EURO^2

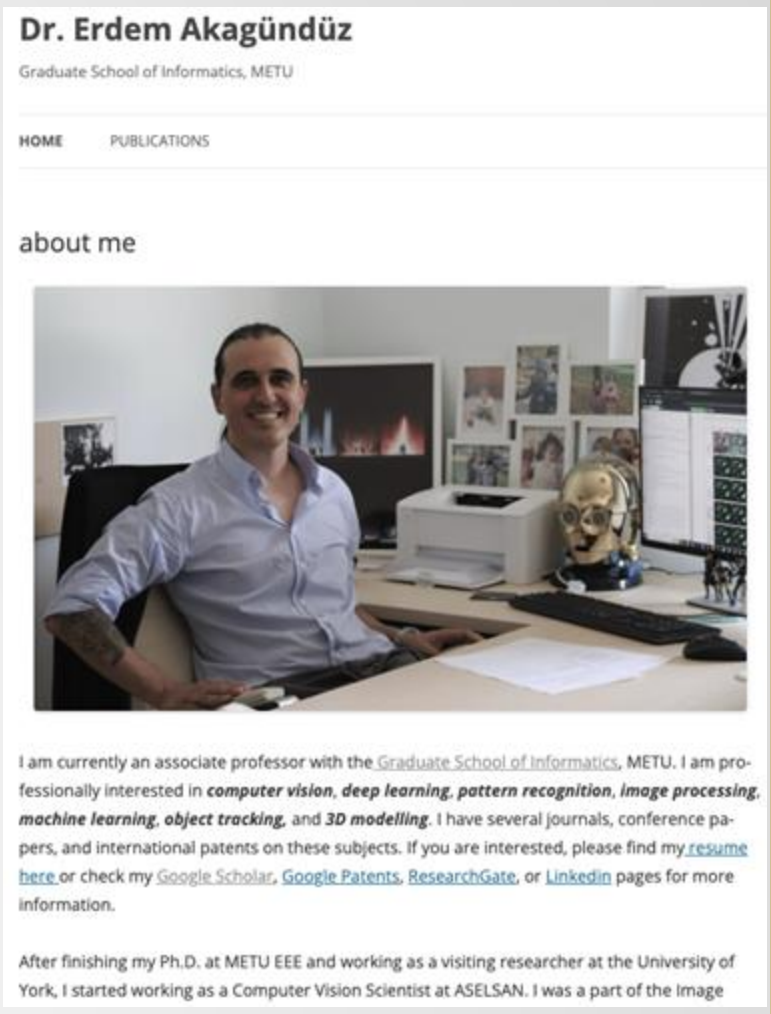
Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Ankara

Meet the Instructor

- Education
  - B.Sc. EEE METU, Turkey (1997-2001)
    B.Sc. Minor Program, Sociology METU (1998-2001)
  - M.Sc. EEE, METU, Turkey
    on "3D Modelling and Graphics" (2001-2004)
  - Ph.D. EEE, METU, Turkey
    on "3D Object/Surface Representation" (2004-2011)
- Professional
  - Research and Teaching Assistant, EEE METU, (2001 - 2008)
  - Visiting Phd Student, the Univ. of York (2008 - 2009)
  - Applied Scientist, ASELSAN, Inc, (2009 - 2016)
  - Research Associate, the Univ. of York, (2016)
  - Assistant Professor, EEE Çankaya Uni. (2018 - 2021)
  - Research Associate, the Univ. of York, (2022-2023)
  - Associate Professor, MMI II METU (2021 - …)

https://blog.metu.edu.tr/akaerdem/



Dr. Erdem Akagündüz

Graduate School of Informatics, METU

HOME PUBLICATIONS

about me

I am currently an associate professor with the Graduate School of Informatics, METU. I am professionally interested in ***computer vision***, ***deep learning***, ***pattern recognition***, ***image processing***, ***machine learning***, ***object tracking***, and ***3D modelling***. I have several journals, conference papers, and international patents on these subjects. If you are interested, please find my resume here or check my Google Scholar, Google Patents, ResearchGate, or Linkedin pages for more information.

