Generative AI

Course Description

- **Foundational Generative AI**: In-depth exploration of generative models, including GANs and diffusion models.
- **Transformer Architectures and LLMs**: Study of transformer and large language model architectures, training methods, and real-world applications.
- **Hands-on Fine-tuning**: Practical experience with fine-tuning techniques like LoRA, QLoRA, and RLHF.
- **Optimization and Evaluation**: Focus on optimization methods and evaluation metrics specific to generative models.
- **Prompt Engineering and RAG**: Exploration of prompt engineering strategies and retrieval-augmented generation (RAG).
- **AI Agents and LangChain**: Introduction to AI agents and frameworks like LangChain.

CLOs

- **Explain foundational principles of Generative AI models**, including GANs, diffusion models, and transformers, and their applications in generating diverse data types.
- **Analyze the architectures and pre-training methodologies of large language models (LLMs)** and apply fine-tuning techniques, including parameter-efficient methods like LoRA and QLoRA, to adapt them for specific domains or tasks.
- **Evaluate optimization methods and metrics** for assessing generative models' quality and performance.
- **Apply Retrieval-Augmented Generation (RAG) techniques and fine-tuning methods** to adapt generative models for specific domains and tasks.
- **Develop prompt engineering strategies** to optimize generative AI outputs for diverse applications.

Books & Learning Material

1. Generative Deep Learning, second edition, Teaching Machines to Paint, Write, Compose, and Play- David Foster Foreword by Karl Friston
2. Generative AI in Practice by Bernard Marr
3. RAG-Driven Generative AI: Build Custom Retrieval Augmented Generation Pipelines with LlamaIndex, Deep Lake, and Pinecone Book by Denis Rothman

Additional references and research papers related to the course.

Grading Policy

20% Midterm exam

35% Final Exam

15% Presentations

~10% Quizzes

Announced & Unannounced quizzes

20% Programming Assignments + HomeWorks

4 to 5 assignments

*** The above grading distribution may change as per discretion of the instructor**

What is Generative AI

- Models that **create new data** (text, images, audio) from learned patterns.
- Example: DALL-E generates art; GPT creates text.

•**Applications**:

- Text Generation: ChatGPT.
- Image Creation: Stable Diffusion.
- Music Composition: OpenAI’s Jukebox.

Applications of Generative AI

Text Generation:

Content creation

Language translation

Summarization

Image Generation:

Art and design

Medical imaging

Video game development

Music Generation:

Composition

Sound effects

Drug Discovery:

Designing new molecules

Image generation

Produces alias-free, photorealistic images suitable for high-end applications.

The network is capable of producing highly detailed and realistic human faces that do not belong to real individuals.

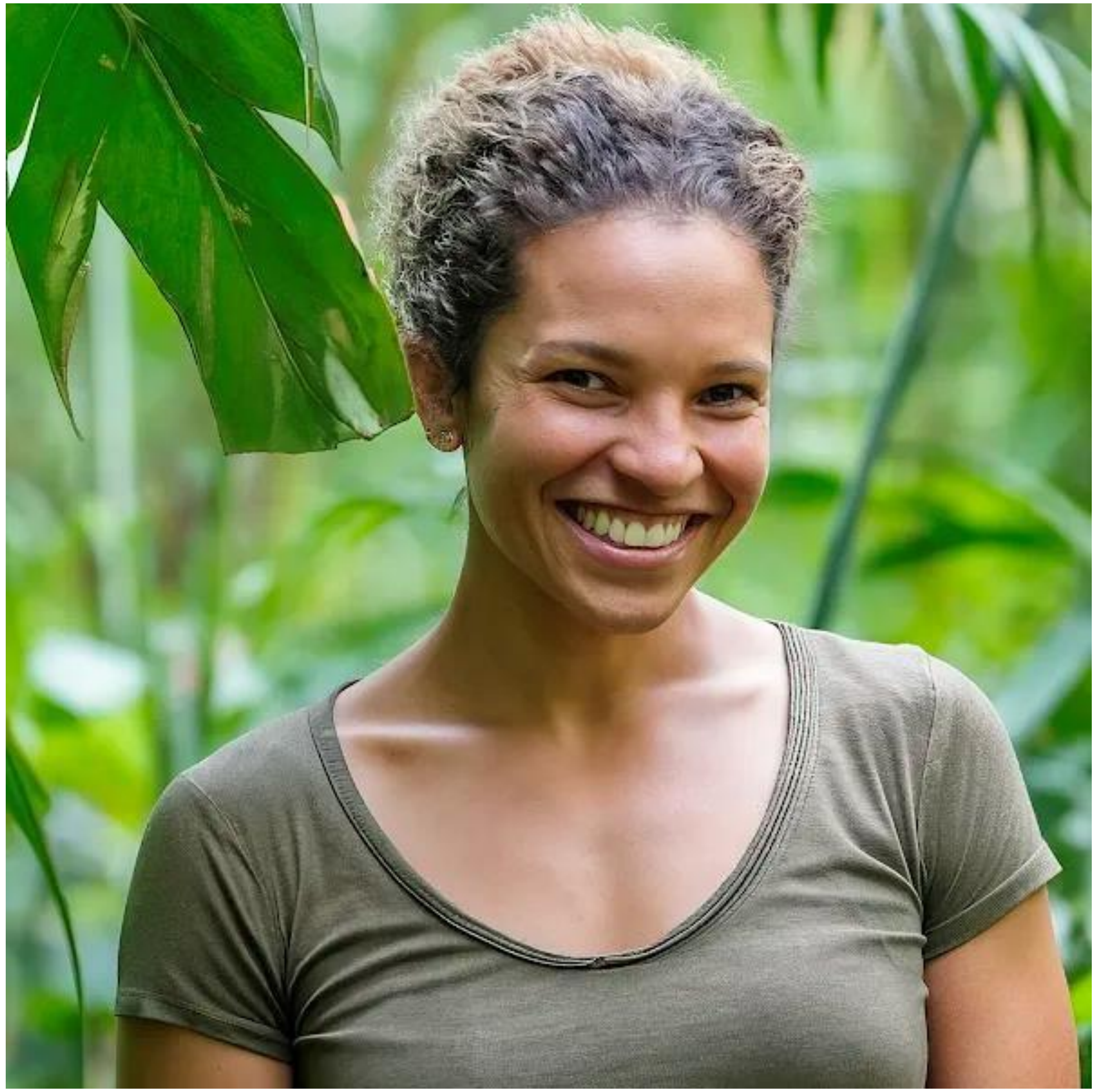
#StyleGAN3 (NVIDIA) Karras, Tero, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Alias-free generative adversarial networks." NeurIPS 2021. [code]

Image generation

#DiT (UC Berkeley, NYU) Peebles, William, and Saining Xie. "Scalable Diffusion Models with Transformers." ICCV 2023.

Text-to-Image (T2I) generation

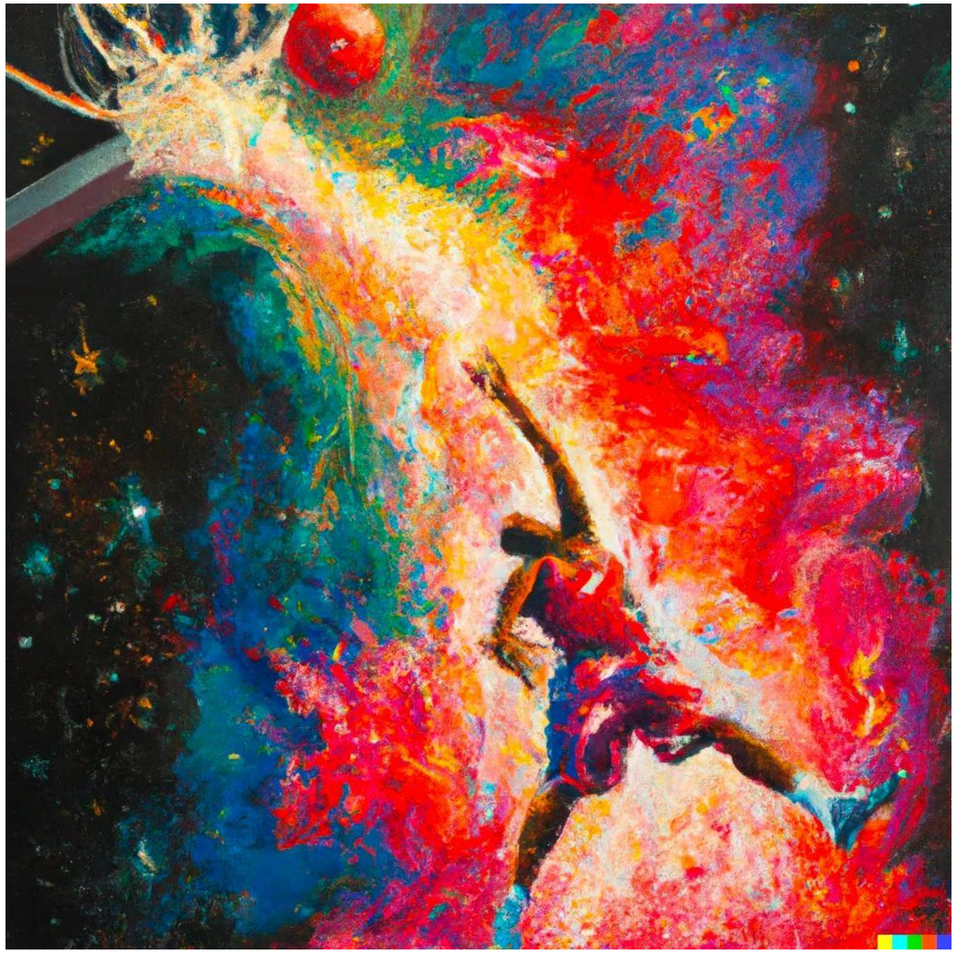
“A shot of a 32-year old female, up and coming conservationist in a jungle; athletic with short, curly hair and a warm smile

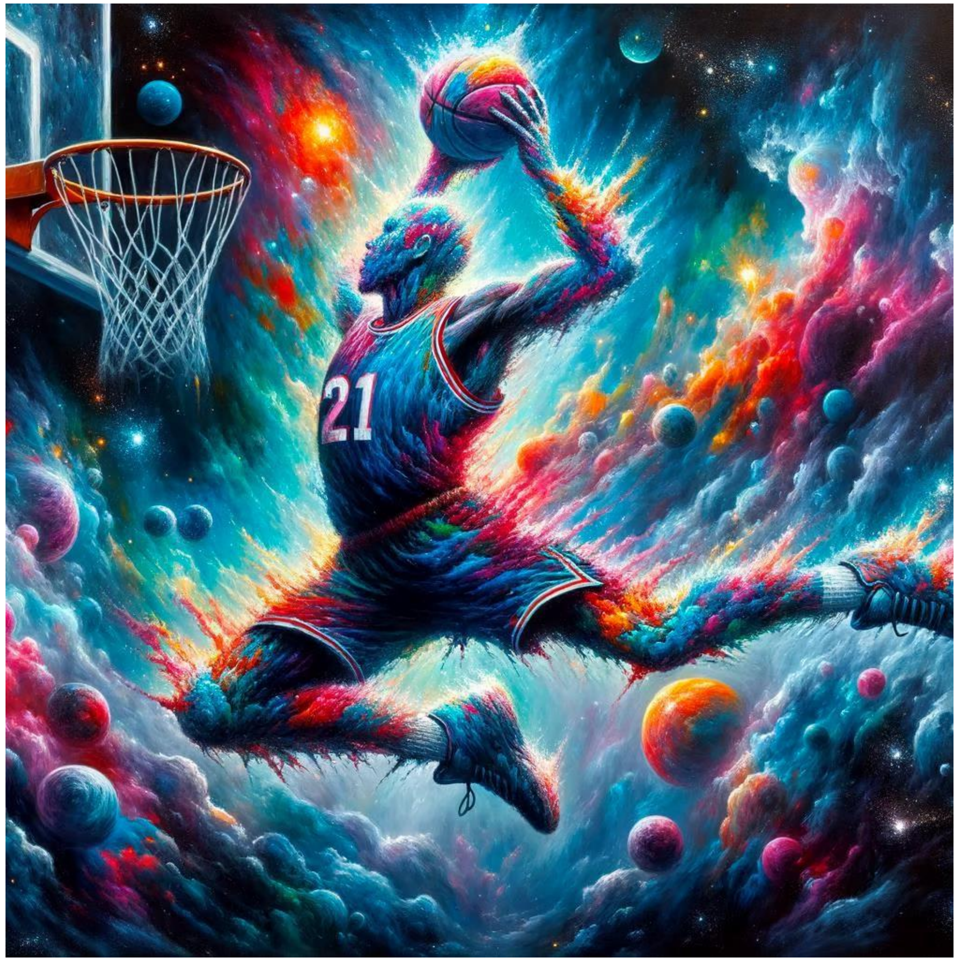


#Imagen2 (Google Deepmind) Blog post

Text-to-Image (T2I) generation

““An expressive oil painting of a basketball player dunking, depicted as an explosion of a nebula.

