Diffusion Models and Flow Matching

Alasdair Newson
anewson@isir.upmc.fr

Introduction

- The past years have seen incredible progress in generative modelling;
- Results have gone from this (2014*):

* ***Generative Adversarial Nets***, *Goodfellow et al, NIPS 2014*

Introduction

- To this (Stable Diffusion, 2022*) !

* ***High-resolution image synthesis with latent diffusion models***, *Rombach et al, CVPR 2022*

Introduction

- Currently, the state-of-the-art is represented by the following methods;
  1. **Diffusion models**/score-based models;
  2. **Flow matching**;
- These are in fact very closely linked;
- This lesson will explore Diffusion Models and Flow Matching;

Introduction

- First, an example of text conditioned generation with Chatgpt
- Prompt: “RDFIA class at sorbonne university, a cool logo“

Introduction

- Video Generation Models (VGMs - Sora, Veo 3 ...) now produce incredible results;

- However, requires **very large models** (billions of parameters)

Introduction

- Goal of all generative models: produce data which "looks like" data in a database;
  - Not sufficient to simply draw a datum from the database;
- Often formulated by modelling the database as samples from an **unknown probability distribution** μ_1;
- To sample from μ_1, we first sample from a simpler distribution μ_0, and apply a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ to the sample

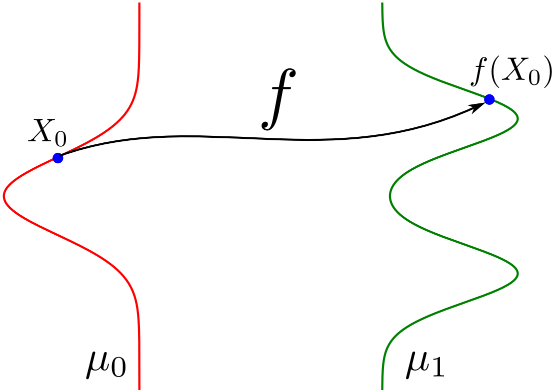
Introduction

- Goal of all generative models: produce data which "looks like" data in a database;
  - Not sufficient to simply draw a datum from the database;
- Often formulated by modelling the database as samples from an **unknown probability distribution** μ_1;
- To sample from μ_1, we first sample from a simpler distribution μ_0, and apply a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ to the sample

Generative Model sampling algorithm

1. Sample $X_0 \sim \mu_0$;
2. $X_1 = f(X_0)$;
3. Return X_1;

Introduction



- Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Diffusion Models and Flow Matching all use this approach;
- The main question is, how to **design f such that $f(X_0) \sim \mu_1$** ?

Introduction

- Diffusion Models and Flow Matching design f in a very different manner to VAEs and GANs;
- f is in fact an **iteration of a neural network** several times;
- The network is viewed as a **denoising** process;

Diffusion Models

Diffusion Models

- Diffusion models[*,†] become image synthesis state-of-the-art;
- Produce incredible results[‡] :

[*] ***Deep Unsupervised Learning using Nonequilibrium Thermodynamics***, *J. Sohl-Dickstein, ICML 2015*
[†] ***Denoising Diffusion Probabilistic Models***, *Ho et al., NIPS 2020*
[‡] ***High-Resolution Image Synthesis with Latent Diffusion Models***, *R. Rombach et al, CVPR 2022*

Diffusion Models

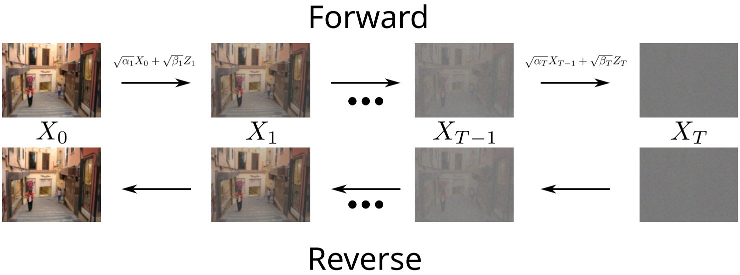
- There are several formulations of diffusion models, with different technical tools;
  - All lead to similar algorithms;
- In this lesson, we present the **Denoising Diffusion Probabilistic Models (Ho et al) version**;
  - Less technical tools, but slightly more complicated;
  - As far as possible, we maintain same notation;
- Other versions : Score-based, Denoising Diffusion Implicit Models;

Diffusion Models

- Main difference between diffusion models and previous generative models (VAEs, GANs):
  - Neural network **applied iteratively**;

Diffusion Models

- Main difference between diffusion models and previous generative models (VAEs, GANs):
  - Neural network **applied iteratively**;
- Core idea: we know how to add noise to an image; if we know how to remove it, we can **synthesise images from noise**
  - "Forward/reverse*" random processes;



We use "reverse" to not confuse with "backward" in backpropagation

Diffusion Models

- Diffusion model algorithm :
  1. Train a neural network to denoise images;
  2. Sample a random Gaussian noise;
  3. Iteratively denoise to produce random synthesised image;
- Sounds simple !
- Well, let us look at the mathematical formulation;

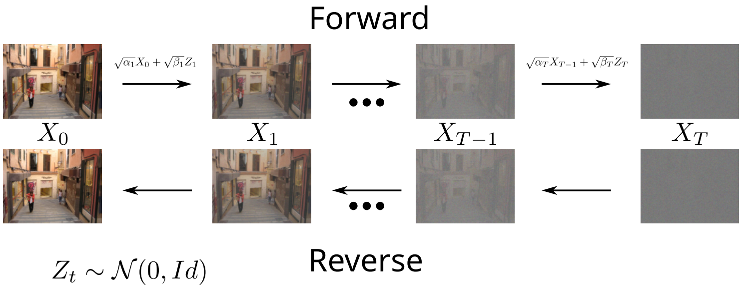
Thanks to Arthur Leclaire and Bruno Galerne for their exceedingly enlightening explanations on this subject !