After finishing my Ph.D. at METU EEE and working as a visiting researcher at the University of York, I started working as a Computer Vision Scientist at ASELSAN. I was a part of the Image

Our Institute

- Associate Professor, MMI II METU
  - MSc, PhD Advisorships,
  - Research Project (TUBITAK) Administrations
  - Industrial Consultancies
  - Teaching
    - DI504 Foundations of Deep Learning
    - MMI711 Sequence Models in Multimedia
    - MMI714 Generative Models in Multimedia
    - IS566 Image Processing Algorithms
    - MMI704 Human Motion Capture, Analysis and Synthesis

https://ii.metu.edu.tr/full-time-faculty

Erdem Akagündüz, Assoc. Prof. Dr.

Multimedia Informatics

Room: B-216, **Phone:** 7886, **Email:** akaerdem[at]metu.edu.tr

Fields of Interest:

Computer Vision, Deep Learning, Pattern Recognition

Yeşim Aydın Son, Assoc. Prof. Dr. (Head of Health Informat

Health Informatics

Room: B-207, **Phone:** 7708, **Email:** yesim[at]metu.edu.tr

Fields of Interest:

bioinformatics; computational biology; genomics; GWAS microarra

research, personalized medicine, medical informatics, genomics,

next generation sequencing, neurogenetics, molecular genetics

Nazife Baykal, Prof. Dr.

Information Systems

Room: A-206, **Phone:** 7701, **Email:** baykal[at]metu.edu.tr

Fields of Interest:

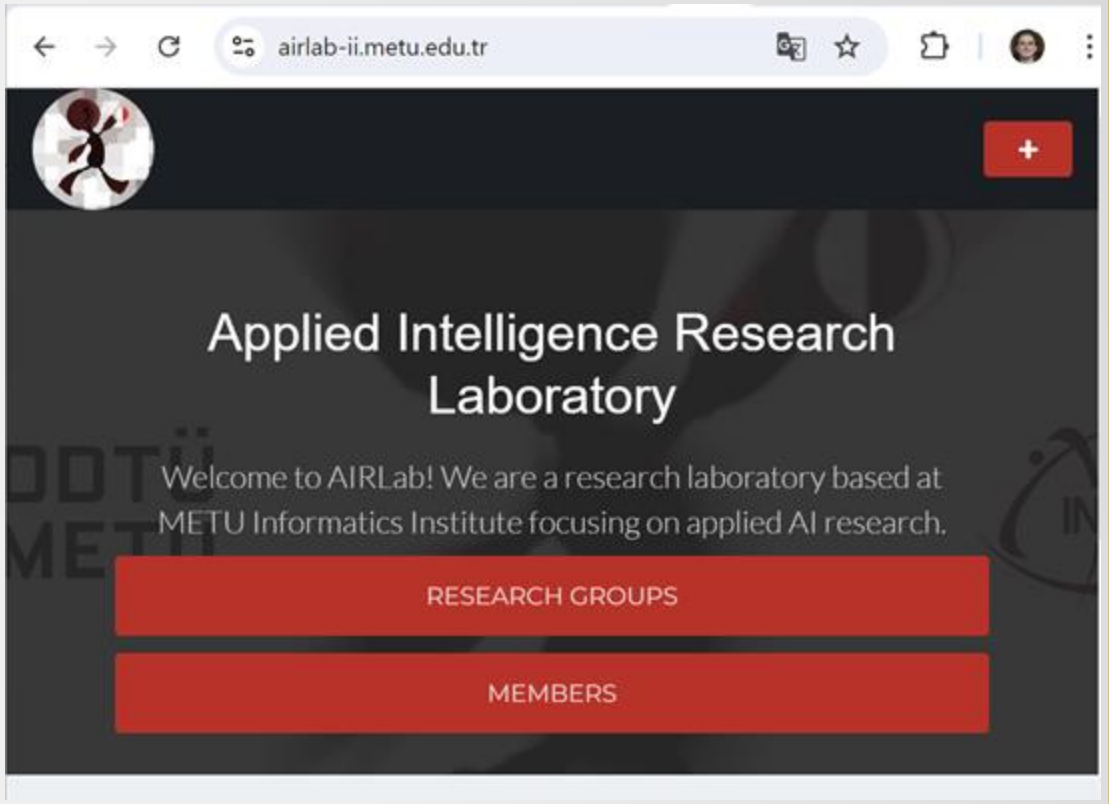
Cem Bozsahin, Prof. Dr.