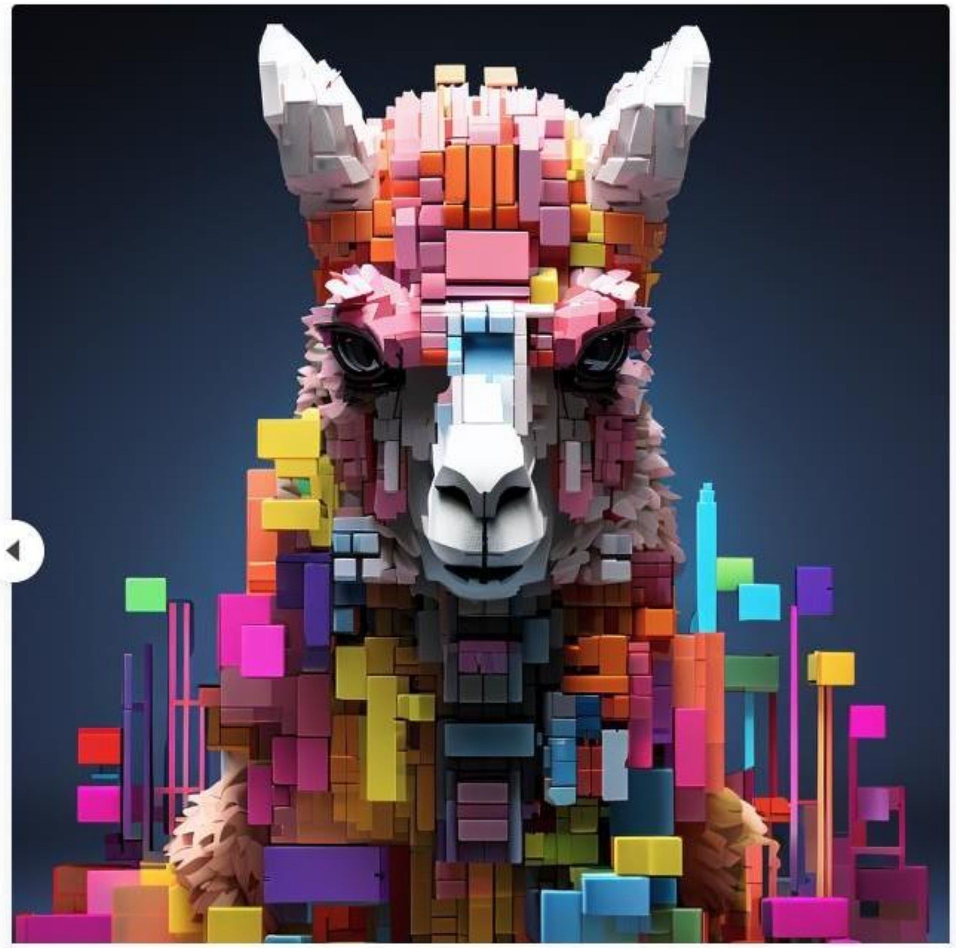


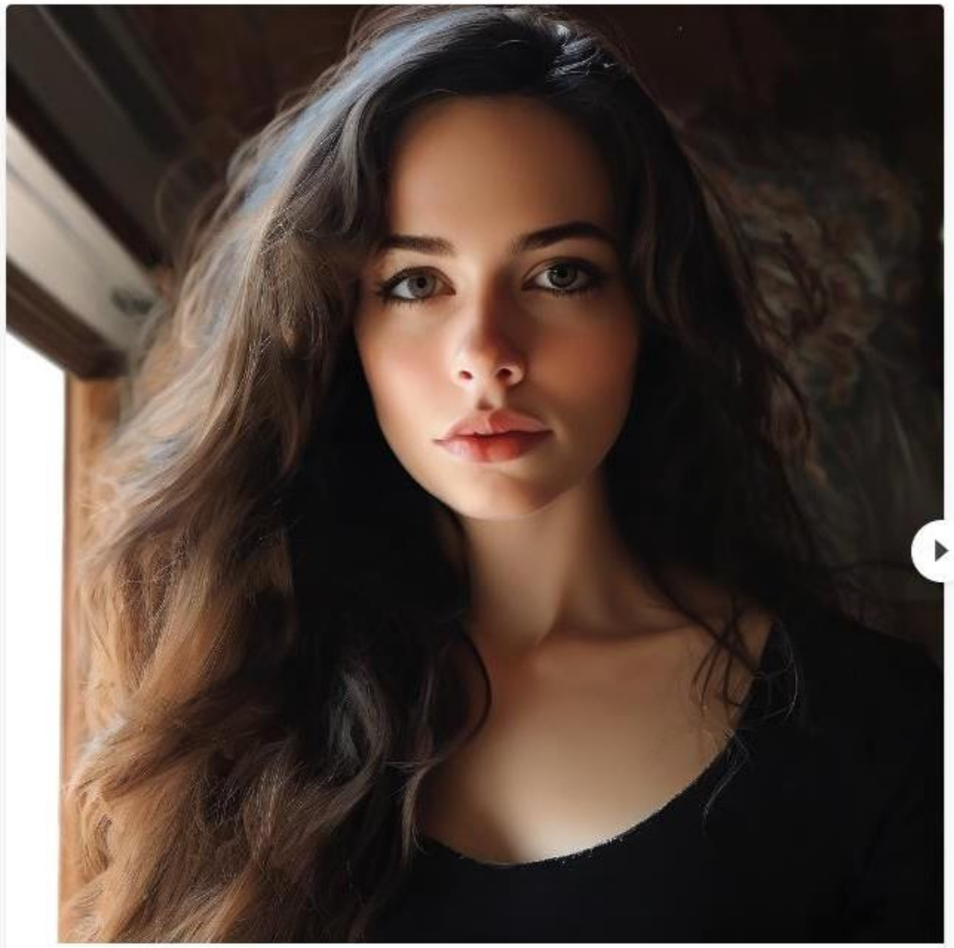


#**DALL-E-2** (OpenAI) Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen "Hierarchical Text-Conditional Image Generation with CLIP Latents." 2022. [blog]

#**DALL·E-3** (OpenAI) James Betker, Gabriel Goh, et al, “Improving Image Generation with Better Captions” 2023 [blog]

Text-to-Image (T2I) generation


A alpaca made of colorful building blocks, cyberpunk.


Real beautiful woman.

#PixArt-alpha Chen, Junsong, Y. U. Jincheng, G. E. Chongjian, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. "PixArt-alpha: Fast Training of Diffusion Transformer for Photorealistic Text-to-Image Synthesis." ICLR 2024

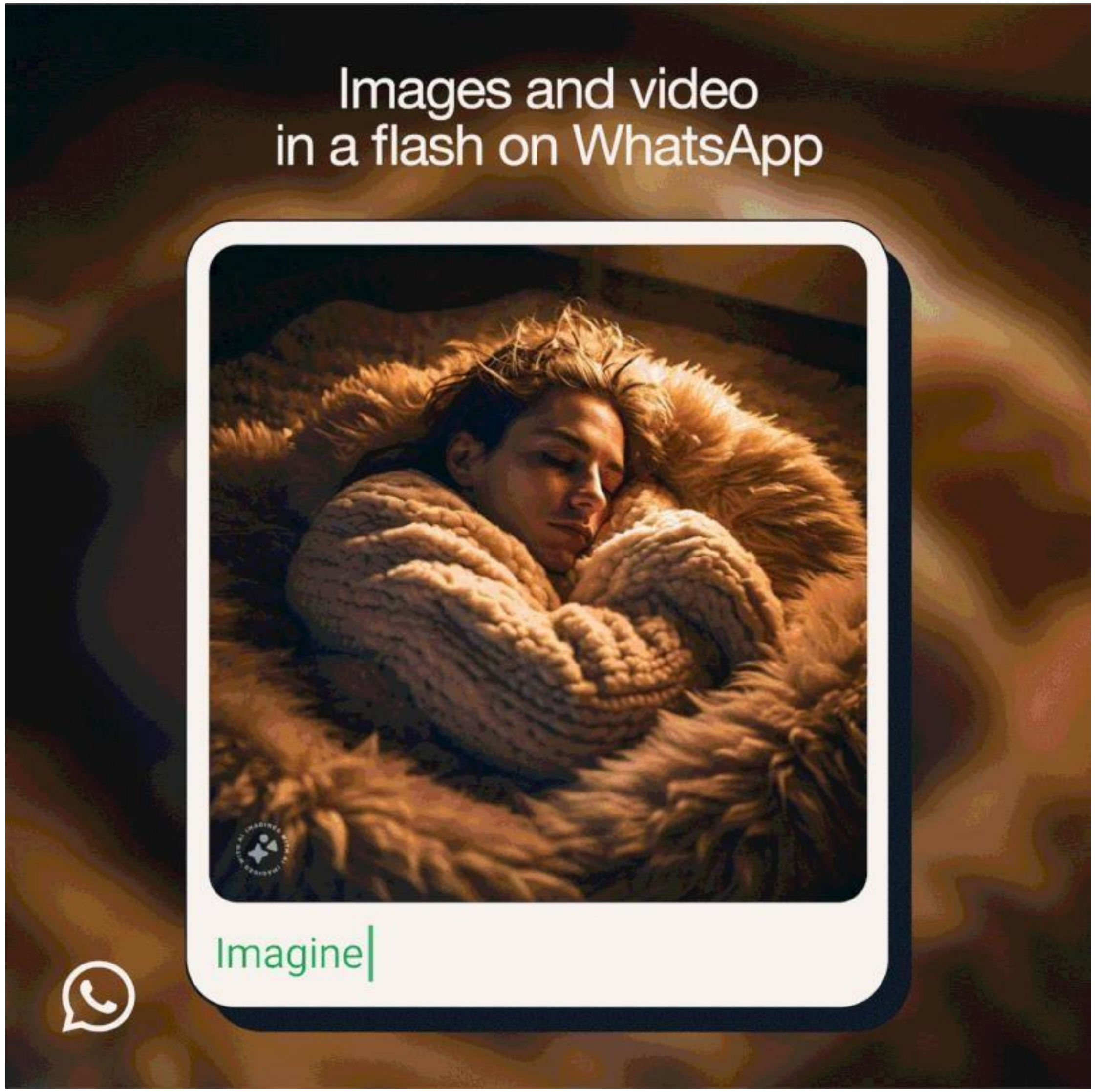
Text-to-Music (T2M) generation

(Stability AI) Evans, Zach, Julian D. Parker, C. J. Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. "Long-form music generation with latent diffusion." arXiv 2024.

Text-to-Video (T2V) generation

#Sora (OpenAI) Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Wing Yin Ng, Ricky Wang, Aditya Ramesh. Video generation models as world simulators. OpenAI 2024.

Text-to-Image (T2I) generation



Albert Pumarola
@AlbertPumarola

PhD 2021
Institut de Robòtica Industrial (IRI)
Universitat Politècnica de Catalunya (UPC)

#Imagine Flash (Meta) Jonas Kohler, Albert Pumarola, Edgar Schönfeld, Artsiom Sanakoyeu, Roshan Sumbaly, Peter Vajda, and Ali Thabet. Imagine Flash: Accelerating Emu Diffusion Models with Backward Distillation. Meta 2024.

Historical Context and Evolution

- **2014:** Ian Goodfellow introduced GANs, allowing machines to generate realistic images by pitting two networks against each other.
- **2017:** Transformers emerged with the Attention is All You Need paper, laying the foundation for models like BERT and GPT.
- **2020:** OpenAI’s GPT-3 stunned the world with its ability to generate coherent, context-aware text.
- **2022:** Diffusion models gained attention with tools like DALL-E and Stable Diffusion, redefining creativity.

Challenges in GenAI

- Ethical Concerns: Bias, deepfakes
- Hallucinations in AI-generated content
- High computational costs (resource intensive)
- Data privacy concerns

Discriminative Vs Generative Model

Machine learning models can be broadly categorized into discriminative and generative models.

Each type of model has different goals, strengths, and applications.

Discriminative Models

- Discriminative models focus on distinguishing between different classes in the data.
- They model the decision boundary between classes.
- Given input data x, predict the label y.
- Learns the $P(y|x)$
- **Examples**:
  - Logistic Regression
  - Neural Networks
  - Convolutional Neural Networks
- **Advantages**:
  - Often achieve higher accuracy on classification tasks.
  - Directly optimize the decision boundary.

Generative Models

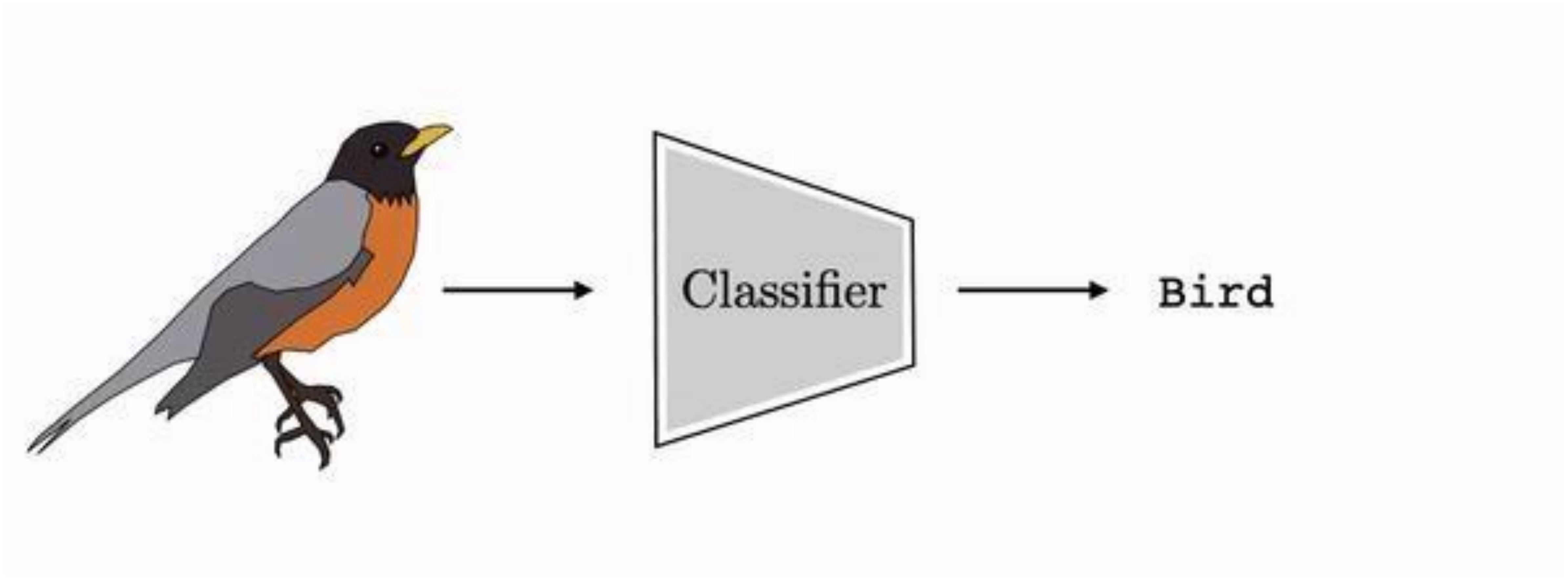
- Generative models focus on modeling the distribution of the data.
- They learn to generate new data points that resemble the training data.
- Model the probability distribution P(x).
- If dataset is labelled, estimates/learn the P(x|y) - data conditioned on specific labels
- **Examples**:
    - Gaussian Mixture Models (GMMs)
    - Hidden Markov Models (HMMs)
    - Variational Autoencoders (VAEs)
    - Generative Adversarial Networks (GANs)
- **Advantages**:
    - Can generate new data samples.
    - Useful for unsupervised learning tasks.

Key Differences

Discriminative	Generative
Learn the Decision Boundary	Learn the distribution of the data.
Model $P(y\|x)$.	Model $P(x)$.
Requires Labelled Data	Can work with unlabeled data.

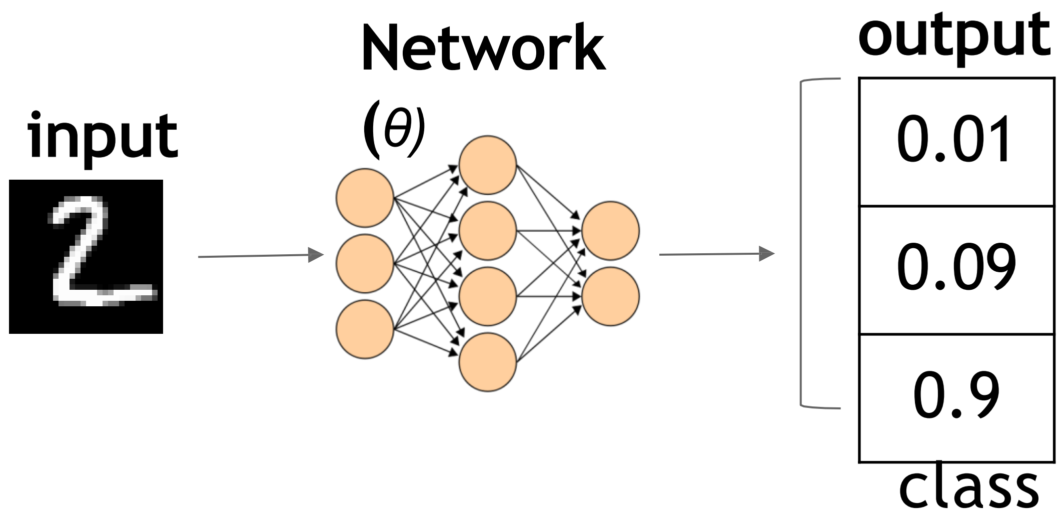
Discriminative vs Generative Models

Philip Isola, Generative Models of Images. MIT 2023.



$P_\theta(Y|X)$: Discriminative Models

Discriminative model: Tell me the probability of some 'Y' responses given 'X' input.

