

Universidad de Cantabria
Instituto de Física de Cantabria (IFCA), CSIC-UC

Modelos de difusión

Jose González-Abad

Aplicaciones

Los **modelos de difusión** han emergido recientemente como una **alternativa** a otros **modelos generativos**:

- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)

Auto-Encoding Variational Bayes

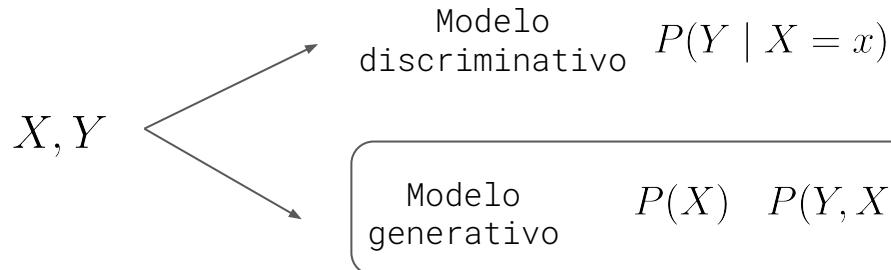
Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,^{*} Mehdi Mirza, Bing Xu, David Warde-Farley,
Sherjil Ozair,[†] Aaron Courville, Yoshua Bengio[‡]
Département d'informatic et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Recordatorio: Los modelos generativos aproximan la distribución de los datos, permitiéndonos generar nuevas muestras a partir de esta



Aplicaciones

¿Para qué se utilizan los modelos de difusión?

Generación de imágenes



Aplicaciones

¿Para qué se utilizan los modelos de difusión?

Generación de imágenes



Figure 18.12 Conditional generation using classifier guidance. Image samples conditioned on different ImageNet classes. The same model produces high quality samples of highly varied image classes. Adapted from Dhariwal & Nichol (2021).

Aplicaciones

¿Para qué se utilizan los modelos de difusión?

Generación de imágenes condicionadas a texto

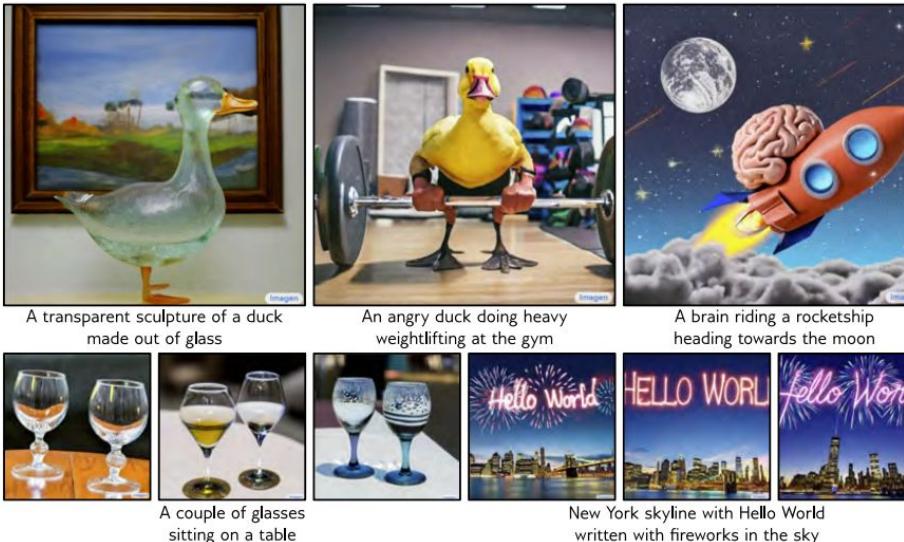


Figure 18.13 Conditional generation using text prompts. Synthesized images from a cascaded generation framework, conditioned on a text prompt encoded by a large language model. The stochastic model can produce many different images compatible with the prompt. The model can count objects and incorporate text into images. Adapted from Saharia et al. (2022b).

Aplicaciones

¿Para qué se utilizan los modelos de difusión?

Inpainting de imágenes



Figure 1.7 Inpainting. In the original image (left), the boy is obscured by metal cables. These undesirable regions (center) are removed and the generative model synthesizes a new image (right) under the constraint that the remaining pixels must stay the same. Adapted from Saharia et al. (2022a).

Aplicaciones

¿Para qué se utilizan los modelos de difusión?

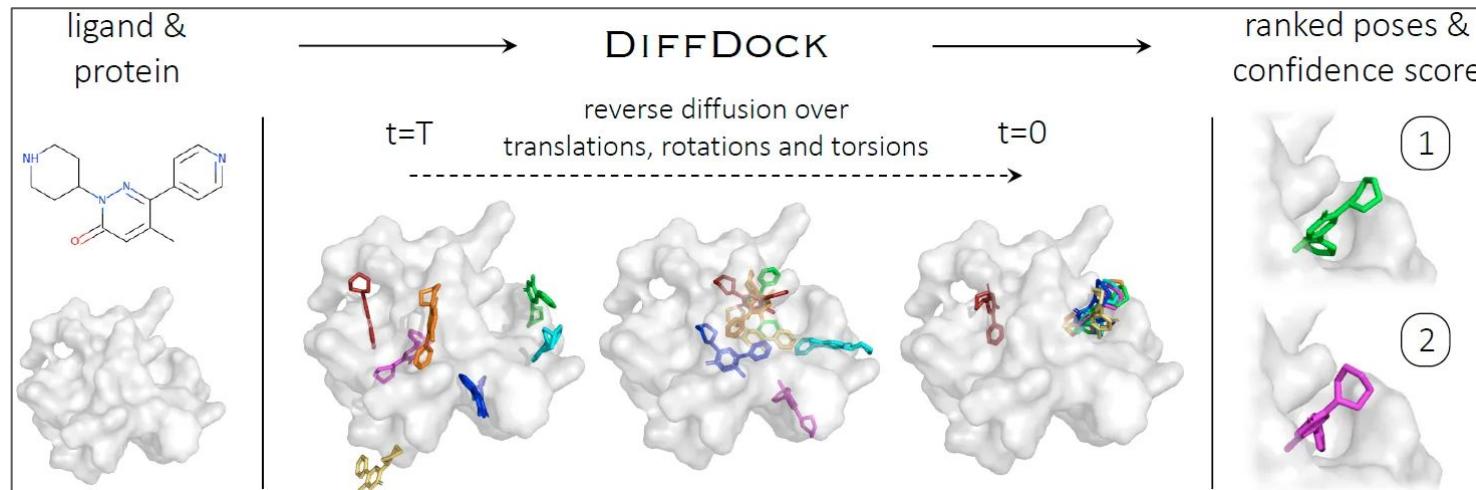
Generación de video



Aplicaciones

¿Para qué se utilizan los modelos de difusión?

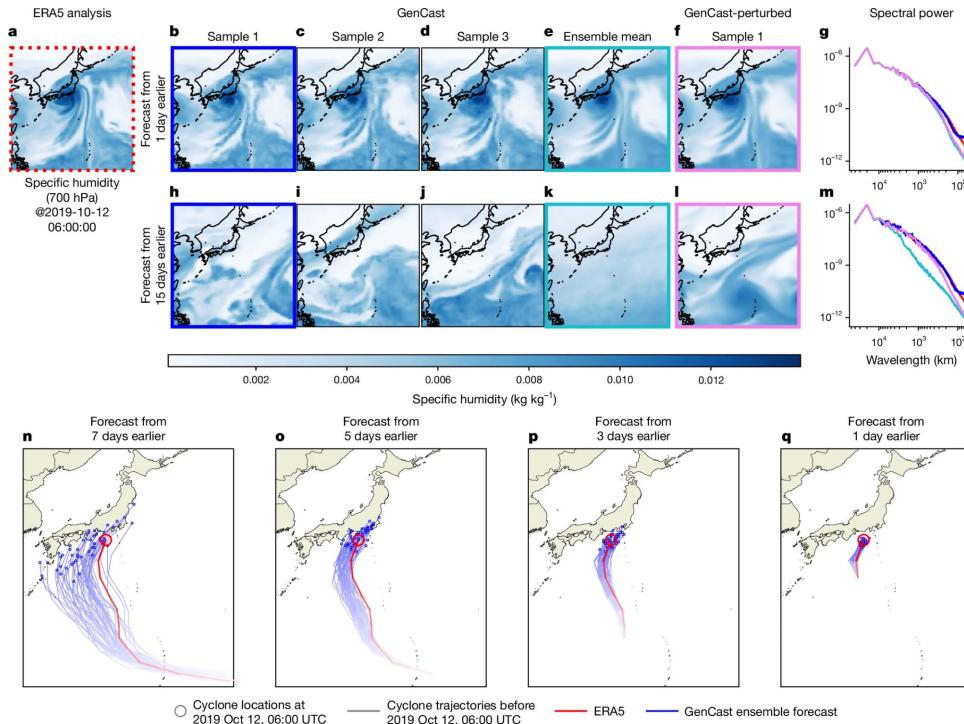
Drug discovery



Aplicaciones

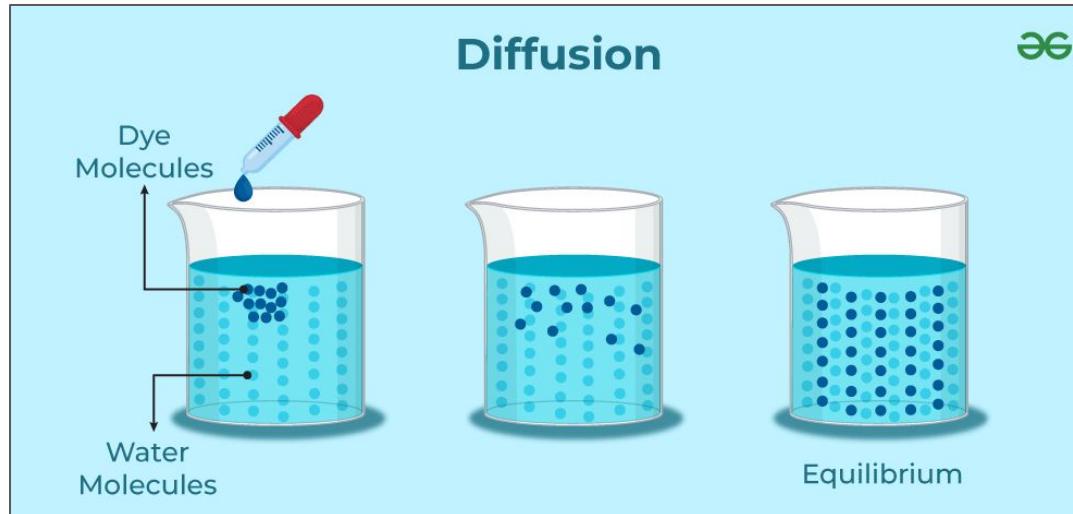
¿Para qué se utilizan los modelos de difusión?

Meteorología/clima



¿Qué es la difusión?

Principio físico que describe cómo las partículas se **dispersan** con el tiempo debido a su movimiento aleatorio (Brownian motion) y su tendencia hacia la **máxima entropía**



¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

Data



¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

Data



¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

Data



¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

Data



¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

Data



¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

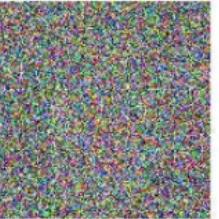
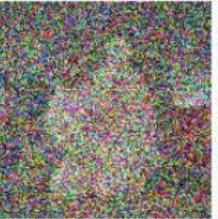
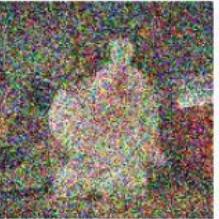
Data



¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

Data



Noise

Destrucción de información → Máxima entropía

¿Qué es la difusión?

En el contexto de los **datos**, la **difusión** se puede ver como un proceso de **destrucción de información** (disolución de información)

Data



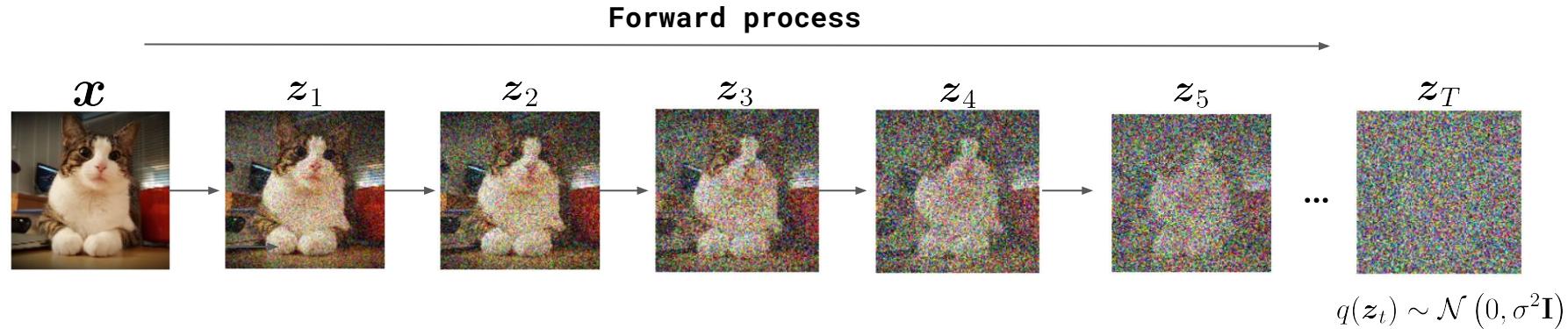
Noise

Destrucción de información → Máxima entropía



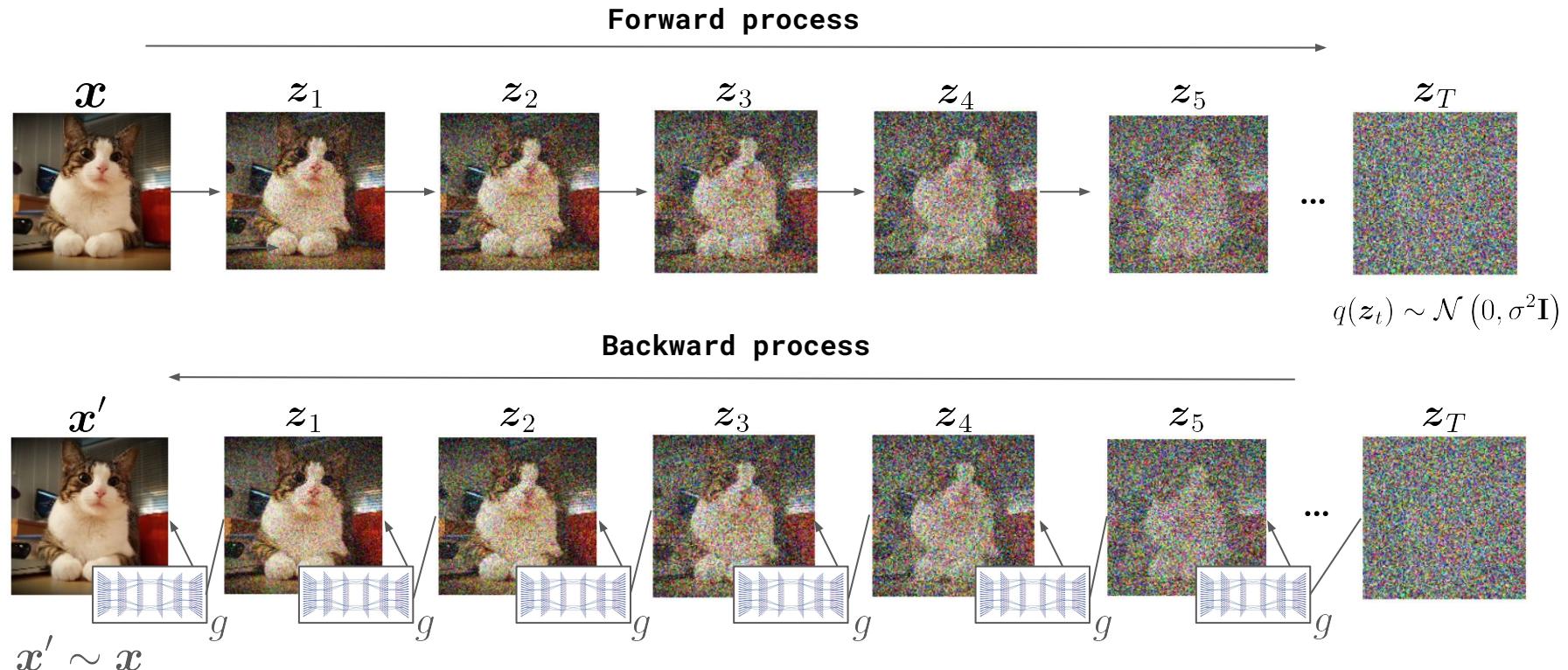
Overview

Primero, en el forward process, añadimos de forma gradual ruido a la imagen de entrada



Overview

Primero, en el forward process, añadimos de forma gradual ruido a la imagen de entrada

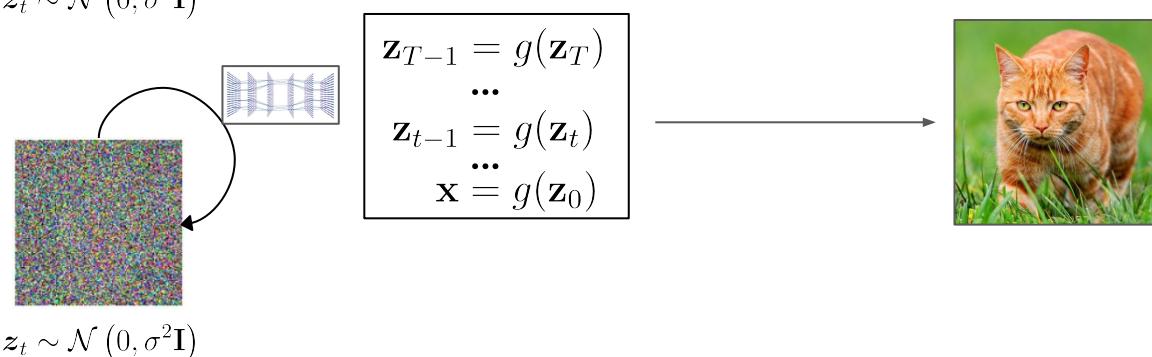
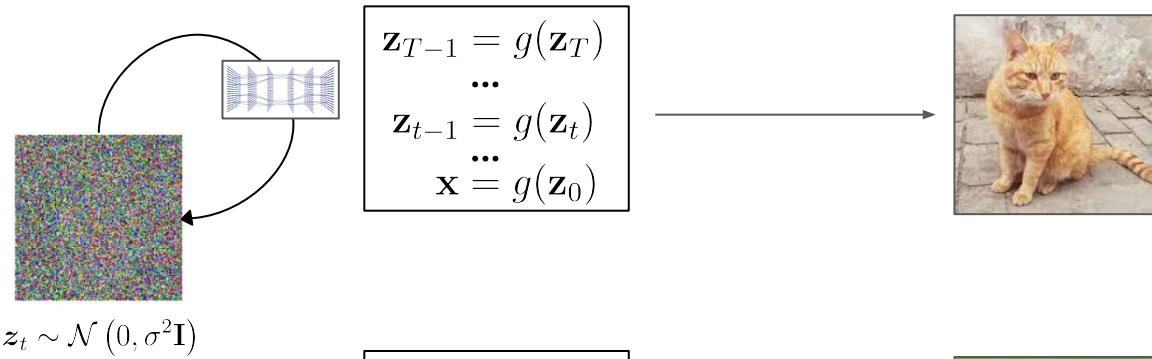