Our Research Laboratory

https://airlab-ii.metu.edu.tr/

- **AIRLab** is home to different research groups focusing on applying AI problems across diverse domains including, but not limited to, computer vision, geoinformatics, earthquake engineering, and decision systems.

- For comprehensive insights into specific projects, publications, and team members, please navigate to the individual pages of each research group.



Preknowledge/Prerequisite(s)

(Must)

- Familiarity with Probability Theory
- Familiarity with Fundamental Machine Learning Concepts

(Optional/Recommended)

- Experience with Deep Neural Networks
- Experience with Information Theory

Preknowledge/Prerequisite(s)

This course is designed for a broad audience,

- including students with basic knowledge of probability theory and machine learning,
- as well as those who are more experienced in deep learning and information theory.

What will you learn?

Fundamental Concepts of Generative Machine Learning

- **Foundational Concepts**: Gain a solid understanding of the core principles of generative modeling, including the necessity of modeling data as distributions within latent spaces.
- **Mathematical and Theoretical Tools**: Learn about essential mathematical concepts, evaluation techniques, and introductory information theory, such as entropy and divergence.
- **Latent Space and Generative Models**: Explore the properties of latent spaces, deep feature spaces, and their roles in self-supervised auto-encoder systems.

Why this course?

Fundamental Concepts of Generative Machine Learning

- Many practitioners in machine learning and deep learning often focus on model training and prediction without fully understanding the concept of generation and its connection to information theory.
- This short course is designed to bridge that gap, providing a dive into the principles of generative modeling and its critical role in understanding and shaping data distributions within the broader context of AI.

Course Outline

PART I: Mathematical Background

- Generation vs. Discrimination in Machine Learning
- Data Distribution, Sampling, Inference and Generation
- Expectation and Likelihood
- Evaluation for Generative Models, Distribution Distances, Divergence and Entropy

Course Outline

EURO^2

PART II: Latent Spaces

- (Curse of) Dimensionality
- Deep Features vs. Latent Spaces
- Latent Space properties, Continuity, Entanglement, etc

Course Outline

PART III: Auto-Encoding

- Autoencoders and Dimensionality Reduction
- Variational Inference and VAEs
- Conclusions

What this course is

- **A Conceptual Foundation:** This course provides a deep understanding of the principles behind generative modeling, focusing on the importance of data generation through distributions.
- **Exploring Latent Spaces**: Gain insights into why modeling data within a latent space is crucial for effective generation and how it underpins various generative models.
- **Connecting with Information Theory**: Understand the relationship between generative modeling and key concepts in information theory, emphasizing why a simple distribution is often necessary.

What this course isn’t

- **Not Just Another Course on GANs, VAEs, or Diffusion Models**: This course does not focus on teaching specific generative model architectures or their technical details.
- **No Architecture-Specific Tutorials**: You won’t find step-by-step guides on how to build GANs or diffusion models here.
- **Not a Tool-Centric Approach**: The course emphasizes the underlying ideas and theory behind generation, rather than the application of specific tools or techniques.

Course Objectives

- **Understand the Core Principles**: Develop a solid grasp of the fundamental concepts in generative modeling, including the role of distributions in generating data.
- **Explore Latent Spaces**: Learn why latent spaces are essential for modeling and generation, and how they differ from deep feature spaces.
- **Evaluate Generative Models**: Understand the evaluation techniques used to assess the performance and accuracy of generative models.
- **Bridge Theory and Practice**: Connect theoretical concepts with practical applications, understanding how they underpin the generative models used in modern AI.

Set Up/Configure/Install …..