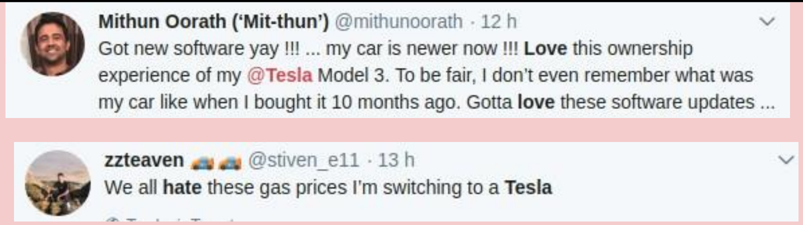
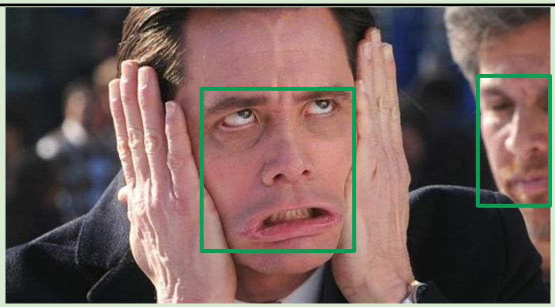
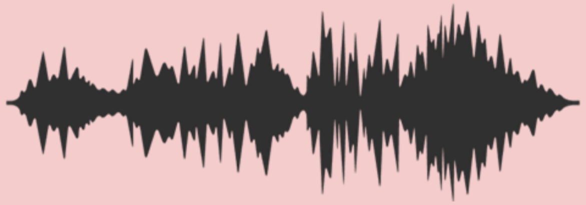


$$P_\theta(Y\,|X = [\text{pixel}_1, \text{pixel}_2, \ldots, \text{pixel}_{784}])$$

Figure credit: Javier Ruiz (UPC TelecomBCN)

$P_\theta(Y|X)$: Discriminative Models

Slide credit:
Albert Pumarola (UPC 2019)

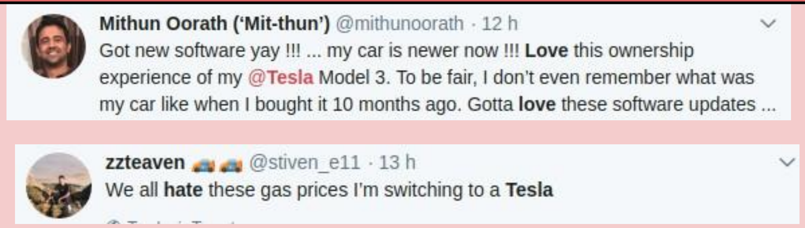
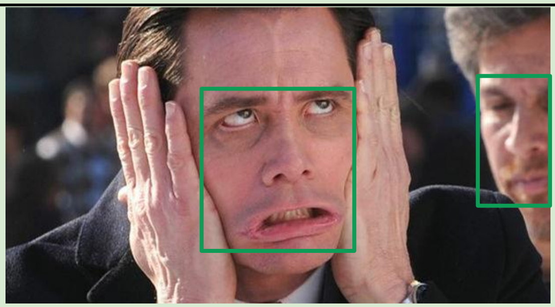
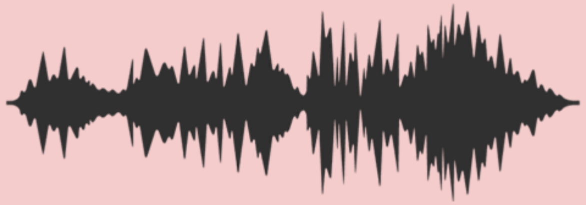
	Classification	Regression
Text	 **Mithun Oorath ('Mit-thun')** @mithunoorath · 12 h Got new software yay !!! ... my car is newer now !!! **Love** this ownership experience of my @Tesla Model 3. To be fair, I don't even remember what was my car like when I bought it 10 months ago. Gotta **love** these software updates ... **zzteaven** @stiven_e11 · 13 h We all **hate** these gas prices I'm switching to a **Tesla**	Prob. of being a Potential Customer
Image	Jim Carrey	
Audio	 What Language?	Speech Translation

Discriminative Modeling
$P_\theta(Y|X)$

X=Data
Y=Labels
θ = Model parameters

$P_\theta(X)$: Generative Models

Slide Concept: Albert Pumarola (UPC 2019)

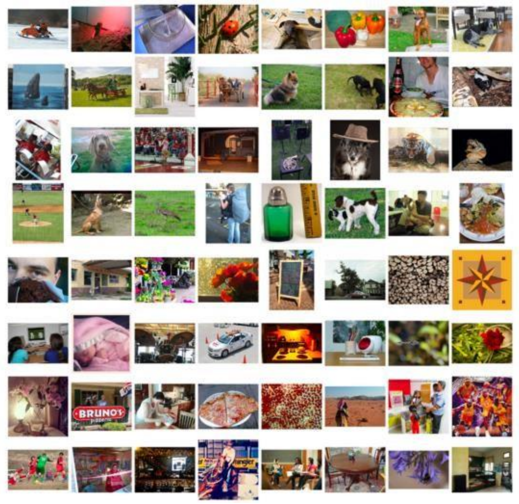
	Classification	Regression	Generative
Text	 **Mithun Oorath ('Mit-thun')** @mithunoorath · 12 h Got new software yay !!! ... my car is newer now !!! **Love** this ownership experience of my @Tesla Model 3. To be fair, I don't even remember what was my car like when I bought it 10 months ago. Gotta **love** these software updates ... **zzteaven** @stiven_e11 · 13 h We all **hate** these gas prices I'm switching to a **Tesla**	Prob. of being a Potential Customer	"What about Ron magic?" offered Ron. To Harry, Ron was loud, slow and soft bird. Harry did not like to think about birds.
Image	Jim Carrey		
Audio	 What Language?	Language Translation	Music Composer and Interpreter MuseNet Sample

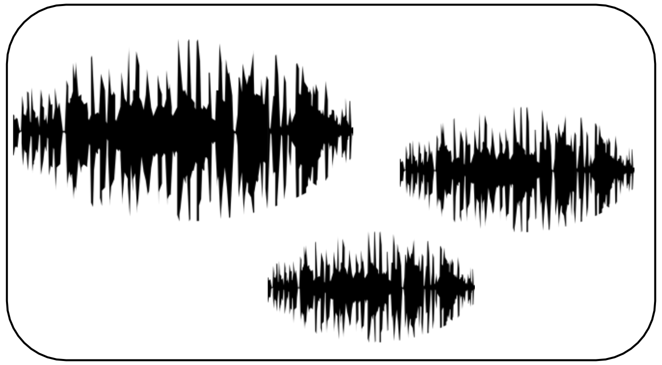
Discriminative Modeling
$P_\theta(Y|X)$

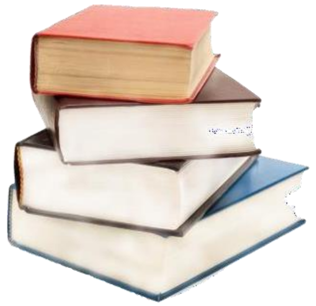
Generative Modeling
$P_\theta(X)$

X=Data
Y=Labels
θ = Model parameters

$P_\theta(X)$: Generative Models





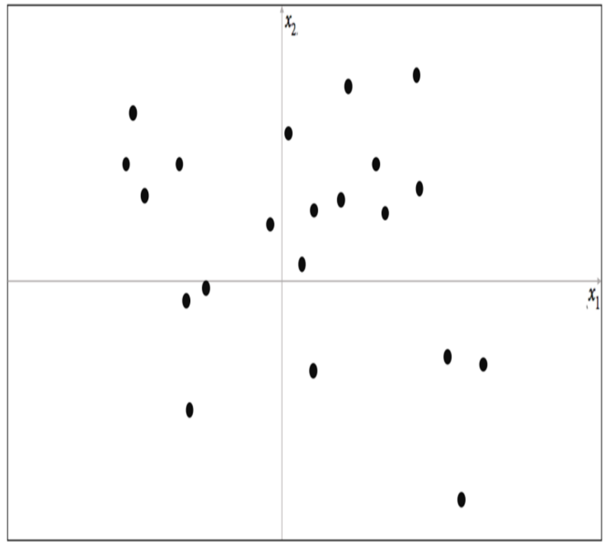


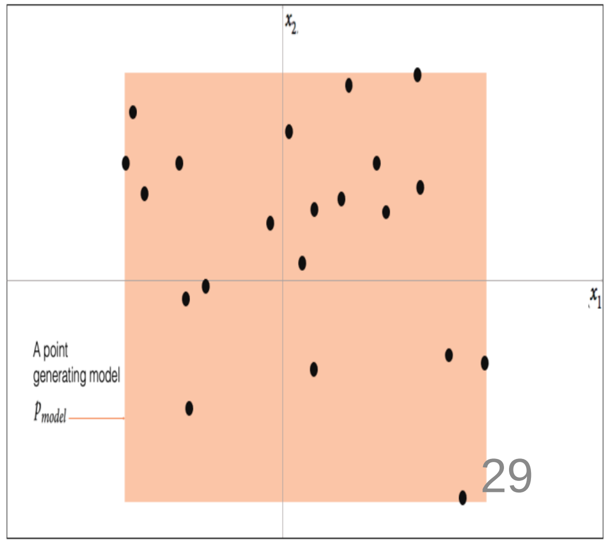
Each real sample x_i comes from an M-dimensional probability distribution *P(X)*.

$$X = \{x_1, x_2, \ldots, x_N\}$$

Generative Modeling Framework

- A rule has generated some set of points
- Let's call the rule P_{data}
- Task is to select a point that has been generated by the same rule
- Try to estimate the rule
- Let's call the estimation P_{model}



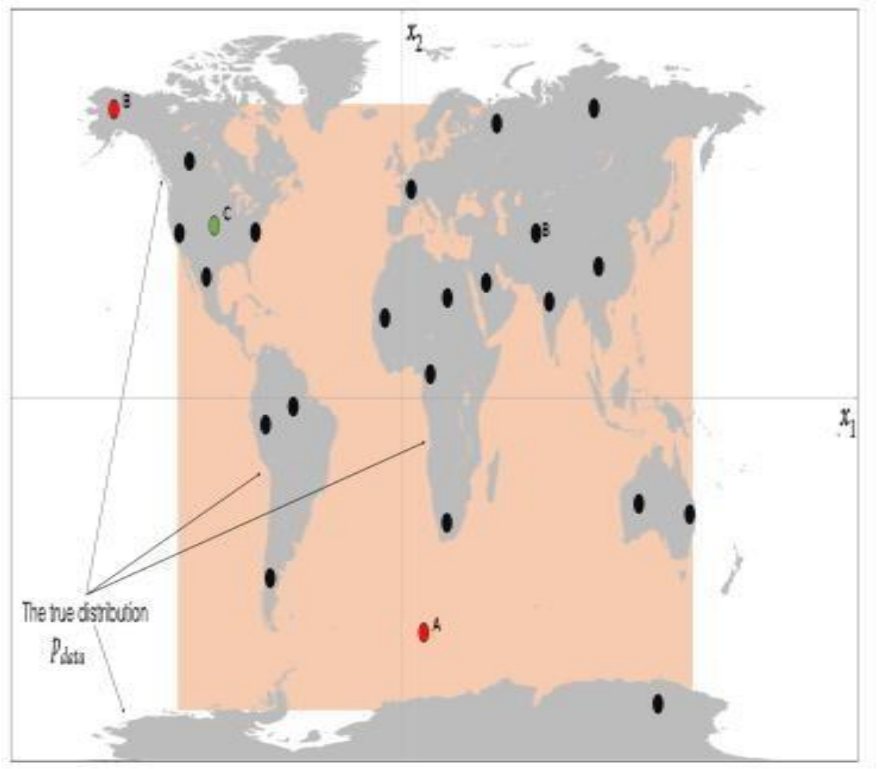


Generative Modelling Framework

- We have a dataset of observations **X**.
- We assume that the observations have been generated according to some unknown distribution, p_{data} .
- We want to build a generative model p_{model} that mimics p_{data} . If we achieve this goal, we can sample from p_{model} to generate observations that appear to have been drawn from p_{data} .
- The desirable properties of *p* are:
  - **Accuracy**: If $p_{model}(x)$ is high, x should look like it has been drawn from p_{data} . If $p_{model}(x)$ is low, x should *not* look like it has been drawn from p_{data} . The model should accurately capture the patterns and characteristics of the real data.
  - **Generation**: It should be possible to easily sample a new observation *x* from $p_{model}(x)$. If sampling is difficult or computationally expensive, the model is less practical.
  - **Representation**: It should be possible to understand how different high-level features (e.g. in our example, continents) are represented by p_{model}.

P_{data} Vs P_{model}

- **Point A** is an observation that is generated by our model but does not appear to have been generated by p_{data} as it's in the middle of the sea.
- **Point B** could never have been generated by p_{model} as it sits outside the orange box.
    - model has some gaps in its ability to produce observations across the entire range of potential possibilities.
- **Point C** is an observation that could be generated by p_{model} and also by p_{data}



Sampling

Our learned model should be able to make up new samples from the distribution, not just copy and paste existing samples!

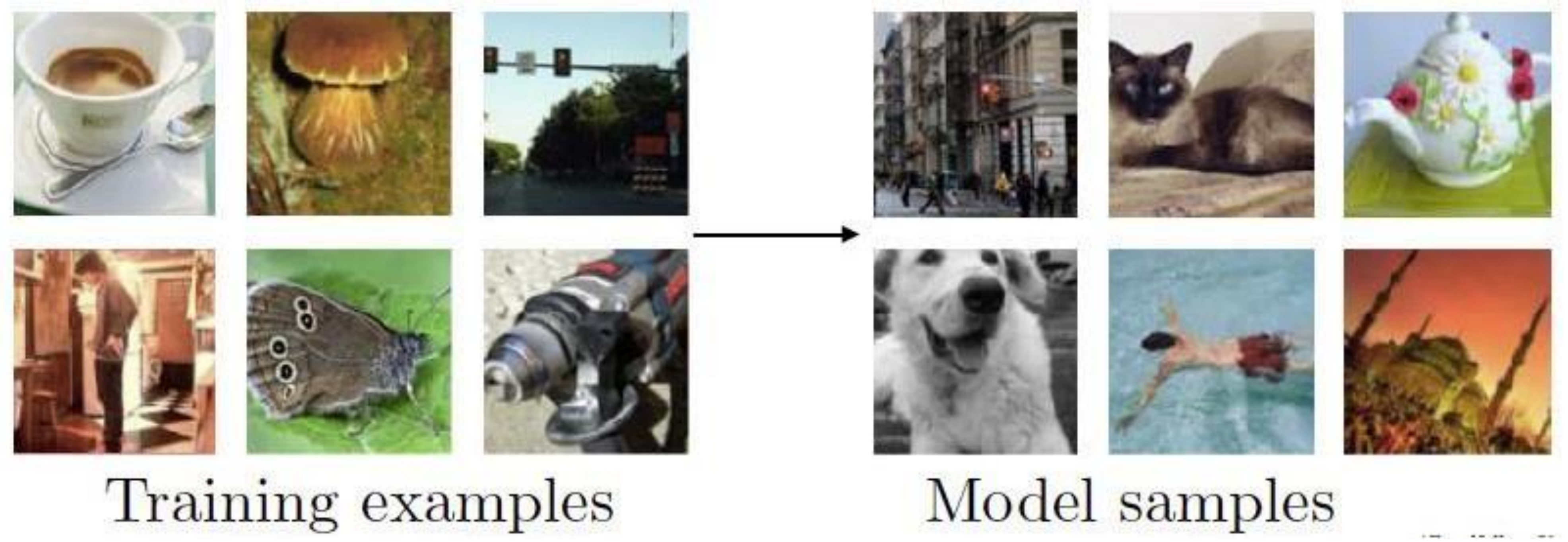
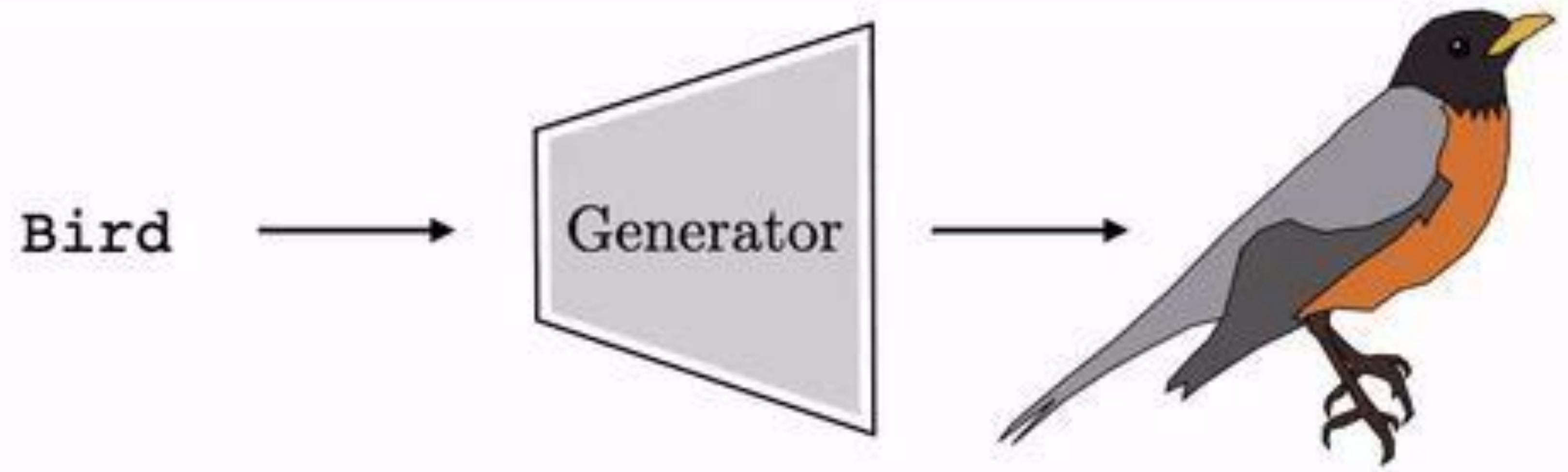


