Generative Models

BIFOLD Aqtivate Workshop | 26. Februar 2024

Khaled Kahouli

BIFOLD | Machine Learning Group – Technische Universität Berlin

Table of contents

- Motivation
- Generative Models
  - Autoregressive Models
  - Variational Autoencoders (VAE)
  - Diffusion Models

Motivation

Supervised vs Unsupervised Models

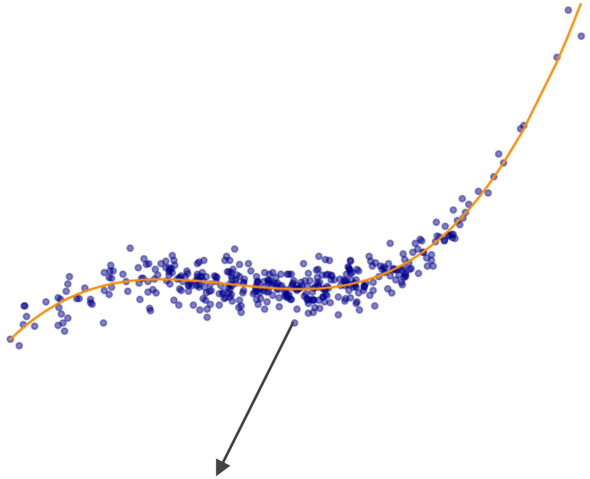
Supervised Models:

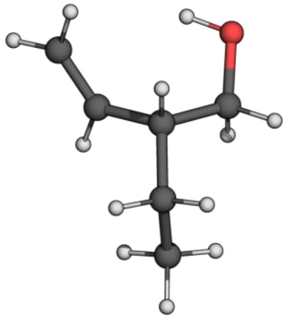
- **Given**: data x and labels y
- **Goal**: estimate the conditional distribution $\boldsymbol{p}(\mathbf{y}|\mathbf{x})$
- **Solution**: Use the training data to learn a mapping function $\boldsymbol{f}\colon\ \mathbf{y} = \boldsymbol{f}(\boldsymbol{x})$

Supervised vs Unsupervised Models

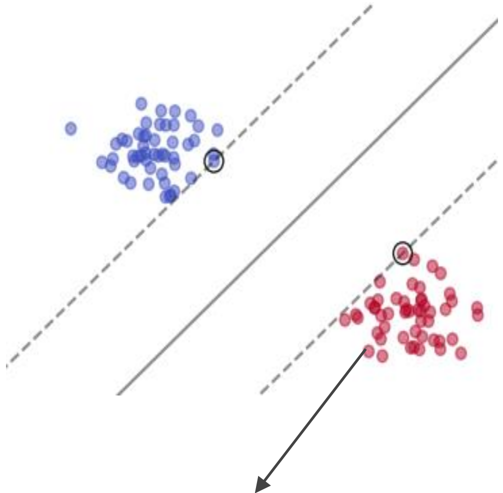
Supervised Models:

- $y \in \mathbb{R}$: ***regression***



$f($  $) = -6534\ eV$

- y are categories: ***classification***



$f($  $) = cat$

Image generated by DALL-E 3

Supervised vs Unsupervised Models

Supervised Models:

- **Problems:**
  - often very *costly* to get the labels y.
    - E.g. calculating the true energy for one molecule takes few hours for small systems with 9 heavy atoms up to several months for large systems like materials.
  - Unlabeled data are usually very *cheap* and *everywhere.*
  - Can be boring: most of the time the solution lies already in the labels.

    (At least you can't get to AGI by merely discriminating between objects in the world)

=> Interest in different tasks than $p(\mathbf{y}|\mathbf{x})$

Supervised vs Unsupervised Models

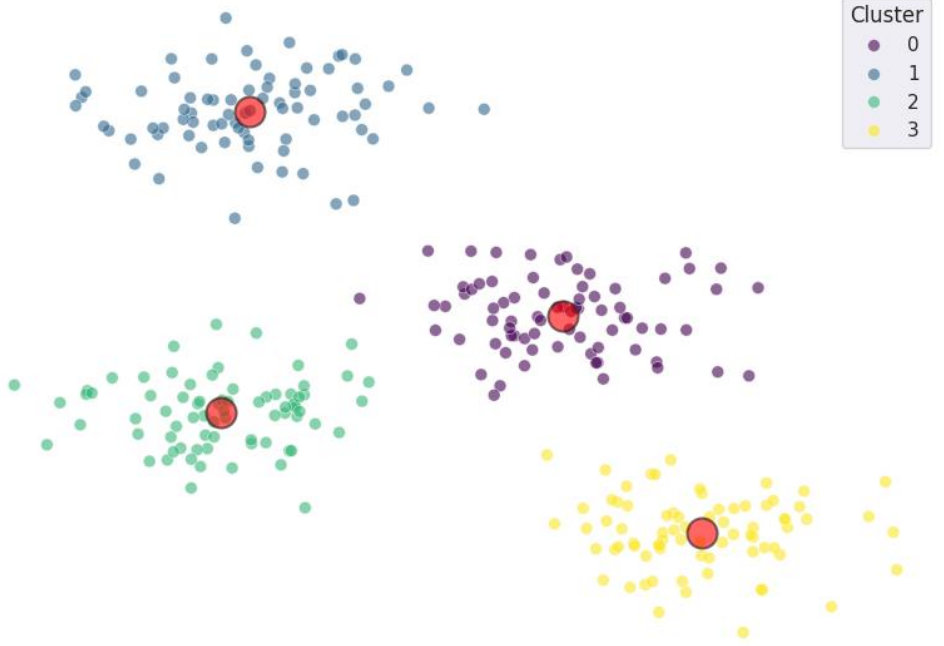
Unsupervised Models:

- **Given:** *unlabeled* data $\mathbf{x}$.
- The unsupervised approach is an umbrella term for different approaches.
- **Abstract Goal:** Reveal the underlying hidden structures of the data.
- (Ultimate Goal: extract some meaning from the data => sounds promising !)

Supervised vs Unsupervised Models

Unsupervised Models:

- **Example tasks:**
  - **Clustering**: identify relevant subgroups.



Supervised vs Unsupervised Models

Unsupervised Models:

- **Example tasks:**
  - **Dimensionality reduction**: the data lies in a low dimensional subspace

    => Manifold Hypothesis

Supervised vs Unsupervised Models

Unsupervised Models:

- **Example tasks:**
  - **Self-supervised learning**:
    - First Learn useful representation by pre-training on unlabeled data
    - Then Use the representation to solve different down stream tasks, e.g. classification

=> less/zero labeled data.

Supervised vs Unsupervised Models

Unsupervised Models:

- **Example tasks:**
  - **Density estimation using generative models:**
    - **Problem:** Most of the time we have data samples x drawn from some distribution $\boldsymbol{p}(\mathbf{x})$. But we do not know $\boldsymbol{p}(\mathbf{x})$ or have access to it.
    - **Solution**:
      - Learn a generative model from the known samples $\mathbf{x}$ to approximate $\boldsymbol{p}(\mathbf{x})$.
      - Model $\boldsymbol{p}(\mathbf{x}, \mathbf{y})$ or $\boldsymbol{p}(\mathbf{x}|\mathbf{y})$, If labeled data $(\mathbf{x}, \mathbf{y})$ are given.
    - Generative models can also solve discriminative tasks:
      - $\boldsymbol{p}(\mathbf{y}|\mathbf{x}) \propto \boldsymbol{p}(\mathbf{x}, \mathbf{y}) = \boldsymbol{p}(\mathbf{x}|\mathbf{y})\boldsymbol{p}(\mathbf{y})$

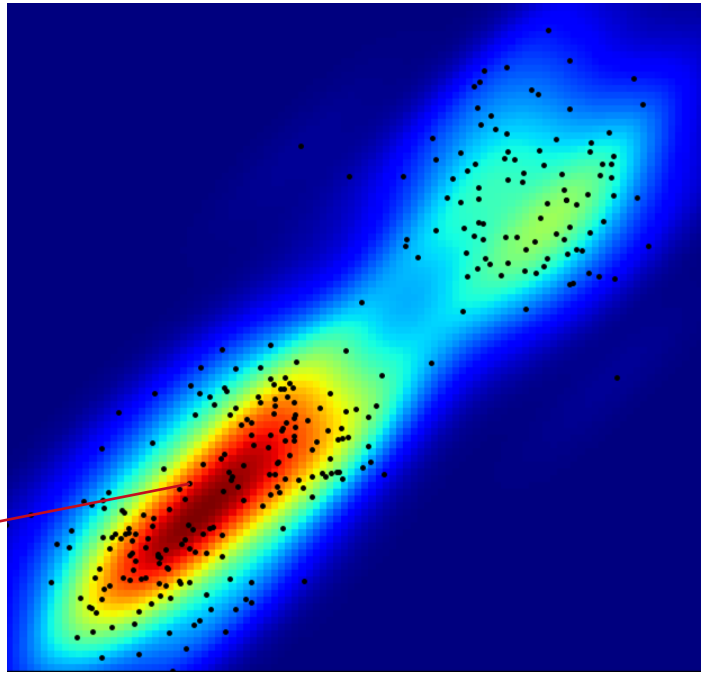
Generative Models – Density Estimation

Two objectives for generative models:

1. Learn $\boldsymbol{p_{model}}(\mathbf{x})$ to approximate the true $\boldsymbol{p_{data}}(\mathbf{x})$ => **Density estimation**

   What is the probability of $\mathbf{x}$ comping from $\boldsymbol{p_{data}}$?