- **No Installation Required:** This course focuses on the theoretical foundations of generative modeling, so you won’t need to install any software or tools.
- **Conceptual Understanding:** Our emphasis is on understanding the underlying principles and ideas, rather than hands-on coding or model implementation.

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

Lesson 1: Mathematical Background

Welcome to Part I: “Mathematical Background”

This part includes four subsections:

- **Generation vs. Discrimination in Machine Learning**
- Data Distribution, Sampling, Inference and Generation
- Expectation and Likelihood
- Evaluation for Generative Models, Distribution Distances, Divergence and Entropy

Discriminative vs Generative Models

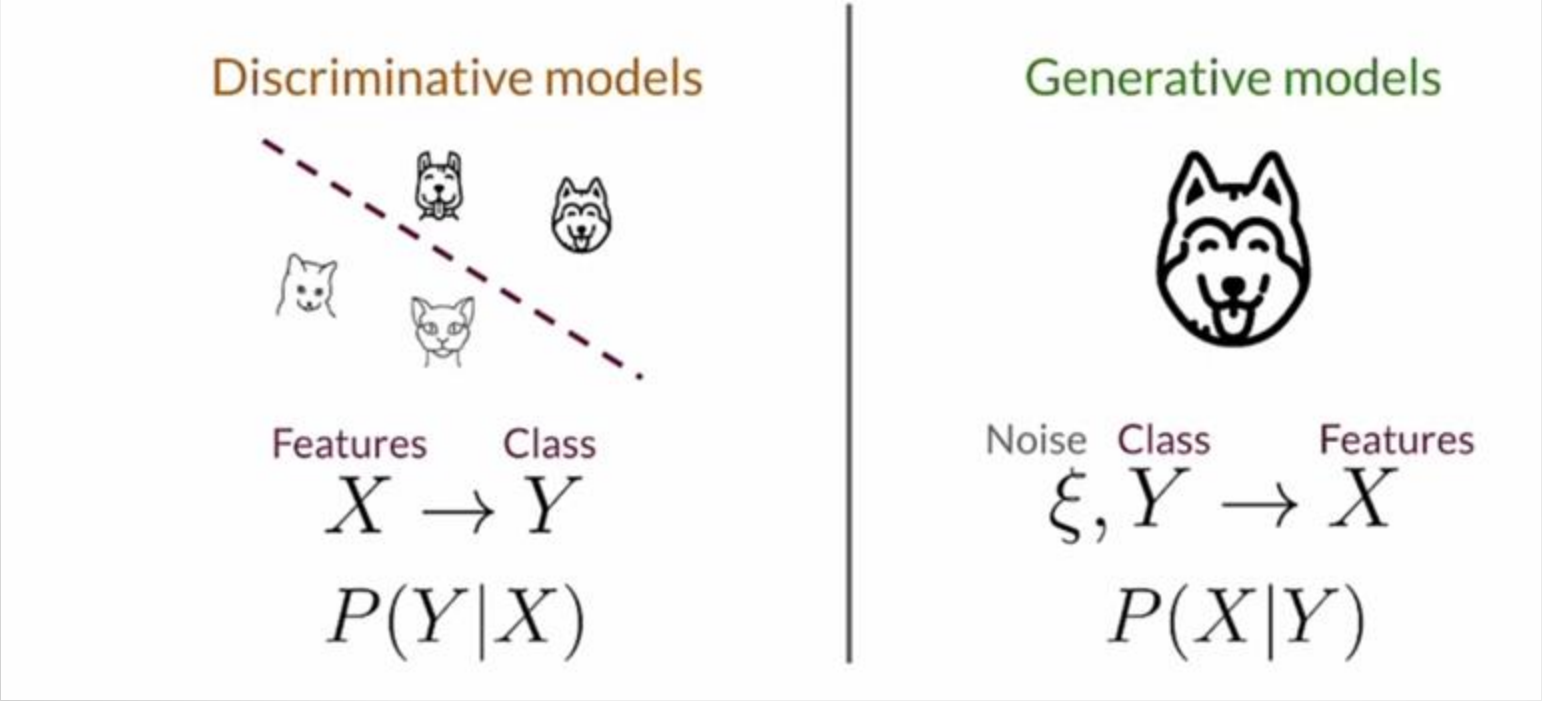
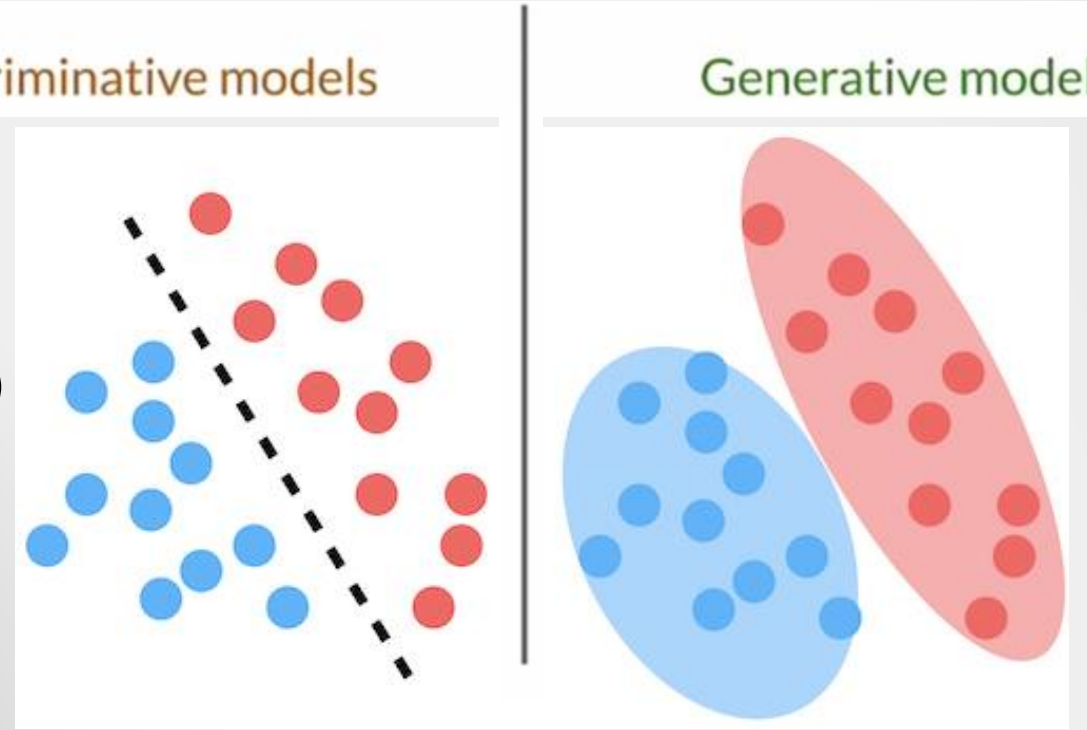