Después, en el backward process eliminamos de forma gradual ese ruido con un modelo de deep learning

Overview

Si conseguimos entrenar un modelo que sea capaz de eliminar el ruido de forma gradual tenemos una *aproximación* de $P(X)$

Para generar muestras de esta distribución bastaría con hacer lo siguiente:



Overview

Un modelo de difusión se puede resumir en tres pasos:

1. **Destruimos**, de forma progresiva, la **información** en una imagen (forward process)
2. **Reconstruimos**, de forma progresiva, la **información** en una imagen (backward process) a través de un modelo g de deep learning
3. Una vez entrenado, partiendo de una distribución normal, reconstruimos la información a través de g , pudiendo **generar nuevas muestras** de la distribución $P(\mathbf{X})$

Forward process

El principal objetivo del forward process es **destruir información** en la imagen de entrada de **forma progresiva**

$$\begin{aligned}\mathbf{z}_1 &= \sqrt{1 - \beta_1} \cdot \mathbf{x} + \sqrt{\beta_1} \cdot \epsilon_1 \\ \mathbf{z}_t &= \sqrt{1 - \beta_t} \cdot \mathbf{z}_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t \quad \forall t \in 2, \dots, T\end{aligned}$$

El ruido se obtiene a través de una normal estándar:

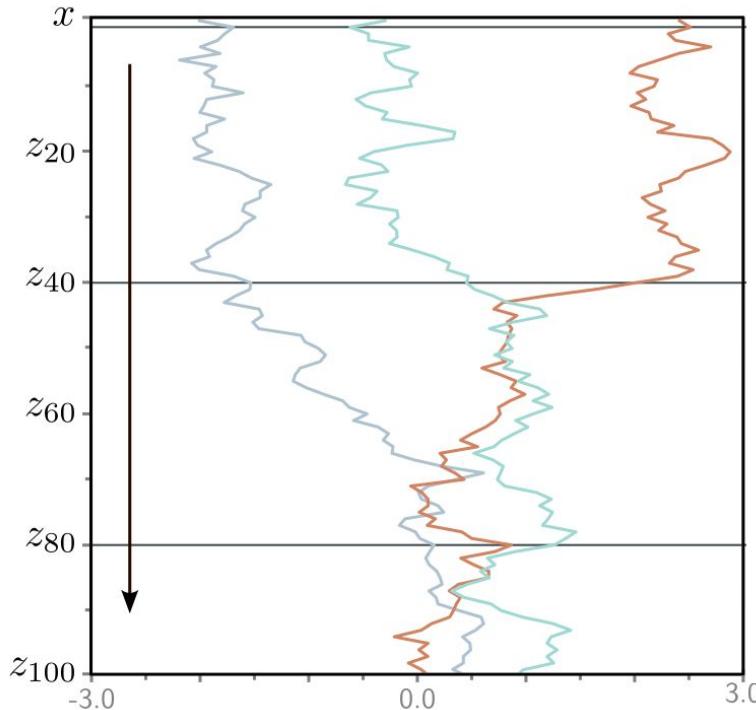
$$\epsilon_t \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

Y β_t se conoce como **noise schedule**. Este parámetro controla cómo de rápido nos movemos hacia la normal:

$$\beta_t \in [0, 1]$$

Forward process (visualización)

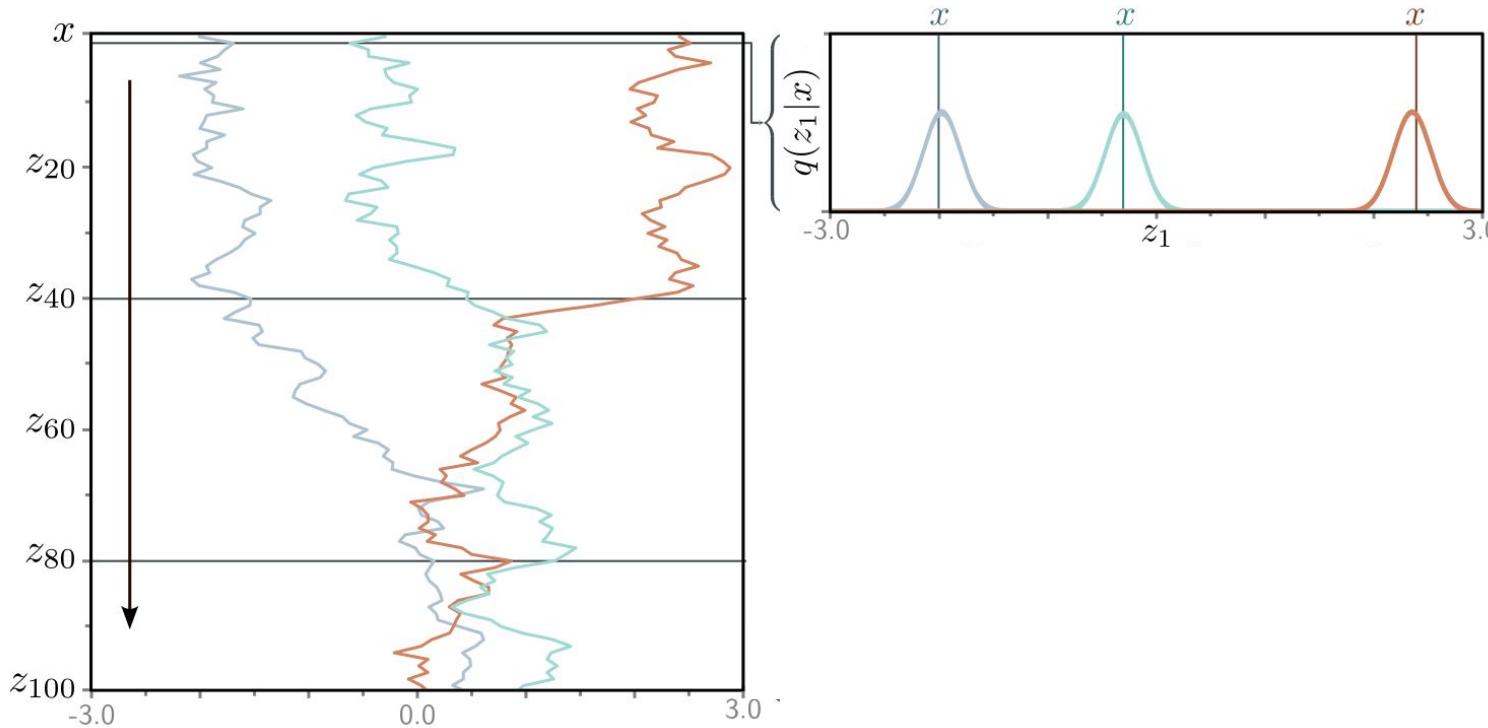
Imaginemos una variable $x \in \mathbb{R}$ a la que le aplicamos el forward process a través de 100 variables latentes $T = 100$



A medida que **avanzamos en el forward process**, esta variable se aproxima a una **normal estándar**

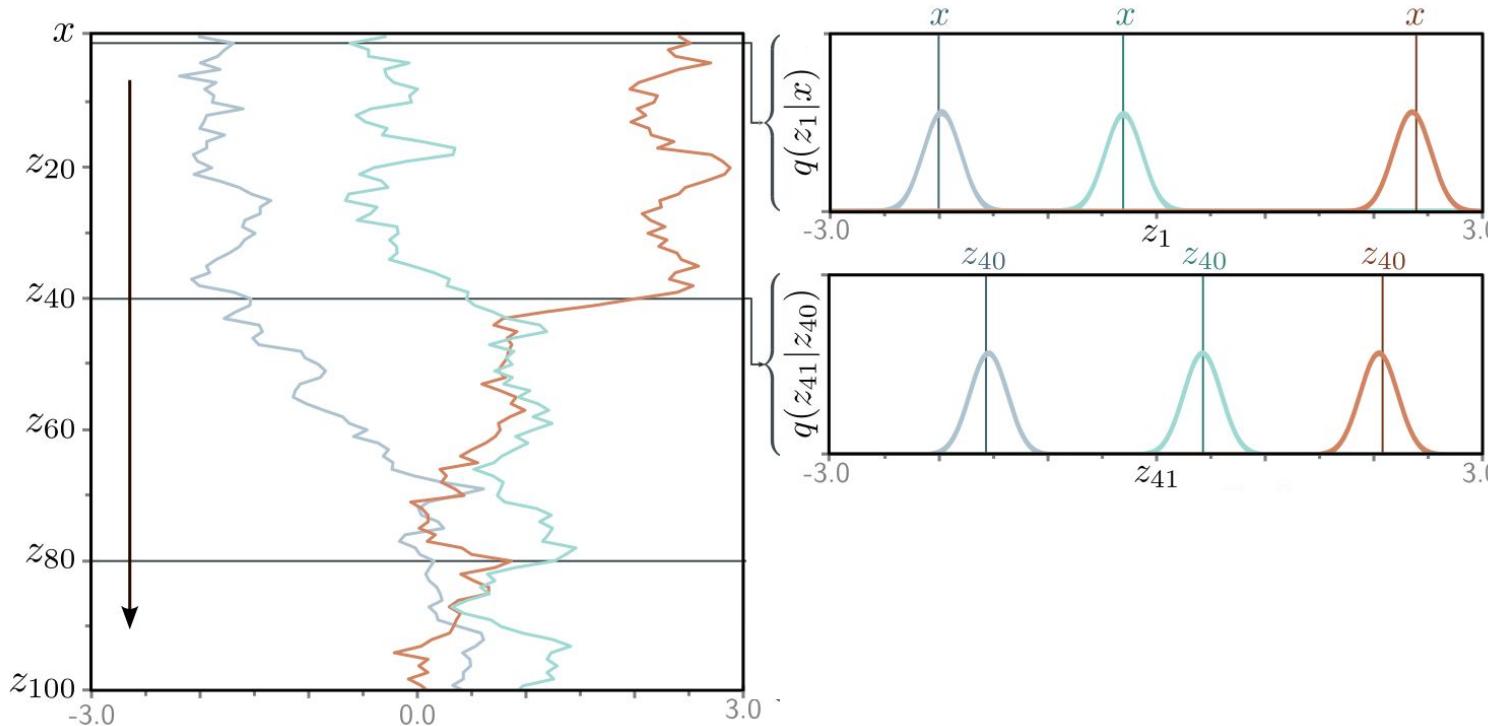
Forward process (visualización)

Imaginemos una variable $x \in \mathbb{R}$ a la que le aplicamos el forward process a través de 100 variables latentes $T = 100$



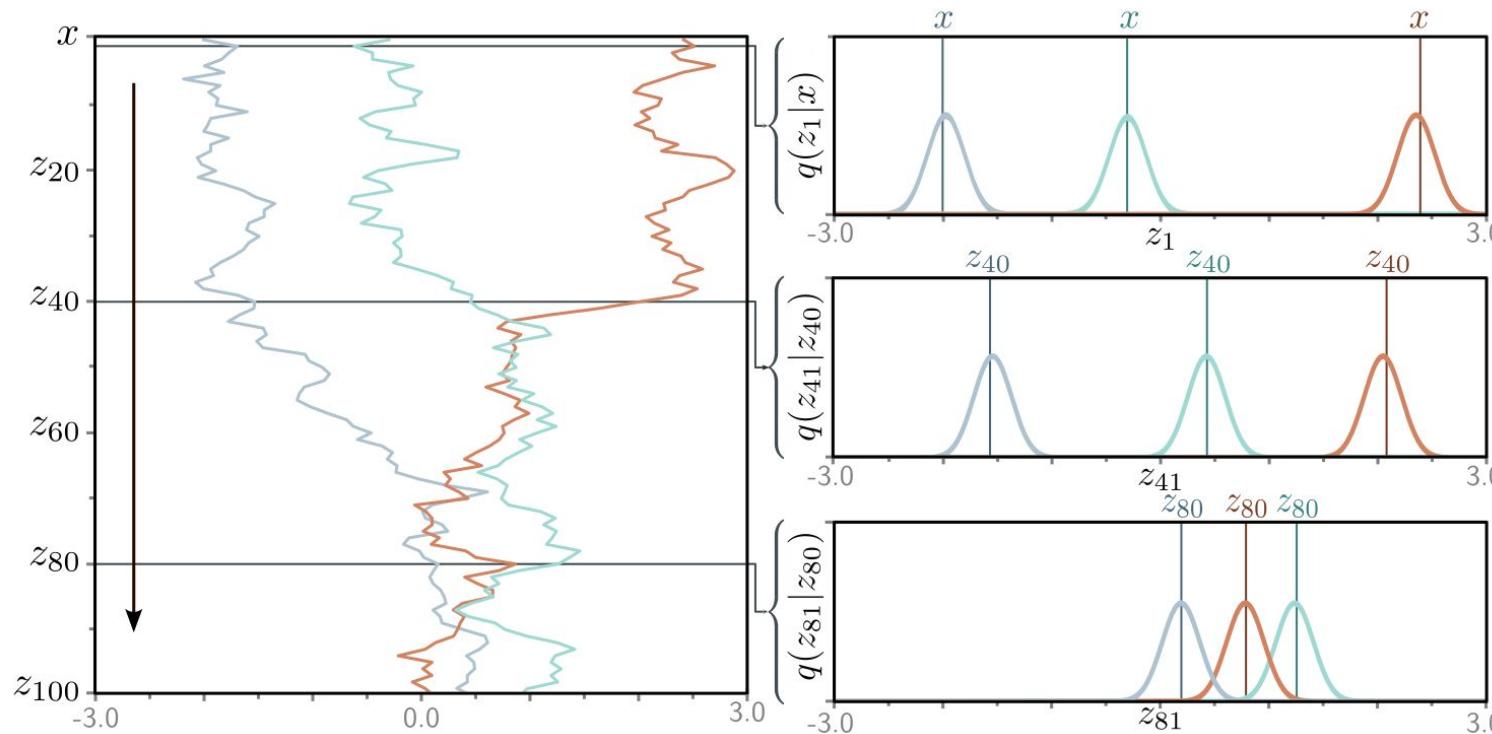
Forward process (visualización)

Imaginemos una variable $x \in \mathbb{R}$ a la que le aplicamos el forward process a través de 100 variables latentes $T = 100$



Forward process (visualización)

Imaginemos una variable $x \in \mathbb{R}$ a la que le aplicamos el forward process a través de 100 variables latentes $T = 100$



Probabilidad conjunta

El proceso del forward diffusion sigue un **proceso de Markov** que se puede expresar de la siguiente manera:

$$\begin{aligned} q(\mathbf{z}_1 \mid \mathbf{x}) &= \text{Norm}_{\mathbf{z}_1}[\sqrt{1 - \beta_1}\mathbf{x}, \beta_1\mathbf{I}] \\ q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) &= \text{Norm}_{\mathbf{z}_t}[\sqrt{1 - \beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}] \quad \forall t \in 2, \dots, T \end{aligned}$$

Probabilidad conjunta

El proceso del forward diffusion sigue un **proceso de Markov** que se puede expresar de la siguiente manera:

$$\begin{aligned} q(\mathbf{z}_1 \mid \mathbf{x}) &= \text{Norm}_{\mathbf{z}_1}[\sqrt{1 - \beta_1}\mathbf{x}, \beta_1\mathbf{I}] \\ q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) &= \text{Norm}_{\mathbf{z}_t}[\sqrt{1 - \beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}] \quad \forall t \in 2, \dots, T \end{aligned}$$

Por lo tanto, con una **distribución conjunta** con la siguiente forma:

$$q(\mathbf{z}_{1\dots T} \mid \mathbf{x}) = q(\mathbf{x} \mid \mathbf{z}_1) \prod_{t=2}^T q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$$

Probabilidad conjunta

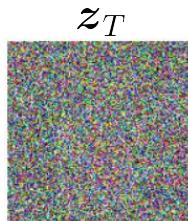
El proceso del forward diffusion sigue un **proceso de Markov** que se puede expresar de la siguiente manera:

$$\begin{aligned} q(\mathbf{z}_1 \mid \mathbf{x}) &= \text{Norm}_{\mathbf{z}_1}[\sqrt{1 - \beta_1}\mathbf{x}, \beta_1\mathbf{I}] \\ q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) &= \text{Norm}_{\mathbf{z}_t}[\sqrt{1 - \beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}] \quad \forall t \in 2, \dots, T \end{aligned}$$

Por lo tanto, con una **distribución conjunta** con la siguiente forma:

$$q(\mathbf{z}_{1\dots T} \mid \mathbf{x}) = q(\mathbf{x} \mid \mathbf{z}_1) \prod_{t=2}^T q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$$

Con un **número suficiente de pasos T** **destruimos toda la información en la imagen**, o lo que es lo mismo:



$$q(\mathbf{z}_t) \sim \mathcal{N}(0, \sigma^2\mathbf{I})$$

Diffusion kernel

Para generar cualquier \mathbf{z}_t es necesario partir de \mathbf{x} y añadir ruido progresivamente. Sin embargo, para valores altos de t , este proceso puede resultar bastante costoso. Afortunadamente, podemos optimizarlo utilizando el **diffusion kernel**.