Figure from NIPS 2016 Tutorial: Generative Adversarial Networks (I. Goodfellow)

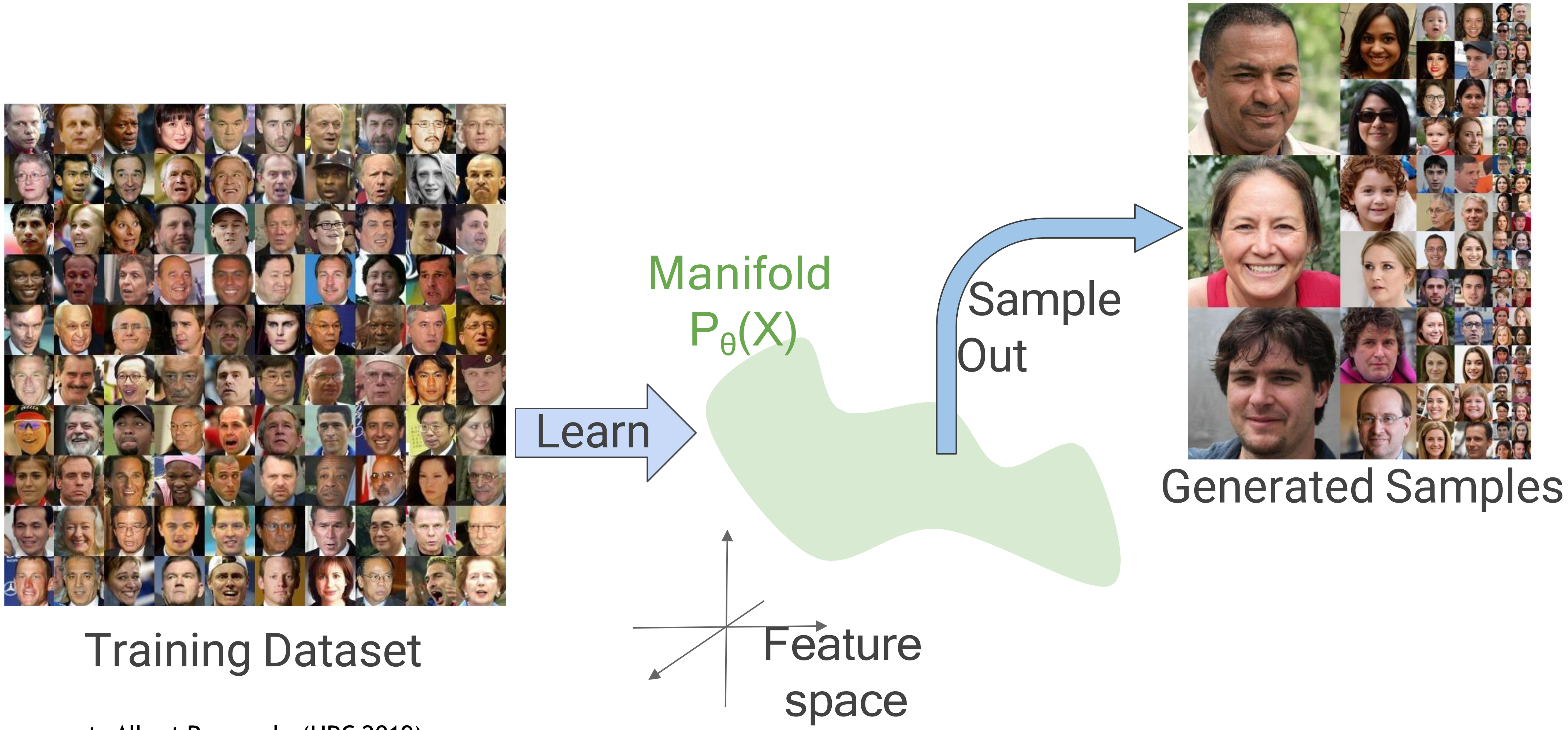
Sampling



Philip Isola, Generative Models of Images. MIT 2023.

Sampling

"Model the data distribution so that we can sample new points out of the distribution"

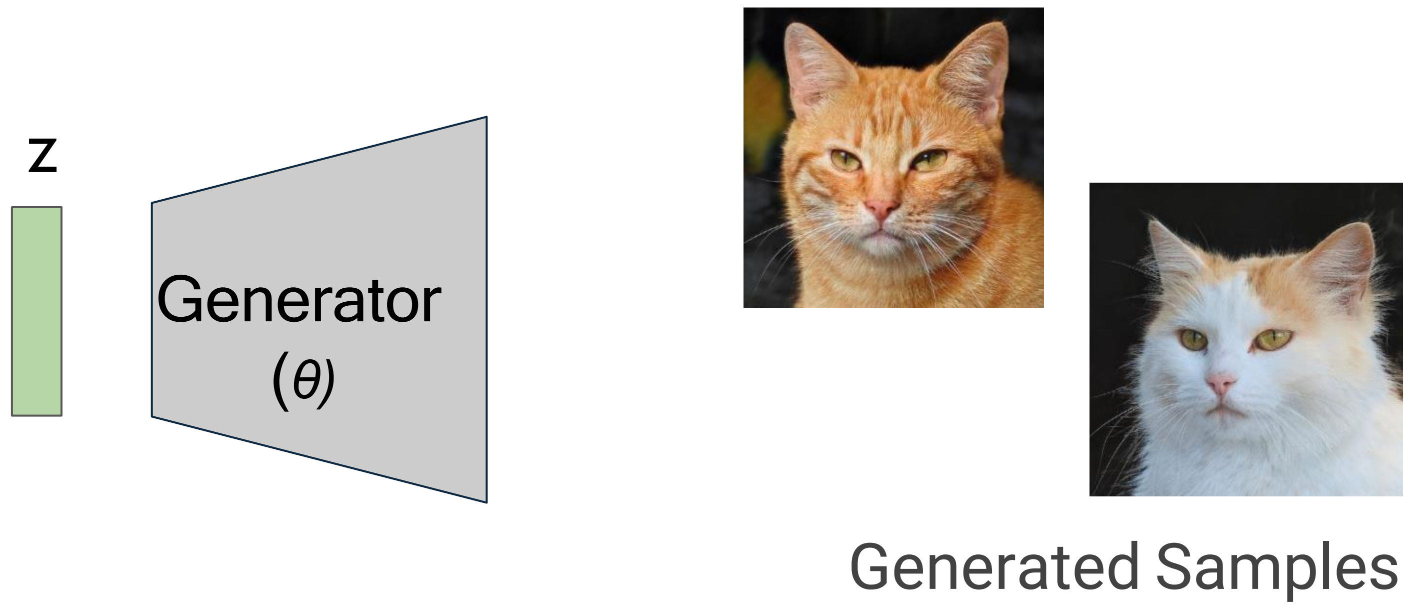


Training Dataset

Generated Samples

Sampling

How could we generate diverse samples from a deterministic deep neural network ?

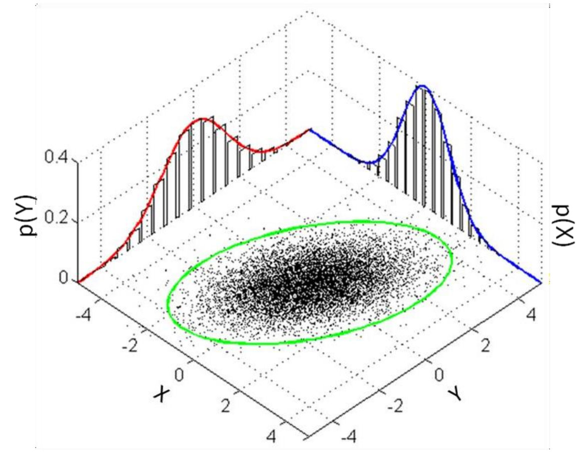


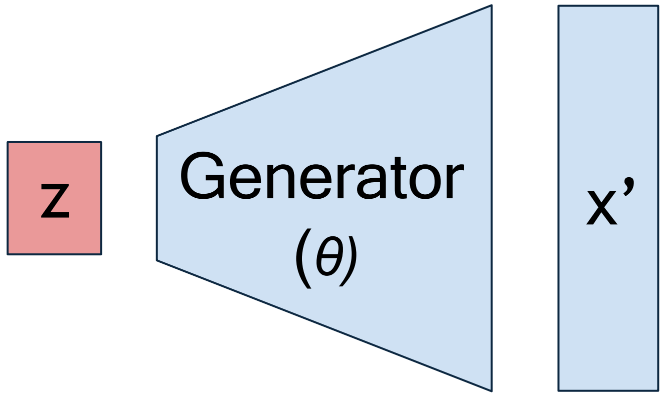
Sampling

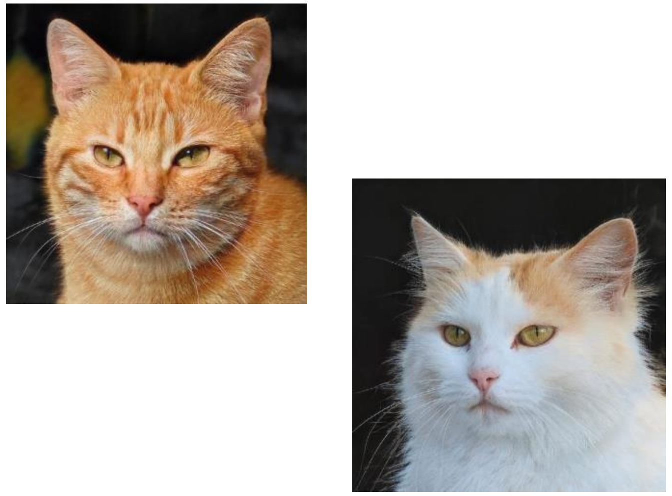
How could we generate diverse samples from a deterministic deep neural network ?

Sample z from a known prior, for example, a multivariate normal distribution N(0, I).

Example: dim(z)=2







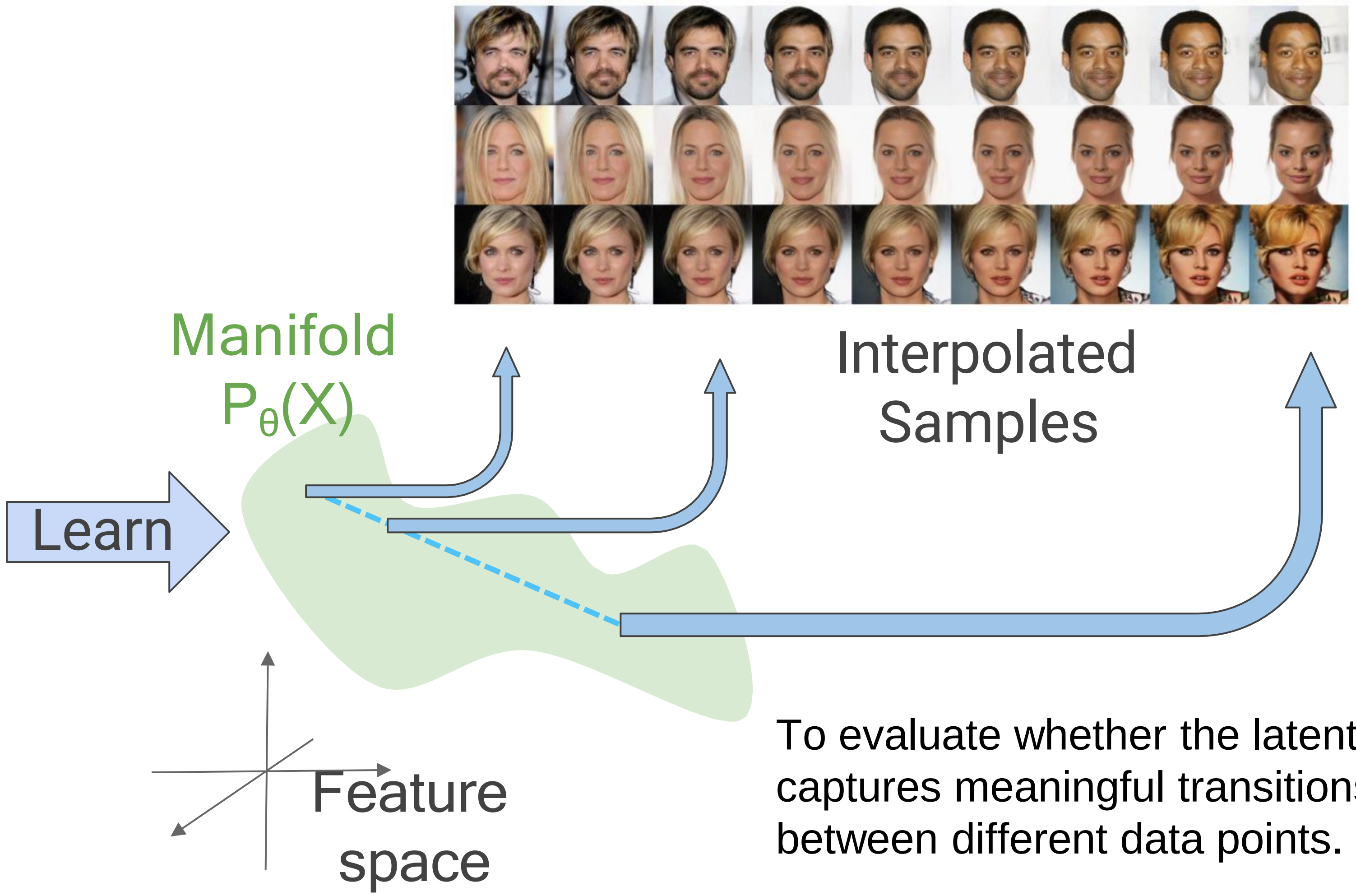
Generated Samples

Interpolation

Interpolation is the process of smoothly transitioning between two points in the latent space to generate intermediate outputs.

Traversing the learned manifold through interpolation.