Image generated by DALL-E 3



$\boldsymbol{p_{model}}(\mathbf{x}|\mathbf{y} = \boldsymbol{cat})$

Generative Models – Sample Generation

Two objectives for generative models:

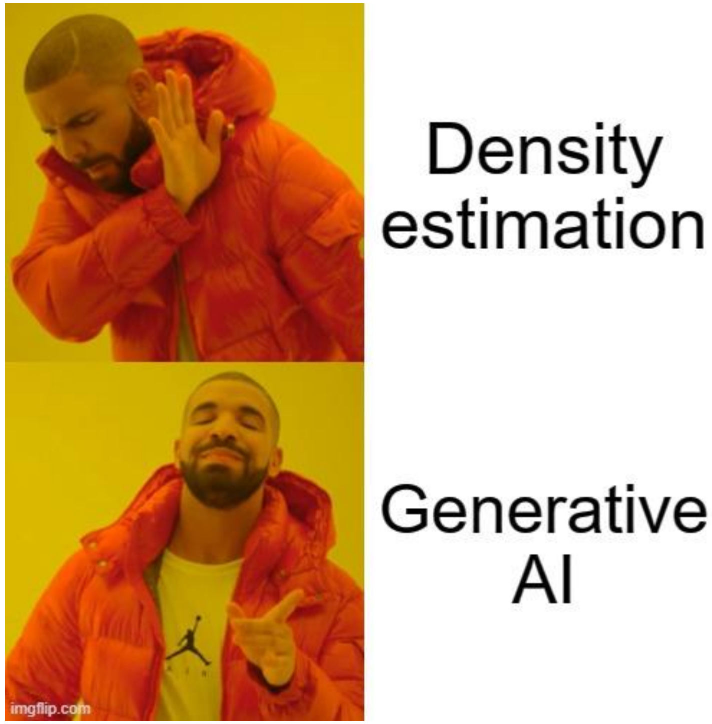
1. Learn $\boldsymbol{p_{model}}(\mathbf{x})$ to approximate the true $\boldsymbol{p_{data}}(\mathbf{x})$ => **Density esimation**
2. Generate new samples $\mathbf{x}$ from $\boldsymbol{p_{model}}(\mathbf{x}) \approx \boldsymbol{p_{data}}(\mathbf{x})$

   Conditional generation $\boldsymbol{p_{model}}(\mathbf{x}|\mathbf{y})$: generate an image of a cat instead of any image

Generative Models – Generative AI

Two objectives for generative models:

1. Learn $\boldsymbol{p_{model}}(\mathbf{x})$ as approximate of the true $\boldsymbol{p_{data}}(\mathbf{x})$ => **Density esimation**
2. Generate new samples $\mathbf{x}$ from $\boldsymbol{p_{model}}(\mathbf{x}) \approx \boldsymbol{p_{data}}(\mathbf{x})$ => **Generative AI** (ChatGPT and Co.)



Generative Models are Hard

Images generated by DALL-E 3

Foward

Inverse

“cat”

Unlike forward problems, inverse problems are 1 to N mappings.

=> **“ill-posed”** problem.

=> Can not define a function: “cat” -> image.

=> Harder to solve!

But Why Generative Models then ?

- Solving the inverse problem implicitly solves the forward problem:
  - Generative models can solve discriminative tasks, e.g. by modeling $p(\mathbf{x}|\mathbf{y})$.
- To generate useful samples, the model needs deep understanding of the patterns in the data.
  - The representation learned by a generative model can be used for different downstream tasks
  - **Disclaimer**: this does not always work!
- Generative models can solve different tasks, e.g. anomaly detection, denoising, inpainting, …
- It is more interesting to discover a new molecule with specific properties than to predict the properties of a given known molecule.
- (ChatGPT is also a generative model)

Generative Models

Types of Generative Models

- **Explicit generative models:** explicitly optimize for $\boldsymbol{p_{model}}(\mathbf{x}) \approx \boldsymbol{p_{data}}(\mathbf{x})$

  => can draw samples $\mathbf{x} \sim \boldsymbol{p_{model}}(\mathbf{x})$

  => Usually evaluating $\boldsymbol{p_{model}}(\mathbf{x})$ is tractable

  => Explicit density estimation

- **Implicit generative models**: define a process/model to sample from $\boldsymbol{p_{model}}(\mathbf{x})$ without explicitly defining it.

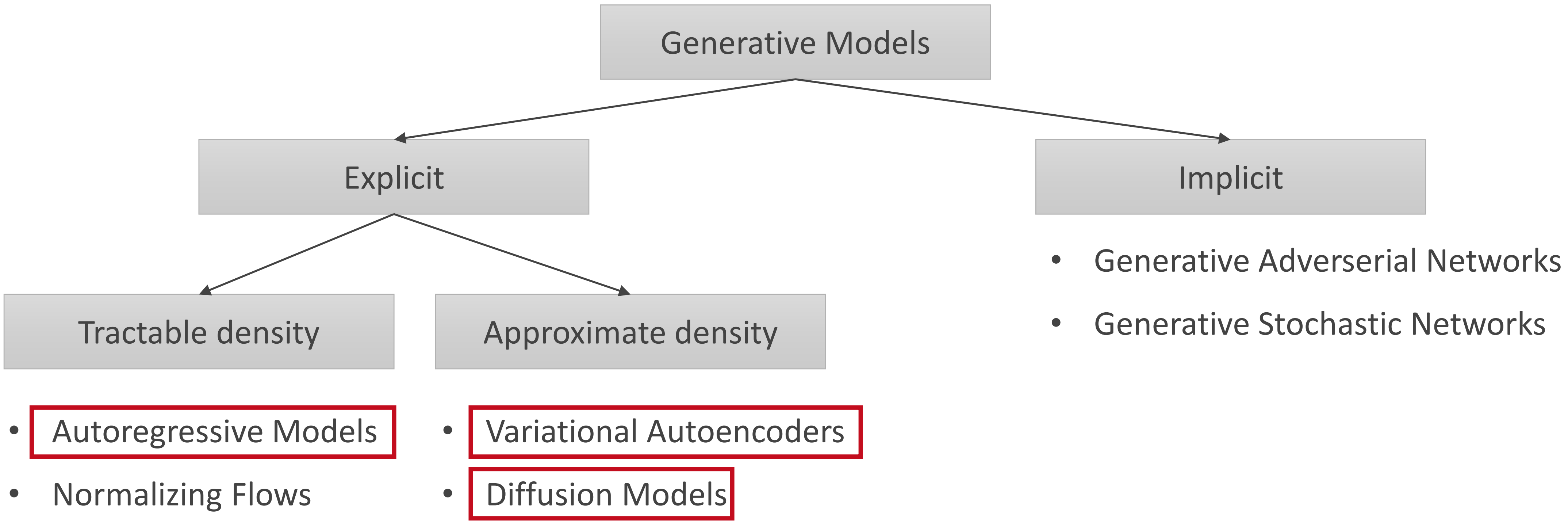
  => efficient sampling from $\boldsymbol{x} \sim \boldsymbol{p_{model}}(\mathbf{x})$

  => Computing $\boldsymbol{p_{model}}(\mathbf{x})$ is intractable

  => Implicit density estimation

Usually, implicit models provide faster sample generation at the cost of an intractable estimation of $\boldsymbol{p_{model}}(\boldsymbol{x})$.

Types of Generative Models



Autoregressive Models

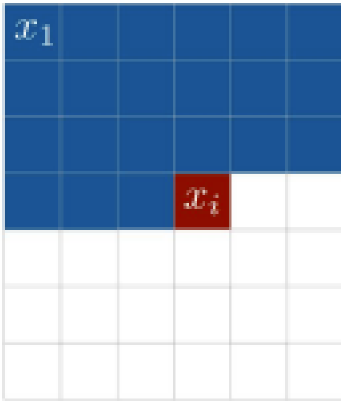
Autoregressive Models

- First introduced as **Fully visible belief networks (FVBN).**
- Given data point $\mathbf{x} = (x_1, x_2, \dots, x_n)$ decompose the joint likelihood using the chain rule:

$$p(\mathbf{x}) = p(x_1, x_2, \dots, x_n)$$

$$= \prod_{i=1}^{n} p(x_i | x_{i-1}, \dots, x_2, x_1)$$

For instacen $\mathbf{x}$ is an image or a molecule and x_i is one pixel or one atom.