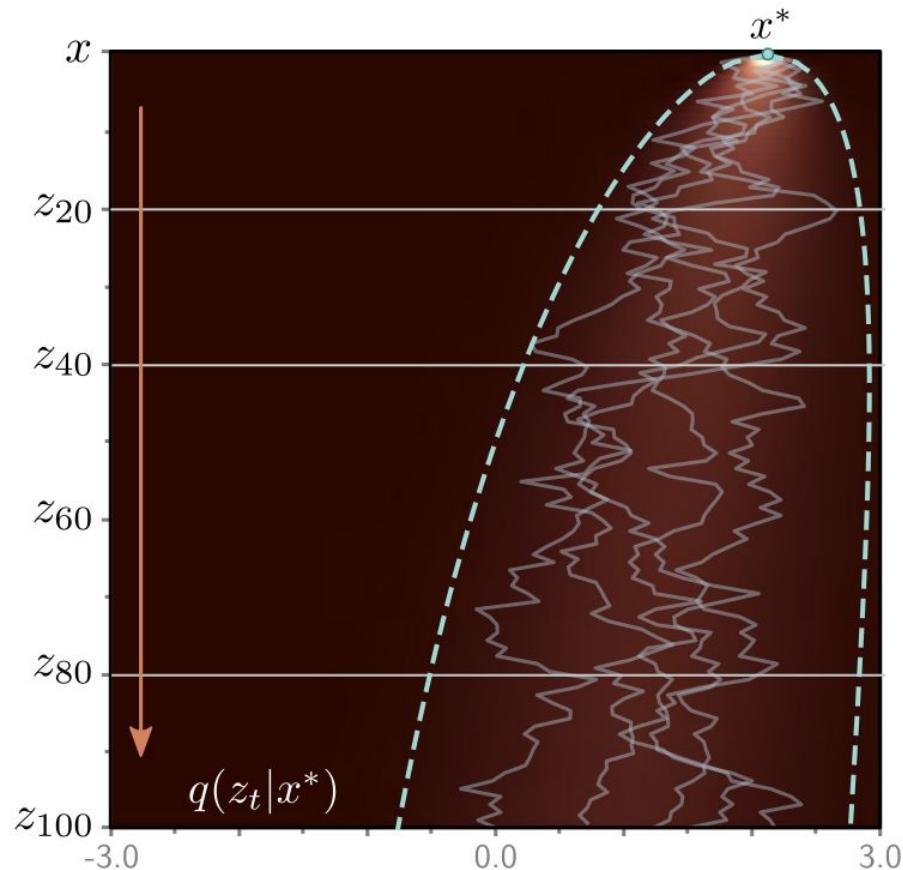
$$\mathbf{z}_t = \sqrt{\alpha_t} \cdot \mathbf{x} + \sqrt{1 - \alpha_t} \cdot \epsilon_t \quad \alpha_t = \prod_{s=1}^t 1 - \beta_s$$

O lo que es lo mismo:

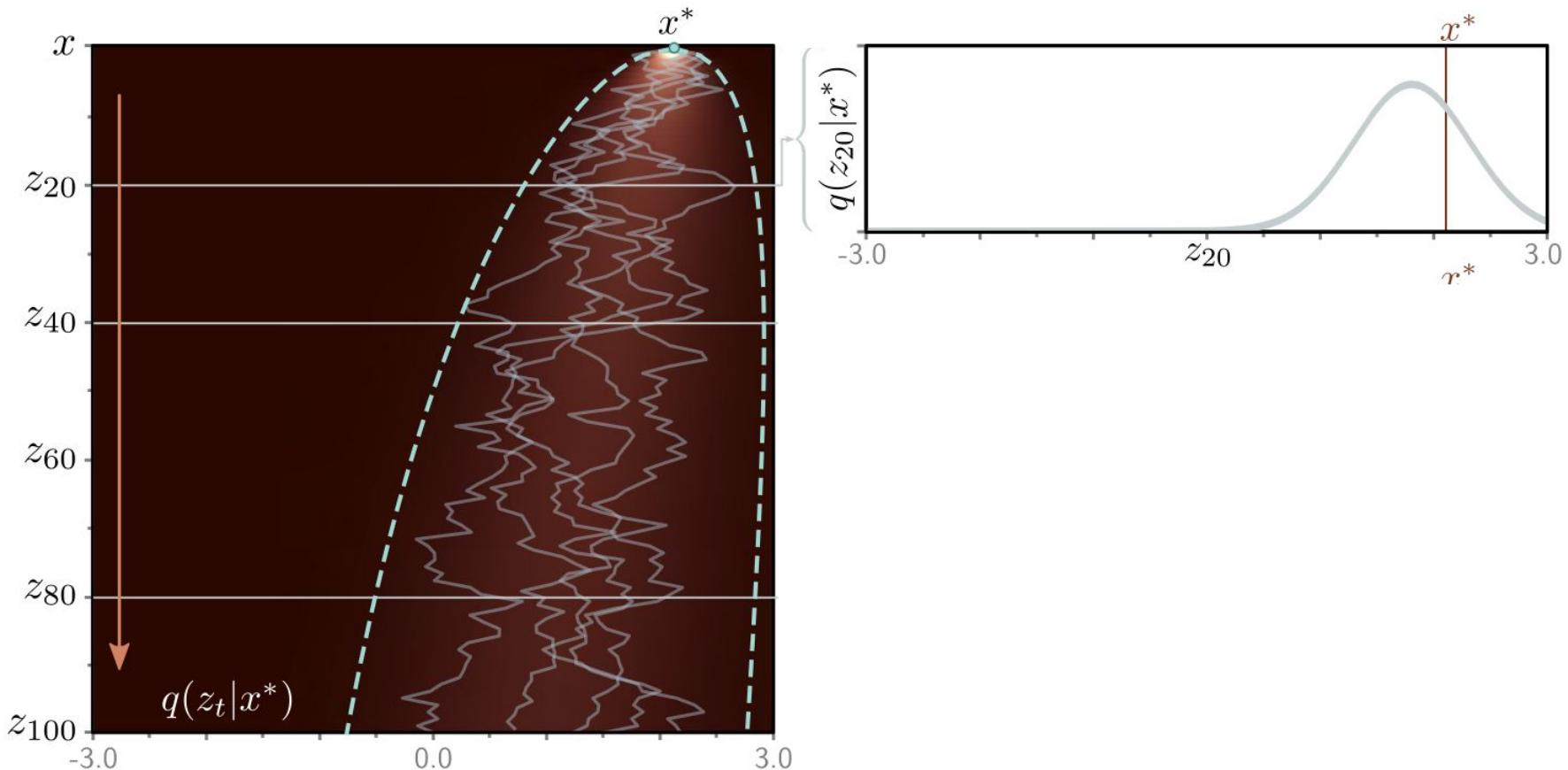
$$q(\mathbf{z}_t \mid \mathbf{x}) = Norm_{\mathbf{z}_t}[\sqrt{\alpha_t} \cdot \mathbf{x}, (1 - \alpha_t)\mathbf{I}]$$

Esto nos permite obtener cualquier variable latente sin necesidad de proceder de forma iterativa desde la variable de entrada

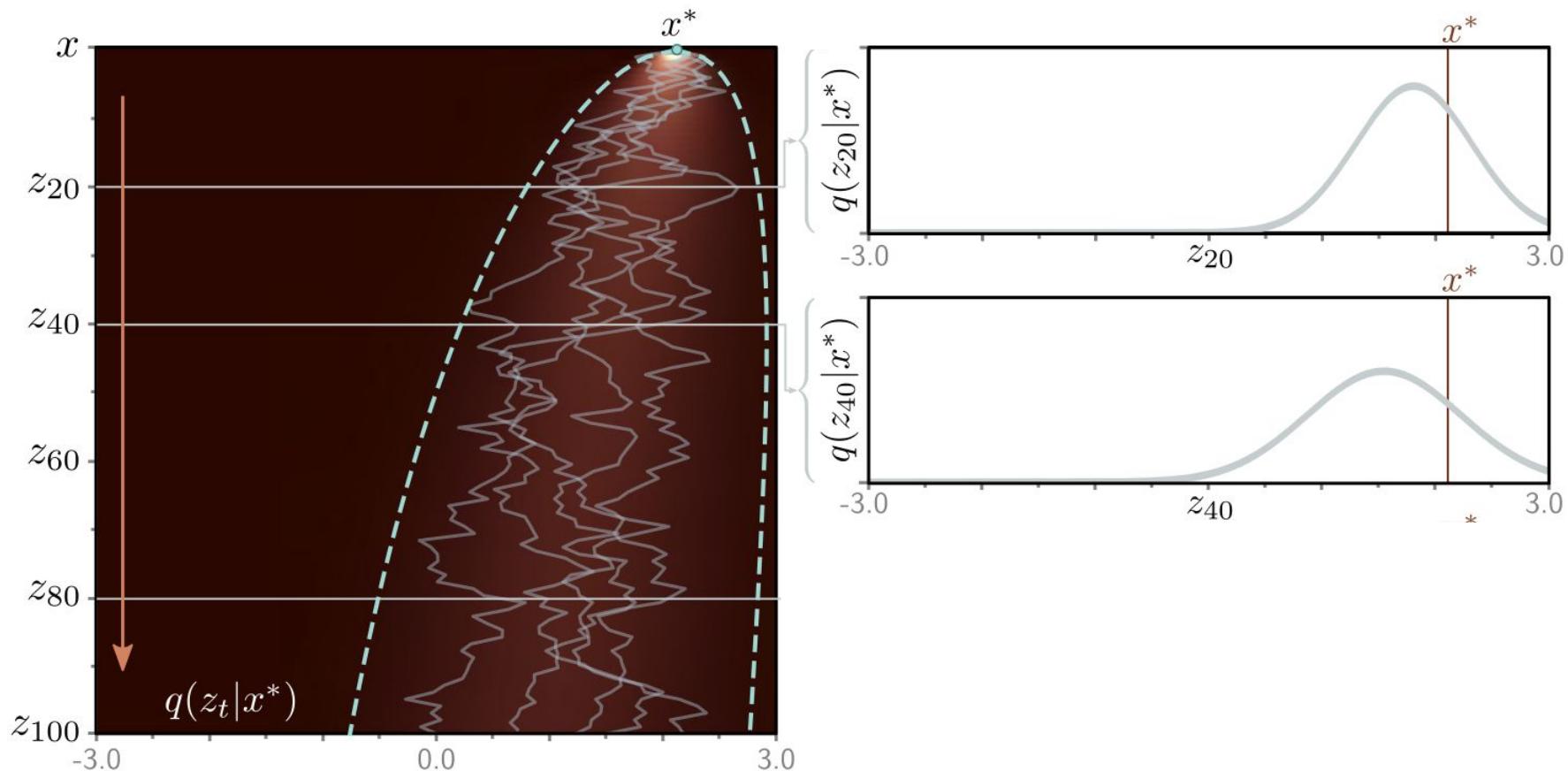
Diffusion kernel (visualización)



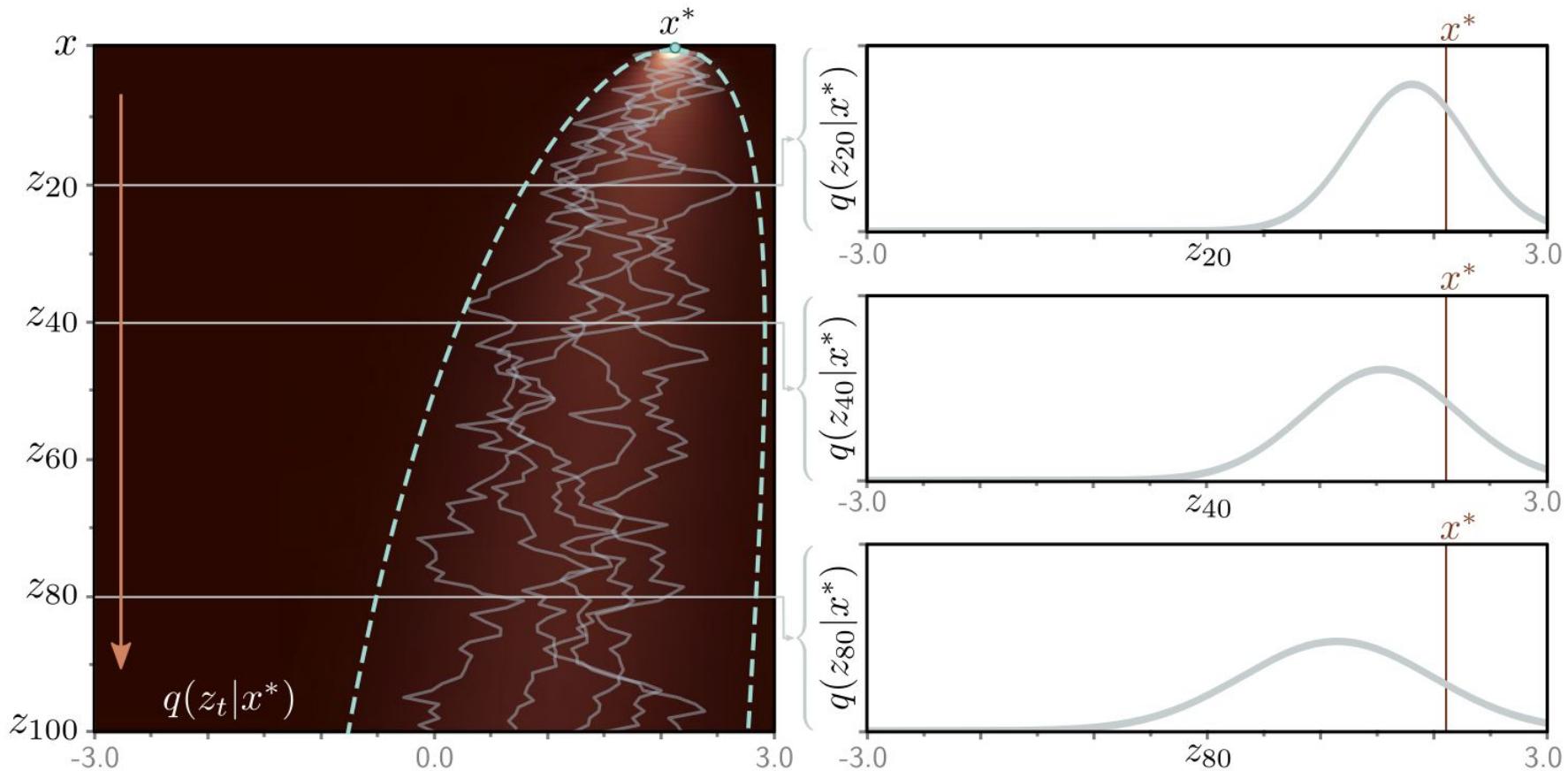
Diffusion kernel (visualización)



Diffusion kernel (visualización)



Diffusion kernel (visualización)



Distribución condicionada

Para **revertir** el proceso de **destrucción de información** necesitamos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$

Distribución condicionada

Para **revertir** el proceso de **destrucción de información** necesitamos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$

Teorema de Bayes

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) =$$

Distribución condicionada

Para **revertir** el proceso de **destrucción de información** necesitamos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$

Teorema de Bayes

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1})q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}$$

Distribución condicionada

Para **revertir** el proceso de **destrucción de información** necesitamos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$

Teorema de Bayes

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}$$

Intractable

Distribución condicionada

Para **revertir** el proceso de **destrucción de información** necesitamos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$

Teorema de Bayes

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}$$

Intractable

Ya que

$$q(\mathbf{z}_{t-1}) = \int q(\mathbf{z}_{t-1} \mid \mathbf{x}) P(\mathbf{x}) d\mathbf{x}$$

Distribución condicionada

Para **revertir** el proceso de **destrucción de información** necesitamos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$

Teorema de Bayes

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}$$

Intractable

Ya que

$$q(\mathbf{z}_{t-1}) = \int q(\mathbf{z}_{t-1} \mid \mathbf{x}) P(\mathbf{x}) d\mathbf{x}$$



No es posible conocer la distribución de $P(\mathbf{X})$

Distribución condicionada

Sin embargo, si conocemos la \mathbf{x} sobre la que partimos, podemos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$

Distribución condicionada

Sin embargo, si conocemos la \mathbf{x} sobre la que partimos, podemos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) =$$

Distribución condicionada

Sin embargo, si conocemos la \mathbf{x} sobre la que partimos, podemos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$

Teorema de Bayes



$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})}{q(\mathbf{z}_t \mid \mathbf{x})}$$

Distribución condicionada

Sin embargo, si conocemos la \mathbf{x} sobre la que partimos, podemos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$

Teorema de Bayes



$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})}{q(\mathbf{z}_t \mid \mathbf{x})} \propto q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})$$

Distribución condicionada

Sin embargo, si conocemos la \mathbf{x} sobre la que partimos, podemos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$

Teorema de Bayes

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})}{q(\mathbf{z}_t \mid \mathbf{x})} \propto \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})}{\downarrow \text{Markov}} \\ q(\mathbf{z}_t \mid \mathbf{z}_{t-1})q(\mathbf{z}_{t-1} \mid \mathbf{x})$$

Distribución condicionada

Sin embargo, si conocemos la \mathbf{x} sobre la que partimos, podemos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$

Teorema de Bayes

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})}{q(\mathbf{z}_t \mid \mathbf{x})} \propto \underbrace{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})}_{\text{Markov}}$$

$$\frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1})}{\text{Diffusion kernel}} \frac{q(\mathbf{z}_{t-1} \mid \mathbf{x})}{\text{Diffusion kernel}}$$

Distribución condicionada

Sin embargo, si conocemos la \mathbf{x} sobre la que partimos, podemos conocer $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$

Teorema de Bayes

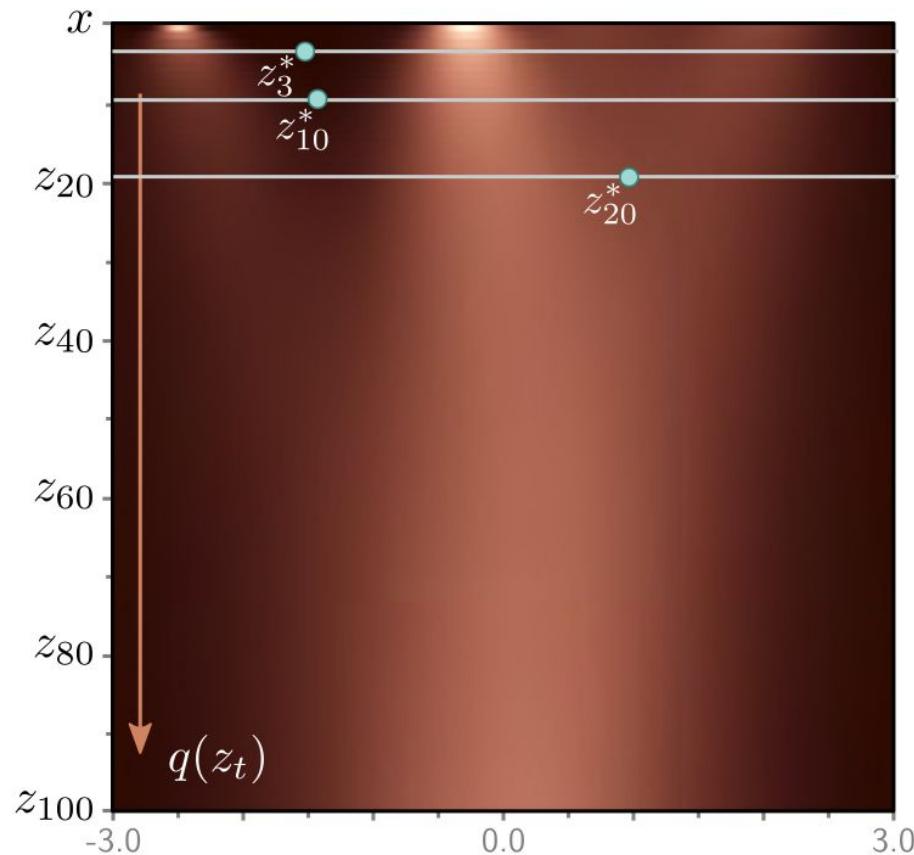
$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1} \mid \mathbf{x})}{q(\mathbf{z}_t \mid \mathbf{x})}$$