Training Dataset



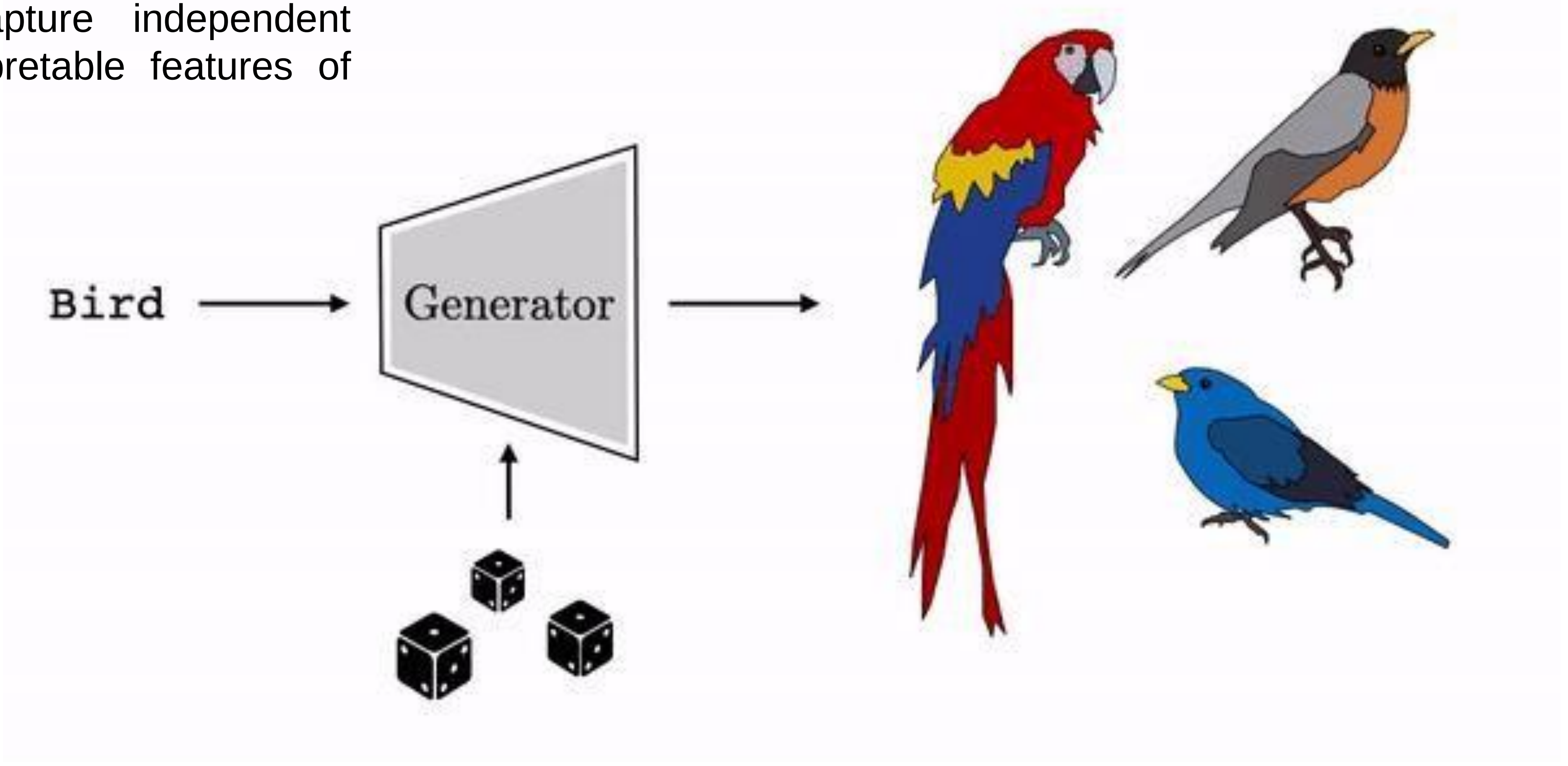
To evaluate whether the latent space captures meaningful transitions between different data points.

Slide concept: Albert Pumarola (UPC 2019)

Disentanglement

If features are **entangled**, changing one latent variable might affect multiple aspects of the generated output in unpredictable ways.

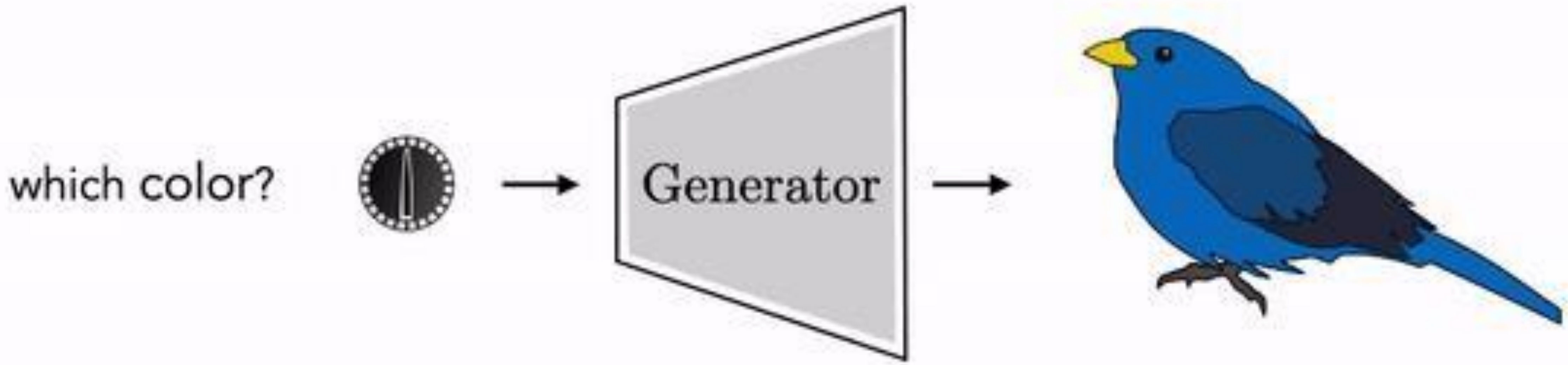
Entanglement refers to how well the latent variables in the model capture independent and interpretable features of the data.



Philip Isola, Generative Models of Images. MIT 2023.

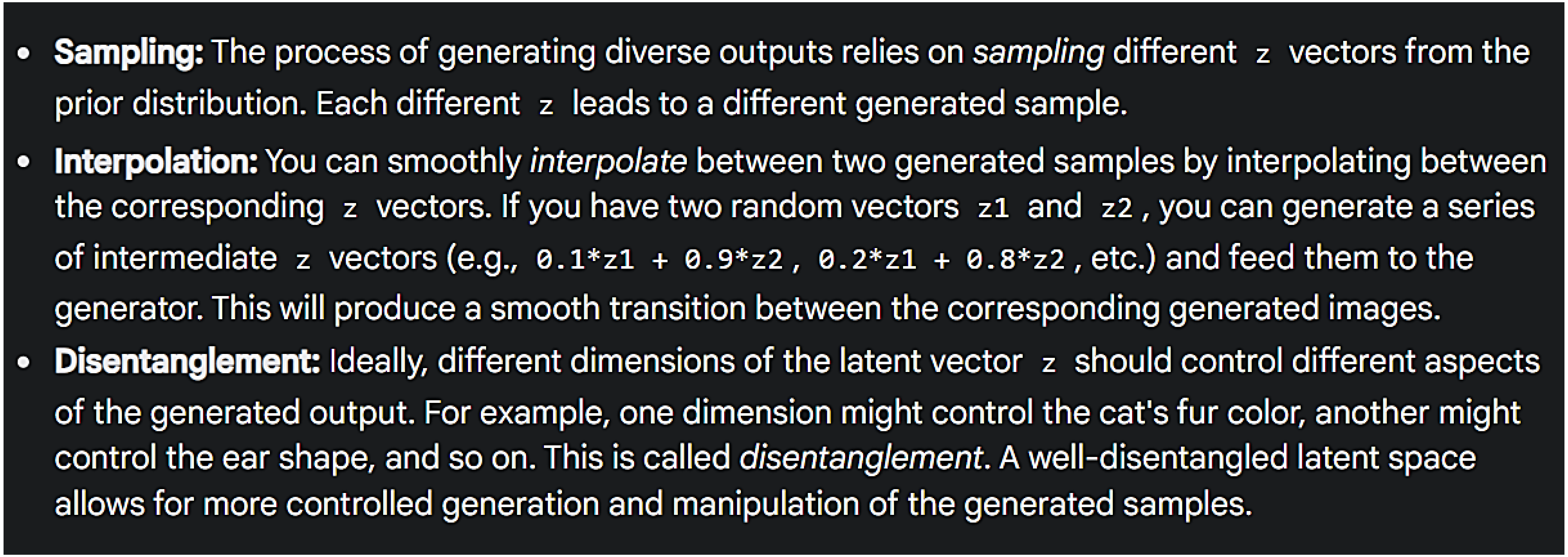
Disentanglement

In **disentangled representation learning** a disentangled latent space might allow you to modify a specific feature (e.g., hair color in an image) without affecting others (e.g., face shape).



High **entanglement** would mean that changing one variable, like "smile," might unintentionally also change "eye color" or "age."

Philip Isola, Generative Models of Images. MIT 2023.



- **Sampling:** The process of generating diverse outputs relies on *sampling* different `z` vectors from the prior distribution. Each different `z` leads to a different generated sample.
- **Interpolation:** You can smoothly *interpolate* between two generated samples by interpolating between the corresponding `z` vectors. If you have two random vectors `z1` and `z2`, you can generate a series of intermediate `z` vectors (e.g., `0.1*z1 + 0.9*z2`, `0.2*z1 + 0.8*z2`, etc.) and feed them to the generator. This will produce a smooth transition between the corresponding generated images.
- **Disentanglement:** Ideally, different dimensions of the latent vector `z` should control different aspects of the generated output. For example, one dimension might control the cat's fur color, another might control the ear shape, and so on. This is called *disentanglement*. A well-disentangled latent space allows for more controlled generation and manipulation of the generated samples.

Variational Autoencoders

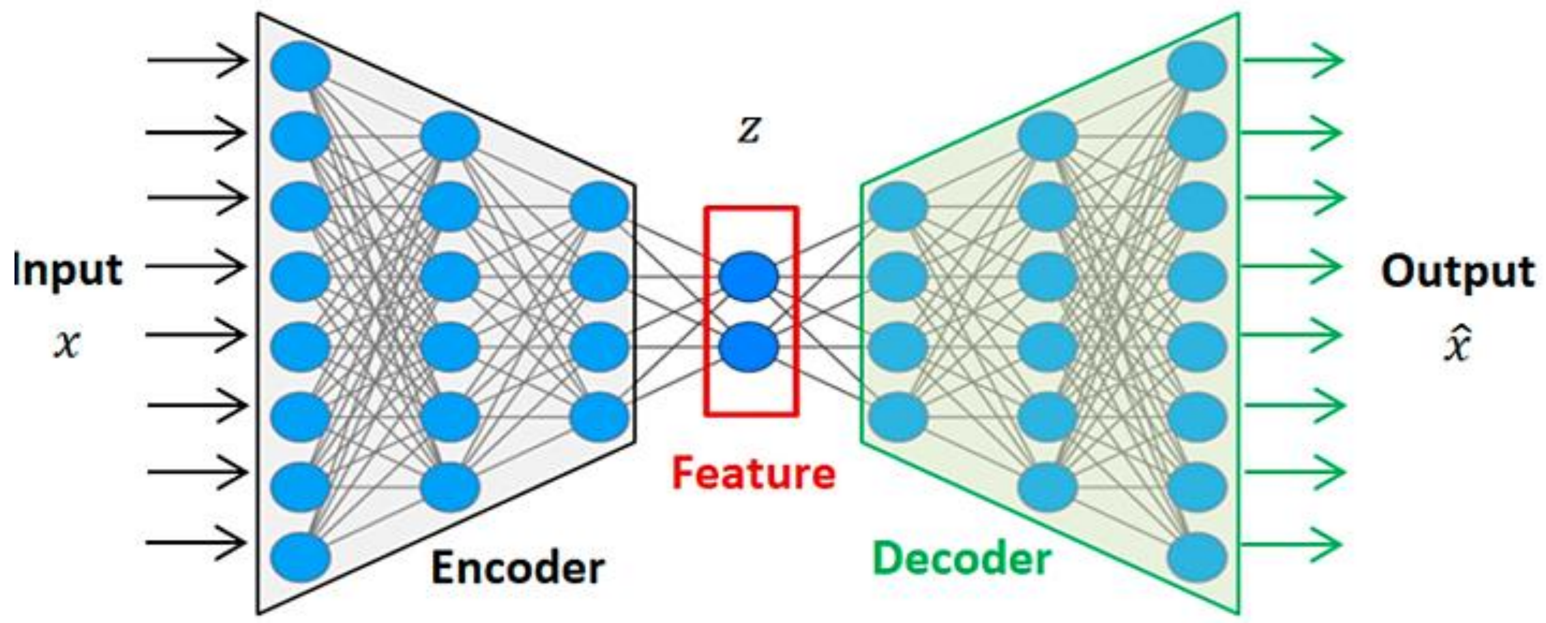
Autoencoders

- Autoencoders are
  - Artificial neural networks
  - Capable of learning efficient representations of the input data, called codings, without any supervision
  - The training set is unlabeled.
- These codings typically have a much lower dimensionality than the input data, making autoencoders useful for dimensionality reduction Autoencoders learn compressed representations of data by mapping it to a lower-dimensional manifold in the feature space.

Autoencoders

An **autoencoder** is a type of neural network designed to learn efficient, compressed representations of data. It consists of two main parts:

1. **Encoder**: Compresses the input data into a lower-dimensional latent space representation.
2. **Decoder**: Reconstructs the original data from the latent space representation.



Autoencoders

- Use for dimensionality reduction
- Act as powerful feature detectors
- Surprisingly, autoencoders work by simply learning to copy their inputs to their outputs
- This may sound like a trivial task, but we will see that constraining the network in various ways can make it rather difficult

Autoencoders

- You can limit the size of the internal representation, or you can add noise to the inputs and train the network to recover the original inputs.
- These constraints prevent the autoencoder from trivially copying the inputs directly to the outputs, which forces it to learn efficient ways of representing the data
- In short, the codings are byproducts of the autoencoder’s attempt to learn the identity function under some constraints

Efficient Data Representations

Which of the following number sequences do you find the easiest to memorize?

- 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
- 50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20

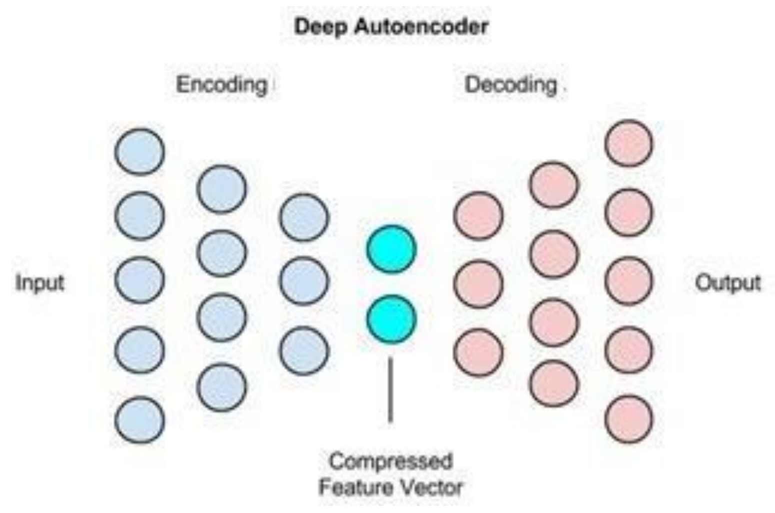
Efficient Data Representations

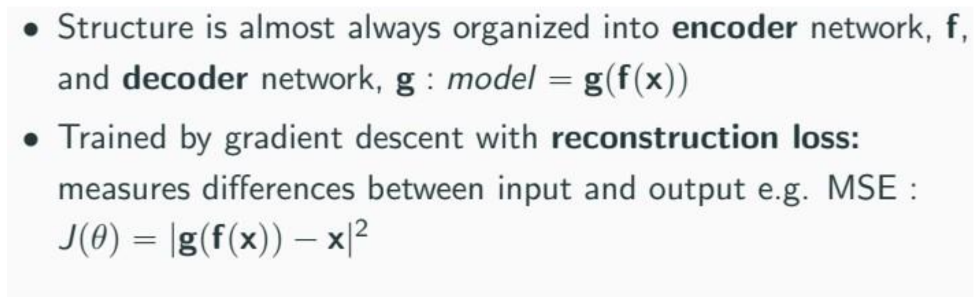
- At first glance, it would seem that the first sequence should be easier, since it is much shorter
- However, if you look carefully at the second sequence, you may notice that it follows two simple rules:
  - Even numbers are followed by their half,
  - And odd numbers are followed by their triple plus one
- This is a famous sequence known as the hailstone sequence

Efficient Data Representations

- Once you notice this pattern, the second sequence becomes much easier to memorize than the first because you only need to memorize the two rules,
  - The first number,
  - And the length of the sequence