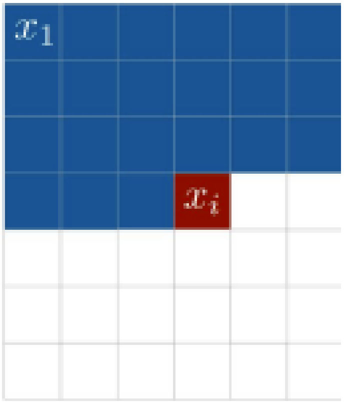
[2]

Autoregressive Models

- First introduced as **Fully visible belief networks (FVBN).**
- Given data point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ decompose the joint likelihood using the chain rule:

$$p(\mathbf{x}) = p\,(x_1, x_2, \ldots, x_n)$$

$$= \prod_{i=1}^{n} p(x_i | x_{i-1}, \ldots, x_2, x_1)$$



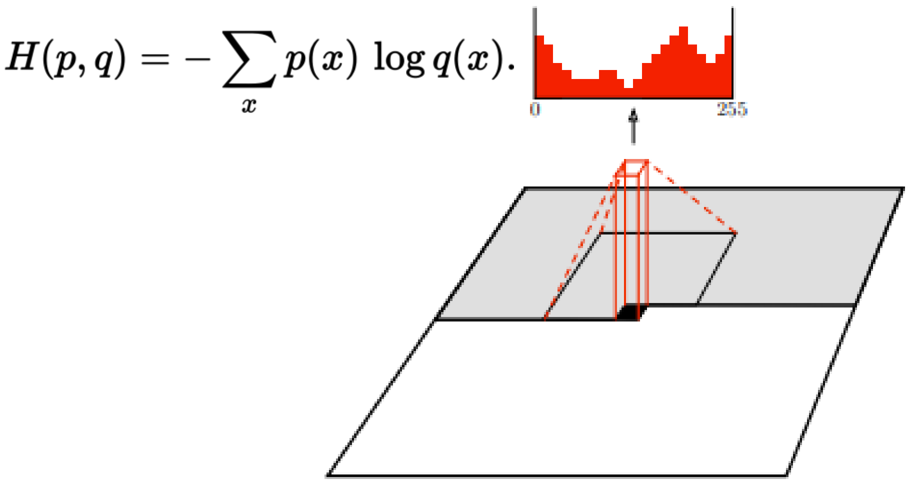
[2]

For instacen $\mathbf{x}$ is an image or a molecule and x_i is one pixel or one atom.

- Minimize the negative log-likelihood (NLL) of the training data $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N$:

$$\min -\sum_{j=1}^{N} \log p(\mathbf{x}^j)$$

PixelCNN [1]



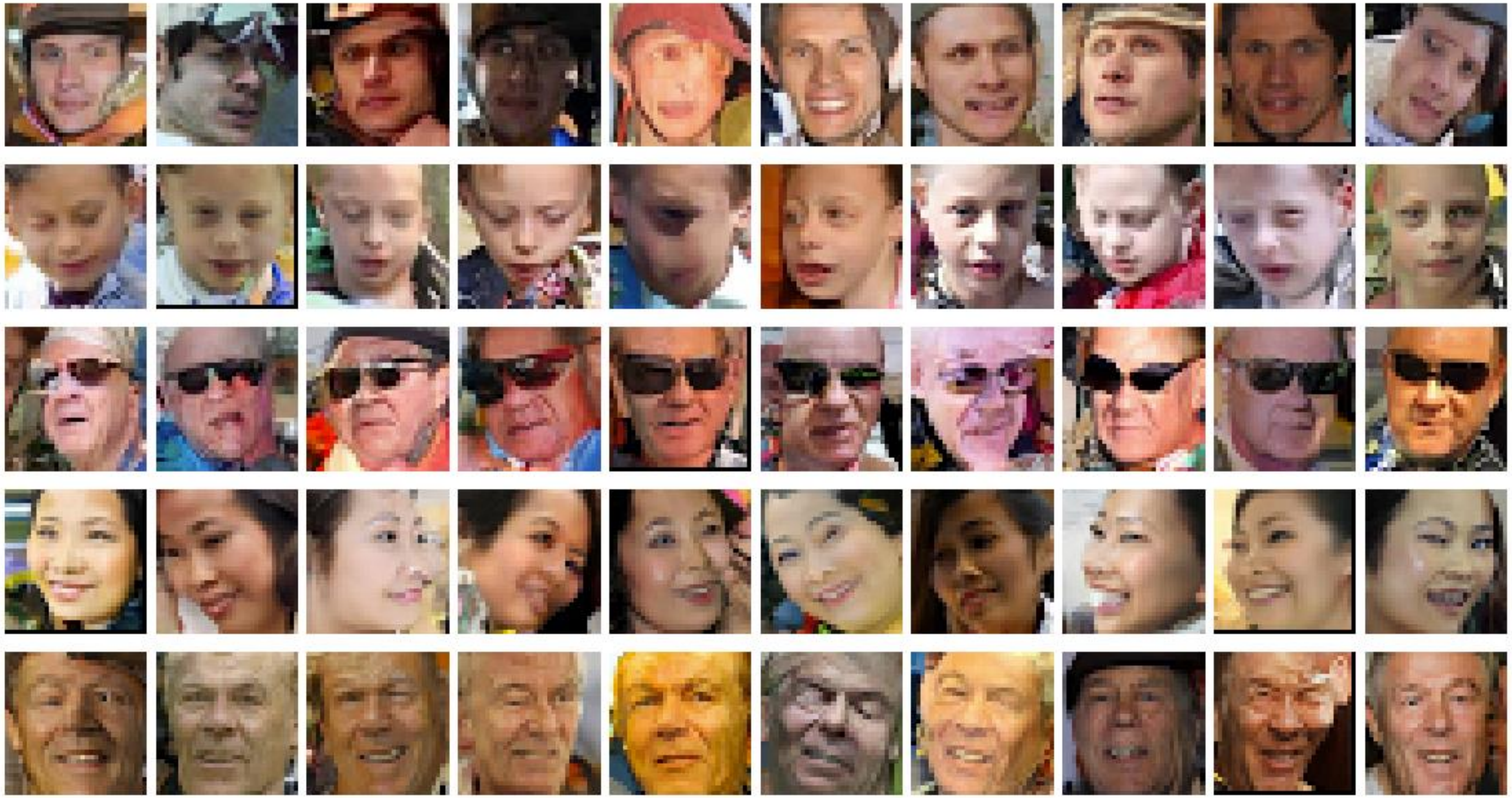
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

[2]

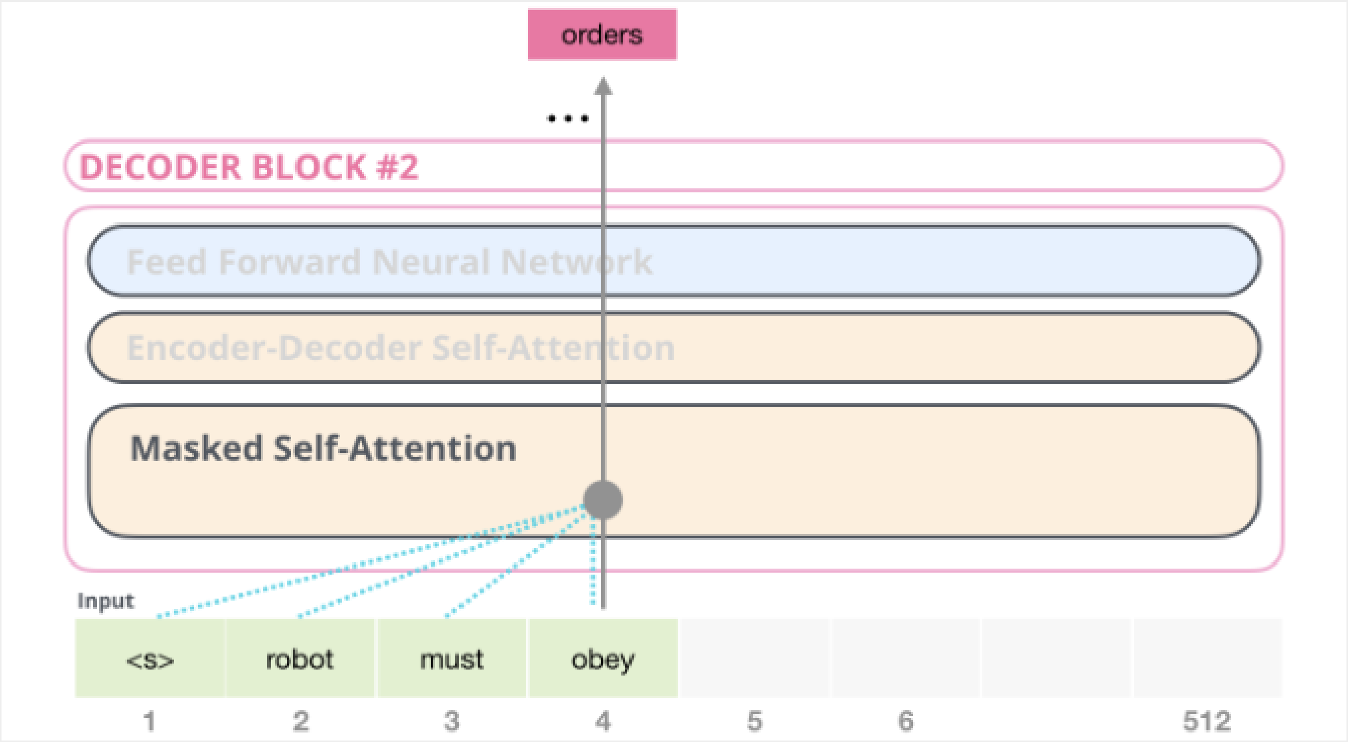
- Generates image pixelwise starting from upper left corner.
- Uses **masked convolutions** to define the current context:
  - Model can not use information from the future $\boldsymbol{x_{i+1}, \dots, x_n}$ only from the past $\boldsymbol{x_1, x_2, \dots, x_{i-1}}$
  - Limit the context to the last j pixels only $\boldsymbol{p(x_i|x_{i-1}, \dots, x_{i-j})}$, e.g. $\boldsymbol{j = 5}$.

    => faster training + can be parallelized
- Use pixelwise **cross-entropy** as loss to minimize the NLL of the training data.