Markov

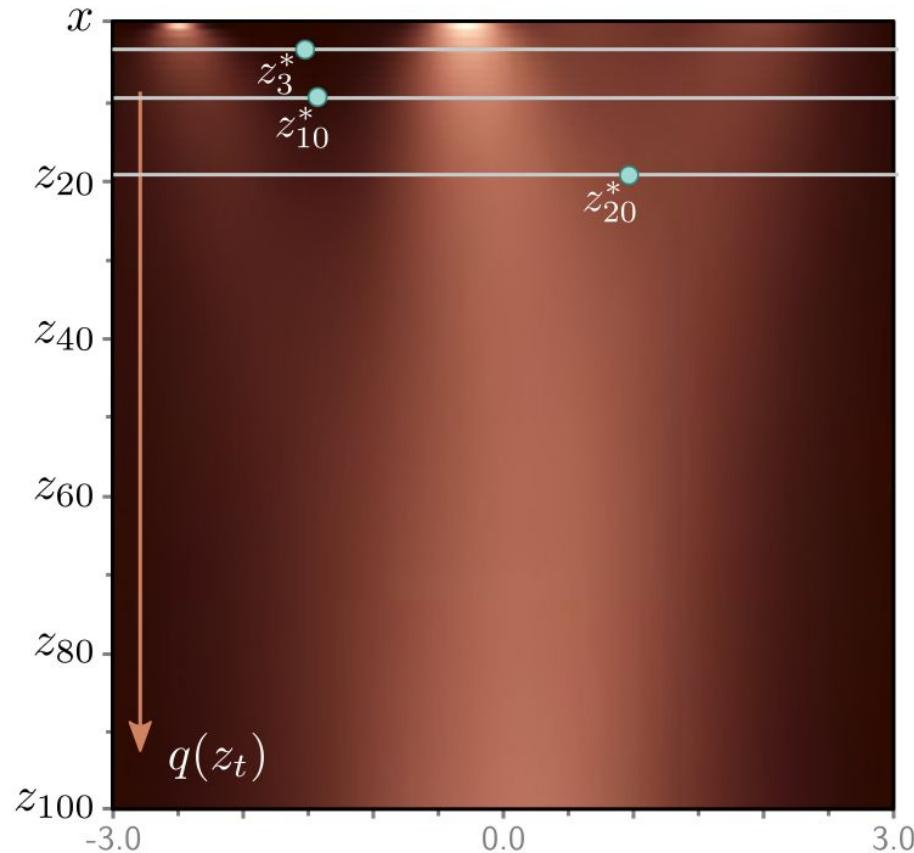
$$\frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1})q(\mathbf{z}_{t-1} \mid \mathbf{x})}{\text{Diffusion kernel}}$$

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = \text{Norm}_{\mathbf{z}_{t-1}} \left[\frac{(1 - \alpha_{t-1})}{1 - \alpha_t} \sqrt{1 - \beta_t} \mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}} \beta_t}{1 - \alpha_t} \mathbf{x}, \frac{\beta_t(1 - \alpha_{t-1})}{1 - \alpha_t} \mathbf{I} \right]$$

Distribución condicionada (visualización)

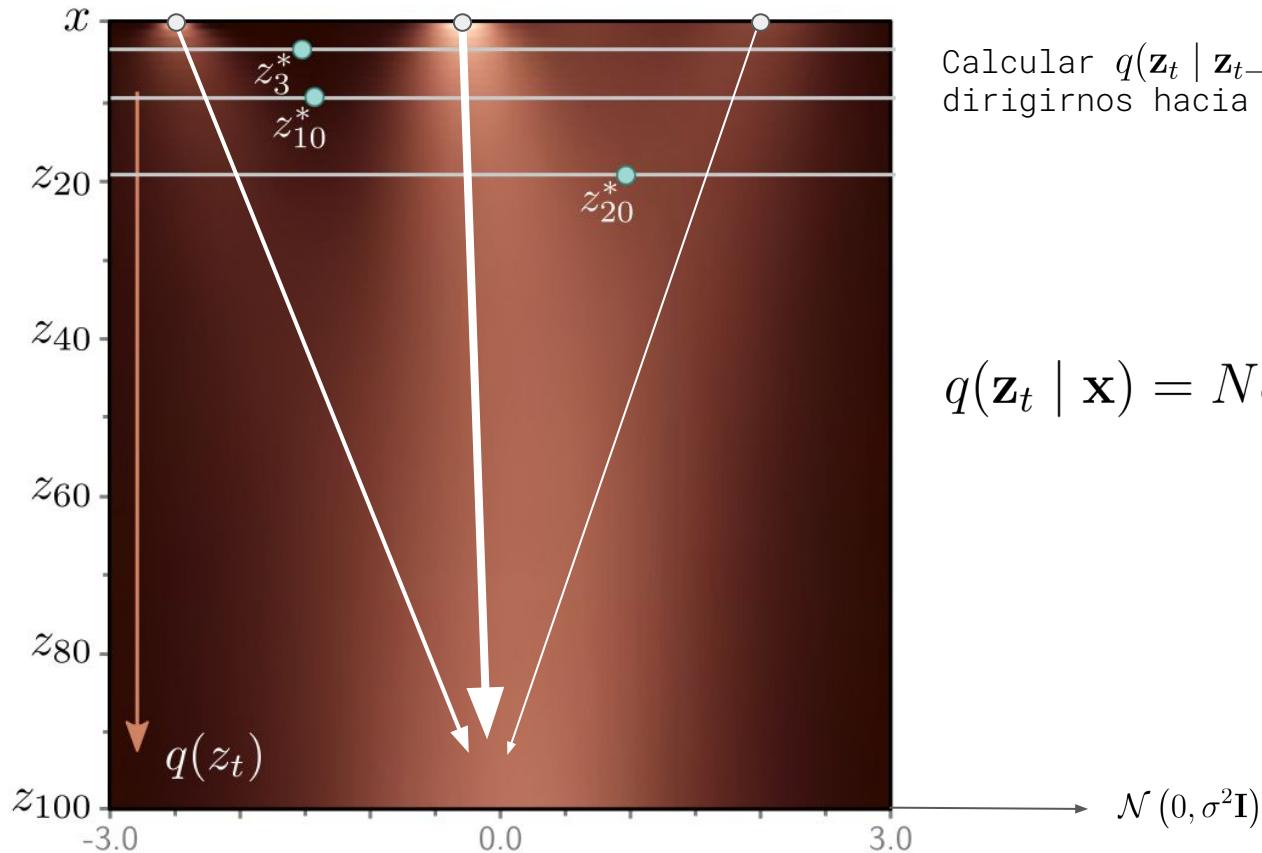


Distribución condicionada (visualización)



Calcular $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ es sencillo, basta con dirigirnos hacia $\mathcal{N}(0, \sigma^2 \mathbf{I})$

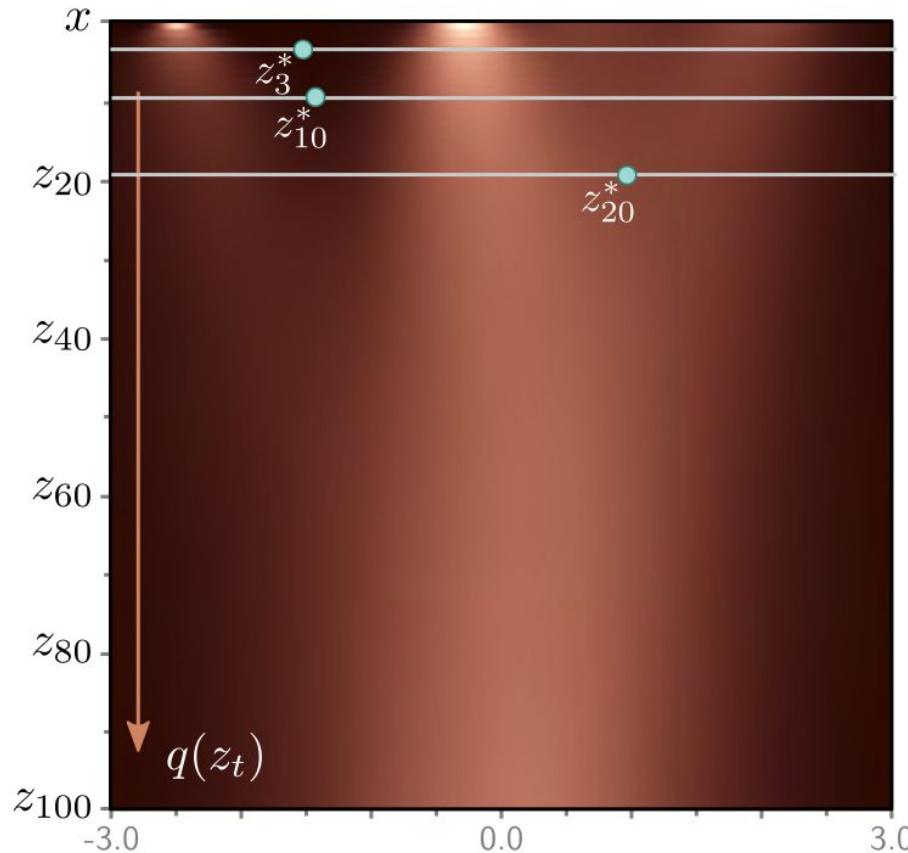
Distribución condicionada (visualización)



Calcular $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ es sencillo, basta con dirigirnos hacia $\mathcal{N}(0, \sigma^2 \mathbf{I})$

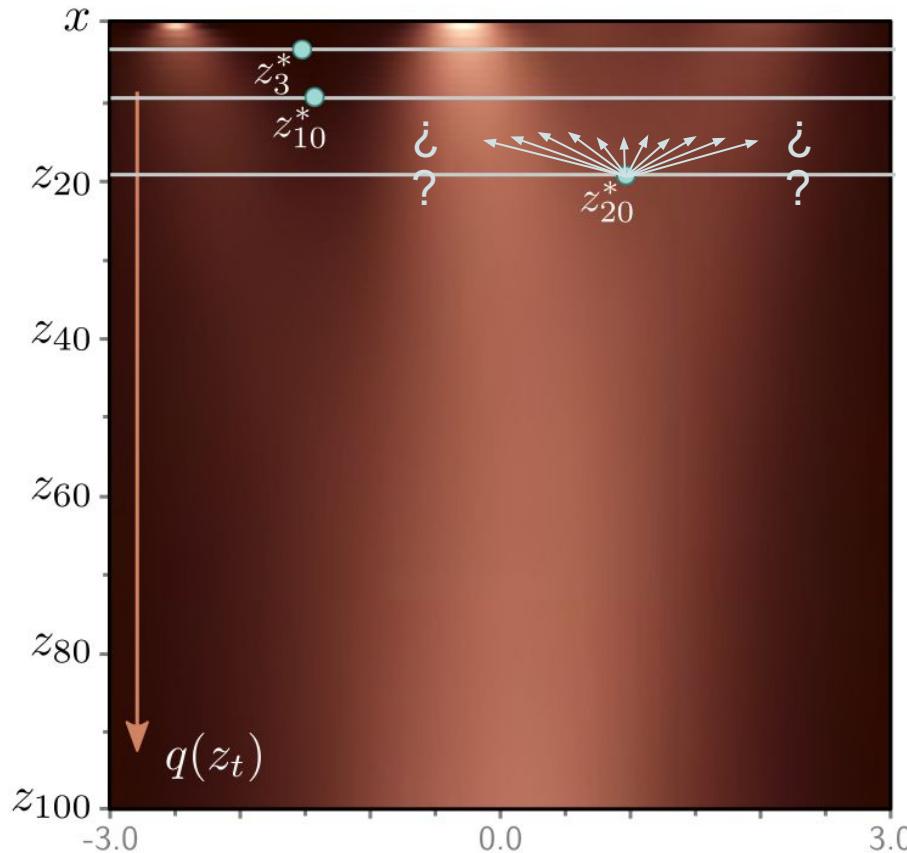
$$q(\mathbf{z}_t \mid \mathbf{x}) = \text{Norm}_{\mathbf{z}_t}[\sqrt{\alpha_t} \cdot \mathbf{x}, (1 - \alpha_t)\mathbf{I}]$$

Distribución condicionada (visualización)



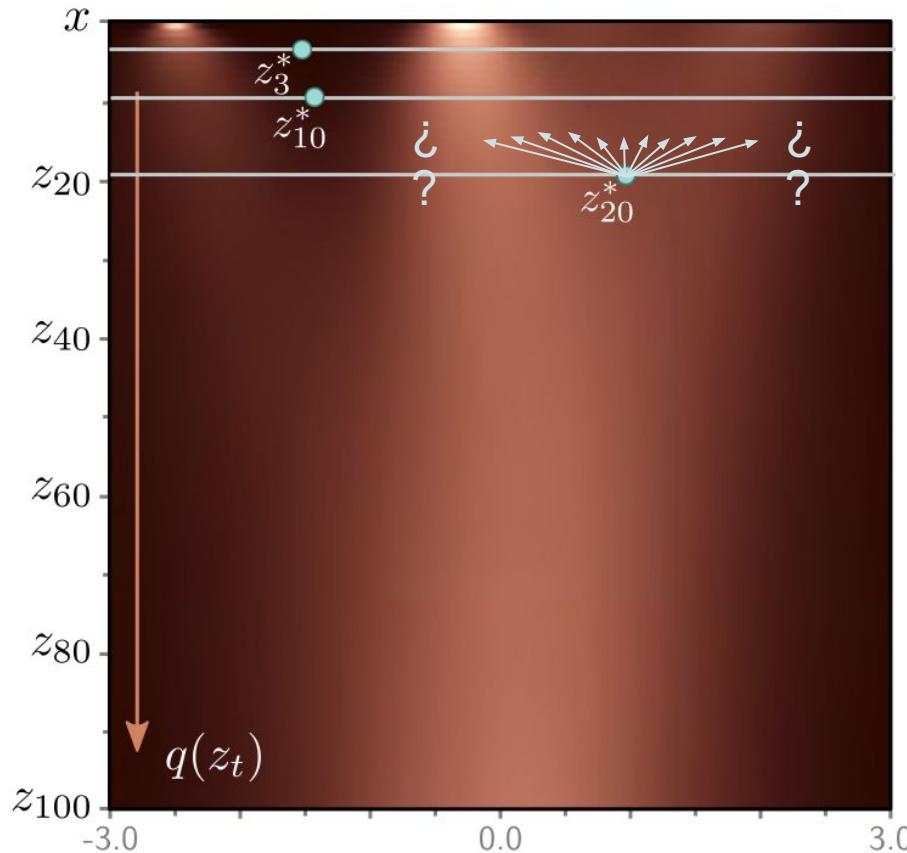
Sin embargo, el camino opuesto $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$ no es tan sencillo de trazar

Distribución condicionada (visualización)



Sin embargo, el camino opuesto $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ no es tan sencillo de trazar

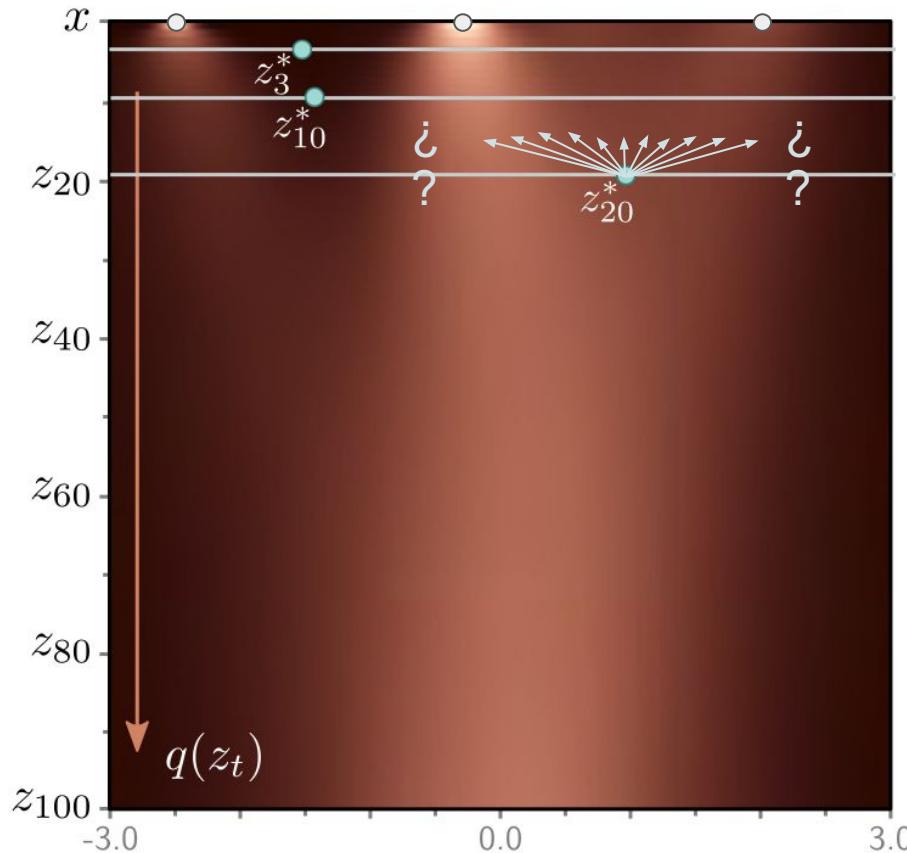
Distribución condicionada (visualización)



Sin embargo, el camino opuesto $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ no es tan sencillo de trazar

De hecho, solo le podemos trazar si conocemos $P(\mathbf{X})$

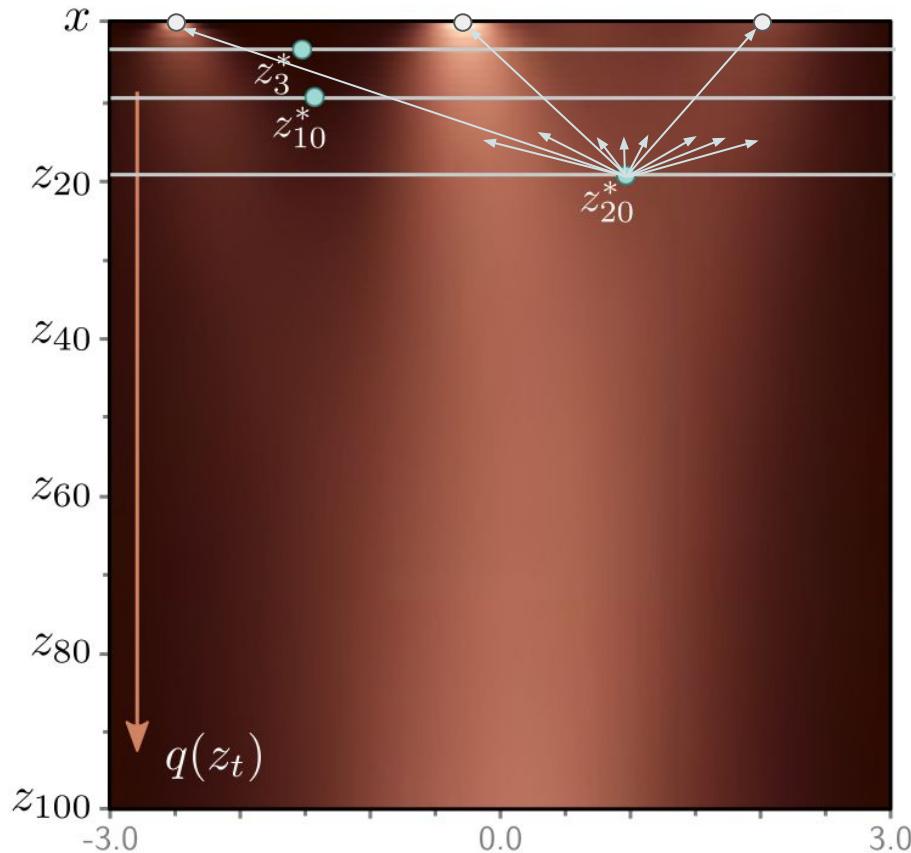
Distribución condicionada (visualización)