Efficient Data Representations



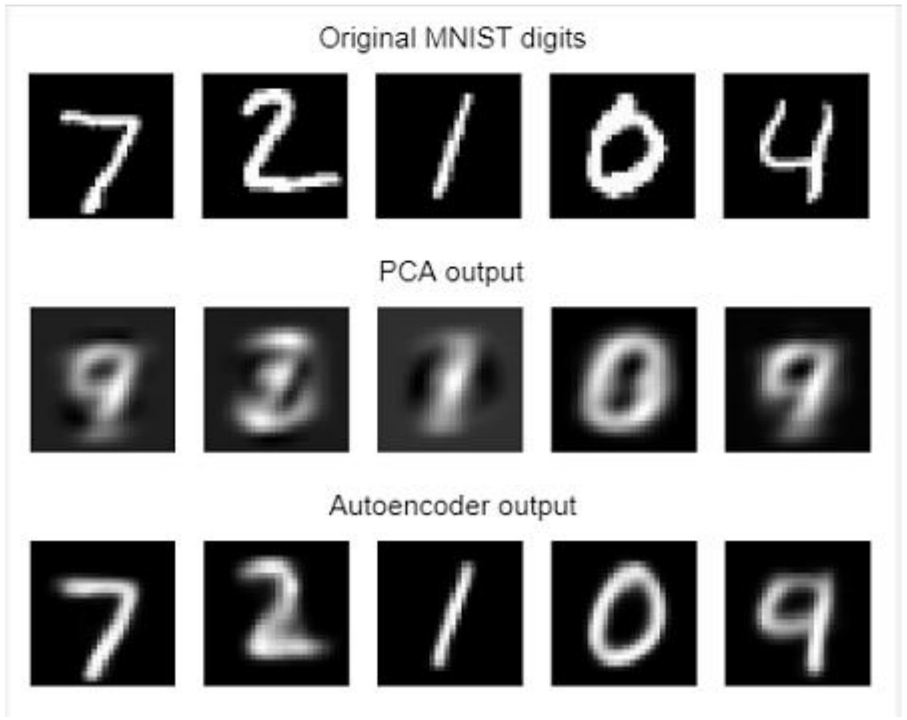


- Structure is almost always organized into **encoder** network, **f**, and **decoder** network, **g** : *model* $= \mathbf{g}(\mathbf{f}(\mathbf{x}))$
- Trained by gradient descent with **reconstruction loss:** measures differences between input and output e.g. MSE : $J(\theta) = |\mathbf{g}(\mathbf{f}(\mathbf{x})) - \mathbf{x}|^2$

- Composition of Autoencoder
- An autoencoder is always composed of two parts:
  - An encoder or recognition network that converts the inputs to an internal representation,
  - Followed by a decoder or generative network that converts the internal representation to the outputs

Applications of AE

- Data Compression (AE vs PCA)
- Image Retrieval
  - Makes search fast
  - Consumes less memory
- Image Denoising
- Image colorization
- Anomaly Detection

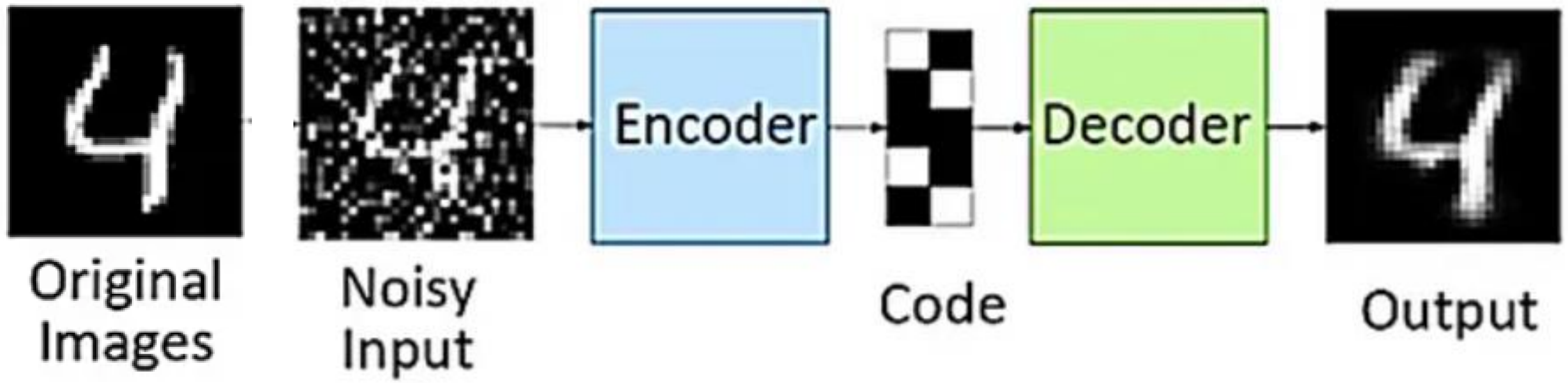


Applications of AE

- Data Compression (AE vs PCA)
- Image Retrieval
  - Once the autoencoder is trained, compute and store the latent representations for all images in the dataset.
  - When a query image is provided, the encoder computes its latent representation z.
  - This *z* is compared to the *stored representations* (using a similarity metric like Euclidean distance or cosine similarity).
  - Makes search fast
  - Consumes less memory

Applications of AE

- Image Denoising
  - Trained AE on images
  - Provide some noisy image during inference
  - Decoder will compute diff with the original image and output an image similar to the original one (denoised)



https://medium.com/@ompramod9921/autoencoders-explained-9196c38af6f6

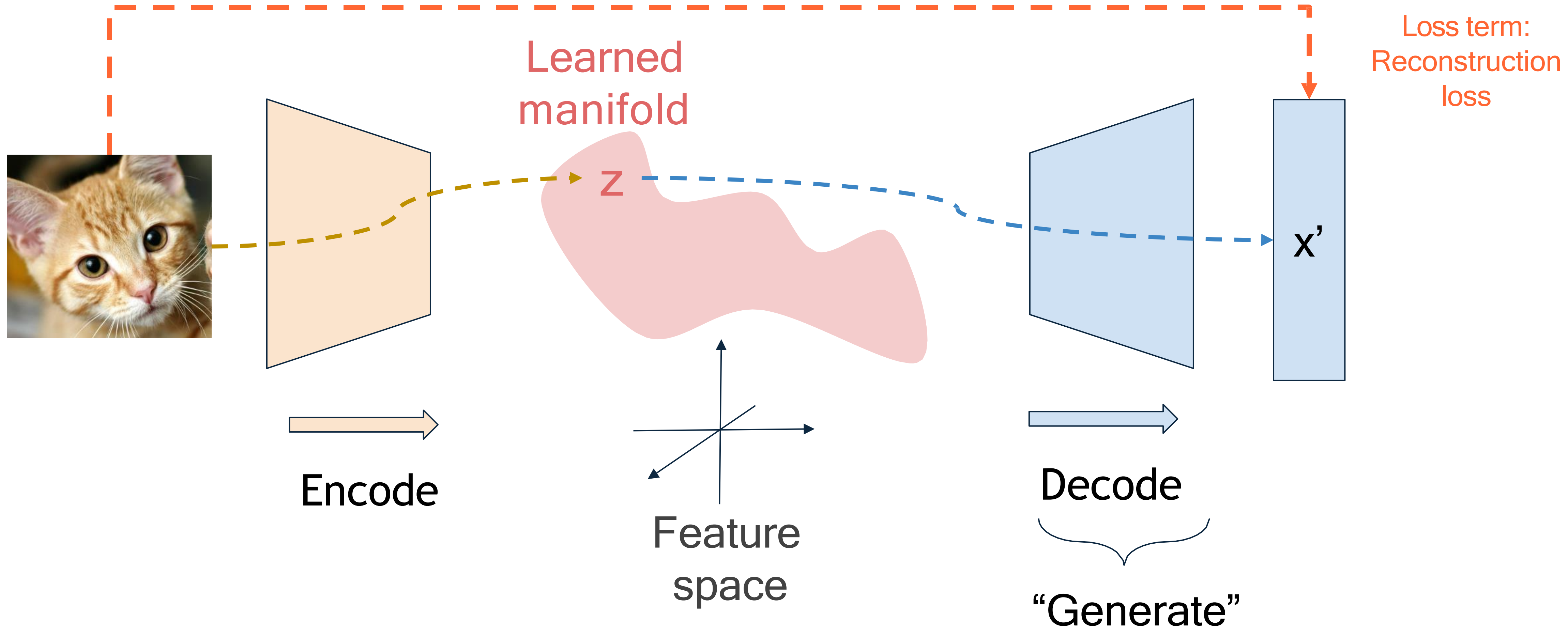
Applications of AE

- Image colorization
  - Input grayscale, output colored image
  - To train the model, paired grayscale and color images are required. The grayscale image serves as the input, while the corresponding color image serves as the target for reconstruction.
- Anomaly Detection

https://medium.com/@ompramod9921/autoencoders-ex

Auto-Encoder (AE)

- Trained with a reconstruction loss.
- Proposed as a pre-training stage for the encoder ("self-supervised learning").

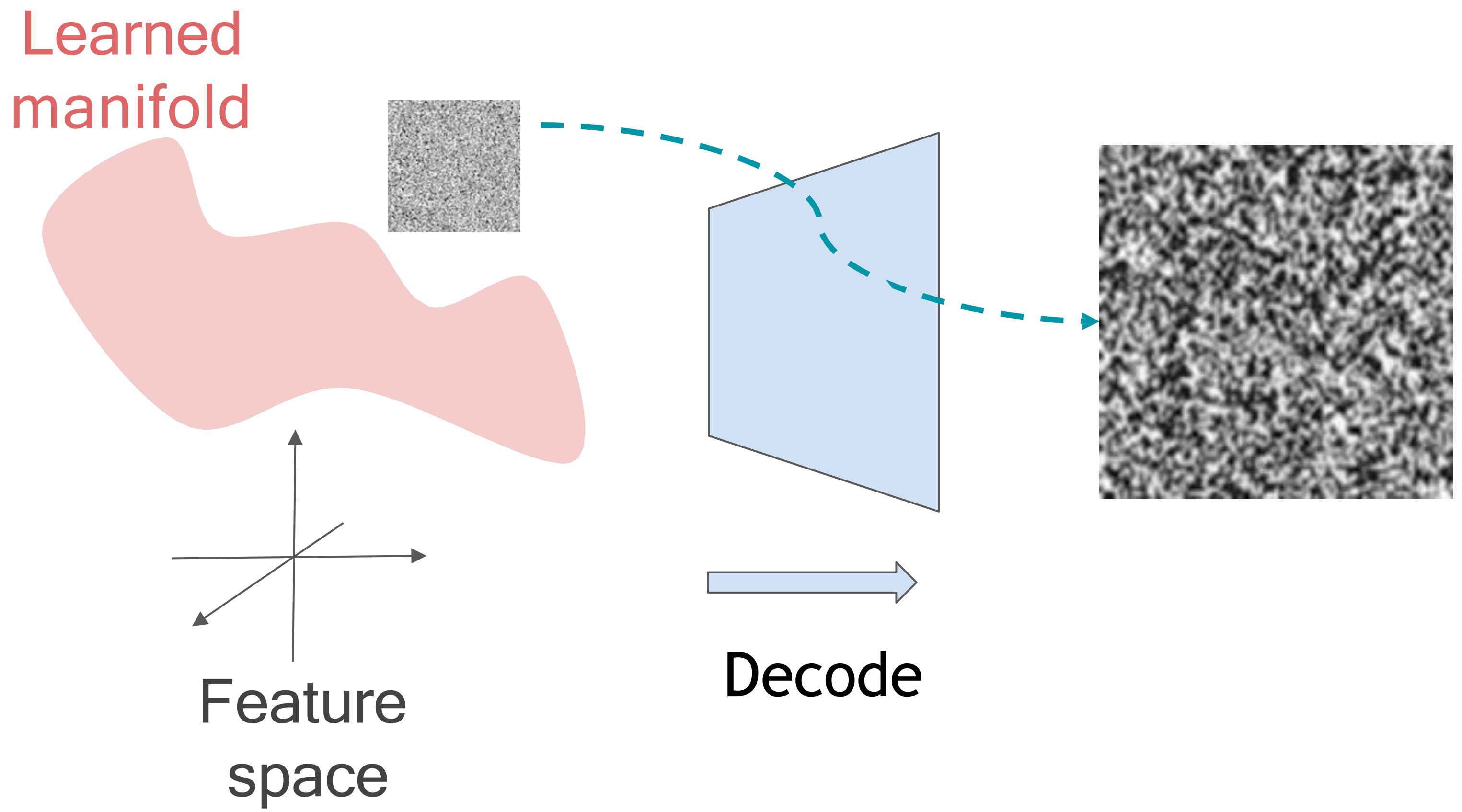


#**AE** Hinton, Geoffrey E., Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 2006.

Auto-Encoder (AE) for generation ?

Could we generate new samples by sampling from a normal distribution and feeding it into the decoder (as in GANs) ?

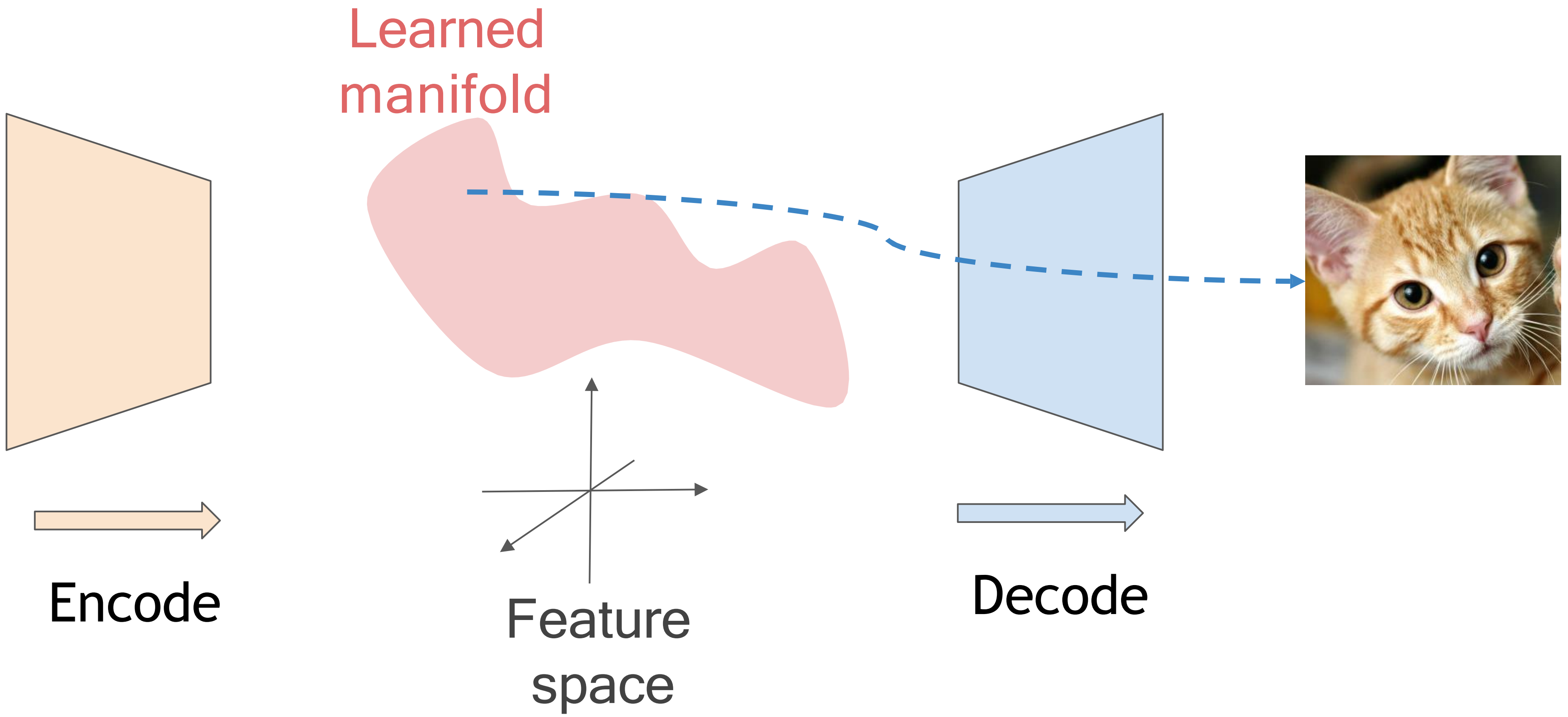
The power of AE comes with its non-linearity.
Adding non-linearity (such as nonlinear activation functions, and more hidden layers) makes AE capable to learn rather powerful representations of the input data in lower dimensions with much less information loss.