figure by coursera.org

Discriminative vs Generative Models

- Logistic regression
- Scalar Vector Machine
- Neural networks (CNNs)
- etc



- Naïve Bayes
- Bayesian networks
- Markov random fields
- Hidden Markov Models
- Gaussian Mixture Models
- etc

“decision boundary”

“distribution”

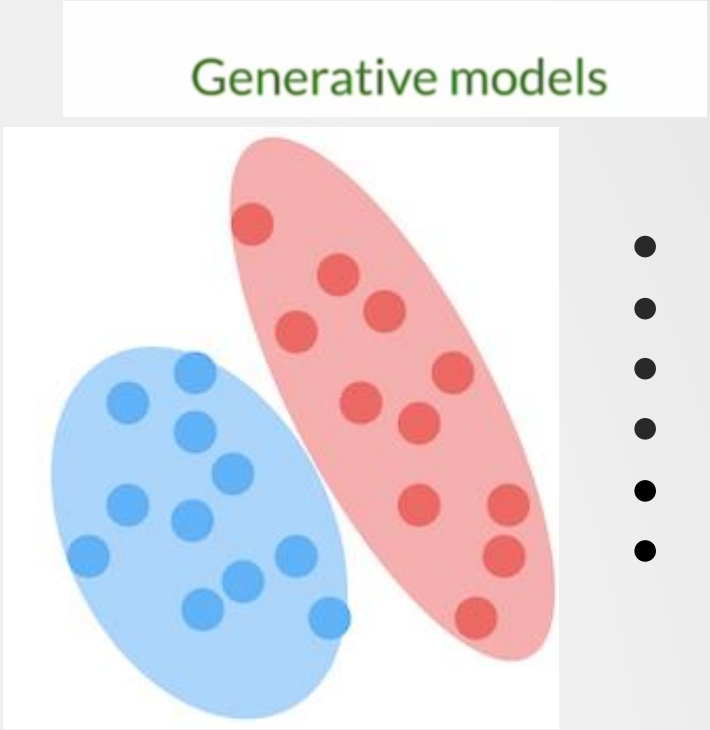
Before the age of GANs, conventional ML scrutinised classifiers into these two categories.