Sin embargo, el camino opuesto $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ no es tan sencillo de trazar

De hecho, solo le podemos trazar si conocemos $P(\mathbf{X})$

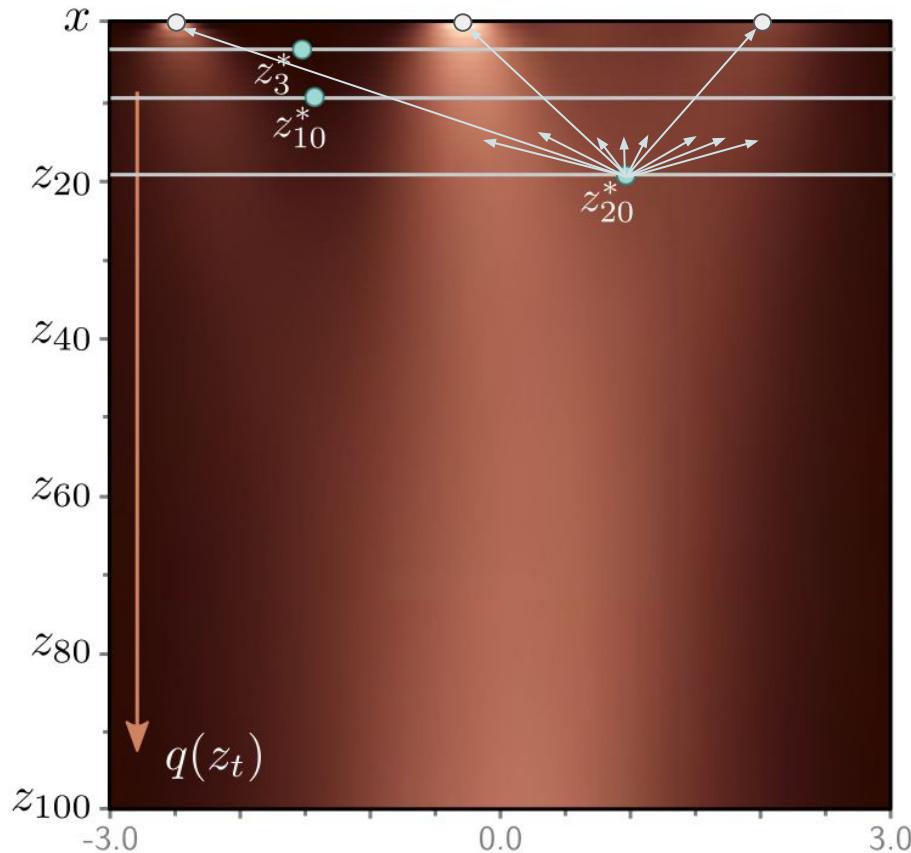
Distribución condicionada (visualización)



Sin embargo, el camino opuesto $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ no es tan sencillo de trazar

De hecho, solo le podemos trazar si conocemos $P(\mathbf{X})$

Distribución condicionada (visualización)



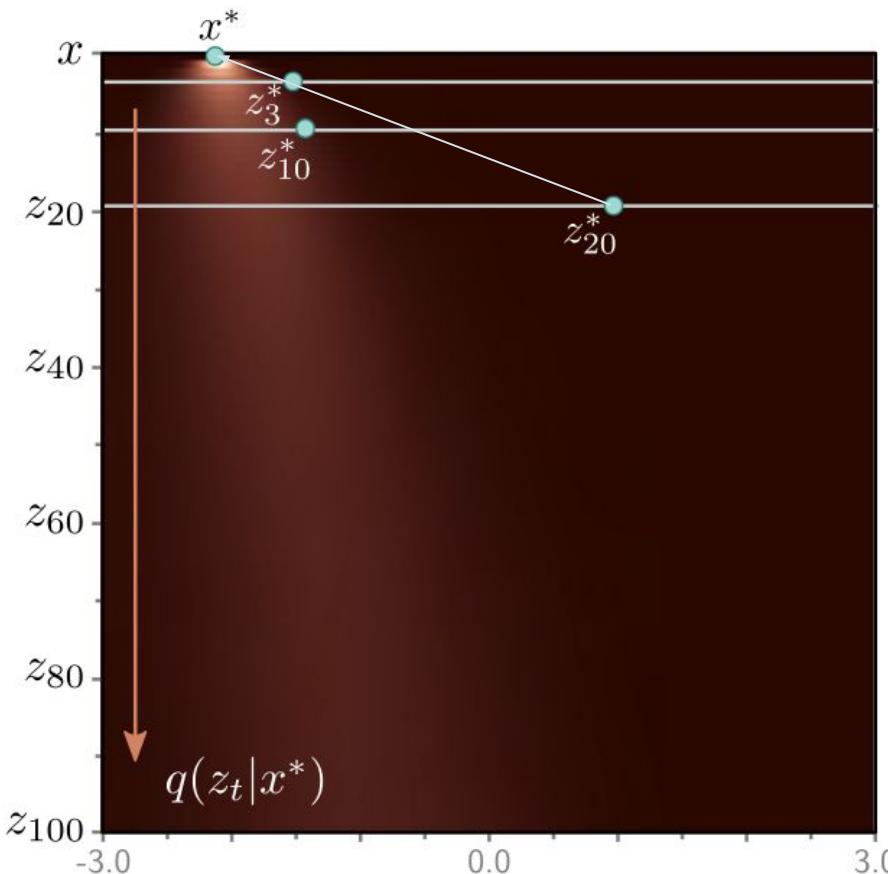
Sin embargo, el camino opuesto $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$ no es tan sencillo de trazar

De hecho, solo le podemos trazar si conocemos $P(\mathbf{X})$

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1})q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}$$

$$q(\mathbf{z}_{t-1}) = \int q(\mathbf{z}_{t-1} \mid \mathbf{x})P(\mathbf{x}) d\mathbf{x}$$

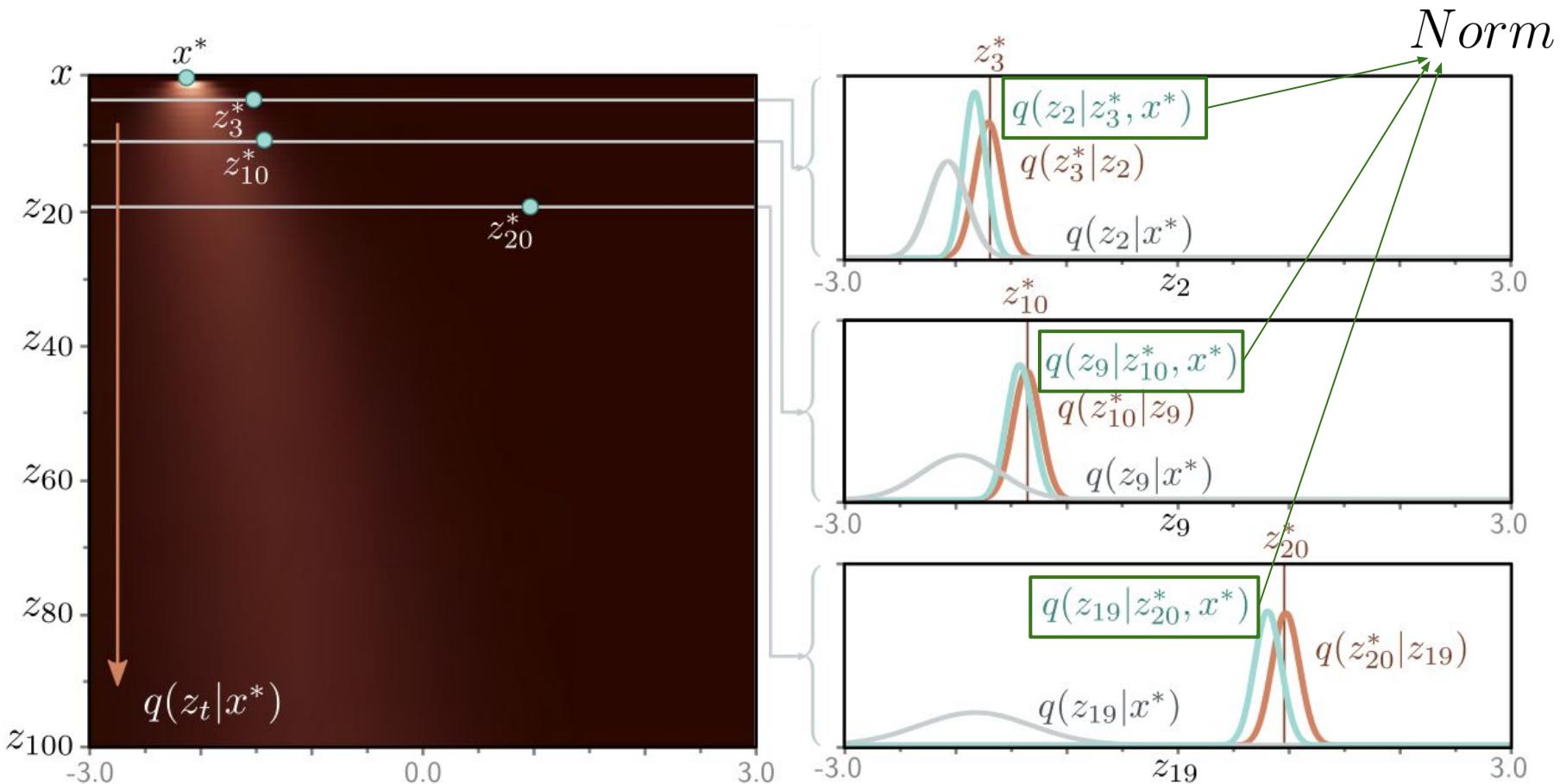
Distribución condicionada (visualización)



Si condicionamos a una \mathbf{x} concreta, el camino a recorrer por $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ queda mucho más claro

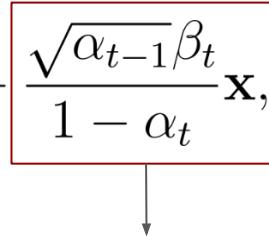
No depende de $P(\mathbf{X})$ sino de $\mathbf{x} \in \mathbf{X}$

Distribución condicionada (visualización)



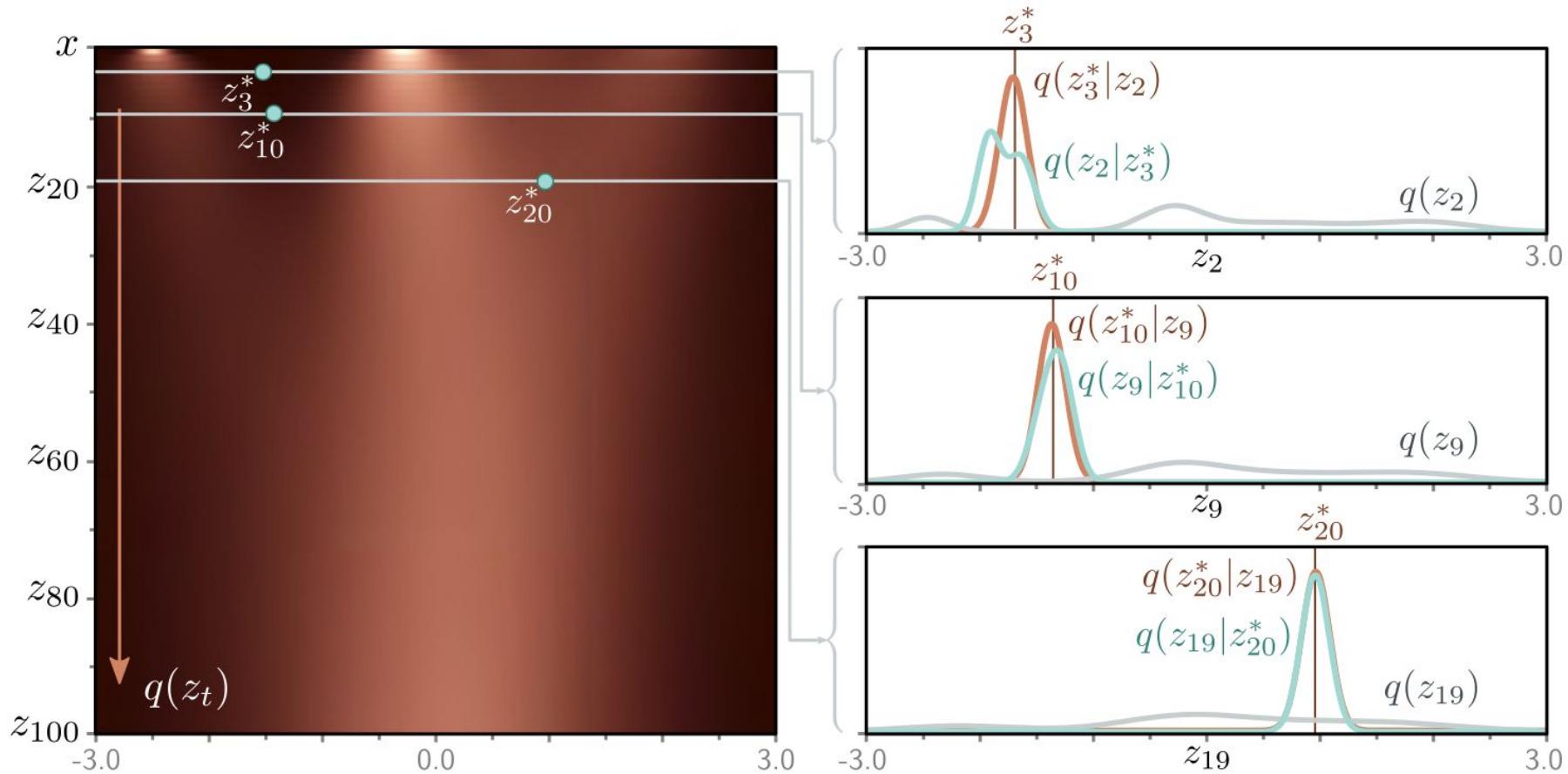
Aproximación final

El **backward process** (reconstrucción de información) se pueden modelar a través de una distribución normal si condicionamos a \mathbf{x} :

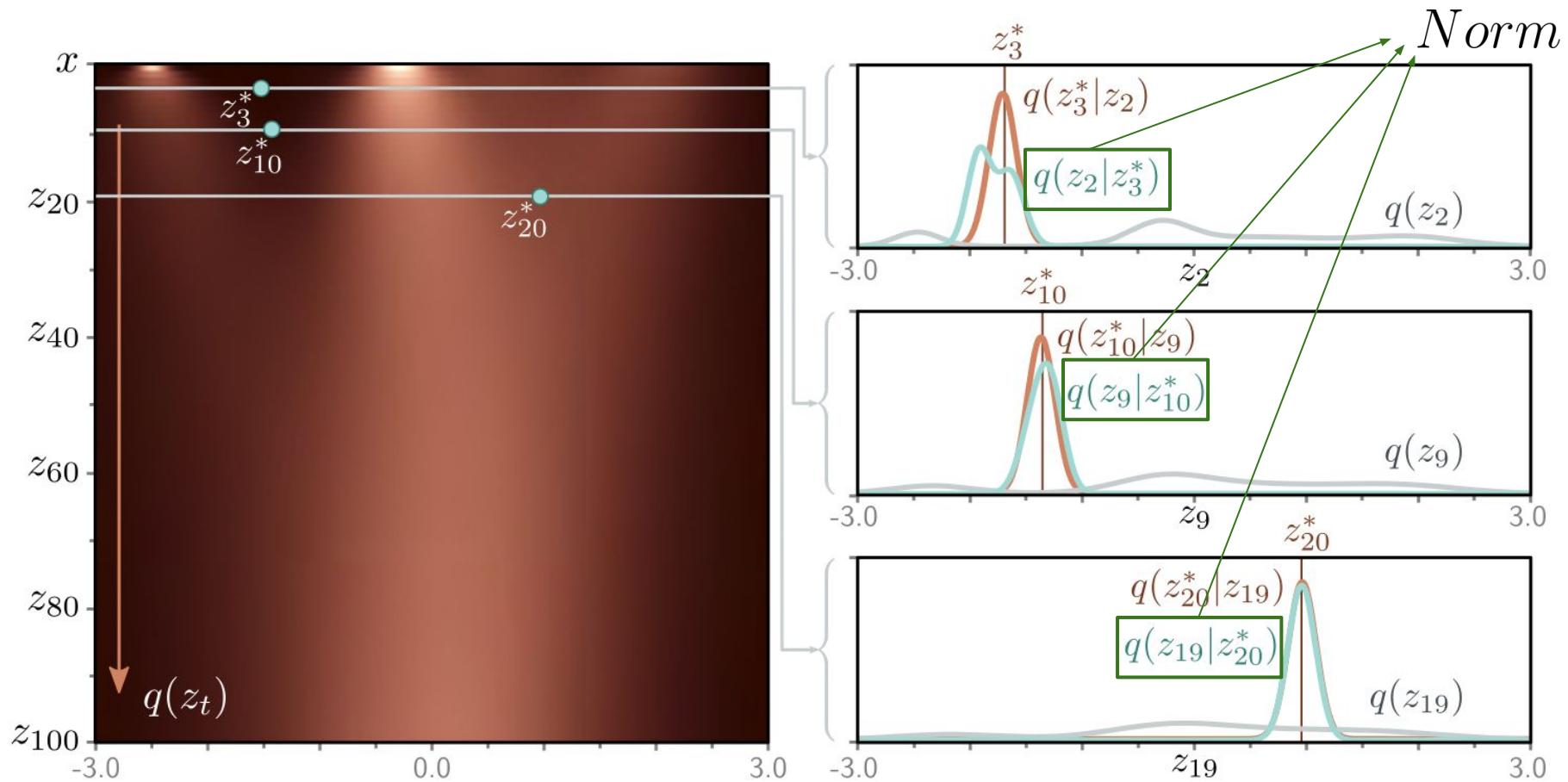
$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = Norm_{\mathbf{z}_{t-1}} \left[\frac{(1 - \alpha_{t-1})}{1 - \alpha_t} \sqrt{1 - \beta_t} \mathbf{z}_t + \boxed{\frac{\sqrt{\alpha_{t-1}} \beta_t}{1 - \alpha_t} \mathbf{x}, \frac{\beta_t (1 - \alpha_{t-1})}{1 - \alpha_t} \mathbf{I}} \right]$$


Sin embargo, esta dependencia nos va a **evitar generar muestras nuevas**, es decir, samplear de la distribución $P(\mathbf{X})$.

Aproximación final

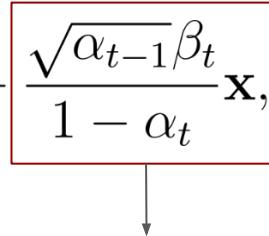


Aproximación final



Aproximación final

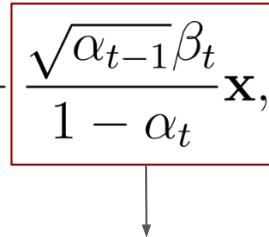
El **backward process** (reconstrucción de información) se pueden modelar a través de una distribución normal si condicionamos a \mathbf{x} :

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = Norm_{\mathbf{z}_{t-1}} \left[\frac{(1 - \alpha_{t-1})}{1 - \alpha_t} \sqrt{1 - \beta_t} \mathbf{z}_t + \boxed{\frac{\sqrt{\alpha_{t-1}} \beta_t}{1 - \alpha_t} \mathbf{x}, \frac{\beta_t (1 - \alpha_{t-1})}{1 - \alpha_t} \mathbf{I}} \right]$$


Sin embargo, esta dependencia nos va a **evitar generar muestras nuevas**, es decir, samplear de la distribución $P(\mathbf{X})$.