No, because the noise (or encoded noise) would be outside the learned manifold.

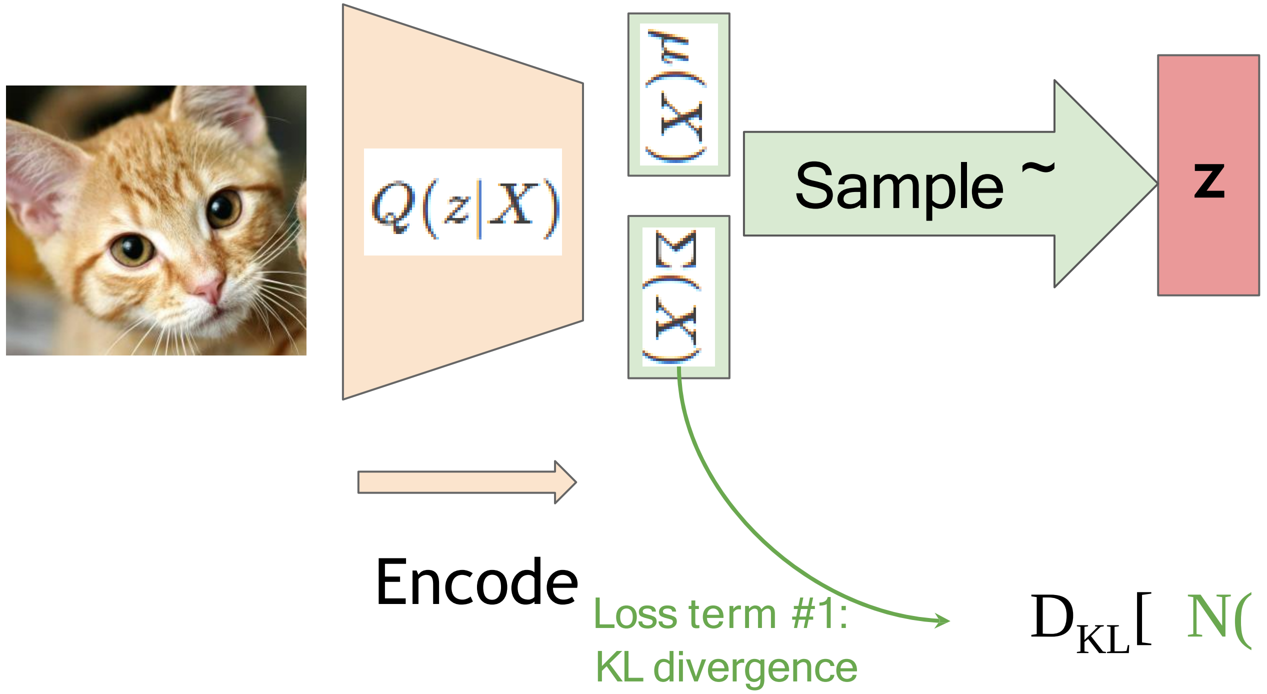
Auto-Encoder (AE) for generation ?

How could we train an AE that would allow to sample from its latent space ?



VAE - Training

Encoder (training only): Instead of mapping the input *x* to a fixed vector, train the encoder to map the image into a multivariate normal distribution N(μ(X), ∑(X)) that should map a standard normal distribution N(0, I) .



$D_{KL}[\ N(\mu(X), \sum(X))\ )\ ||\ N(0, I)\ ]$

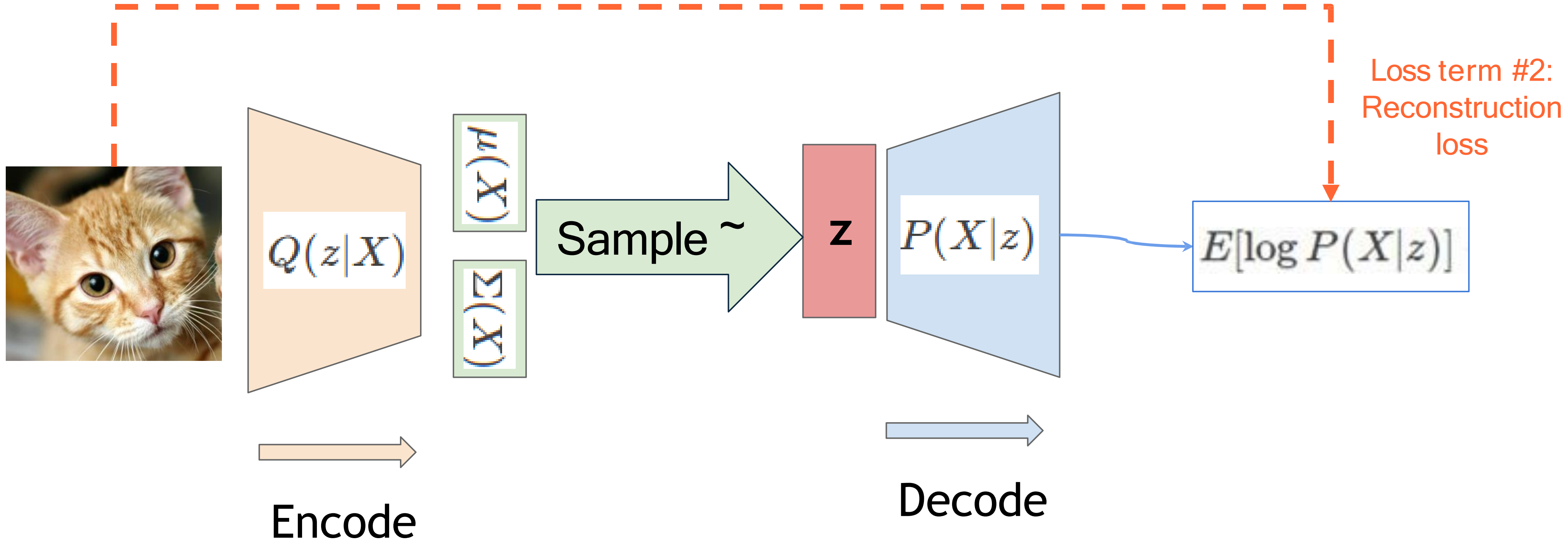
Mapping z to a range of values (prob dist.) instead of discrete/fixed values as in VAE.

The VAE imposes a constraint on this latent distribution forcing it to be a normal distribution. This constraint makes sure that the latent space is regularized.

Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." ICLR 2014.

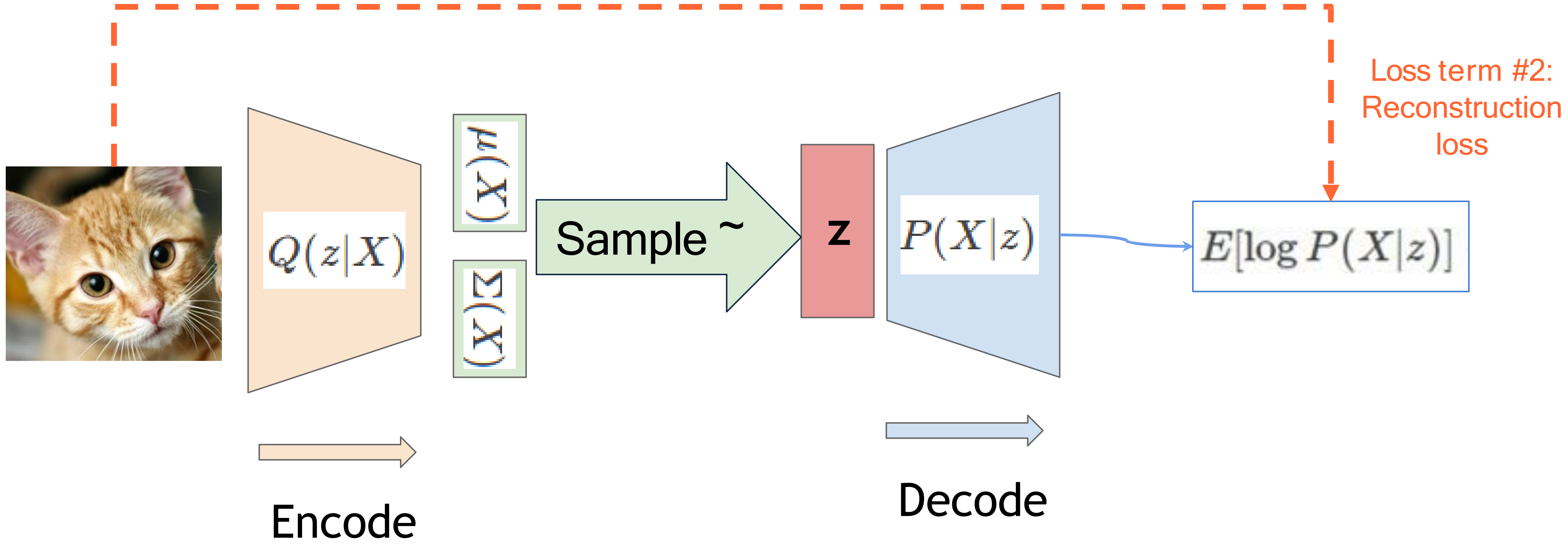
VAE - Training

Decoder: Reconstruct the input data from z ~ N(μ(X), ∑(X)).



VAE - Training

Do you see any challenge in this scheme if use deep neural networks to estimate Q(z|X) and P(X|z) ?



Random Sampling & Backprop:
In a VAE, the latent vector z is *sampled* from a distribution generated by the encoder. This random sampling introduces a problem for training the encoder using backpropagation.

Backpropagation requires calculating gradients (how much the output changes with respect to the input). However, you can't directly calculate gradients through a random sampling operation because it's not a deterministic function.

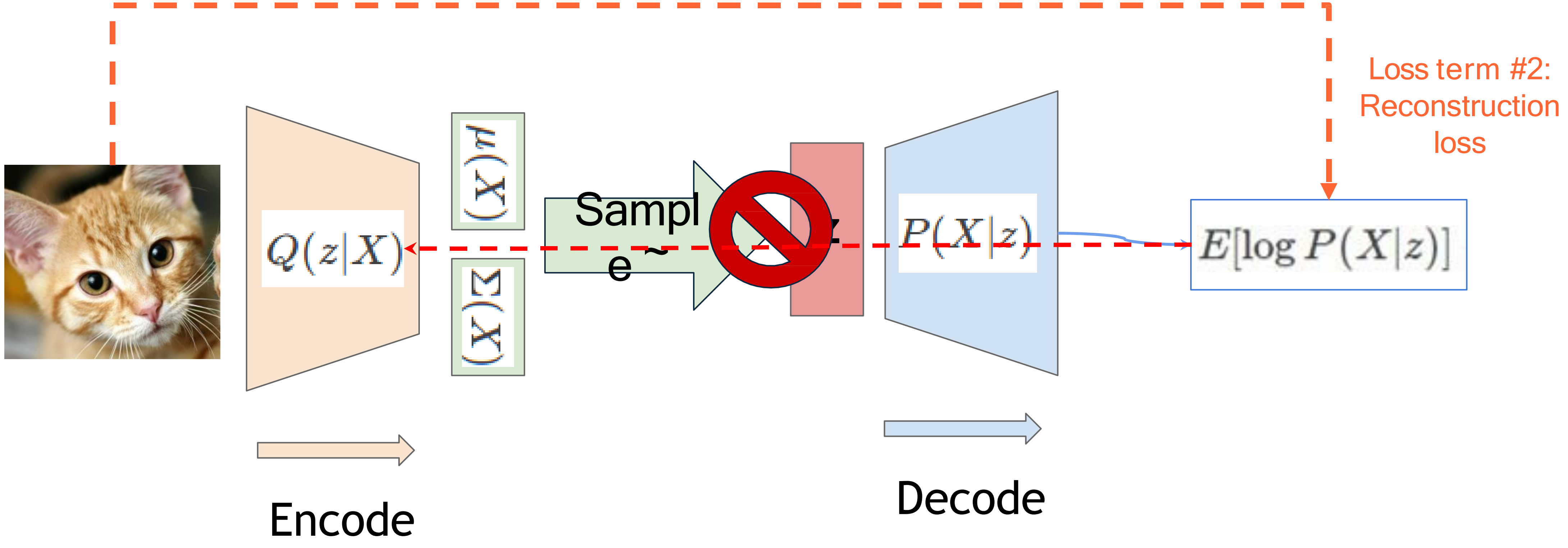
Random sampling is essential for VAEs because it allows them to generate diverse outputs. If you didn't sample and just used the mean of the latent distribution, the VAE would always generate the same output.

$$z = \mu_x + \sigma_x \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

The reparameterization trick solves this problem by re-expressing the sampling process in a way that makes it differentiable. Instead of directly randomly sampling from the encoder's output distribution, take the mean and unit variance from this, which is deterministic and add the stochastic/random element from epsilon, which is random noise sampled from normal distribution.

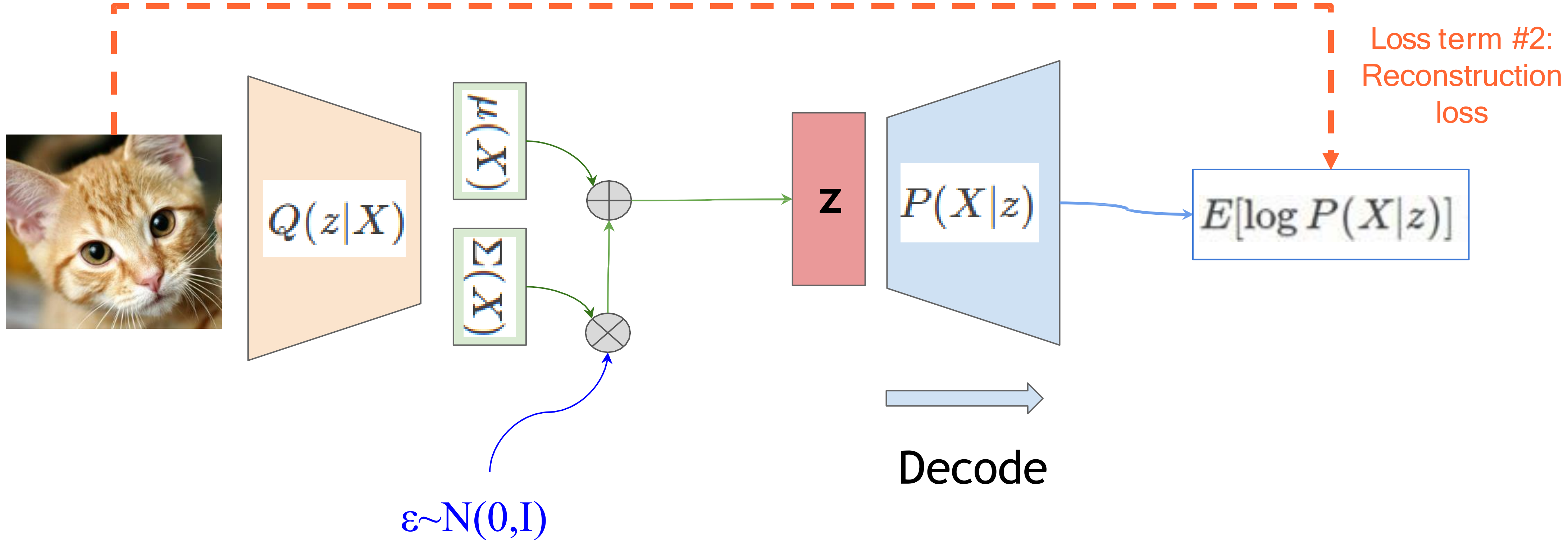
VAE - Training

Challenge: We cannot backprop through "Sampling" because it is not a differentiable operation.