Forward process

- Diffusion models first set up a **forward** process: $(X_0, X_1, X_2, \ldots, X_T)$
  - X_t's are images with an increasing amount of noise;

Forward process

- Diffusion models first set up a **forward** process: $(X_0, X_1, X_2, \ldots, X_T)$
  - X_t's are images with an increasing amount of noise;
- We define $X_t = \sqrt{\alpha_t} X_{t-1} + \sqrt{\beta_t} Z_t$
  - Z_t's are independent Gaussian noises: $z_t \sim \mathcal{N}(0, \beta_t)$;
  - (α_t, β_t) are scalars;



Diffusion Models

- Let q_0 be the probability distribution of X_0 (noiseless image);
  - Note that this corresponds to μ_1 in our original formulation;
- Formally, we wish to draw a sample from q_0 (same goal as any generative model);

Diffusion Models

- Let q_0 be the probability distribution of X_0 (noiseless image);
  - Note that this corresponds to μ_1 in our original formulation;
- Formally, we wish to draw a sample from q_0 (same goal as any generative model);
- First, note that the forward process $(X_0, X_1, X_2, \dots, X_T)$ forms a **Markov Chain** (X_t only depends on X_{t-1}):

$$q(X_1, \dots, X_T | X_0) = q(X_T | X_{T-1}) \; q(X_{T-1} | X_{T-2}) \; \dots \; q(X_1 | X_0) \quad (1)$$

Diffusion Models

- Let q_0 be the probability distribution of X_0 (noiseless image);
  - Note that this corresponds to μ_1 in our original formulation;
- Formally, we wish to draw a sample from q_0 (same goal as any generative model);
- First, note that the forward process $(X_0, X_1, X_2, \dots, X_T)$ forms a **Markov Chain** (X_t only depends on X_{t-1}):

$$q(X_1, \dots, X_T | X_0) = q(X_T | X_{T-1})\ q(X_{T-1} | X_{T-2})\ \dots\ q(X_1 | X_0) \quad (1)$$

- Also, by the definition of $q(X_t | X_{t-1})$:

$$q(X_t | X_{t-1}) = \mathcal{N}(X_t; \sqrt{\alpha} X_{t-1}, \sqrt{\beta_t} Id) \quad (2)$$

- Finally, Ho et al set $\alpha_t = \sqrt{1 - \beta_t}$;

Diffusion Models

- We have, recursively:

$$
\begin{aligned}
X_t &= \sqrt{1-\beta_t}X_{t-1} + \sqrt{\beta_t}Z_t \\
&= \sqrt{1-\beta_t}\left(\underbrace{\sqrt{1-\beta_{t-1}}X_{t-2} + \sqrt{\beta_{t-1}}Z_{t-1}}_{X_{t-1}}\right) + \sqrt{\beta_t}Z_t \\
&= \sqrt{\alpha_t\alpha_{t-1}}X_{t-2} + \sqrt{(1-\beta_t)\beta_{t-1}}Z_{t-1} + \sqrt{\beta_t}Z_t \\
&= \sqrt{\alpha_t\alpha_{t-1}}X_{t-2} + \sqrt{(\alpha_t)(1-\alpha_{t-1})}{\color{red}Z_{t-1}} + \sqrt{1-\alpha_t}{\color{red}Z_t}
\end{aligned}
\tag{3}
$$

- Reminder: ${\color{red}Z_{t-1}}$ and ${\color{red}Z_t}$ are i.i.d normal variables;
- NB: we have converted β_t to α_t for conveniance of formulae;

Diffusion Models

- Recall: sum of two independent Gaussian r.v. is also a Gaussian r.v.
  - Mean and **variance** are both summed;
- Thus, we have $\left(\sqrt{(\alpha_t)(1-\alpha_{t-1})}Z_{t-1} + \sqrt{1-\alpha_t}Z_t\right) \sim \mathcal{N}(0, 1-\alpha_t\alpha_{t-1})$, and we can write:

$$X_t = \sqrt{\alpha_t\alpha_{t-1}}X_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_t, \tag{4}$$

- where $\epsilon_t \sim \mathcal{N}(0, Id)$

Diffusion Models

- Recall: sum of two independent Gaussian r.v. is also a Gaussian r.v.
  - Mean and **variance** are both summed;
- Thus, we have $\left(\sqrt{(\alpha_t)(1-\alpha_{t-1})}Z_{t-1} + \sqrt{1-\alpha_t}Z_t\right) \sim \mathcal{N}(0, 1-\alpha_t\alpha_{t-1})$, and we can write:

$$X_t = \sqrt{\alpha_t\alpha_{t-1}}X_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_t, \tag{4}$$

- where $\epsilon_t \sim \mathcal{N}(0, Id)$
- More generally, wrt X_0, we can write:

$$X_t = \sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, \tag{5}$$

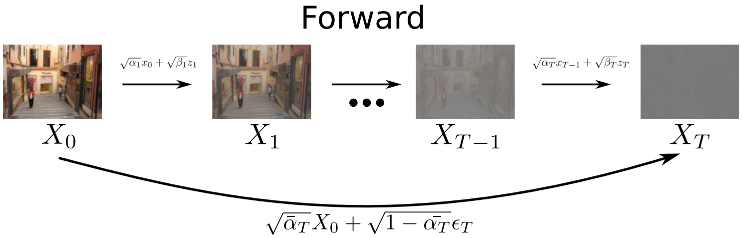
- where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

Diffusion Models

- Intuitively, ϵ_t is the noise which produces X_t from the initial image X_0

$$X_t = \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t, \tag{6}$$

- Thus, we can skip to the end of the Markov chain !



Reverse process

- Now, **how do we go in the reverse direction ?**
  - Recall: we want to do this to randomly synthesise images;
- This is not so easy: we only want to remove noise from X_t to X_{t-1}
  - Most denoisers estimate X_0 directly;
  - Furthermore, they are not trained on very high noise levels (meaningless above a certain noise level);
- To solve this, let us look at the mathematical formulation more carefully;

Diffusion Models

- Formally, reverse process is again a Markov chain $(X_T, X_{T-1}, \ldots, X_0)$;
  - This is also chosen to be a series of Gaussian r.v.'s;
  - We note p_θ the prob. distribution of the backward process;
  - This is the process to be learned;

Diffusion Models

- Formally, reverse process is again a Markov chain $(X_T, X_{T-1}, \dots, X_0)$;
  - This is also chosen to be a series of Gaussian r.v.'s;
  - We note p_θ the prob. distribution of the backward process;
  - This is the process to be learned;
- We have $p_\theta(X_T, \dots, X_0) = p(X_T) \prod_{t=T}^{1} p_\theta(X_{t-1}|X_t)$;

$$p_\theta(X_{t-1}|X_t) = \mathcal{N}(X_{t-1}; \mu_\theta(X_t, t), \Sigma_\theta(X_t, t)) \tag{7}$$

- **How to calculate the mean μ_θ and covariance Σ_θ of this reverse process, for each t ?**
- DDPM does this by maximising $\mathbb{E}\left[X_0 \sim q_0\right] \log p_\theta(X_0)$;

Diffusion Models

- We first note a special property of the Markov chain when the X_t's are Gaussian r.v.'s:
  - If (X_0, X_t) are known, X_{t-1} is also Gaussian !
- Why is this true ?