Aproximación final

El **backward process** (reconstrucción de información) se pueden modelar a través de una distribución normal si condicionamos a \mathbf{x} :

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) = \text{Norm}_{\mathbf{z}_{t-1}} \left[\frac{(1 - \alpha_{t-1})}{1 - \alpha_t} \sqrt{1 - \beta_t} \mathbf{z}_t + \boxed{\frac{\sqrt{\alpha_{t-1}} \beta_t}{1 - \alpha_t} \mathbf{x}, \frac{\beta_t(1 - \alpha_{t-1})}{1 - \alpha_t} \mathbf{I}} \right]$$


Sin embargo, esta dependencia nos va a **evitar generar muestras nuevas**, es decir, samplear de la distribución $P(\mathbf{X})$.

Si suponemos que el **número de pasos es suficientemente grande**, es decir, $T \gg 1$, y que el **noise schedule satisface** $\beta_t \approx 0$ (cercano a cero) tenemos la siguiente aproximación:

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) \sim \text{Norm}_{\mathbf{z}_{t-1}}$$

Entrenamiento del modelo de difusión

El **forward process** (destrucción de información) se implementa a través del **diffusion kernel**:

$$q(\mathbf{z}_t \mid \mathbf{x}) = Norm_{\mathbf{z}_t}[\sqrt{\alpha_t} \cdot \mathbf{x}, (1 - \alpha_t)\mathbf{I}]$$

El **backward process** (reconstrucción de información) se puede aproximar a través de una distribución normal:

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}) \sim Norm_{\mathbf{z}_{t-1}}$$

Con esto tenemos los ingredientes necesarios para **entrenar** un modelo de difusión

Entrenamiento

Forward process $T = 3$

$\mathbf{x} \in X$



$$\alpha_t = \prod_{s=1}^t 1 - \beta_s$$

Entrenamiento

Forward process $T = 3$

$$\alpha_t = \prod_{s=1}^t 1 - \beta_s$$

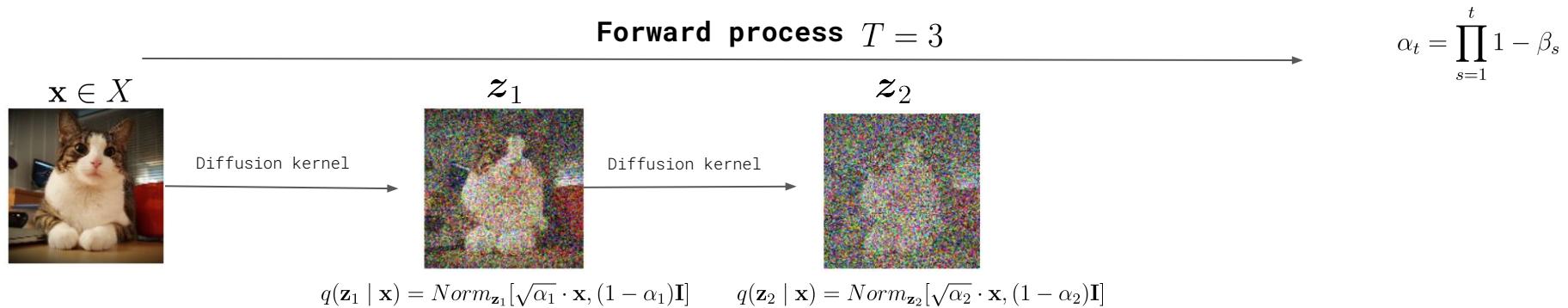


Diffusion kernel

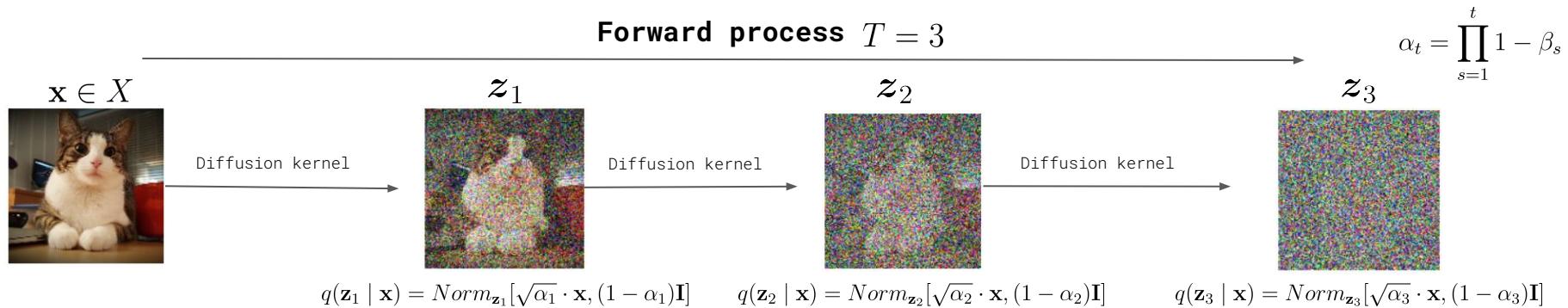


$$q(\mathbf{z}_1 \mid \mathbf{x}) = \text{Norm}_{\mathbf{z}_1}[\sqrt{\alpha_1} \cdot \mathbf{x}, (1 - \alpha_1)\mathbf{I}]$$