VAE - Training

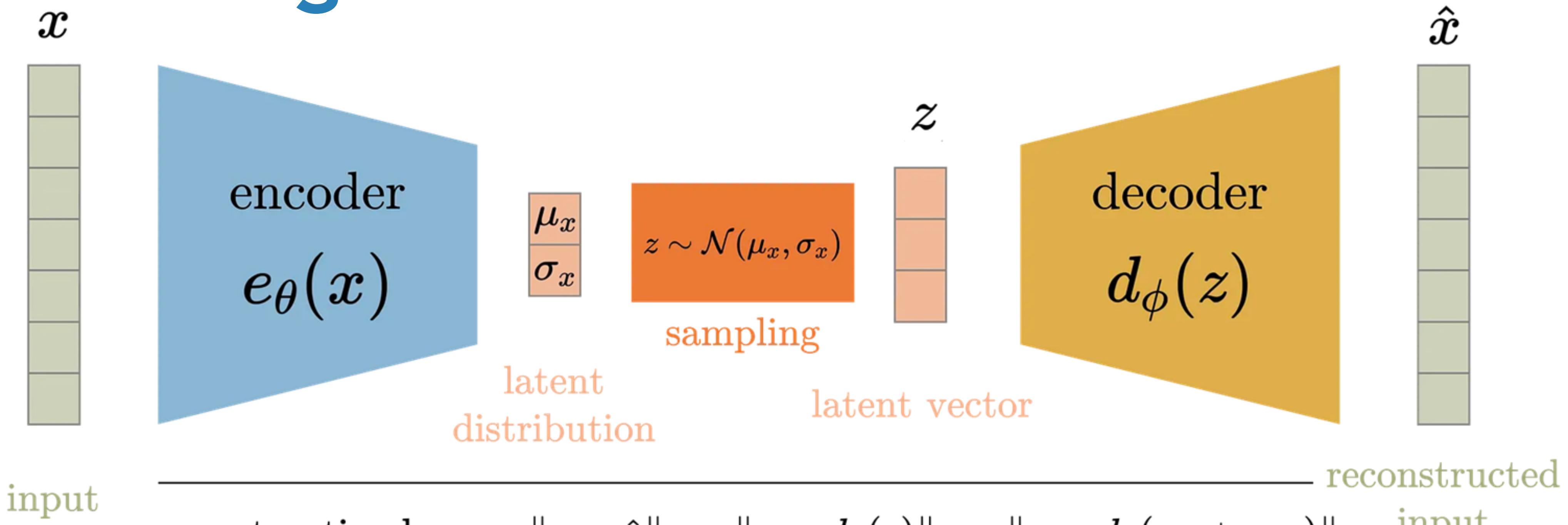
Sampling: The **reparametrization trick** uses an auxiliary variable $\varepsilon \sim N(0,I)$ to obtain a sample z from the distribution N(μ, ∑).



Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." ICLR 2014.

VAE - Training

Figure: Aqeel Anwar



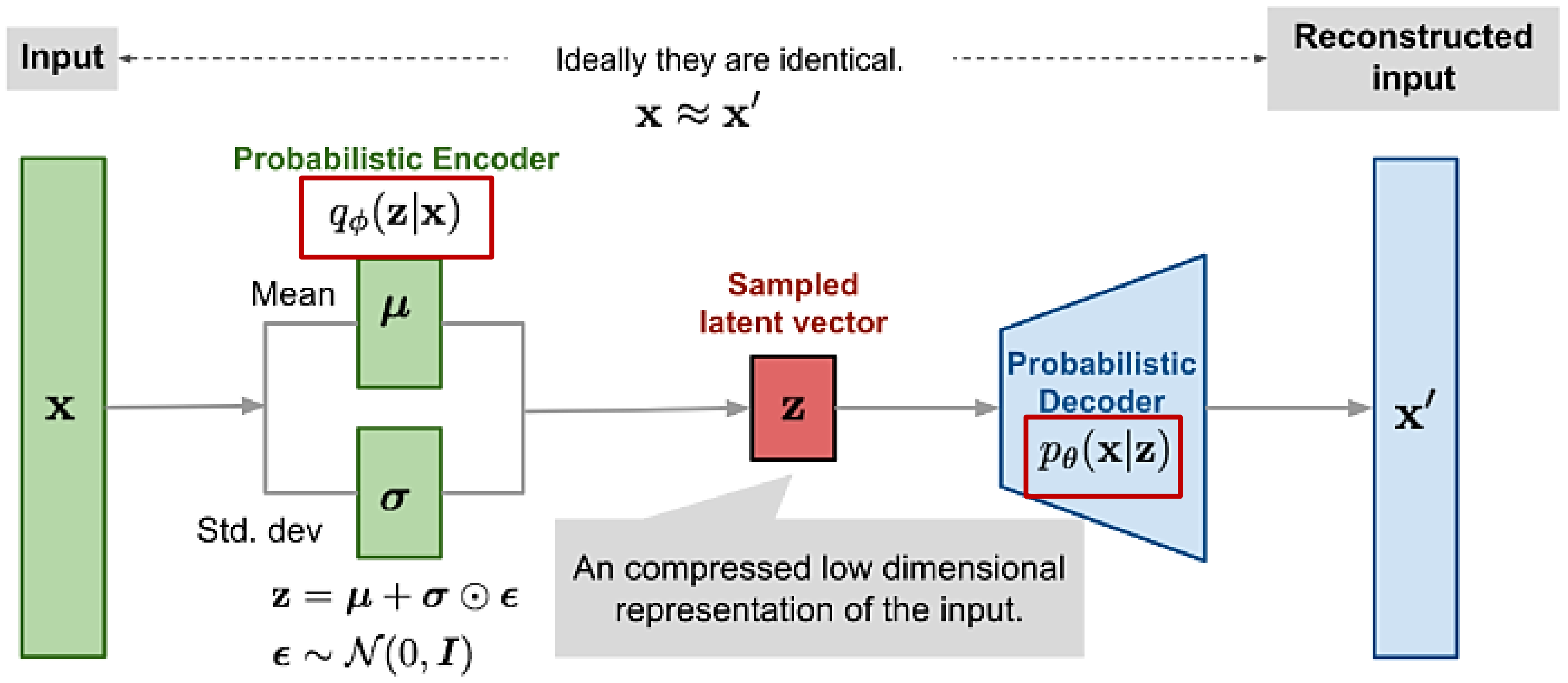
$$\text{reconstruction loss} = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(\mu_x + \sigma_x \epsilon)\|_2$$

$$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\textit{similarity loss} = \textit{KL Divergence} = D_{KL}(\overset{\text{Latent distribution}}{\underline{\mathcal{N}(\mu_x, \sigma_x)}} \parallel \underline{\mathcal{N}(\mathbf{0}, \mathbf{I})})\text{Prior distribution}$$

$$\textit{loss} = \textit{reconstruction loss} + \textit{similarity loss}$$

The **reconstruction error**, is the mean squared loss of the input and reconstructed output.
The **similarity loss** is the KL divergence between the latent space distribution and standard gaussian (zero mean and unit variance).
The loss function is then the sum of these two losses.


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{x}$
Sampled latent vector
$\mathbf{z}$
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
An compressed low dimensional representation of the input.

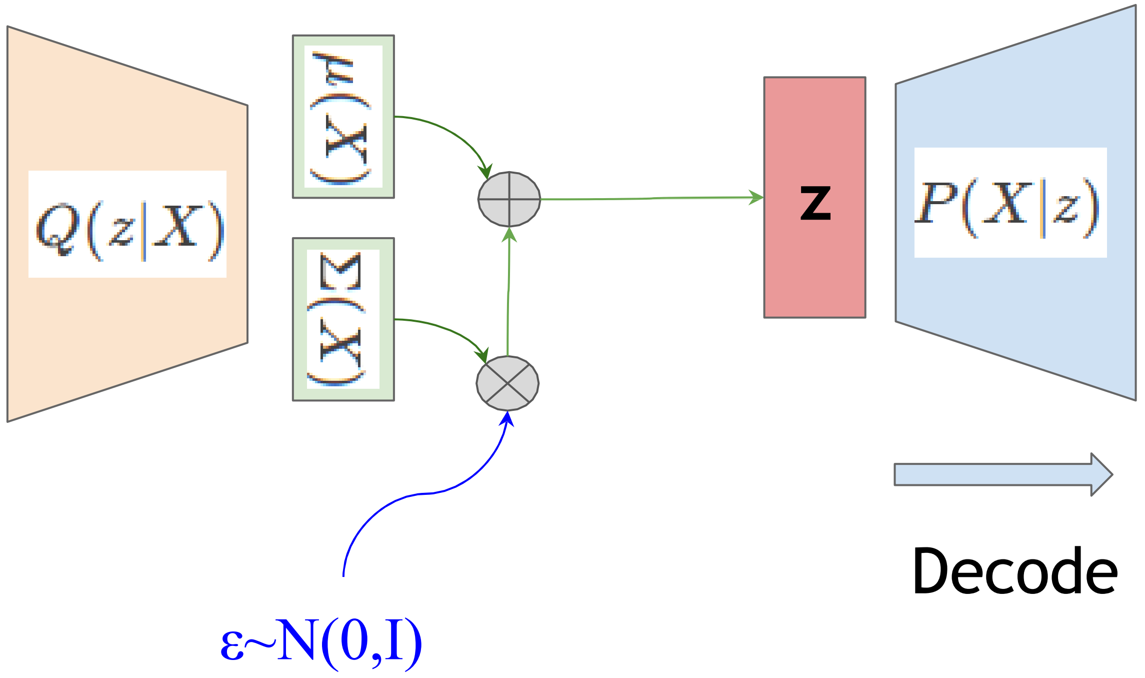
$$L(\theta, \phi) = \underbrace{-\mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x|z)\right]}_{\text{Negative Reconstruction Loss}} + \underbrace{D_{\mathrm{KL}} \left(q_\phi(z|x) \| p(z)\right)}_{\text{KL Divergence Regularization}}$$

Regularizes the posterior qϕ(z|x) to be close to the prior p(z), typically a standard Gaussian N(0,I).

The reparameterization trick z = μ + σ * ε allows us to sample from this distribution by adding a random noise term ε to the mean μ, scaled by the standard deviation.

VAE Inference

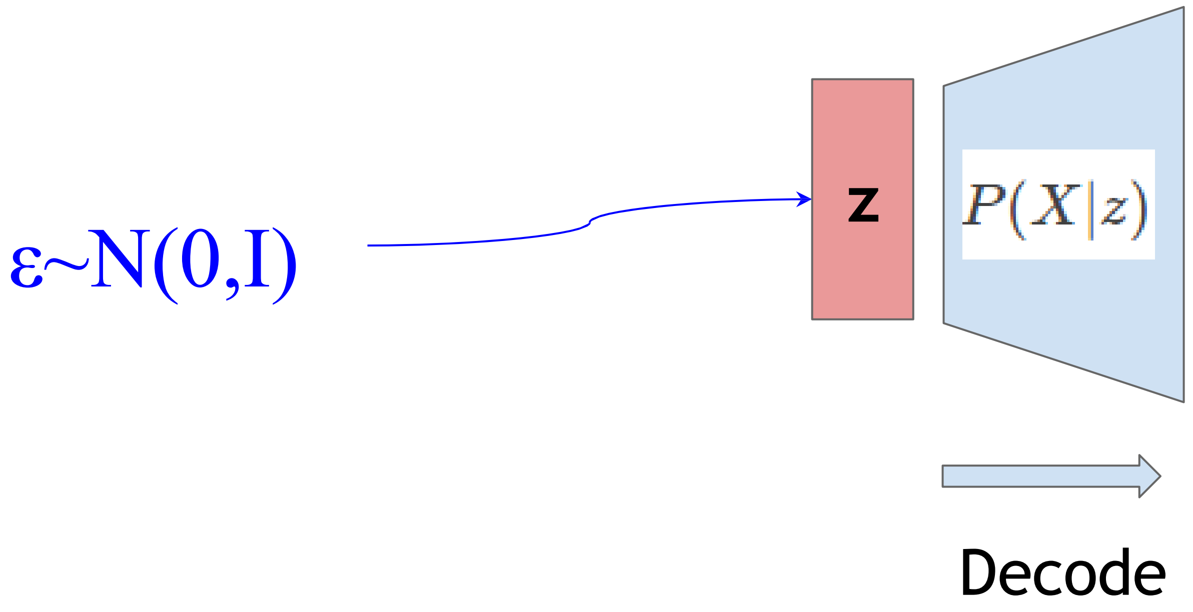
How should be modify the training scheme to generate new samples from a trained VAE ?



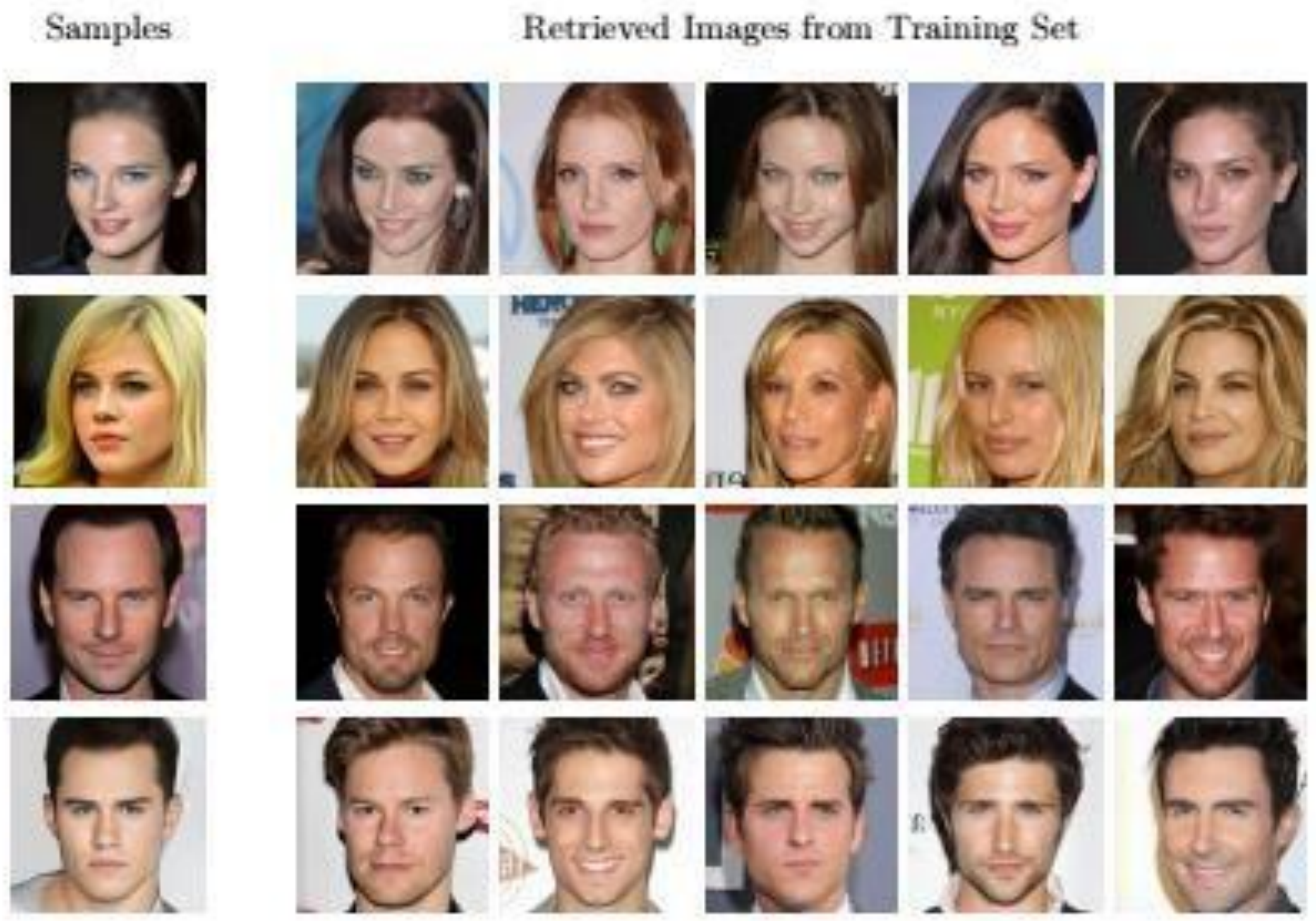
Generative behaviour

How should be modify the training scheme to generate new samples from a trained VAE ?

We can sample from our prior $\varepsilon \sim N(0,I)$, discarding the encoder path.



Generative behaviour



#NVAE Vahdat, Arash, and Jan Kautz. "NVAE: A deep hierarchical variational autoencoder." NeurIPS 2020. [code]