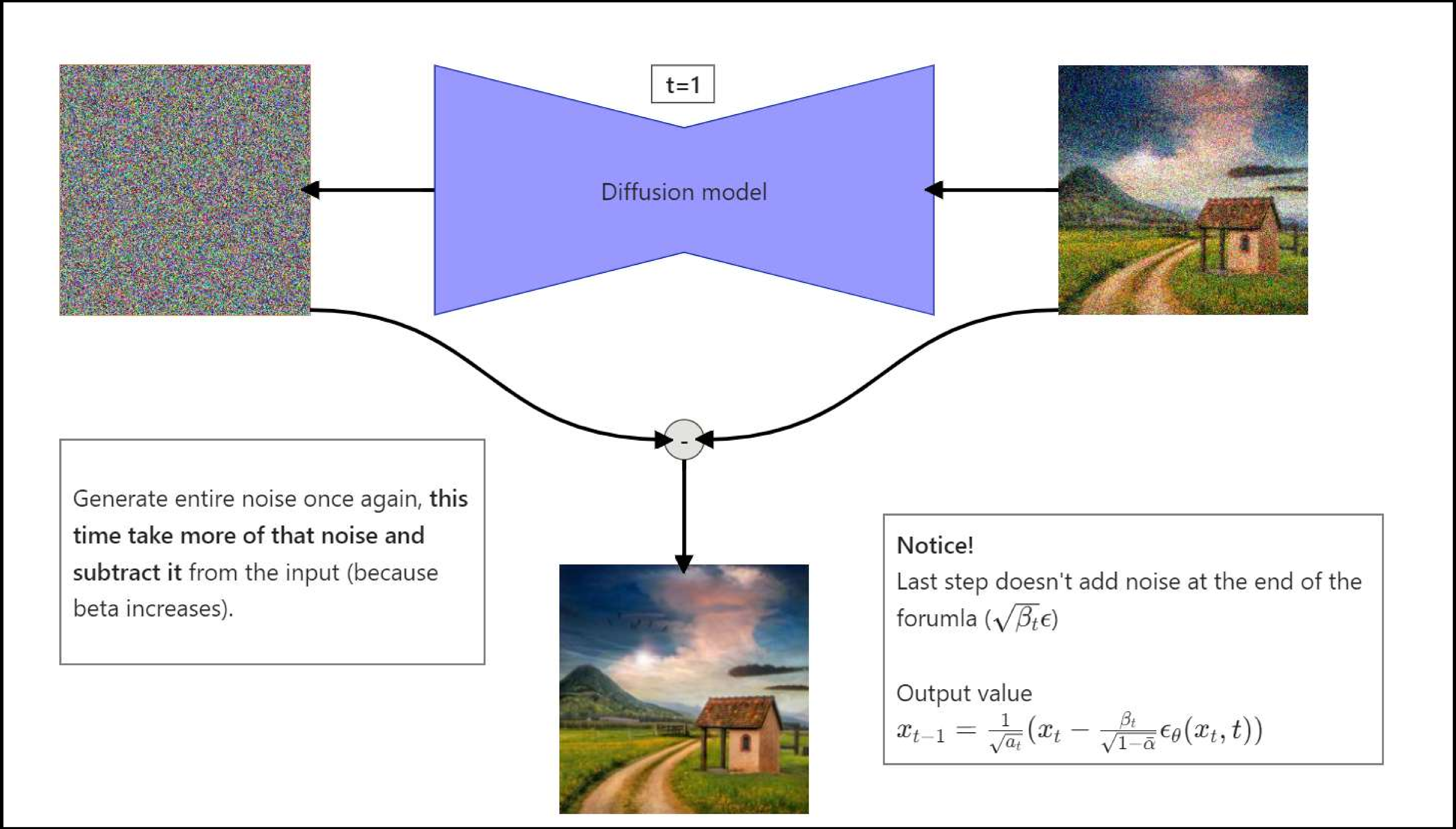

t=6
Diffusion model
-
Generate the entire noise once again, **this time take more of that noise and subtract it** from the input (because beta increases).
Output value
$x_{t-1} = \frac{1}{\sqrt{a_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(x_t, t)) + \sqrt{\beta_t}\epsilon$



Generate the entire noise once again, **this time take more of that noise and subtract it** from the input (because beta increases).



Output value
$$x_{t-1} = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(x_t, t)\right) + \sqrt{\beta_t}\epsilon$$


t=1
Diffusion model
-
Generate entire noise once again, **this time take more of that noise and subtract it** from the input (because beta increases).
Notice!
Last step doesn't add noise at the end of the forumla ($\sqrt{\beta_t}\epsilon$)
Output value
$x_{t-1} = \frac{1}{\sqrt{a_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(x_t, t))$

Loss function

Algorithm 1 Training

repeat

$\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$

$\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$

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

$\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

until converged

Algorithm 2 Sampling

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

for $t = T, \ldots, 1$ **do**

$\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$

$\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$

end for

return $\mathbf{x}_0$

Why not just use GANs