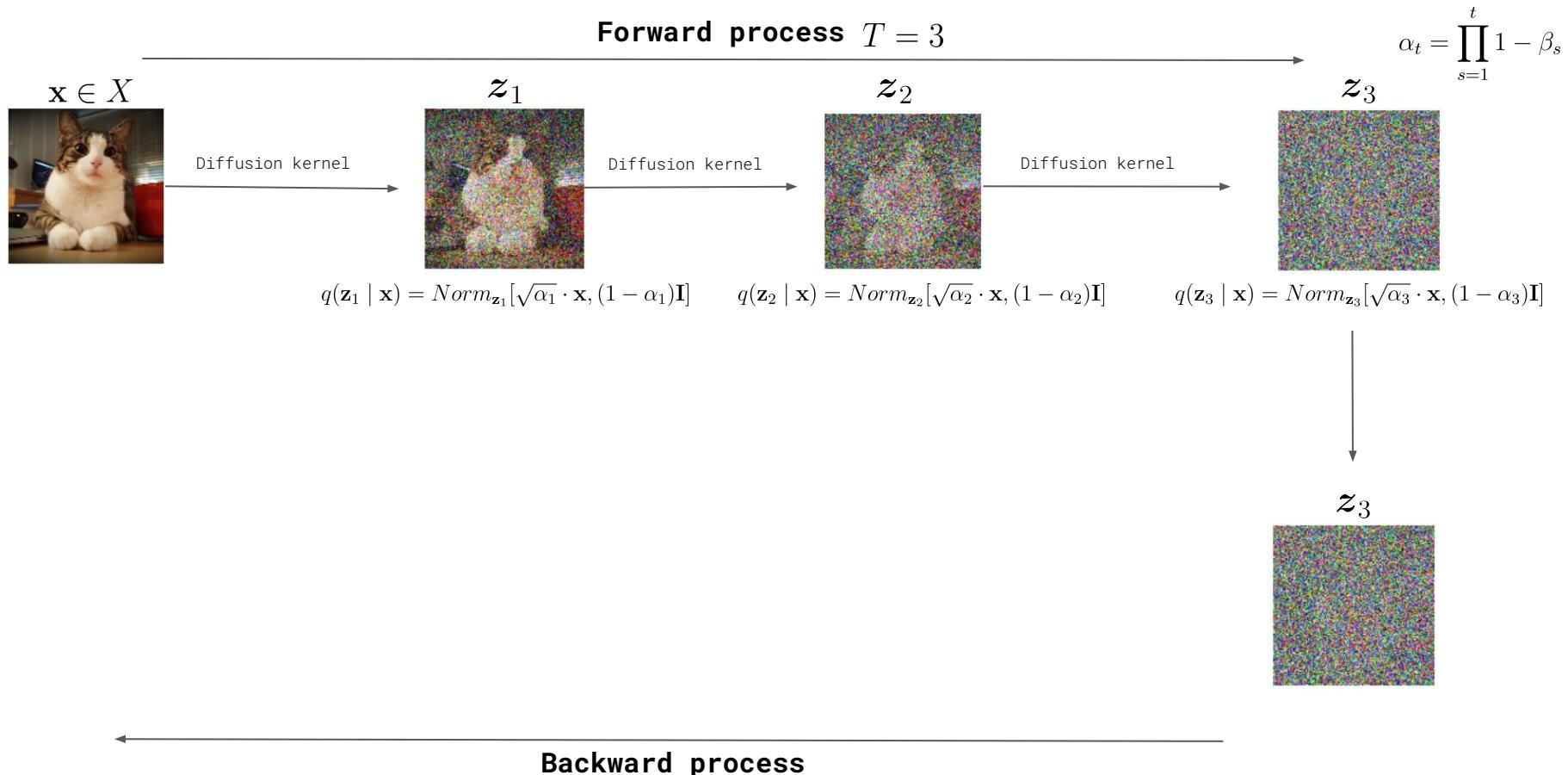
Entrenamiento



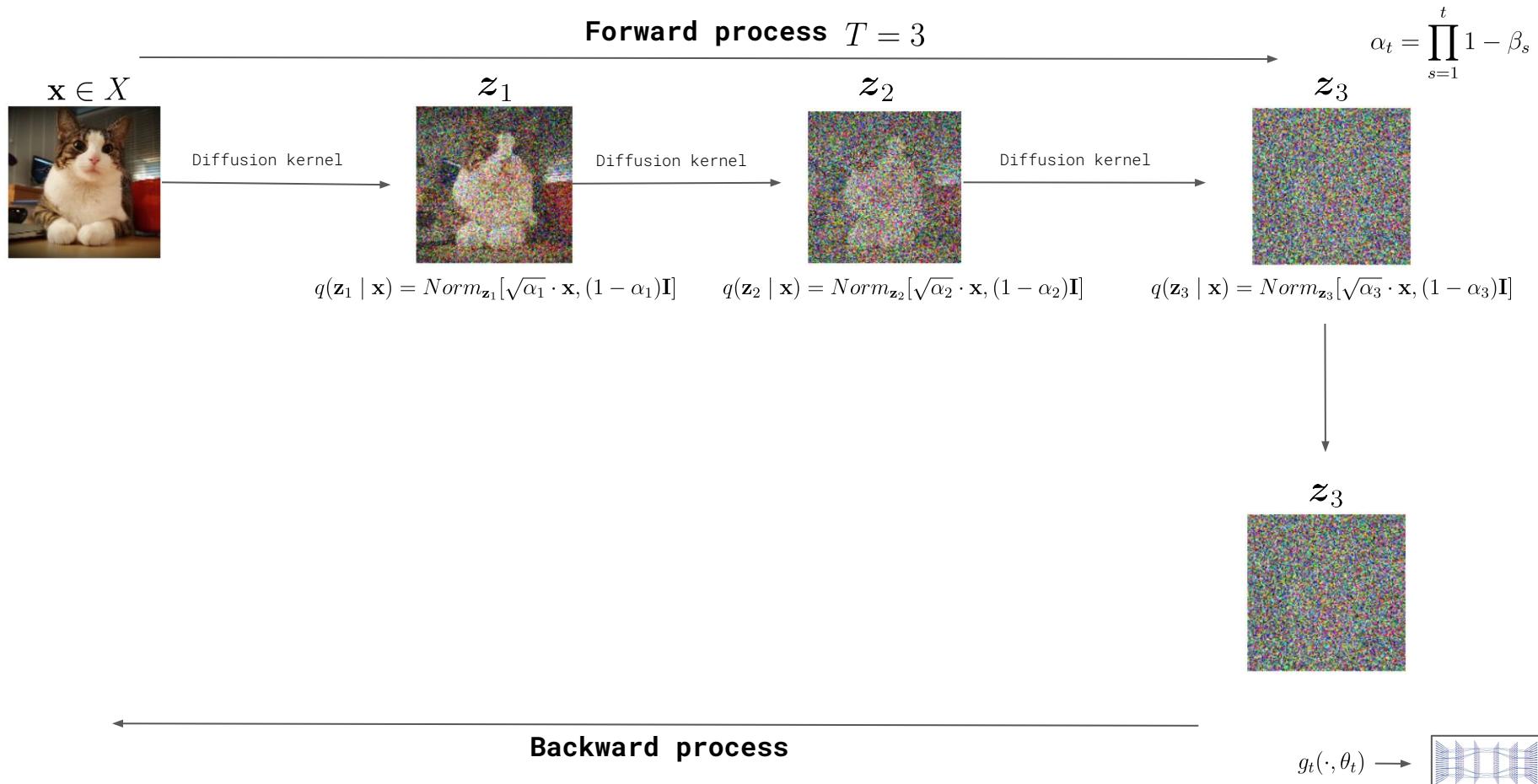
Entrenamiento



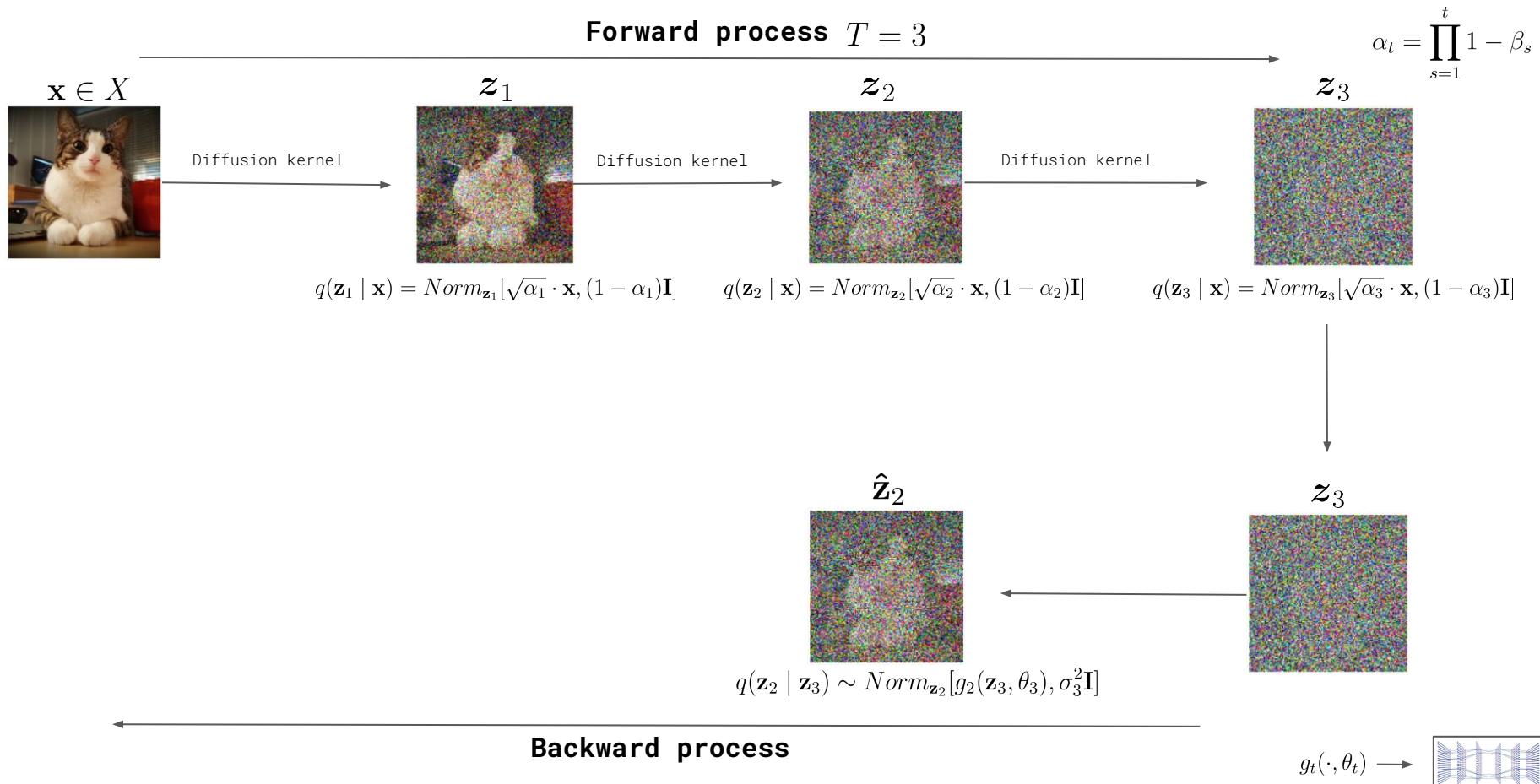
Entrenamiento



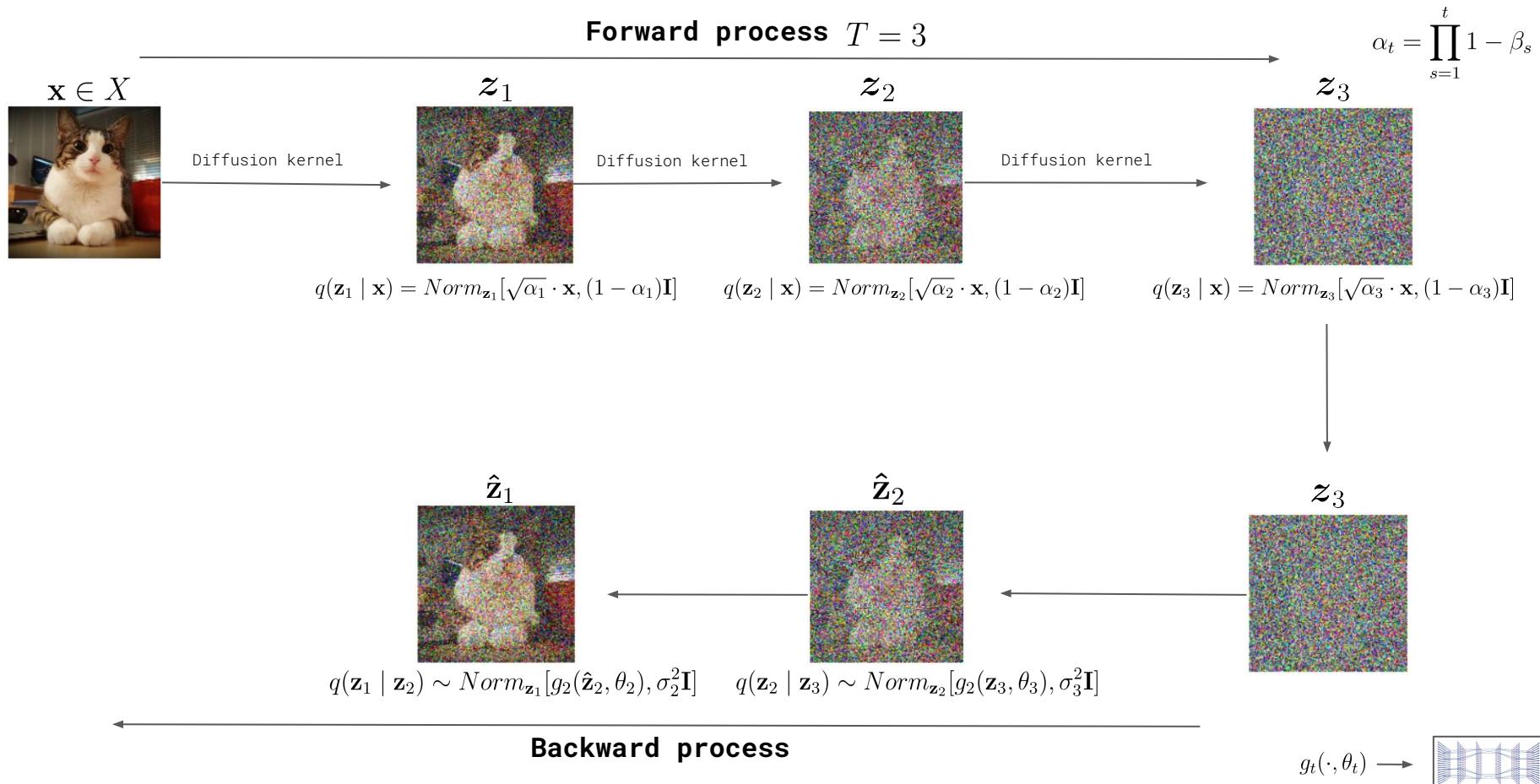
Entrenamiento



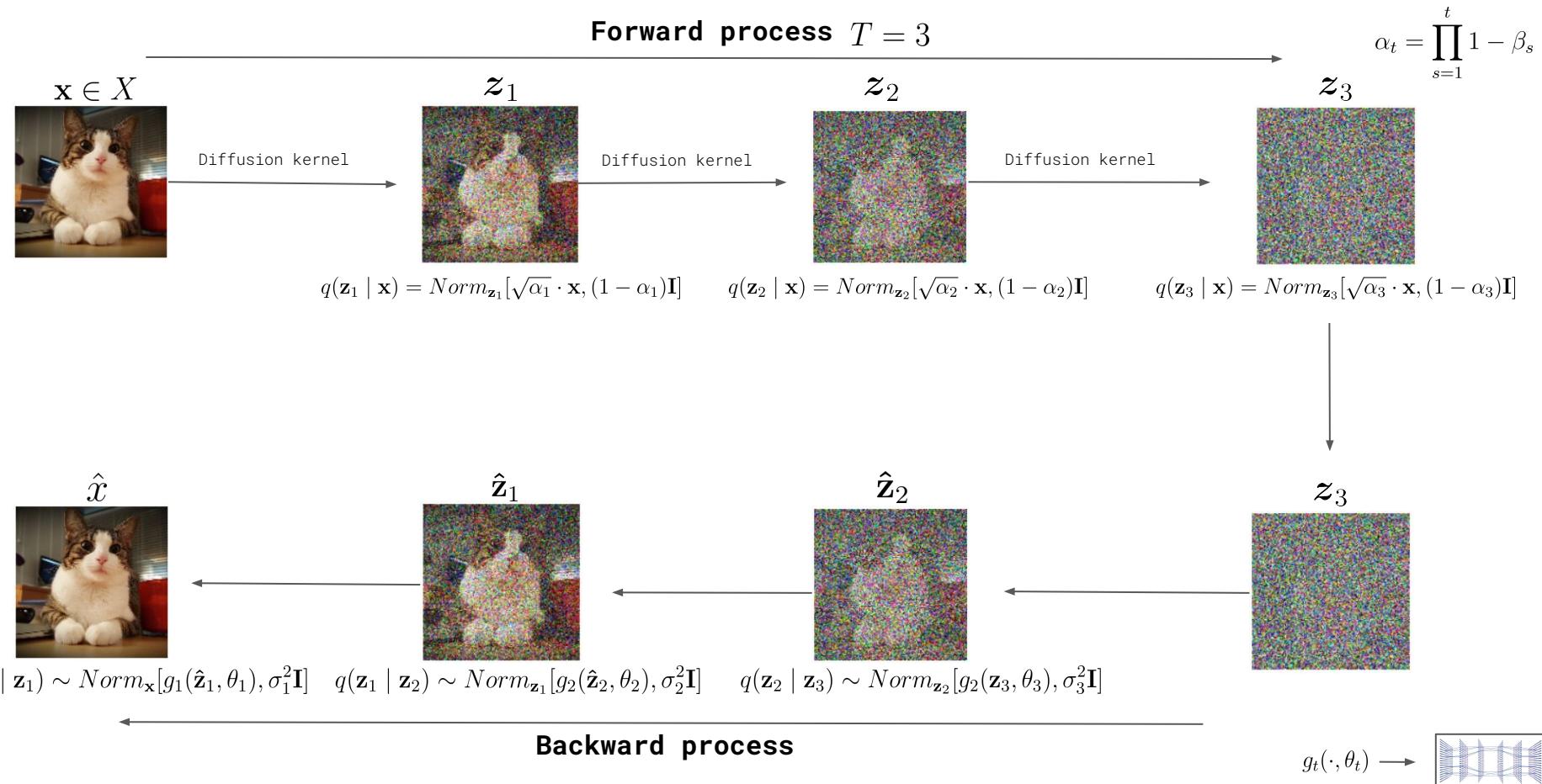
Entrenamiento



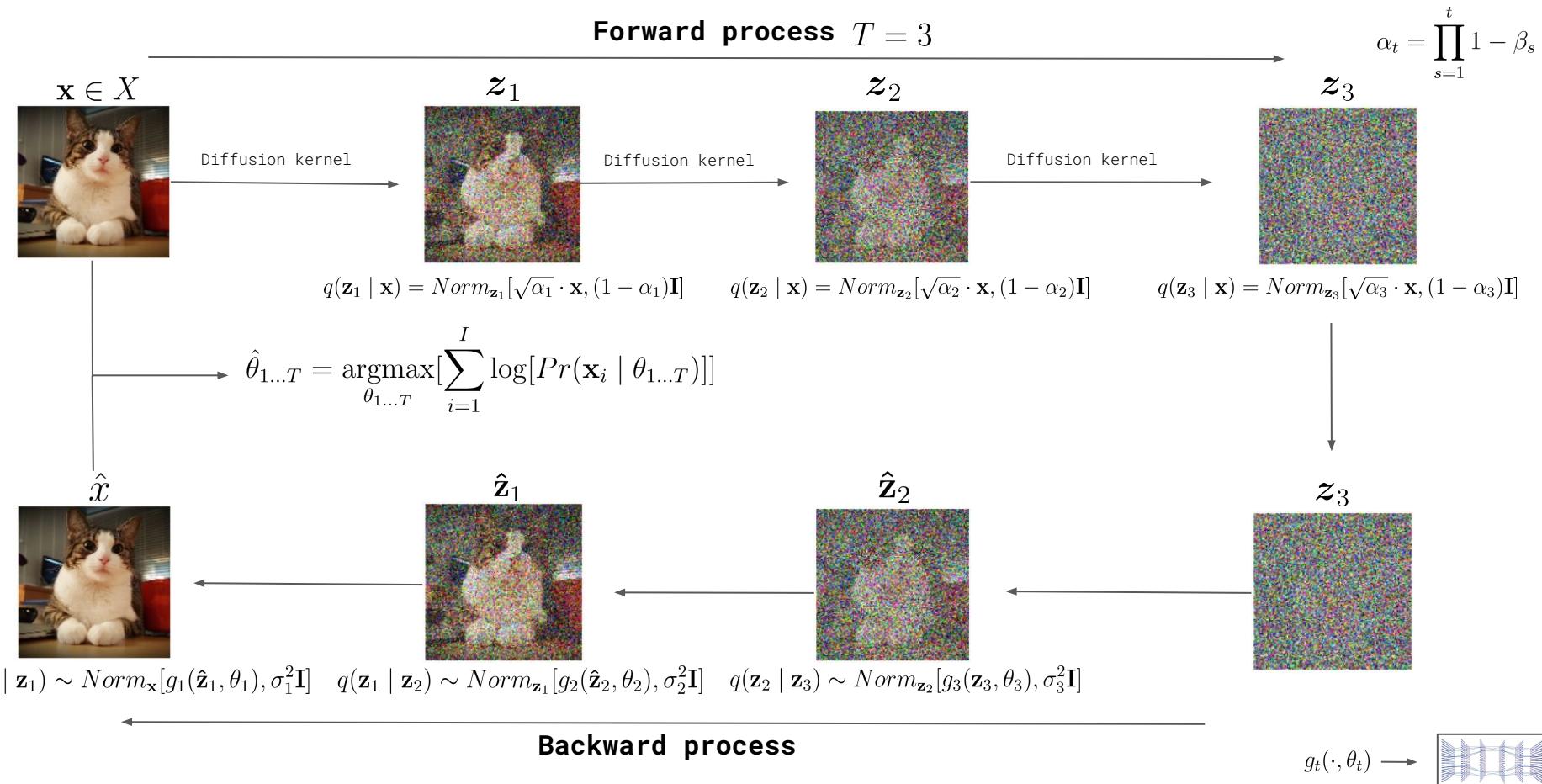
Entrenamiento



Entrenamiento



Entrenamiento

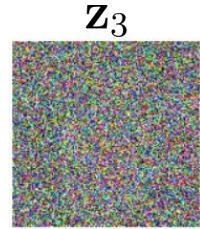


Inferencia

Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:

Inferencia

Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:



$$q(\mathbf{z}_3) \sim Norm[0, \mathbf{I}]$$

Inferencia

Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:

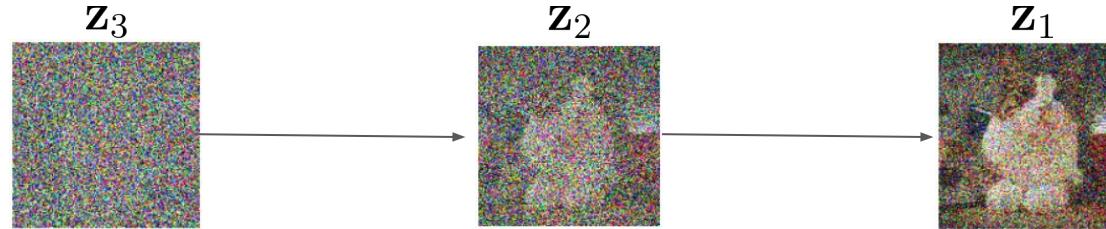


$$q(\mathbf{z}_3) \sim Norm[0, \mathbf{I}]$$

$$q(\mathbf{z}_2 \mid \mathbf{z}_3) \sim Norm_{\mathbf{z}_2}[g_3(\mathbf{z}_3, \theta_3), \sigma_3^2 \mathbf{I}]$$

Inferencia

Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:



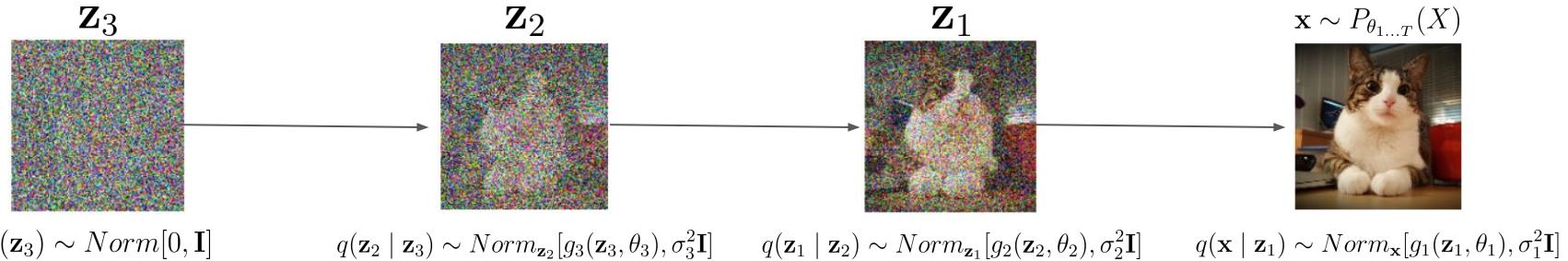
$$q(\mathbf{z}_3) \sim Norm[0, \mathbf{I}]$$

$$q(\mathbf{z}_2 \mid \mathbf{z}_3) \sim Norm_{\mathbf{z}_2}[g_3(\mathbf{z}_3, \theta_3), \sigma_3^2 \mathbf{I}]$$

$$q(\mathbf{z}_1 \mid \mathbf{z}_2) \sim Norm_{\mathbf{z}_1}[g_2(\mathbf{z}_2, \theta_2), \sigma_2^2 \mathbf{I}]$$

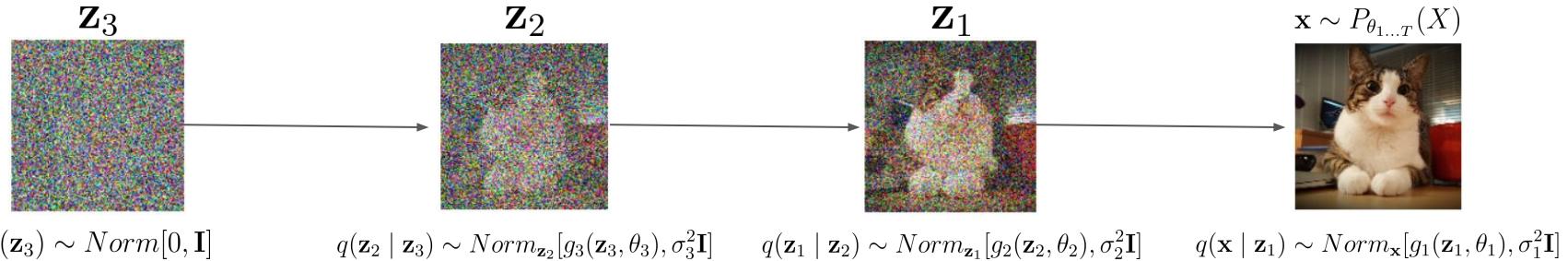
Inferencia

Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:



Inferencia

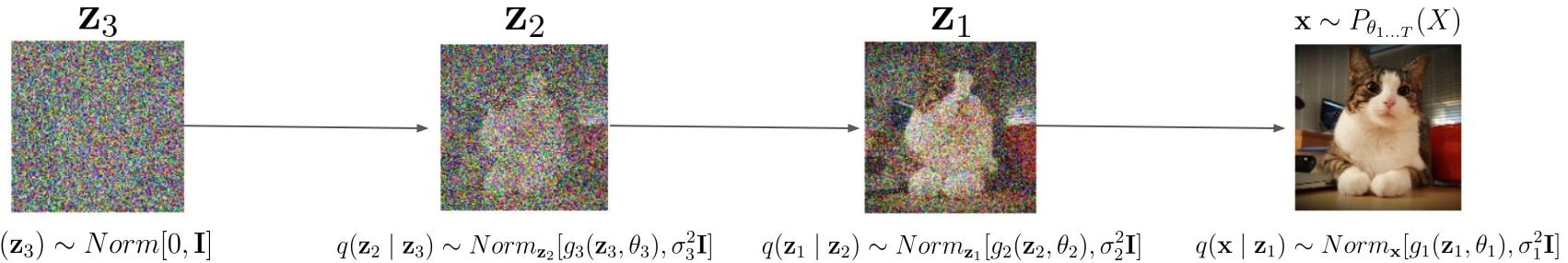
Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:



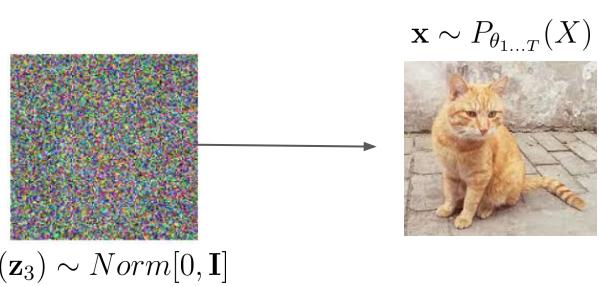
Si hago la reconstrucción con un \mathbf{z}_3 distinto obtengo distintas imágenes:

Inferencia

Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:

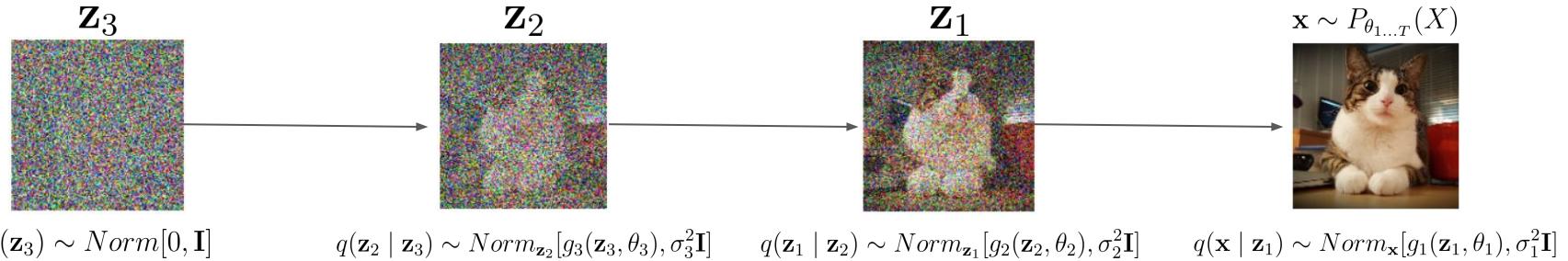


Si hago la reconstrucción con un \mathbf{z}_3 distinto obtengo distintas imágenes:



Inferencia

Con nuestros modelos $g_t(\cdot, \theta_t)$ ya entrenados:



Si hago la reconstrucción con un \mathbf{z}_3 distinto obtengo distintas imágenes:



Reparametrización

Con la formulación actual más algunas aproximaciones (ELBO) es posible entrenar un modelo de difusión, sin embargo, para hacer el proceso **más eficiente** se recurren a una **reparametrización** de la **función de pérdida** y, por lo tanto, del **modelo**

Reparametrización

Con la formulación actual más algunas aproximaciones (ELBO) es posible entrenar un modelo de difusión, sin embargo, para hacer el proceso **más eficiente** se recurren a una **reparametrización** de la **función de pérdida** y, por lo tanto, del **modelo**

El nuevo modelo $f_t(\cdot, \phi_t)$ predice el ruido ϵ_t que se suma a \mathbf{x} durante el forward process:

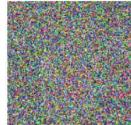
Reparametrización

Con la formulación actual más algunas aproximaciones (ELBO) es posible entrenar un modelo de difusión, sin embargo, para hacer el proceso **más eficiente** se recurren a una **reparametrización** de la **función de pérdida** y, por lo tanto, del **modelo**

El nuevo modelo $f_t(\cdot, \phi_t)$ predice el ruido ϵ_t que se suma a \mathbf{x} durante el forward process:



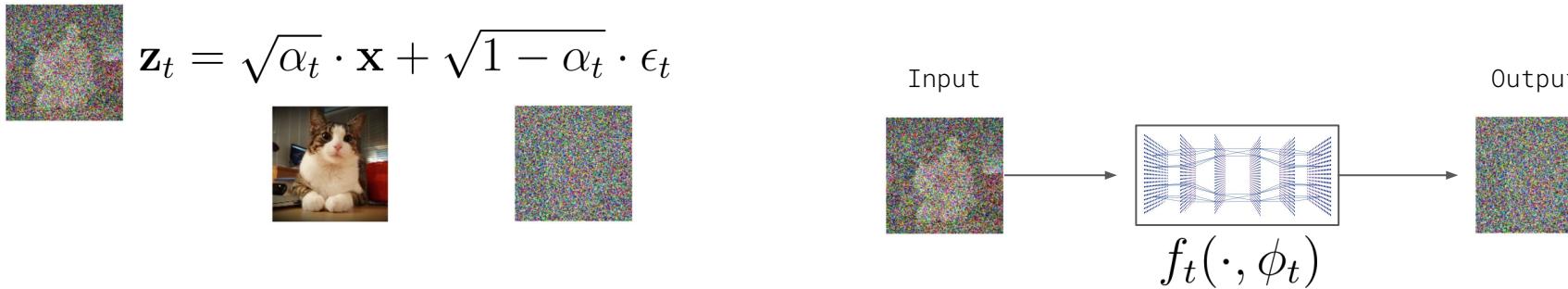
$$\mathbf{z}_t = \sqrt{\alpha_t} \cdot \mathbf{x} + \sqrt{1 - \alpha_t} \cdot \epsilon_t$$



Reparametrización

Con la formulación actual más algunas aproximaciones (ELBO) es posible entrenar un modelo de difusión, sin embargo, para hacer el proceso **más eficiente** se recurren a una **reparametrización** de la **función de pérdida** y, por lo tanto, del **modelo**

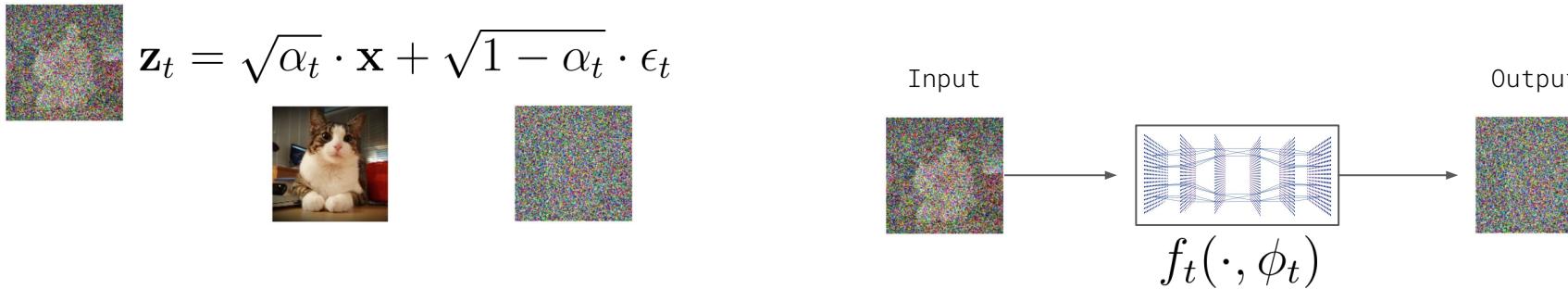
El nuevo modelo $f_t(\cdot, \phi_t)$ predice el ruido ϵ_t que se suma a \mathbf{x} durante el forward process:



Reparametrización

Con la formulación actual más algunas aproximaciones (ELBO) es posible entrenar un modelo de difusión, sin embargo, para hacer el proceso **más eficiente** se recurren a una **reparametrización** de la **función de pérdida** y, por lo tanto, del **modelo**

El nuevo modelo $f_t(\cdot, \phi_t)$ predice el ruido ϵ_t que se suma a \mathbf{x} durante el forward process:

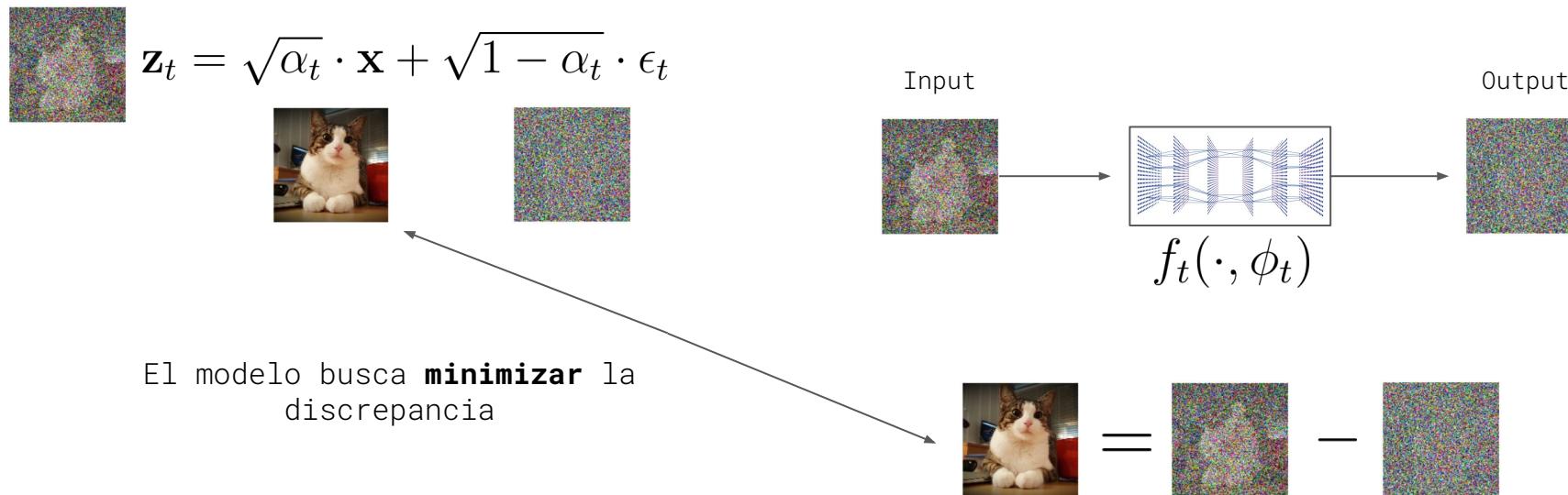


$$\text{Original Image} = \text{Noisy Image} - \text{Predicted Noise}$$

Reparametrización

Con la formulación actual más algunas aproximaciones (ELBO) es posible entrenar un modelo de difusión, sin embargo, para hacer el proceso **más eficiente** se recurren a una **reparametrización** de la **función de pérdida** y, por lo tanto, del **modelo**

El nuevo modelo $f_t(\cdot, \phi_t)$ predice el ruido ϵ_t que se suma a \mathbf{x} durante el forward process:



Reparametrización

Esta reparametrización nos lleva a un **entrenamiento del modelo de difusión simple y directo**

Algorithm 18.1: Diffusion model training

Input: Training data \mathbf{x}

Output: Model parameters ϕ_t

repeat

for $i \in \mathcal{B}$ **do**

 // For every training example index in batch

$t \sim \text{Uniform}[1, \dots T]$

 // Sample random timestep

$\epsilon \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$

 // Sample noise

$\ell_i = \left\| \mathbf{g}_t \left[\sqrt{\alpha_t} \mathbf{x}_i + \sqrt{1 - \alpha_t} \epsilon, \phi_t \right] - \epsilon \right\|^2$

 // Compute individual loss

 Accumulate losses for batch and take gradient step

until converged

Reparametrización

Y de la correspondiente **inferencia**

Algorithm 18.2: Sampling

Input: Model, $\mathbf{g}_t[\bullet, \phi_t]$

Output: Sample, \mathbf{x}

```

 $\mathbf{z}_T \sim \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$                                 // Sample last latent variable
for  $t = T \dots 2$  do
     $\hat{\mathbf{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t} \sqrt{1-\beta_t}} \mathbf{g}_t[\mathbf{z}_t, \phi_t]$     // Predict previous latent variable
     $\epsilon \sim \text{Norm}_{\epsilon}[\mathbf{0}, \mathbf{I}]$                                          // Draw new noise vector
     $\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t \epsilon$                                      // Add noise to previous latent variable
     $\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}} \mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1} \sqrt{1-\beta_1}} \mathbf{g}_1[\mathbf{z}_1, \phi_1]$  // Generate sample from  $\mathbf{z}_1$  without noise
  
```

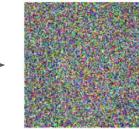
Modelo de deep learning

Para reducir la carga computacional, en vez de entrenar un modelo distinto por cada $t \in T$, **entrenamos un único modelo**

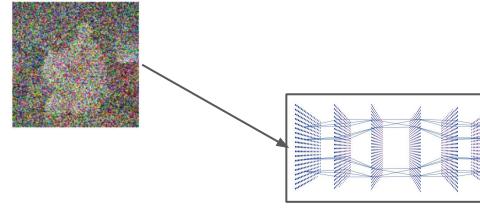
Input



Output



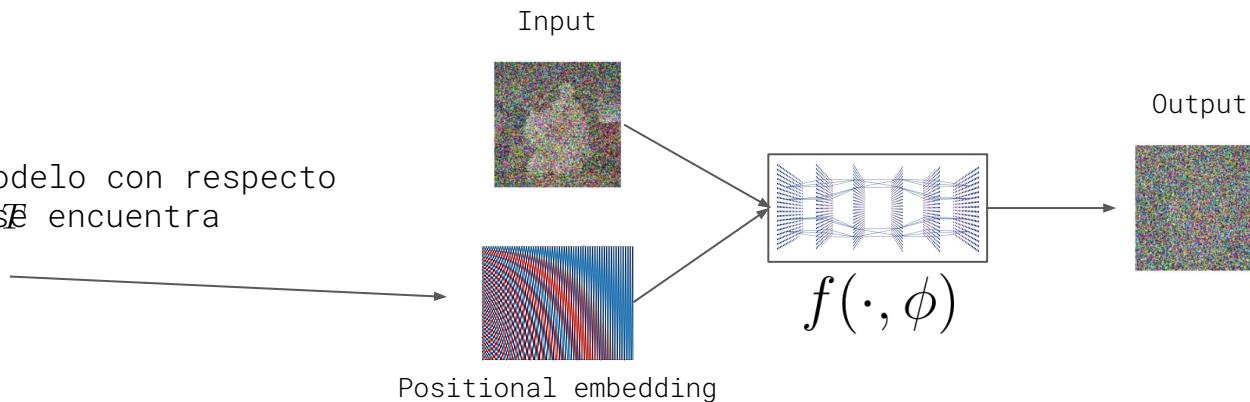
$$f(\cdot, \phi)$$



Modelo de deep learning

Para reducir la carga computacional, en vez de entrenar un modelo distinto por cada $t \in T$, **entrenamos un único modelo**

Da **contexto** al modelo con respecto
al t en el que se encuentra



De esta forma el modelo sabe el ruido que debe predecir en base al paso dentro del backward process en el que se encuentra

Modelo de deep learning

Como modelo $f(\cdot, \phi)$ se suele recurrir a las **U-Nets**

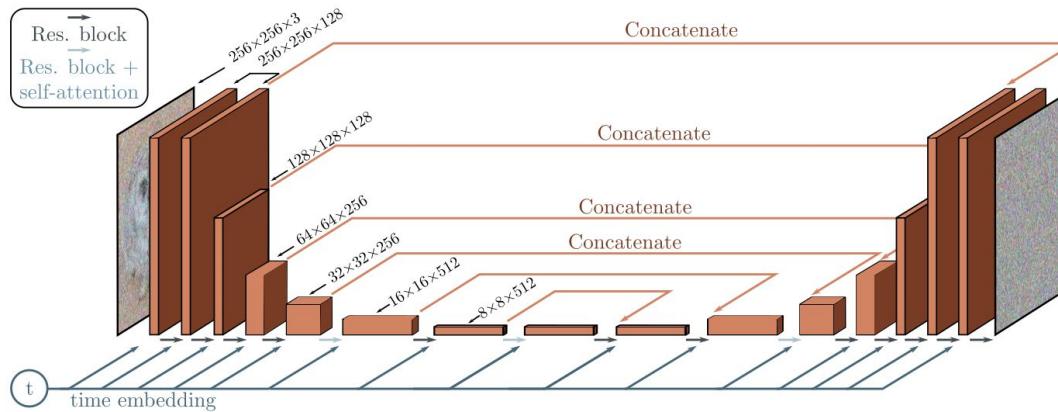


Figure 18.9 U-Net as used in diffusion models for images. The network aims to predict the noise that was added to the image. It consists of an encoder which reduces the scale and increases the number of channels and a decoder which increases the scale and reduces the number of channels. The encoder representations are concatenated to their partner in the decoder. Connections between adjacent representations consist of residual blocks, and periodic global self-attention in which every spatial position interacts with every other spatial position. A single network is used for all time steps, by passing a sinusoidal time embedding (figure 12.5) through a shallow neural network and adding the result to the channels at every spatial position of the U-Net.

Conclusiones

- Los modelos de difusión nos permiten aproximar $P(\mathbf{X})$ **reconstruyendo la destrucción de información** (backward process) hecha sobre $\mathbf{x} \in P(\mathbf{X})$ (forward process)
- Si el **número de pasos** en esta destrucción es lo **suficientemente grande**, las reconstrucciones se aproximan a una **distribución normal**
- Tras aplicar varias **reparametrizaciones**, **el proceso de entrenamiento se simplifica**, con una función de coste equivalente a un Mean Squared Error (MSE)
- Los modelos de difusión permiten generar datos con gran **realismo** y **variedad**
- El único inconveniente es que es necesario aplicar el modelo de deep learning (U-Net) T veces para generar una única muestra



Existen soluciones a este problema, por ejemplo, **Stable Diffusion**

Preguntas?

Jose González-Abad
gonzabad@ifca.unican.es

¿Cómo condicionar?

El modelo de difusión presentado hoy nos permite aproximar $P(\mathbf{X})$, es decir generar muestras de esa distribución



Figure 18.12 Conditional generation using classifier guidance. Image samples conditioned on different ImageNet classes. The same model produces high quality samples of highly varied image classes. Adapted from Dhariwal & Nichol (2021).

Pero... ¿y si quiero controlar que imágenes genero?, es decir, ¿si quiero hacer una **generación condicionada**?

¿Cómo condicionar?

Pero... ¿y si quiero controlar que imágenes genero?, es decir, ¿si quiero hacer una **generación condicionada**?

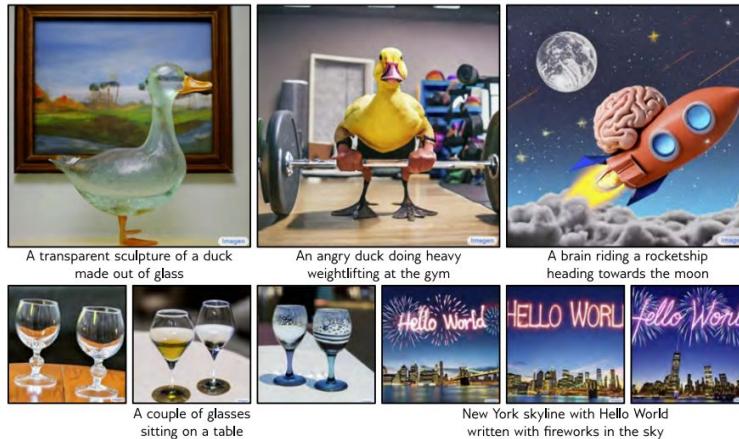


Figure 18.13 Conditional generation using text prompts. Synthesized images from a cascaded generation framework, conditioned on a text prompt encoded by a large language model. The stochastic model can produce many different images compatible with the prompt. The model can count objects and incorporate text into images. Adapted from Saharia et al. (2022b).

¿Cómo lo puedo hacer?

¿Cómo condicionar?

Supongamos que:

x →

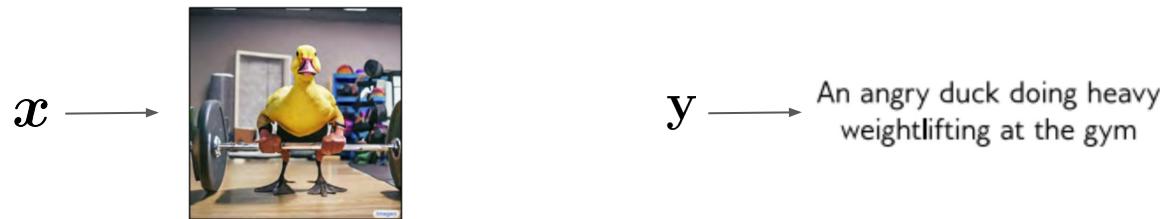


y →

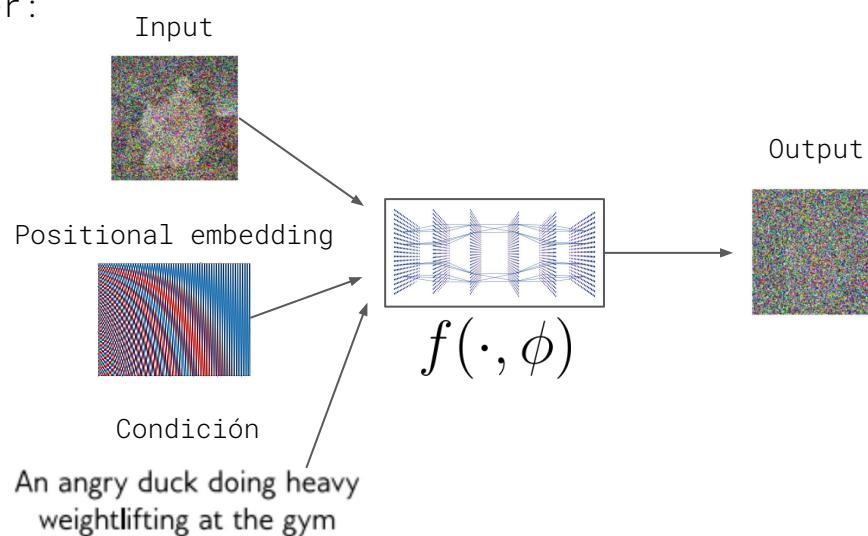
An angry duck doing heavy weightlifting at the gym

¿Cómo condicionar?

Supongamos que:



Entonces podemos hacer:



Referencias

- Prince, S. J. (2023). Understanding deep learning. MIT press.
- <https://openai.com/sora/>
- Alakhdar, A., Poczos, B., & Washburn, N. (2024). Diffusion models in de novo drug design. *Journal of Chemical Information and Modeling*, 64(19), 7238-7256.
- Price, I., Sanchez-Gonzalez, A., Alet, F., Andersson, T. R., El-Kadi, A., Masters, D., ... & Willson, M. (2025). Probabilistic weather forecasting with machine learning. *Nature*, 637(8044), 84-90.
- <https://www.geeksforgeeks.org/diffusion-biology/>
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10684-10695).