Diffusion Models

- We first note a special property of the Markov chain when the X_t's are Gaussian r.v.'s:
  - If (X_0, X_t) are known, X_{t-1} is also Gaussian !
- Why is this true ?
  - The joint distribution $\mathbb{P}(X_T, \ldots, X_1, X_0)$ is Gaussian;
  - For any random variables X, Y, with $\mathbb{P}(X, Y)$ Gaussian, then $\mathbb{P}(X|Y)$ also Gaussian;
  - More generally, for any collection of jointly Gaussian r.v's, then conditioning on one or more of them also gives a Gaussian;

- We have $q(X_{t-1}|X_0, X_t) = \mathcal{N}(X_{t-1}; \tilde{\mu}_t(X_t, X_0), \tilde{\beta}_t Id)$, with

$$\tilde{\mu}(X_t, X_0) = \frac{\sqrt{\bar{\alpha}_{t-1}\beta_t}}{1-\bar{\alpha}_t} X_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} X_t \tag{8}$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \tag{9}$$

- **Closed form** !

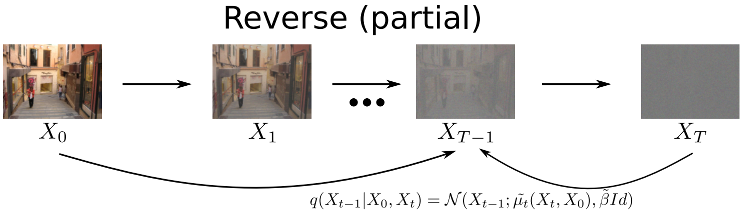
Diffusion Models

- We have $q(X_{t-1}|X_0, X_t) = \mathcal{N}(X_{t-1}; \tilde{\mu}_t(X_t, X_0), \tilde{\beta}_t Id)$, with

$$\tilde{\mu}(X_t, X_0) = \frac{\sqrt{\bar{\alpha}_{t-1}\beta_t}}{1-\bar{\alpha}_t} X_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} X_t \quad (8)$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \quad (9)$$

- **Closed form** !



Diffusion Models

- Why is this useful ? We don't know X_0 in practice: indeed, X_0 is what we are trying to produce!

Diffusion Models

- Why is this useful ? We don't know X_0 in practice: indeed, X_0 is what we are trying to produce!

- In fact, we can **train a NN to estimate X_0**, and then use this to sample $q(X_{t-1}|X_0, X_t)$;
  - Indeed, estimating X_0 is the common goal of denoisers;
  - Easier than estimating X_{t-1} directly from x_t;

- Thus, knowing X_t and an estimation of X_0, we can sample X_{t-1};

Diffusion Models

- Why is this useful ? We don't know X_0 in practice: indeed, X_0 is what we are trying to produce!

- In fact, we can **train a NN to estimate X_0**, and then use this to sample $q(X_{t-1}|X_0, X_t)$;
  - Indeed, estimating X_0 is the common goal of denoisers;
  - Easier than estimating X_{t-1} directly from x_t;

- Thus, knowing X_t and an estimation of X_0, we can sample X_{t-1};

- So, our algorithm is now :
  1. Train denoiser;
  2. Sample Gaussian noise;
  3. Iterate:
     - Estimate X_0 using denoiser(X_t);
     - Sample X_{t-1}, using X_T and estimate of X_0

- So, how is the denoiser trained ?

Diffusion Models

- We train a network to maximise $p_\theta(X_0)$, using the ELBO, as in the VAE;
- Slightly more complicated form due to the Markov chain setting:

$$\log(p_\theta(X_0)) \geq \mathbb{E}\left[\log p_\theta(X_0|X_1)\right] - KL\left(q(X_T|X_0)||p_\theta(X_T)\right) \tag{10}$$

$$- \sum_{t>1} KL\left(q(X_{t-1}|X_0, X_t)||p_\theta(X_{t-1}|X_t)\right) \tag{11}$$

- The terms in blue are known, and do not intervene in the optimisation;
- So, what is $KL\left(q(X_{t-1}|X_0, X_t)||p_\theta(X_{t-1}|X_t)\right)$?

Diffusion Models

- Recall that $p_\theta(X_{t-1}|X_t)$ was chosen to be Gaussian
  - We just don't know the mean and variance yet;
- Thus, since $p_\theta(X_{t-1}|X_t)$ and $q(X_{t-1}|X_0, X_t)$ are both Gaussian, we have a closed form solution for $KL\left(q(X_{t-1}|X_0, X_t)||p_\theta(X_{t-1}|X_t)\right)$

$$KL\left(q(X_{t-1}|X_0, X_t)||p_\theta(X_{t-1}|X_t)\right) = \mathbb{E}_q\left[\frac{1}{\beta_t}\|\underbrace{\tilde{\mu}(X_t, X_0)}_{\substack{\text{Known if}\\ X_t, X_0 \text{ known}}} - \overbrace{\mu_\theta(X_t, t)}^{\text{Neural net}}\|_2^2\right] \tag{12}$$

Diffusion Models

- Recall that $p_\theta(X_{t-1}|X_t)$ was chosen to be Gaussian
  - We just don't know the mean and variance yet;
- Thus, since $p_\theta(X_{t-1}|X_t)$ and $q(X_{t-1}|X_0, X_t)$ are both Gaussian, we have a closed form solution for $KL\left(q(X_{t-1}|X_0, X_t)||p_\theta(X_{t-1}|X_t)\right)$

$$KL\left(q(X_{t-1}|X_0, X_t)||p_\theta(X_{t-1}|X_t)\right) = \mathbb{E}_q\left[\frac{1}{\beta_t}\|\underbrace{\tilde{\mu}(X_t, X_0)}_{\substack{\text{Known if}\\ X_t, X_0 \text{ known}}} - \overbrace{\mu_\theta(X_t, t)}^{\text{Neural net}}\|_2^2\right] \tag{12}$$

- During a training process, we select pairs (X_t, X_0) to train $\mu_\theta(X_t, t)$;
- In practice, role of the network: **estimate X_0 from X_t**;

Diffusion Models

- Final note: the authors of DDPM reformulate the loss such that the **network estimate the noise** ϵ_t, rather than X_0
  - Why do they do this ? Because they report that this gives better results (see Section 3.2) ...
- Since $X_t = \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t$ (Equation (6)), it is equivalent to estimate X_0 or ϵ_t from X_t;
  - Knowing X_0 gives us ϵ_t directly, and vice cersa