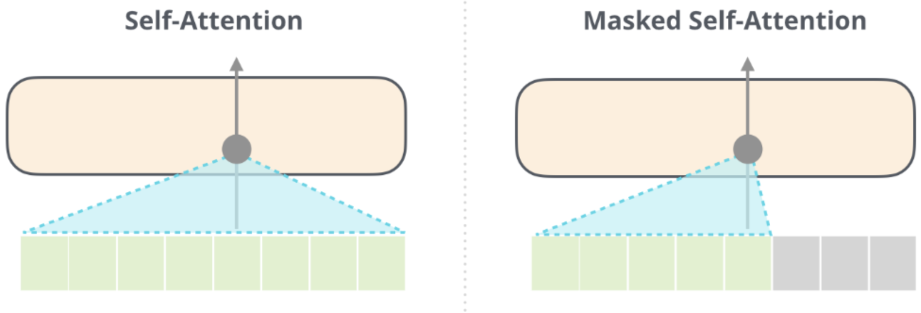
PixelCNN [1]



Generative Pre-trained Transformers (GPT 1-4)



Source: https://jalammar.github.io/illustrated-gpt2/



- Autoregressive generative models that are pre-trained to predict the next token conditioned on previous tokens
- Predict the one-hot encoded vector over the vocabulary + cross entropy loss.
- + Reinforcement learning through human feedback (RLHF) = ChatGPT

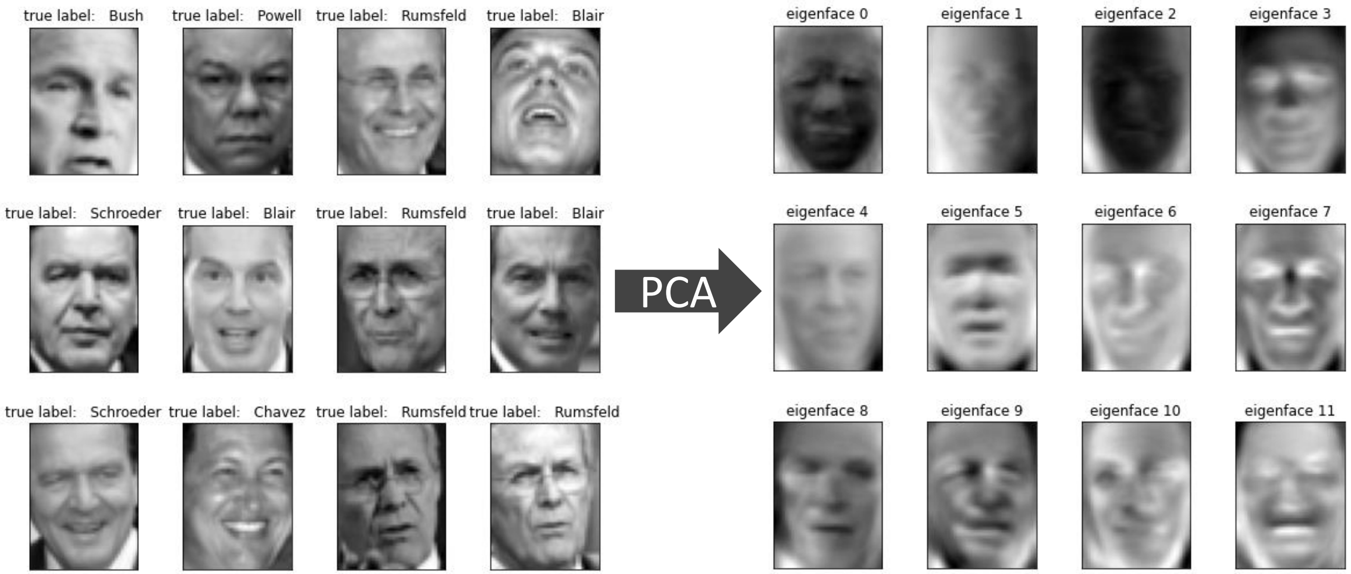
Autoregressive Models

- **Pros:**
  - Easy to optimize
  - Exact and **tractable likelihood** estimation
  - Usually achieves higher negative log-likelihood than other generative models
- **Cons:**
  - Sequential generation in input space => scales badly with sample size => **slow**.
  - Error can build-up during sampling and can not be fixed later.
  - Usually, worse sample quality than other models (exception: language models).

Variational Autoencoder (VAE)

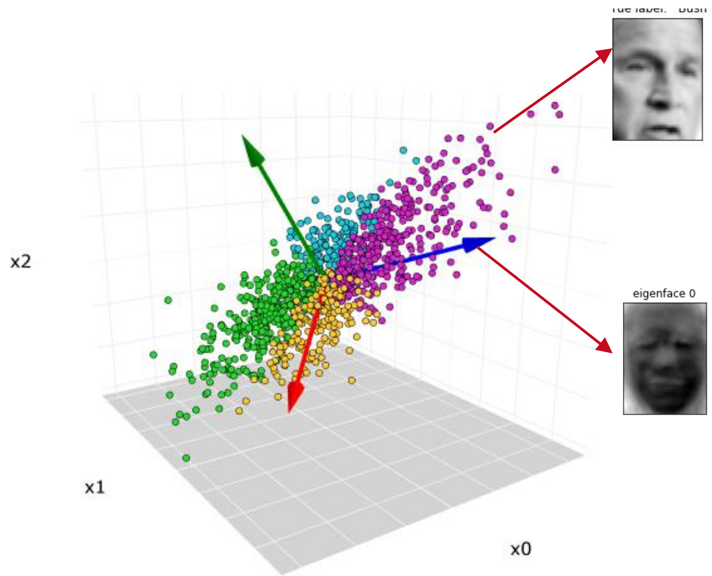
Latent Variable Models

- **Idea:**
  - **Manifold Hypothesis:** In a high-dimensional input space, the data are actually located in a manifold in a low-dimensional sub-space.
  - This low-dimensional subspace is often referred to as the **latent space.**



input space (Dim: 32x32)

latent space (Dim: 11)



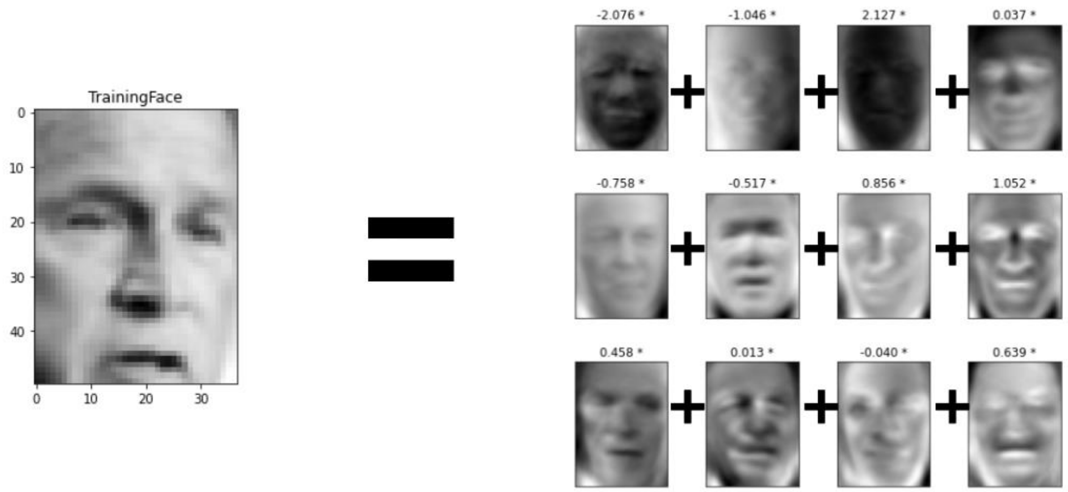
Faces Images: https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/
Right plot: towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d

Latent Variable Models

- **Idea:**
  - Data point **x** can be generated from few latent factors / variables **z.**
    - Usually, we can describe an image using few words, much less than the number of pixels.
  - Most of the time the latent variables **z** are independent.

    => No need for dependencies among pixels .

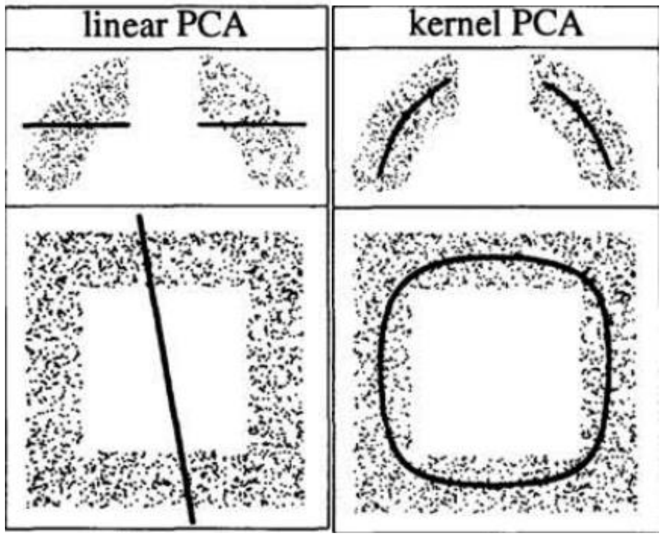
    => Generate all the pixels at once.



Faces Images: https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/

Latent Variable Models

- **Questions:**
    1. How to define and optimize the latent variables **z?**
    2. Given **z** how to generate meaningful data **x** on the data manifold.
- PCA can extract latent variables. Yet it is a linear model and assumes the data to be Gaussian.
- Kernel PCA: Non-linear model but requires manual hand-crafted kernels.