Diffusion Models

- Final note: the authors of DDPM reformulate the loss such that the **network estimate the noise ϵ_t**, rather than X_0
  - Why do they do this ? Because they report that this gives better results (see Section 3.2) ...
- Since $X_t = \sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t$ (Equation (6)), it is equivalent to estimate X_0 or ϵ_t from X_t;
  - Knowing X_0 gives us ϵ_t directly, and vice cersa
- Final training loss (after much simplification):

$$\mathcal{L} = \mathbb{E}_{t,X_0,\epsilon}\left[\left\|\epsilon - f_\theta\left(\underbrace{\sqrt{\bar{\alpha}}X_0 + \sqrt{1-\bar{\alpha}_t}\epsilon}_{X_t}, t\right)\right\|_2^2\right] \tag{13}$$

- ϵ : noise to estimate with NN denoiser f_θ;

Diffusion Models

- Once the training is carried out, we can sample $p_\theta(X_{t-1}|X_t)$ using the following formula:

$$X_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(X_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} f_\theta(X_t, t)\right) + \tilde{\beta} Z_t, \qquad (14)$$

- where $Z_t \sim \mathcal{N}(0, Id)$ is a Gaussian noise;

Diffusion model summary

Algorithm 1 Diffusion model training

- Repeat following until converged:
  1. $X_0 \sim q(X_0)$ (take example X_0 from database);
  2. $t \sim \text{Uniform}(1, \dots, T)$;
  3. $\epsilon \sim \mathcal{N}(0, Id)$;
  4. One optimiser step on $\nabla_\theta \left\| \epsilon - f_\theta \left(\sqrt{\bar{\alpha}} X_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \right) \right\|_2^2$

Algorithm 2 Diffusion model testing (synthesis)

- $X_T \sim \mathcal{N}(0, Id)$;
- For $t = T, \dots, 1$, do
  1. $Z \sim \mathcal{N}(0, Id)$, if $t > 1$, else $Z = 0$;
  2. $X_{t-1} = \frac{1}{\sqrt{\alpha}_t} \left(X_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} f_\theta(X_t, t) \right) + \tilde{\beta} Z_t$

Diffusion models summary

- Idea: if we can reverse a noise process, then we can synthesise random images;
- Forward process $X_{t-1} \to X_t$ is easy to sample: we just add Gaussian noise a certain number of timesteps;
- Reverse process $X_t \to X_{t-1}$ is more difficult to sample;

Diffusion models summary

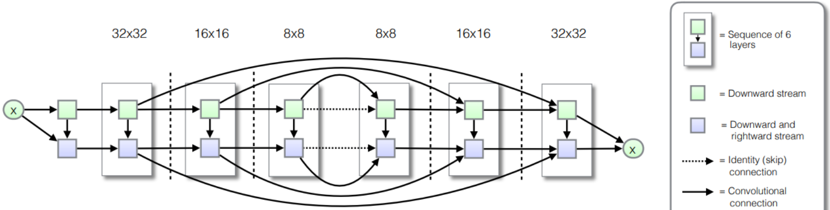
- Idea: if we can reverse a noise process, then we can synthesise random images;
- Forward process $X_{t-1} \to X_t$ is easy to sample: we just add Gaussian noise a certain number of timesteps;
- Reverse process $X_t \to X_{t-1}$ is more difficult to sample;
- We first note that, if we know (X_0, X_t), then $q_\theta(X_{t-1}|X_0, X_t)$ is Gaussian;
  - Thus, we need to estimate X_0 first: job of a denoiser;
- We formulate the ELBO of $\log p_\theta(X_0)$ such that $KL\left(q(X_{t-1}|X_0, X_t)||p_\theta(X_{t-1}|X_t)\right)$ appears.
  - Meaning : the Gaussian reverse process should be as close as possible to $q(X_{t-1}|X_0, X_t)$;

Diffusion Models

- A network f_θ is trained to predict X_0 (or, equivalently, ϵ_t) from any X_t;
- The synthesis starts with a noise image, and these two steps are iterated:
  1. Use f_θ to estimate X_0 (or ϵ_t);
  2. Use X_t and the estimation of X_0 to sample X_{t-1}

Diffusion Models

- Architecture of f_θ (in DDPM) is often a U-Net, more precisely "PixelCNN++"*
  - This is a common type of architecture for denoising;



* ***Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications***, *T. Salimans et al, arXiv:1701.05517, 2017*

Original DDPM results

DDPM results

Stable diffusion

- Diffusion models have many different variants;
- One of the most famous is "Stable Diffusion"*
- Carries out diffusion in a pretrained latent space;
- Conditional on a textual input (we do not explain this here)

* ***High-Resolution Image Synthesis with Latent Diffusion Models***, *R. Rombach et al, CVPR 2022*

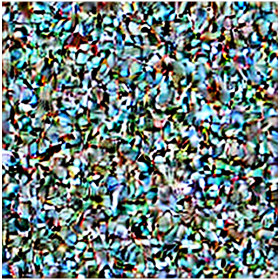
Diffusion Models

Stable diffusion

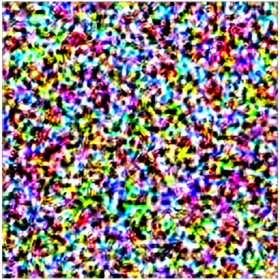
- Produces incredible results:

- Example of iterations of stable diffusion;
- Text input: "Link fighting with Ganon"
- Recall: "$X_t = aX_0 + b\varepsilon_t$"

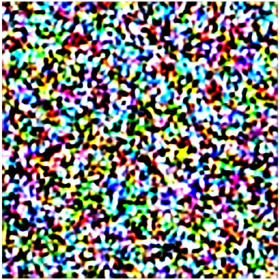
Noise ε_t



Estimation* of X_0



X_t



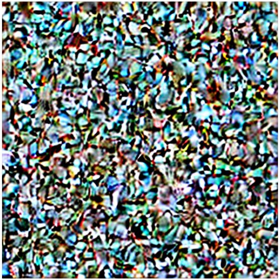
$t = 50$

Stable Diffusion is in a latent space, so this is not exactly correct here

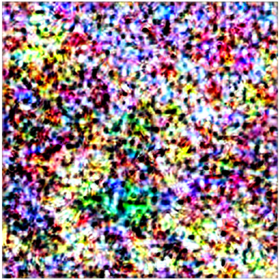
Diffusion Models

- Example of iterations of stable diffusion;
- Text input: "Link fighting with Ganon"
- Recall: "$X_t = aX_0 + b\varepsilon_t$"