Latent Variable Models

- **Questions:**

  1. How to define and optimize the latent variables **z?**

  2. Given **z** how to generate meaningful data **x** on the data manifold.

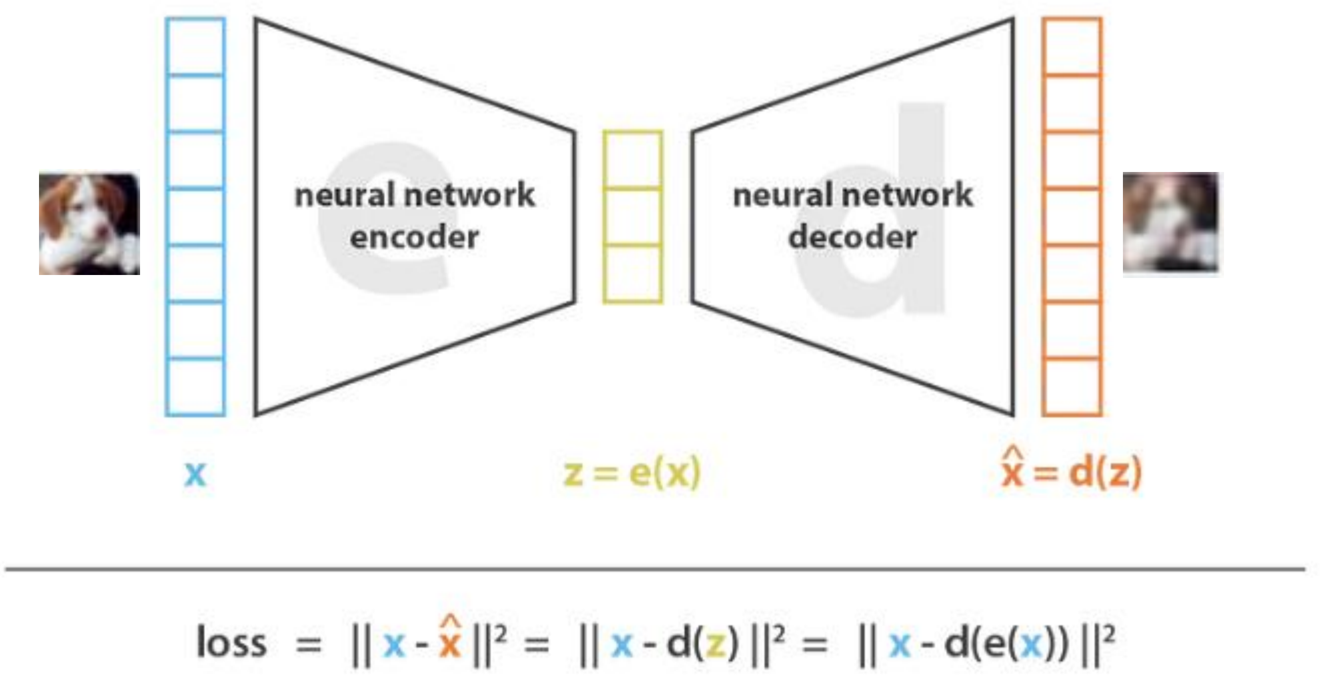
- PCA can extract latent variables. Yet it is a linear model and assumes the data to be Gaussian.

- Kernel PCA: Non-linear model but requires manual hand-crafted kernels.

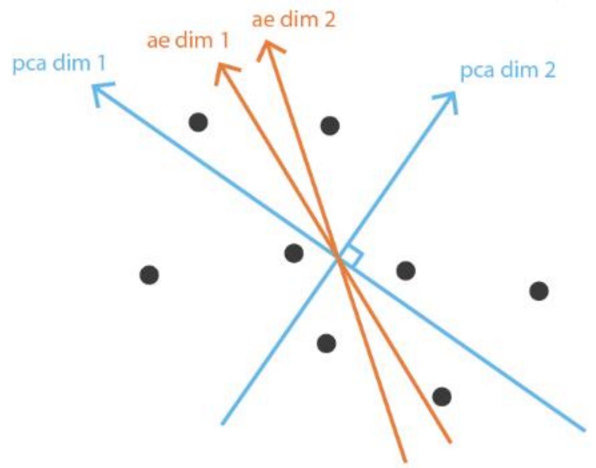
- Can we use neural networks to solve complex non-linear problems ?

  => Yes: **Autoencoders** as non-linear PCA.

Autoencoders



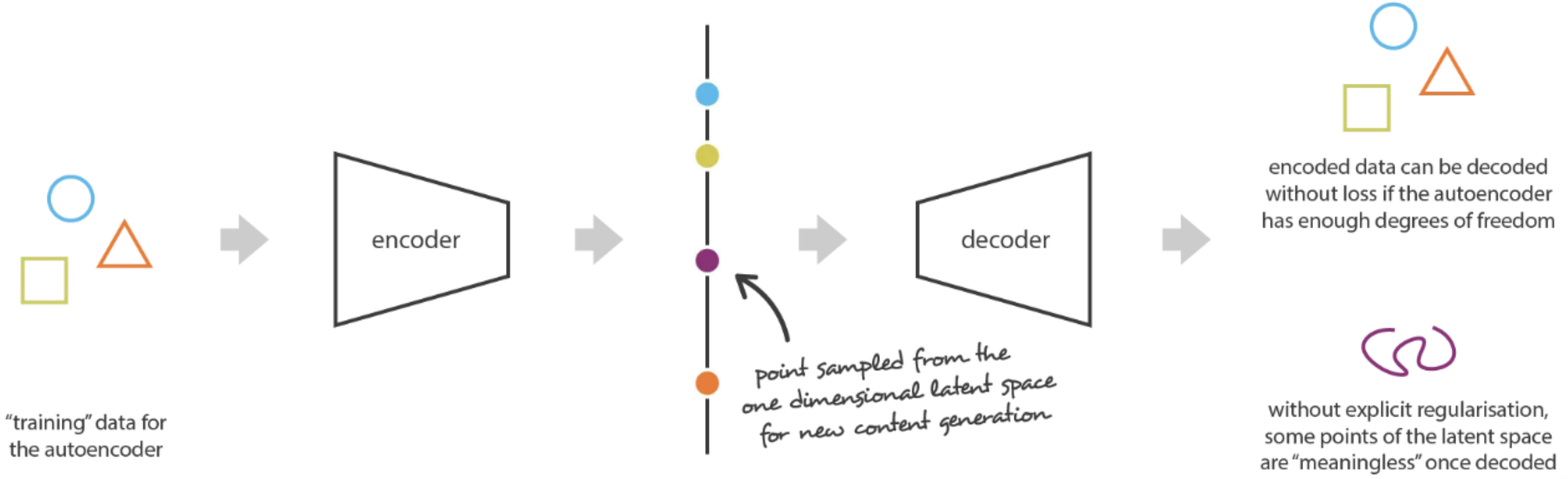
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73



- First encode the data into a low-dimensional latent space using an Encoder NN, then decode it using a Decoder NN such that the reconstruction loss is minimal .
- Setting the encoder NN to be linear $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$ and the decoder to be the inverse $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$, we get PCA.

Autoencoders

- **Questions:**
  1. How to define and optimize the latent variables **z?** ✓
  2. Given **z** how to generate meaningful data **x** on the data manifold.
- Autoencoders can learn a meaningful latent features, but the resulting latent space is not **organized** or **regularized** for generative purposes.



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders

- For the decoder to generate meaningful samples from randomly sampled points in the latent space we need a regularization effect.

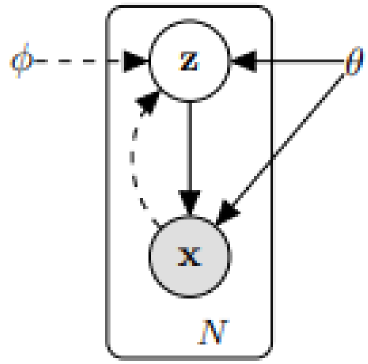
  Can we make an Autoencoder a generative model ?

  => Yes: Variation Autoencoder (VAEs)

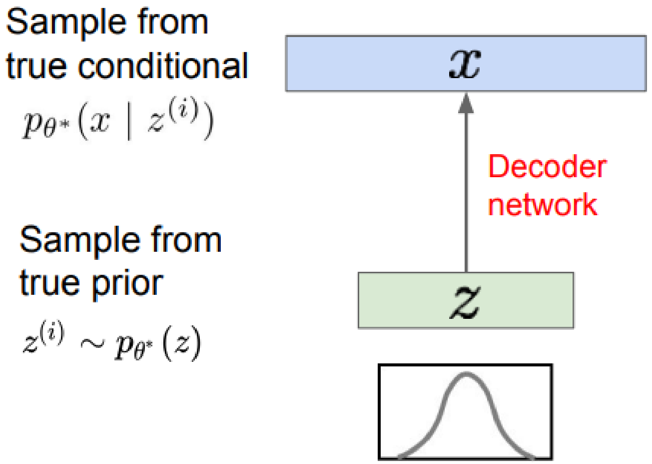
Variational Autoencoders [3]

- Define a probabilistic graphical model over latent variable $\boldsymbol{z}$.

$$p_\theta(x) = \int \underbrace{p_\theta(z)}_{\text{latent prior}} \underbrace{p_\theta(x|z)}_{\text{Decoder Likelihood}} dz$$



- $\boldsymbol{z}$ must be simple for maximal regularization, and provide tractable sampling, e.g. Gaussian.