Noise ε_t



Estimation* of X_0



X_t

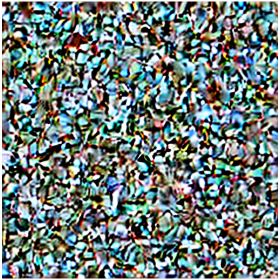
$t = 40$

** Stable Diffusion is in a latent space, so this is not exactly correct here*

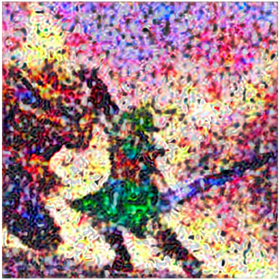
Diffusion Models

- Example of iterations of stable diffusion;
- Text input: "Link fighting with Ganon"
- Recall: "$X_t = aX_0 + b\varepsilon_t$"

Noise ε_t



Estimation* of X_0



X_t

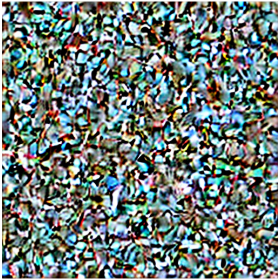
$t = 30$

**Stable Diffusion is in a latent space, so this is not exactly correct here*

Diffusion Models

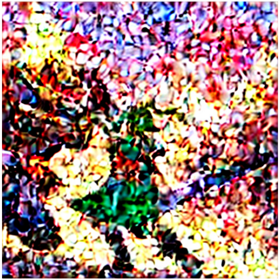
- Example of iterations of stable diffusion;
- Text input: "Link fighting with Ganon"
- Recall: "$X_t = aX_0 + b\varepsilon_t$"

Noise ε_t



Estimation* of X_0

X_t



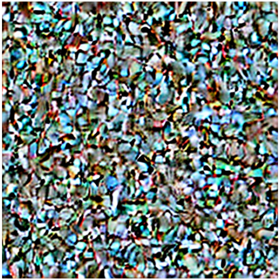
$t = 20$

** Stable Diffusion is in a latent space, so this is not exactly correct here*

Diffusion Models

- Example of iterations of stable diffusion;
- Text input: "Link fighting with Ganon"
- Recall: "$X_t = aX_0 + b\varepsilon_t$"

Noise ε_t



Estimation* of X_0

X_t

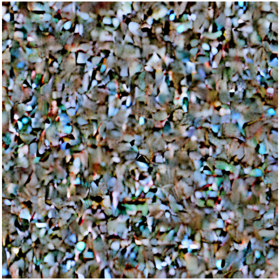
$t = 10$

** Stable Diffusion is in a latent space, so this is not exactly correct here*

Diffusion Models

- Example of iterations of stable diffusion;
- Text input: "Link fighting with Ganon"
- Recall: "$X_t = aX_0 + b\varepsilon_t$"

Noise ε_t



Estimation* of X_0

X_t

$t = 1$

Stable Diffusion is in a latent space, so this is not exactly correct here

Advantages of diffusion models

- More stable training wrt to GANs, which require a discriminator;
- Due to the sampling at each time step t, one initial noise x_T can produce many different outputs:

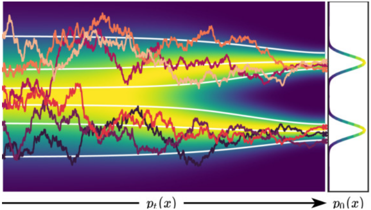


Illustration from Song et al 2021*

- VAEs and GANs produce (mostly) the same image for each initial noise

* *Score-Based Generative Modeling Through Stochastic Differential Equations, Song et al, ICLR 2021*

Disadvantage of diffusion models

- Networks f_θ tend to be **huge** !
  - Why is this ? Because f_θ has to denoise at a very wide range of noise levels (even when there is only noise)
- Theory can be quite complicated;
  - Not always explained or implemented clearly (various practical techniques)
  - Theory and practice are often not aligned;

Flow Matching

Flow matching

- Although frameworks differ, **flow matching** and diffusion are **almost identical** in practice;
- As for Diffusion Models, f is an **iteration** of a neural network;
- Formulated in terms of **trajectories** between μ_0 and μ_1, instead of a Markov chain (Diffusion Models);
  - However, leads to almost identical setting: **noising/denoising** images;

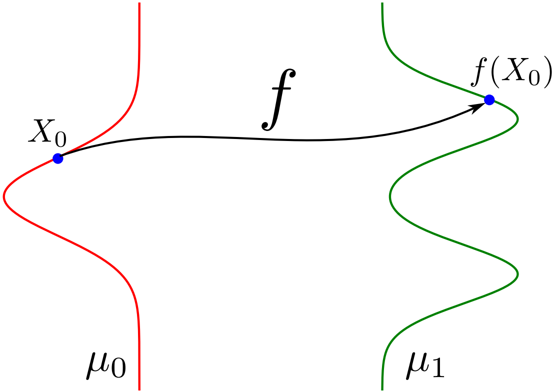
Flow matching

- Although frameworks differ, **flow matching** and diffusion are **almost identical** in practice;
- As for Diffusion Models, f is an **iteration** of a neural network;
- Formulated in terms of **trajectories** between μ_0 and μ_1, instead of a Markov chain (Diffusion Models);
  - However, leads to almost identical setting: **noising/denoising** images;

- Core idea of flow matching: use simple trajectories between μ_0 and μ_1 to learn f;
- In particular, we teach the network using **straight paths between samples $X_0 \sim \mu_0$ and $X_1 \sim \mu_1$**;

Introduction

- Recall of initial idea;

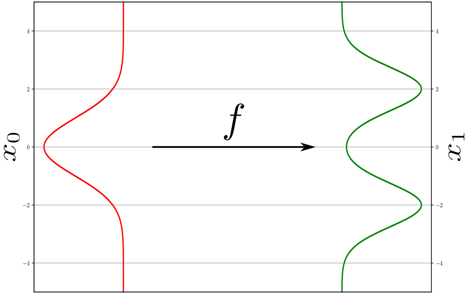


- Before looking into Flow Matching in more detail, we need to recall a notion from probability theory;

Flow matching

Pushforward measure

- Let X be a random variable, following distribution μ;
- The **pushforward measure** $f\#\mu$ is the **distribution** of $f(X)$;



- More formally, $f\#\mu$ is defined as the measure such that, for all sets B,

$$f\#\mu(B) := \mu(f^{-1}(B)) \tag{15}$$

Flow matching

- Back to flow matching. Let $(X_0, X_1) \sim \pi$ be two random variables;
- π is the joint distribution of (X_0, X_1), such that the marginals are μ_0 and μ_1 respectively;

Flow matching

- Back to flow matching. Let $(X_0, X_1) \sim \pi$ be two random variables;
- π is the joint distribution of (X_0, X_1), such that the marginals are μ_0 and μ_1 respectively;