Variational Autoencoders [3]

- How to train the model's parameters θ.
  - Answer: maximize the data likelihood => minimize the NLL.

$$\mathbf{argmin} - \sum_{j=1}^{N} \log p_{\theta}(\mathbf{x}^{j})$$

- Computing the integral is intractable:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Variational Inference

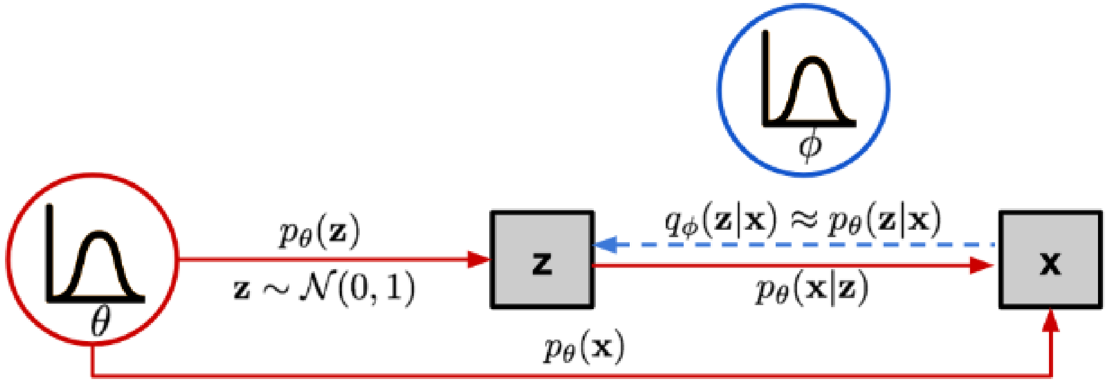
$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$



https://lilianweng.github.io/posts/2018-08-12-vae/

Derivation from: http://cs231n.stanford.edu/slides/2021/lecture_12.pdf

Variational Inference

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling (need some trick to differentiate through sampling).

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Derivation from: http://cs231n.stanford.edu/slides/2021/lecture_12.pdf

Variational Lower Bound

- We want to maximize the log-likelihood of the data and minimize the difference between the real and estimated posterior distributions

$$\log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

- We can derive the VAE loss, which the Variational Lower Bound (VLB) or the Evidence lower bound (ELBO)

$$\begin{aligned} L_{\mathrm{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\ &= \boxed{-\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))} \\ \theta^*, \phi^* &= \arg\min_{\theta,\phi} L_{\mathrm{VAE}} \end{aligned}$$

- The VLB is a lower bound for the true log-likelihood => **approximative explicit density model.**

$$-L_{\mathrm{VAE}} = \log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})$$

Derivation from: https://lilianweng.github.io/posts/2018-08-12-vae/

Reparametrization trick

$$
\begin{aligned}
L_{\mathrm{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= \boxed{-\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\
\theta^*, \phi^* &= \arg\min_{\theta, \phi} L_{\mathrm{VAE}}
\end{aligned}
$$

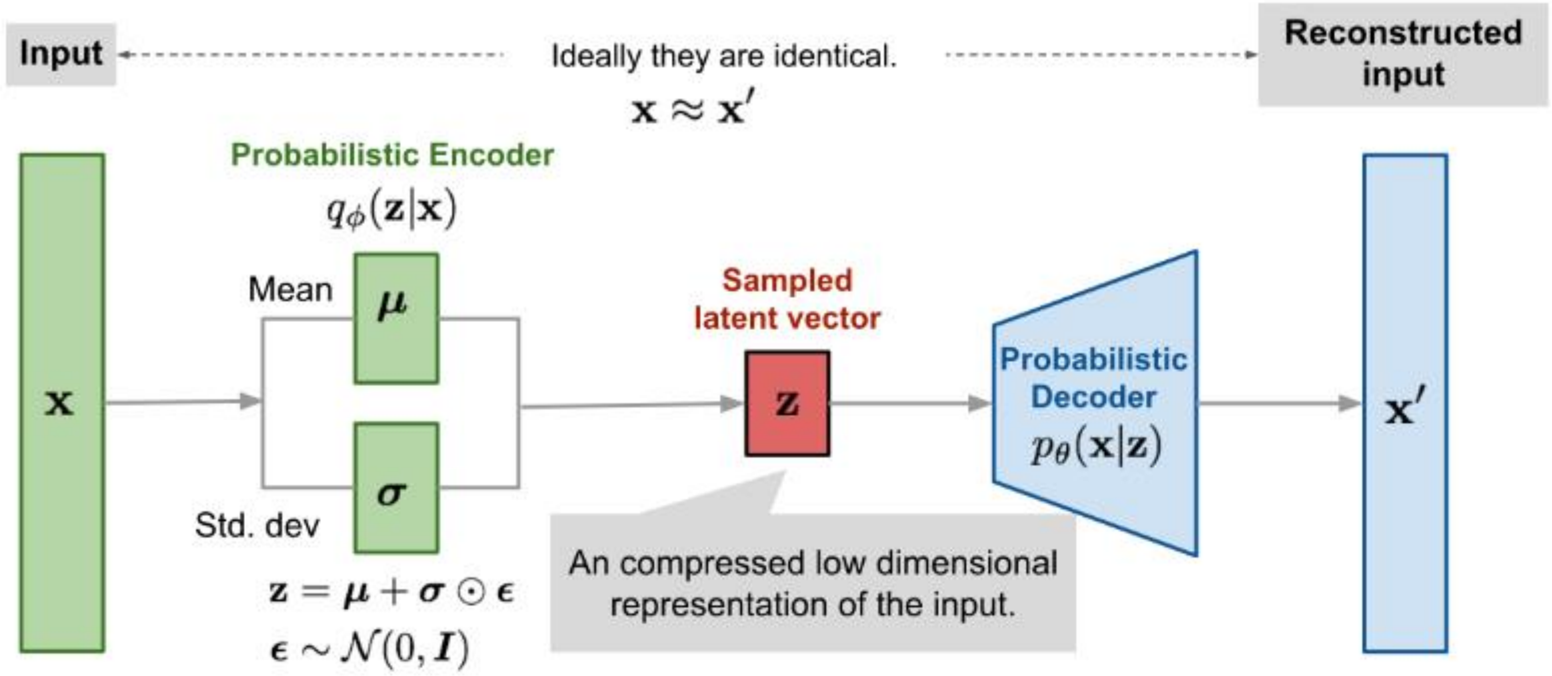
Sampling is not differentiable

- Define the posterior to be Gaussian similar to the prior and use the reparametrization trick

$$
\begin{aligned}
\mathbf{z} &\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)} \boldsymbol{I}) \\
\mathbf{z} &= \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}) \qquad \text{; Reparameterization trick.}
\end{aligned}
$$

All together

During sampling:



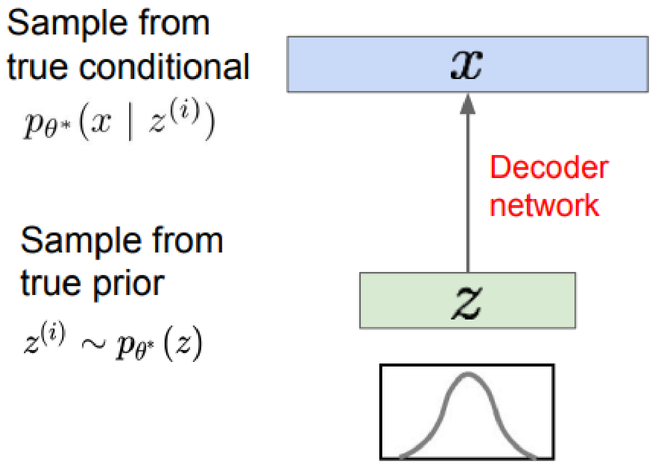
https://lilianweng.github.io/posts/2018-08-12-vae/

$$-\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

– The mean and the standard deviation are predicted by a Neural networks with two output heads.

All together

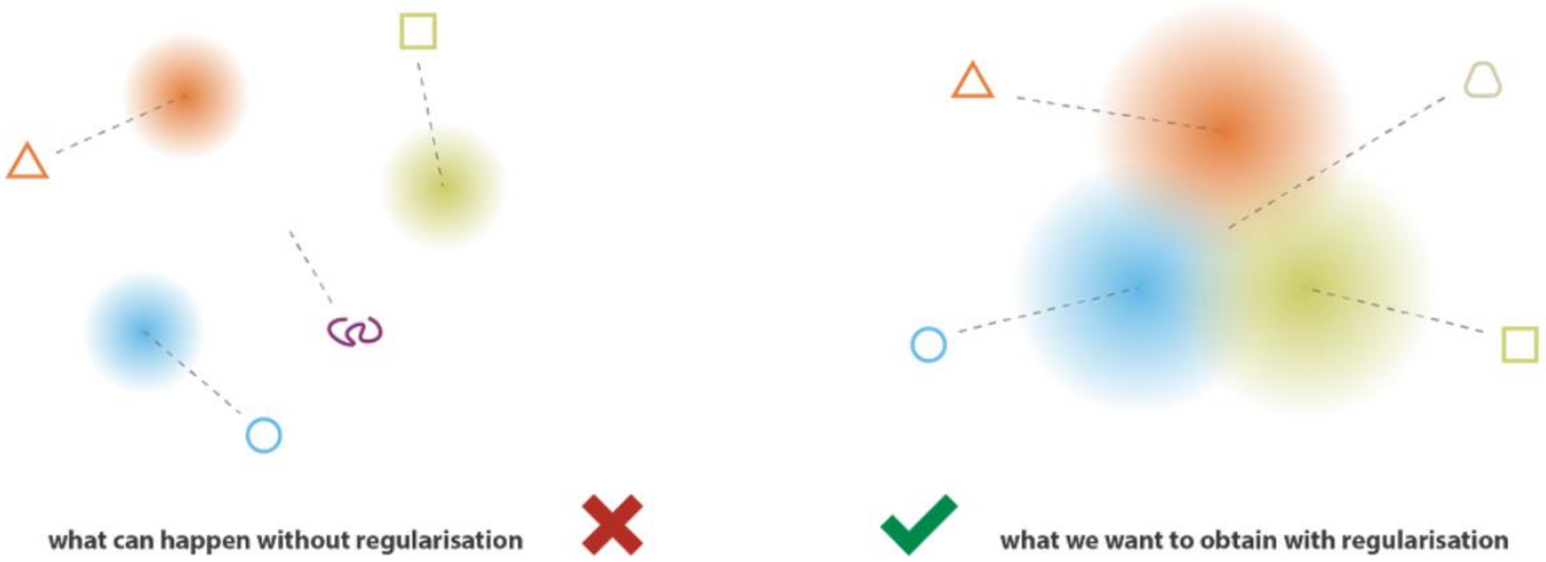
During sampling:



http://cs231n.stanford.edu/slides/2021/lecture_12.pdf

– First sample from the latent variable z using the reparametrization trick and then generate samples with the decoder.

Plain Autoencoders vs VAE



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

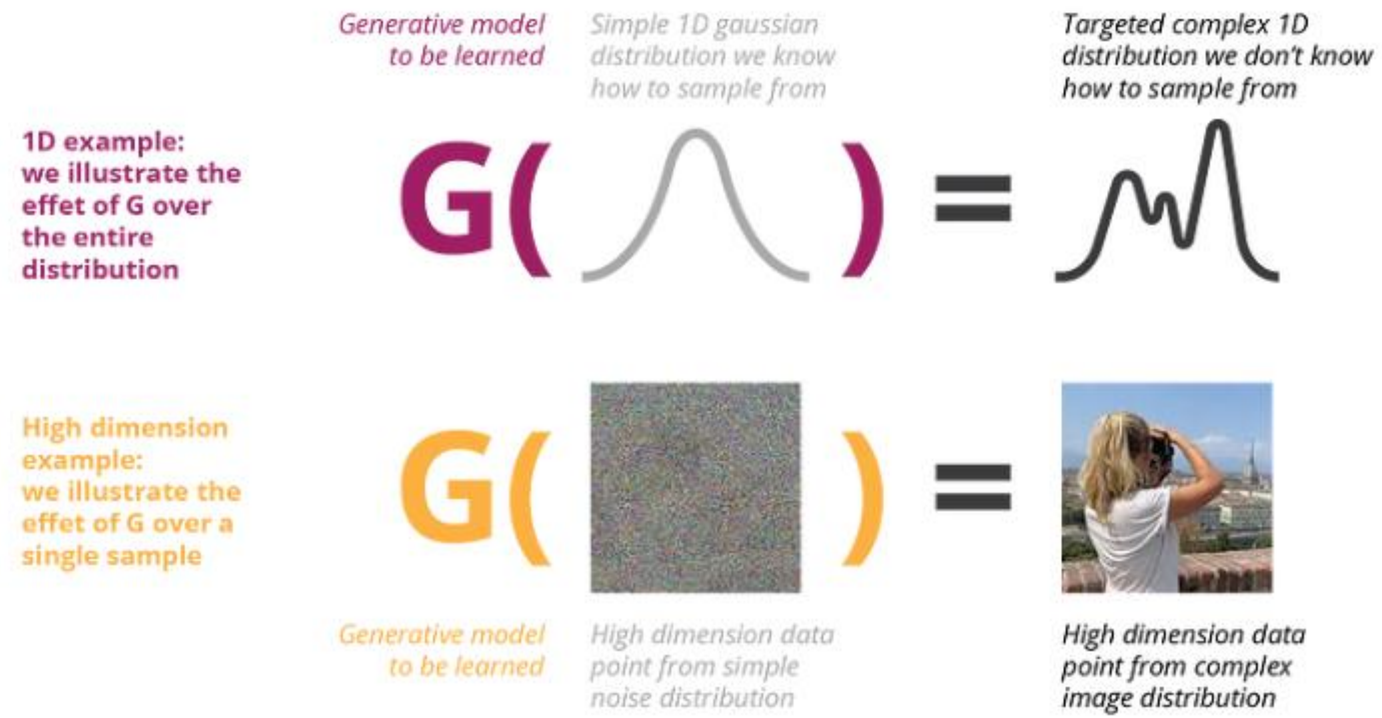
Diffusion Models

Diffusion Models

- **Previously:**
  - **VAE:** encode the input space into regularized latent space using a trainable encoder.
  - **Problem:** training the encoder against the decoder lead to unstable training or bad sample quality.
- **Diffusion Models:**
  - Use a fix forward process as encoder and learn to reverse it using a parametrized model.

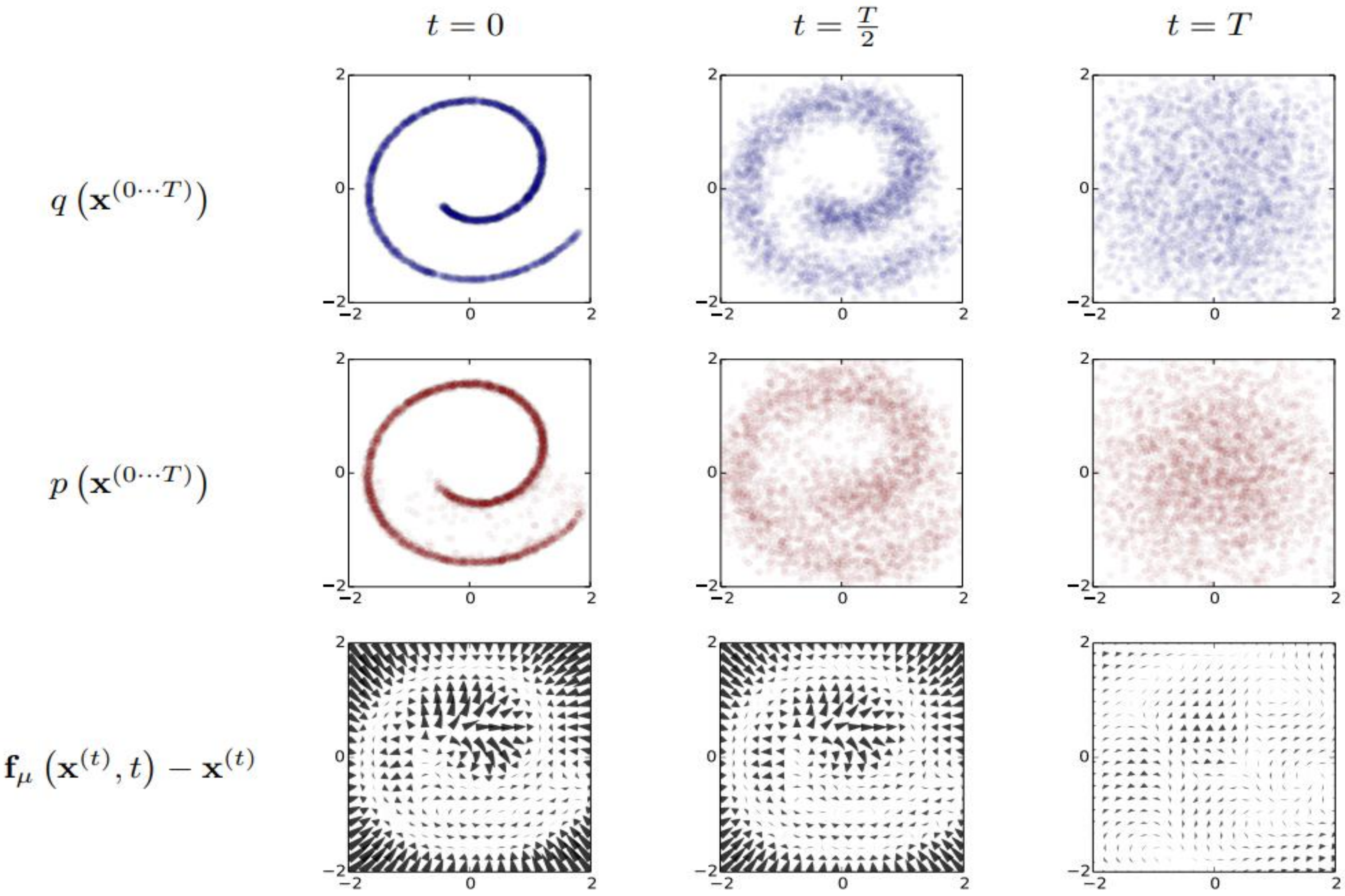
Diffusion Models

- Similar to VAE, they generate novel samples from a target distribution by sampling from a simple source distribution. Unlike VAE, the latent distribution has the same dimensions as the input.