Interpolation X_t

- We define the **interpolation** between X_0 and X_1:

$$X_t := (1-t)X_0 + tX_1 \tag{16}$$

- Define the interpolation function $g_t(x, y) := (1-t)x + ty$, we have:

$$X_t = g_t(X_0, X_1) \tag{17}$$

- Let ρ_t be the distribution of X_t. We can write this:

$$\rho_t = g_t \# \pi \tag{18}$$

- ρ_t is sometimes referred to as a **probability path**;

Flow matching

- Illustration of the interpolation between two probability distributions;

Flow matching

- This interpolation is useful because at $t = 0$ and $t = 1$ we have:

$$g_0(X_0, X_1) := (1 - 0)X_0 + 0X_1 = X_0 \tag{19}$$
$$g_1(X_0, X_1) := (1 - 1)X_0 + 1X_1 = X_1. \tag{20}$$

- Therefore, $\rho_0 = \mu_0$, $\rho_1 = \mu_1$;
- Thus, ρ_t verifies the correct distributions at the beginning and end (unsurprisingly);
  - If we can draw a sample X_t, we can draw a sample $X_1 \sim \mu_1$;

Flow matching

- This interpolation is useful because at $t = 0$ and $t = 1$ we have:

$$g_0(X_0, X_1) := (1 - 0)X_0 + 0X_1 = X_0 \tag{19}$$

$$g_1(X_0, X_1) := (1 - 1)X_0 + 1X_1 = X_1. \tag{20}$$

- Therefore, $\rho_0 = \mu_0$, $\rho_1 = \mu_1$;
- Thus, ρ_t verifies the correct distributions at the beginning and end (unsurprisingly);
    - If we can draw a sample X_t, we can draw a sample $X_1 \sim \mu_1$;
- Unfortunately, we have to be able to sample from both μ_0 and μ_1 to produce X_t, so unusable as such;
- We have to find an indirect way to determine X_t and ρ_t: we will use a **flow**

Flow matching

- Flow matching consists in defining a **flow** which represents ρ_t;
- A flow is a function from $\mathbb{R}^d$ to $\mathbb{R}^d$ which is determined by a **velocity field**;
  - Originally, flows represent fluids in fluid dynamics;

Flow

- Let $\phi_t : \mathbb{R}^d \mapsto \mathbb{R}^d$ be a function such that:

$$\begin{aligned} \frac{d\phi_t(x)}{dt} &= v_t(x) && \text{Velocity field} \\ \phi_0(x) &= x && \text{Starting point: identity at 0} \end{aligned} \tag{21}$$

- v_t is a smooth function which defines the motion of the flow;

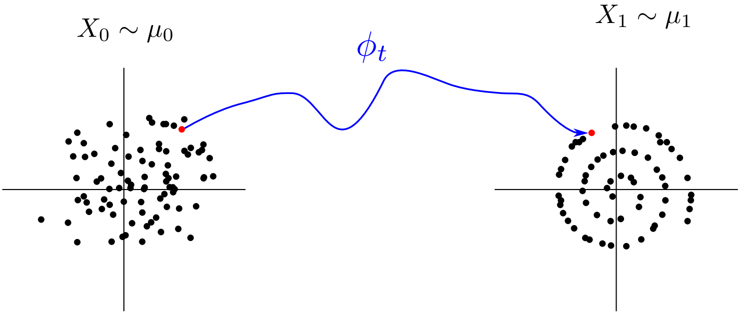
Flow matching

- Once v_t is established, ϕ_t is determined **uniquely** as the solution of the flow equation (21)
- Thus, for a given $t \in (0, 1)$ we have:

$$\phi_t(x) = x + \int_0^t v_\tau(x_\tau) d\tau \tag{22}$$

Flow matching

- Core idea of flow matching: if we choose v_t correctly, we can use the flow ϕ_t to "transport" samples from X_0 to X_1;
- This avoids having to determine X_t, ρ_t directly, we only need μ_0 and ϕ_t;



- Actually, this is similar to a GAN: a function f to transport μ_0 to μ_1;
- Main difference: ϕ_t given by an integration over time of a function v_t;

Flow matching

- Flow matching proposes to define $v_t(x)$ as the **conditional expectation** of the velocity knowing X_t;

Flow matching velocity field

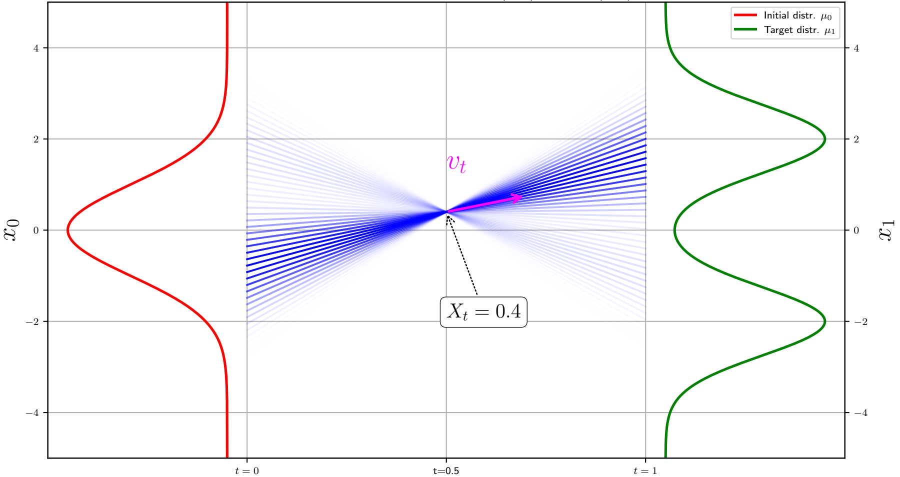
- Let $(X_0, X_1) \sim \pi$ such that the marginals are μ_0, μ_1;
- The flow matching velocity field is defined as:

$$v_t(x) := \mathbb{E}\left[X_1 - X_0 | X_t = x\right] \tag{23}$$

- Indeed, $\frac{\partial g_t}{\partial t}(X_0, X_1) = \frac{\partial}{\partial t}\left((1-t)X_0 + tX_1\right) = X_1 - X_0$;
- Fix a time t and a position x, and calculate the **average velocity of all straight paths** passing through x;

Flow matching

Illustration of velocity $v_t(x) = E[X_1 - X_0 | X_t = x]$, with $x = 0.4$, with $t = 0.5$
Path transparency $\propto \mu_0(x_0) \times \mu_1(x_1)$



Flow matching

- So, we now have a way to establish a flow ϕ_t;
- Let $\tilde{\rho}_t$ be the probability path defined with μ_0 and ϕ_t:

$$\tilde{\rho}_t = \phi_t \# \mu_0 \tag{24}$$

- $\tilde{\rho}_t$ is the distribution of $\phi_t(X_0)$, with $X_0 \sim \mu_0$;

Flow matching

- So, we now have a way to establish a flow ϕ_t;
- Let $\tilde{\rho}_t$ be the probability path defined with μ_0 and ϕ_t:

$$\tilde{\rho}_t = \phi_t \# \mu_0 \tag{24}$$

- $\tilde{\rho}_t$ is the distribution of $\phi_t(X_0)$, with $X_0 \sim \mu_0$;
- Main question: **is v_t correctly designed** to ensure that, for all $t \in [0, 1]$

$$\tilde{\rho}_t = \rho_t, \quad \text{almost everywhere} \quad ? \tag{25}$$

- Why ? Because ρ_t has the good properties ($\rho_1 = \mu_1$), but not necessarily $\tilde{\rho}_t$
- **We need to have $\tilde{\rho}_1 = \mu_1$**, otherwise the flow is useless;
- For this, we turn to the **continuity equation**;

Flow matching

Continuity equation

- Let ρ_t be the density of a flow (ie the probability path), and v_t the velocity field of this flow. Then we have:

$$\frac{\partial \rho_t}{\partial t} + div(\rho_t v_t) = 0 \tag{26}$$

- We say that the couple (ρ_t, v_t) solves the continuity equation;

- The continuity equation will allow us to prove that $\tilde{\rho}_t = \rho_t$, a.e.;

Flow matching

- We also know that the solution to the continuity equation is **unique**, given a fixed initial condition;

Characterisation and uniqueness of solutions to the continuity equation

- Let $v_t : \mathbb{R}^d \rightarrow \mathbb{R}^d$ be a velocity field and ϕ_t the corresponding flow, and consider some initial distribution μ_0;
- Then the distribution $\phi_t \# \mu_0$ and v_t solve the continuity equation;
- Furthermore, with initial condition μ_0, **the solution (ρ_t) to the continuity equation is unique**;

Flow matching

Proposition: (ρ_t, v_t) solve the continuity equation

- Recall that $\rho_t = g_t \# \pi$;
- It can be shown that (ρ_t, v_t) **solve the continuity equation**;
- This means that v_t indeed leads to a flow ϕ_t such that:

$$\phi_t \# \mu_0 = \tilde{\rho}_t \qquad \text{by definition} \tag{27}$$

$$= \rho_t \quad a.e. \tag{28}$$

- We know that $\tilde{\rho}_t = \rho_t$ a.e. because solution to the continuity equation is **unique**;

- In summary: we can sample from $\tilde{\rho}_t$ by using $\phi_t(X_0)$, and it happens that $\tilde{\rho}_t = \rho_t$;
- Thus, $\phi_1(X_0) \sim \mu_1$, achieving our original goal !!

Flow matching summary - the story so far

1. Calculate $v_t(x) = \mathbb{E}\left[X_1 - X_0 | X_t = x\right]$;
2. Sample $X_0 \sim \mu_0$;
3. $X_1 = \phi_t(X_0) = X_0 + \int_0^1 v_t(X_0) d_t$

Flow matching summary - the story so far

1. Calculate $v_t(x) = \mathbb{E}\left[X_1 - X_0 | X_t = x\right]$;
2. Sample $X_0 \sim \mu_0$;
3. $X_1 = \phi_t(X_0) = X_0 + \int_0^1 v_t(X_0) d_t$

Remaining questions

1. How to calculate $v_t(x) = \mathbb{E}\left[X_1 - X_0 | X_t = x\right]$?
   - Not trivial, since we do not know μ_1;
2. How to calculate $\int_0^1 v_t(X_0) d_t$?
   - Numerical approximation of integral;

Calculating $v_t(x) = \mathbb{E}\left[X_1 - X_0 | X_t = x\right]$

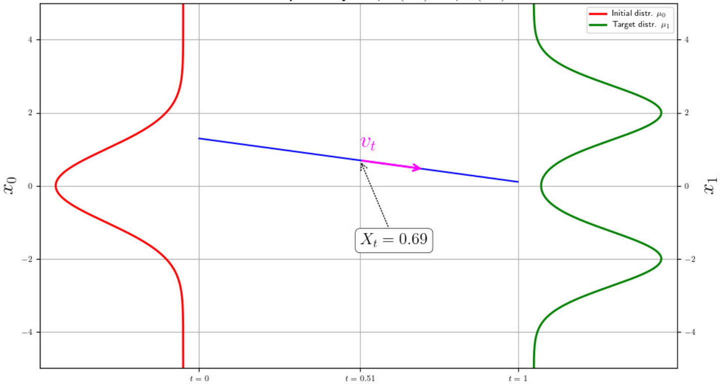
- Unsurprisingly, we use a **neural network** to approximate $\mathbb{E}\left[X_1 - X_0 | X_t = x\right]$;

Approximation of v_t

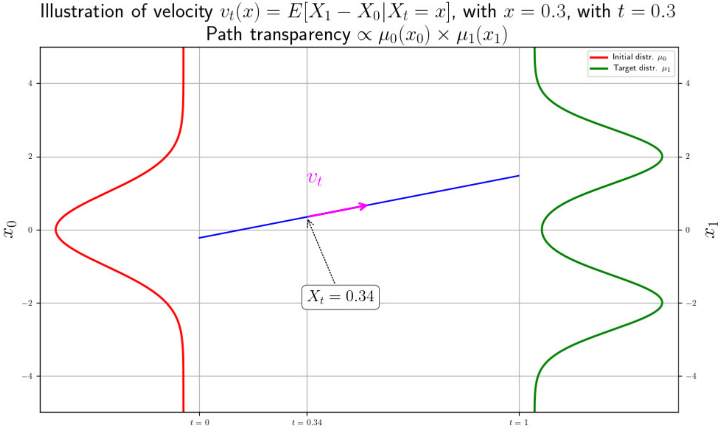
for $i = 1$ to N **do**
  Draw $t \in \mathcal{U}([0,1])$
  Draw $X_0 \sim \mu_0$
  Draw X_1 from the database
  $X_t = (1-t)X_0 + tX_1$
  Minimise$_\theta$ $\|(X_1 - X_0) - f_\theta(X_t, t)\|_2^2$
end for