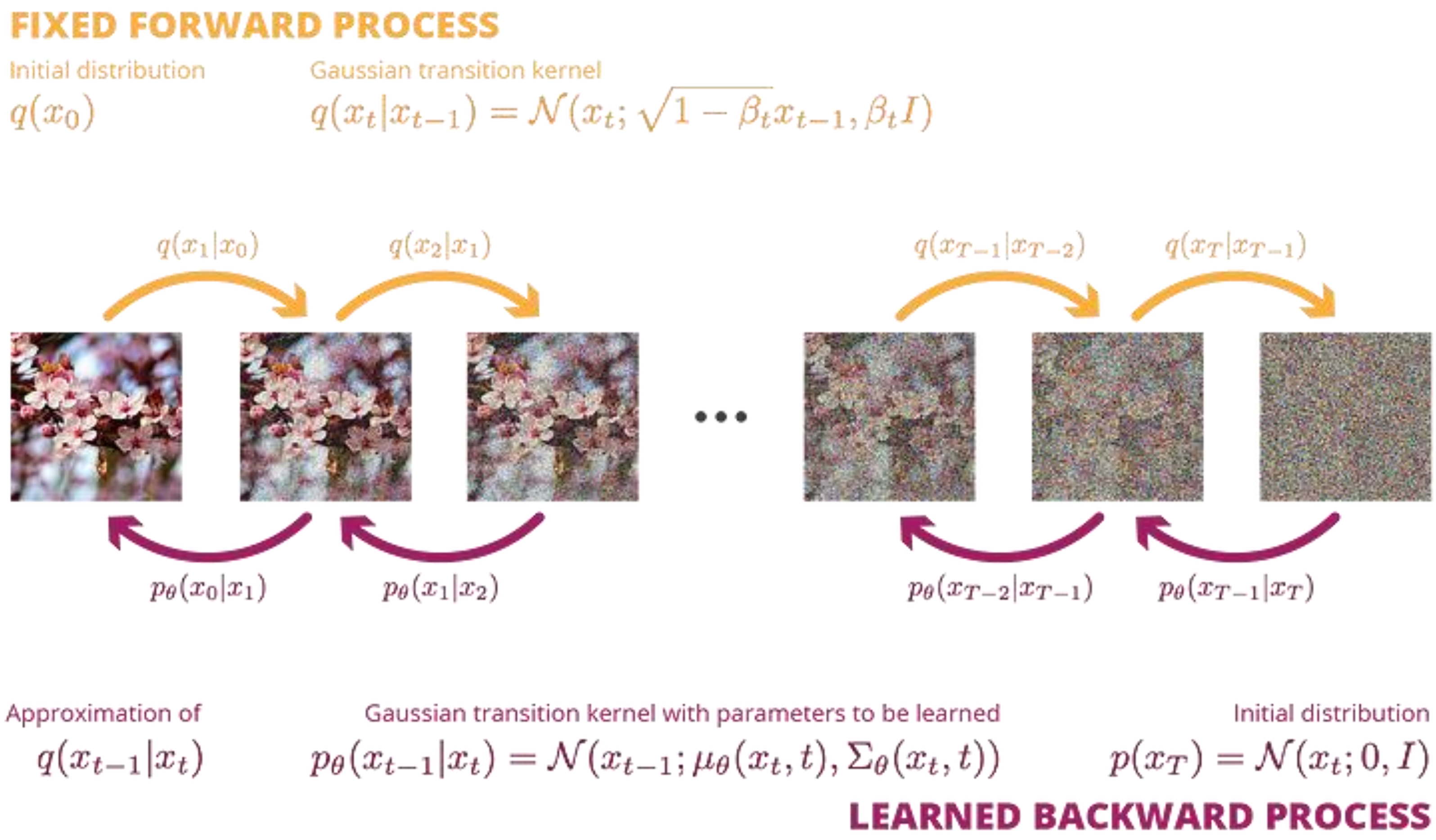


https://towardsdatascience.com/understanding-diffusion-probabilistic-models-dpms-1940329d6048

Diffusion Models: main idea [4]



Denoising Diffusion Probabilistic Models [5]



The Fixed Forward Trajectory

- Given a data point $x_0 \sim q(x)$, we iteratively add a small amount of Gaussian noise in T steps, producing a sequence $x_1, x_2, ..., x_T$ of noisy samples.
- The step size, i.e. diffusion rate, is controlled by The noise schedule $\{\beta_t\}_{t=0}^{T}$, $\beta_0 < \beta_1 < ... < \beta_T$
- This results in a diffusion process of the form:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

- Using the reparametrization trick, this allows for efficient sampling at random time step t:

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

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

where $\bar{\alpha} = \prod_{i=0}^{t} \alpha_i$, $\alpha_t = 1 - \beta_t$ and $\epsilon \sim \mathcal{N}(0, I)$

The learned Backward Trajectory

- During sampling we are interested in the reverse path, i.e. $q(x_{t-1}|x_t)$.
- But reversing the process is very difficult and intractable :(.
- Trick: if β_t is small enough, i.e. T is large (around 1000), then the forward and reverse trajectory has an identical functional form.
- We can approximate the reverse process $q(x_{t-1}|x_t)$ by learning a model $p_\theta(x)$:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

  with μ_θ and Σ_θ are the learned model (e.g. neural networks).
- To sample new data point, first we sample randomly from $p_\theta(x_T) \sim \mathcal{N}(0, I)$ and then use $p_\theta(x_{t-1}|x_t)$ to generate $x_0 \sim p_\theta(x_0) \approx q(x_0)$.

The learned Backward Trajectory

- During sampling we are interested in the reverse path, i.e. $q(x_{t-1}|x_t)$.
- But reversing the process is very difficult and intractable :(.
- Trick: if β_t is small enough, i.e. T is large (around 1000), then the forward and reverse trajectory has an identical functional form.
- We can approximate the reverse process $q(x_{t-1}|x_t)$ by learning a model $p_\theta(x)$:

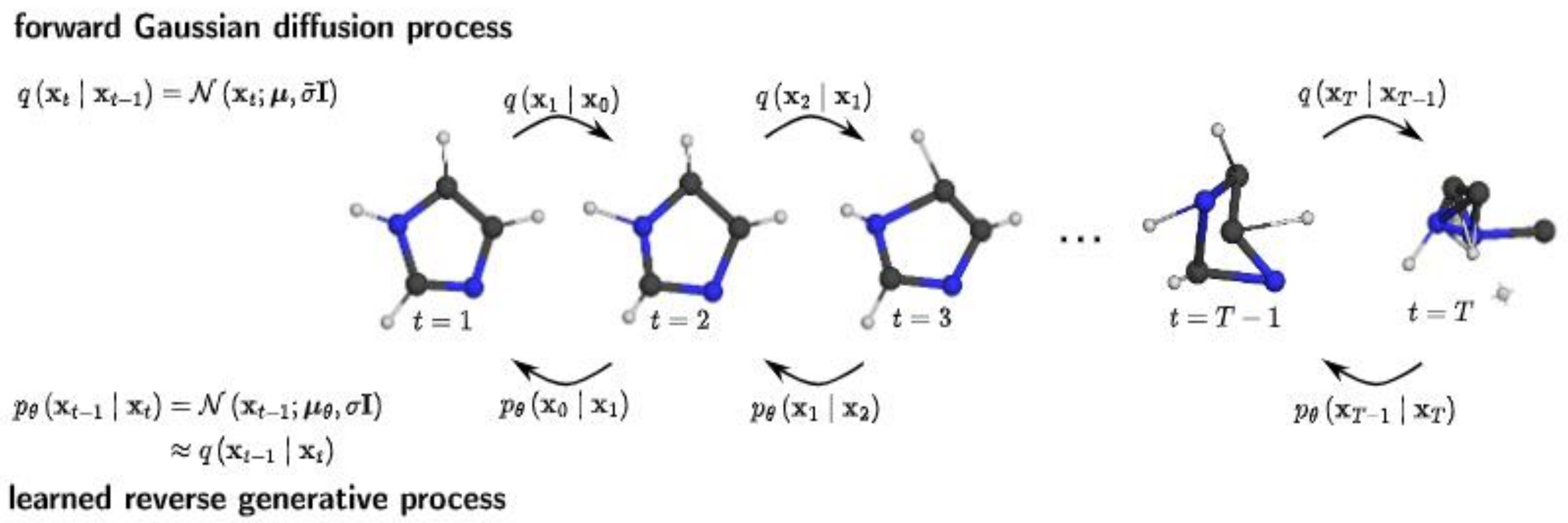
$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

  with μ_θ and Σ_θ are the learned model (e.g. neural networks).
- To sample new data point, first we sample randomly from $p_\theta(x_T) \sim \mathcal{N}(0, I)$ and then use $p_\theta(x_{t-1}|x_t)$ to generate $x_0 \sim p_\theta(x_0) \approx q(x_0)$.

Applications: High quality text to image

– All state-of-the-art image generation models like DALL-E and Imagen are using diffusion models.

Applications: Drug discovery



Conclusion

Conclusion

- Generative models are used for:
  - Density estimation
  - Sample generation
- We can differentiate between explicit and implicit models.
- Autoregressive models offer tractable log-likelihood but have slow sampling process.
- VAEs can sample all pixels at once using variational inference over latent variables.
- Diffusion models are multistep VAEs but with fixed encoding process that functions in the input space.

References

[1] Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., & Graves, A. (2016). Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, *29*.

[2] Van Den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016, June). Pixel recurrent neural networks. In *International conference on machine learning* (pp. 1747-1756). PMLR.

[3] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[4] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015, June). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning* (pp. 2256-2265). PMLR.

[5] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, 6840-6851.