Training f_θ

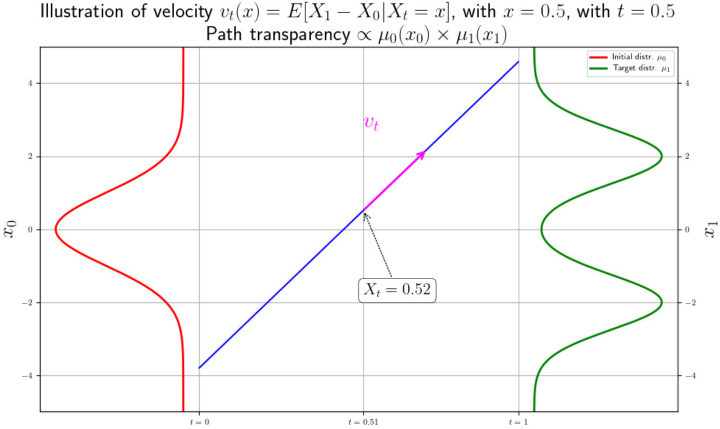
Illustration of velocity $v_t(x) = E[X_1 - X_0 | X_t = x]$, with $x = 0.7$, with $t = 0.5$
Path transparency $\propto \mu_0(x_0) \times \mu_1(x_1)$



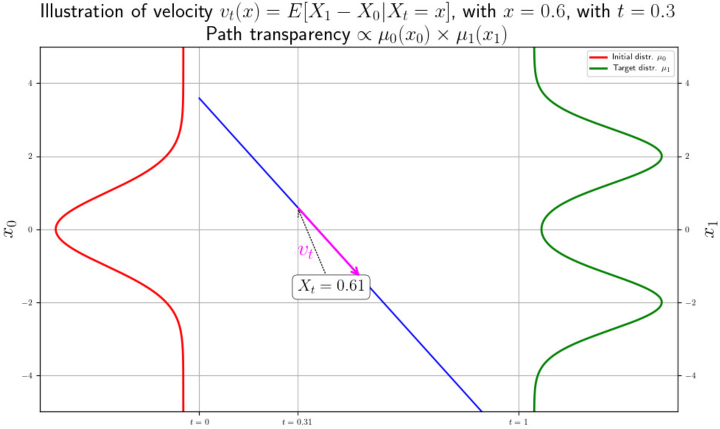
Training f_θ



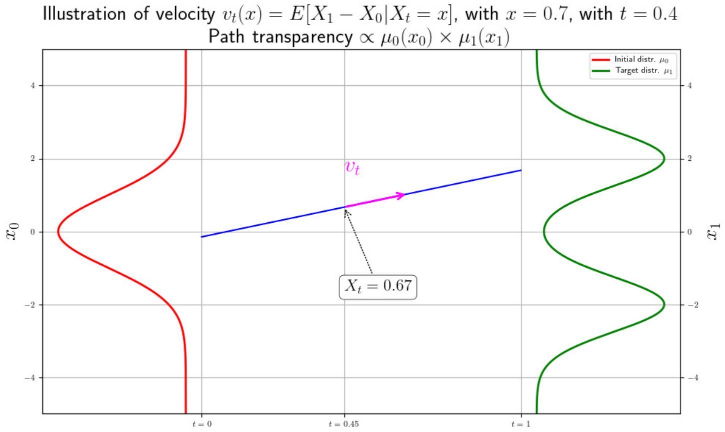
Training f_θ



Training f_θ



Training f_θ



Flow matching

Numerical integration

- Determining $\phi_1(X_0) = \int_0^1 v_t(X_0) d_t$ requires a numerical integration;
- Simplest option, Euler scheme

Flow matching

Numerical integration

- Determining $\phi_1(X_0) = \int_0^1 v_t(X_0) d_t$ requires a numerical integration;
- Simplest option, Euler scheme

Euler scheme for numerical integraion

- Let $N > 0$ be the number of numerical integration steps;
- Let $t_1, \dots, t_N$ be a sequence of discrete time steps:
  - In general, $t_i = \frac{i}{N}$ (but this could be modified)

$X_0 \sim \mathcal{N}(0, Id)$
$X = X_0$
for $i = 1$ to $N - 1$ **do**
  $X = X + (t_{i+1} - t_i) f_\theta(X, t_i)$
end for
Return $\phi_1(X_0) \leftarrow X$

Flow matching

Translation between Diffusion Models and Flow Matching terms

Meaning	**Diffusion Models (DM)**	**Flow Matching (FM)**
Data sample	X_0	X_1
Latent / noise sample	X_T	X_0
Intermediate state	$X_t = \alpha_t X_0 + \sigma_t \varepsilon$	$X_t = \alpha_t X_1 + \beta_t X_0$
Noise variable	$\varepsilon_t \sim \mathcal{N}(0, I)$	X_0
Time variable	$t \in [0, T]$ (diffusion)	$t \in [0, 1]$ (interpolation)
Predicted quantity	$\varepsilon_\theta(X, t)$	$v_\theta(X, t)$

Conclusion

- Diffusion Models and Flow Matching are **extremely similar**;
- Major differences:
  - No noise between X_t and X_{t-1} in diffusion: deterministic from X_0;
  - Diffusion can never reach complete noise, requires infinite T;
- Flow Matching formulation **simpler to explain**, although mathematics behind it may be more sophisticated;

References

A few references

Some legitimate questions !

- Why can't we just train a network to predict/sample x_{t-1} directly from x_t ?
  - Why do we have to sample it *indirectly* via x_0 ?

Some legitimate questions !

- Why can't we just train a network to predict/sample x_{t-1} directly from x_t ?
  - Why do we have to sample it *indirectly* via x_0 ?
  - Answer: it is **difficult for the network to predict the same image with slightly less noise** $(x_t \rightarrow x_{t-1})$;
  - If you could do this, diffusion models would be much simpler (less maths);

Some legitimate questions !

- Why can't we just train a network to predict/sample x_{t-1} directly from x_t ?
  - Why do we have to sample it *indirectly* via x_0 ?
  - Answer: it is **difficult for the network to predict the same image with slightly less noise** ($x_t \rightarrow x_{t-1}$);
  - If you could do this, diffusion models would be much simpler (less maths);
- **Why is it better to carry out an iterative diffusion process**, rather than just one step (as in VAEs/GANs) ?

Some legitimate questions !

- Why can't we just train a network to predict/sample x_{t-1} directly from x_t ?
  - Why do we have to sample it *indirectly* via x_0 ?
  - Answer: it is **difficult for the network to predict the same image with slightly less noise** $(x_t \rightarrow x_{t-1})$;
  - If you could do this, diffusion models would be much simpler (less maths);
- **Why is it better to carry out an iterative diffusion process**, rather than just one step (as in VAEs/GANs) ?
  - This is currently a subject of research;