Generative Classifier

A “generative classifier” is a statistical model of the joint probability distribution:

$$P(X,Y)$$

- on given observable variable $\mathbf{X}$
  - Observation $\stackrel{\text{def}}{=}$ Feature
- and target variable $\mathbf{Y}$



“distribution”

- Naïve Bayes
- Bayesian networks
- Markov random fields
- Hidden Markov Models
- Gaussian Mixture Models
- etc

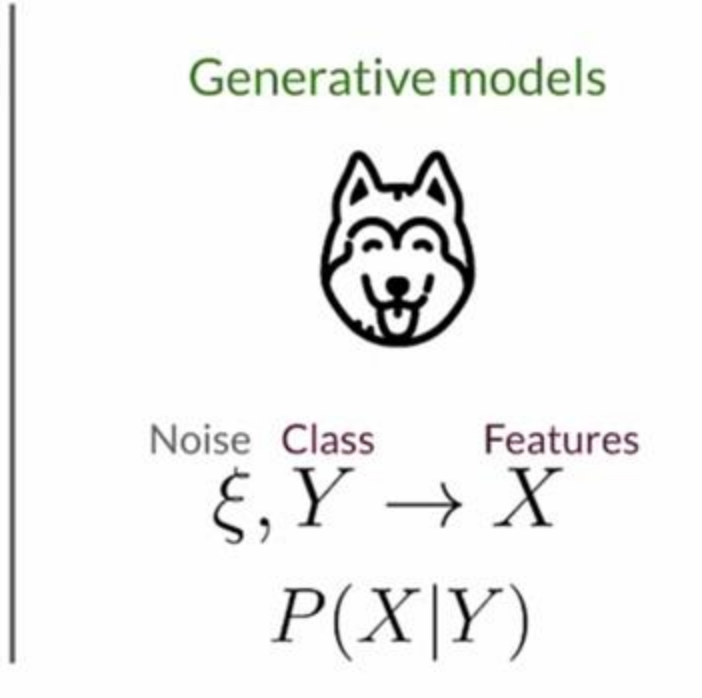
Generative Classifier (vs Generation)

A “generative classifier” is a statistical model of the joint probability distribution:

$$P(X,Y)$$

- on given observable variable $\mathbf{X}$
  - Observation $\stackrel{\text{def}}{=}$ Feature
- and target variable $\mathbf{Y}$

“*When a new observation is fed to a generative classifier, it tries to predict which class would have most likely generated the given observation.*"



“distribution”

- Naïve Bayes
- Bayesian networks
- Markov random fields
- Hidden Markov Models
- Gaussian Mixture Models
- etc

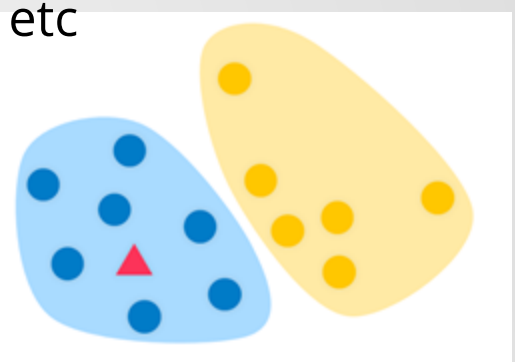
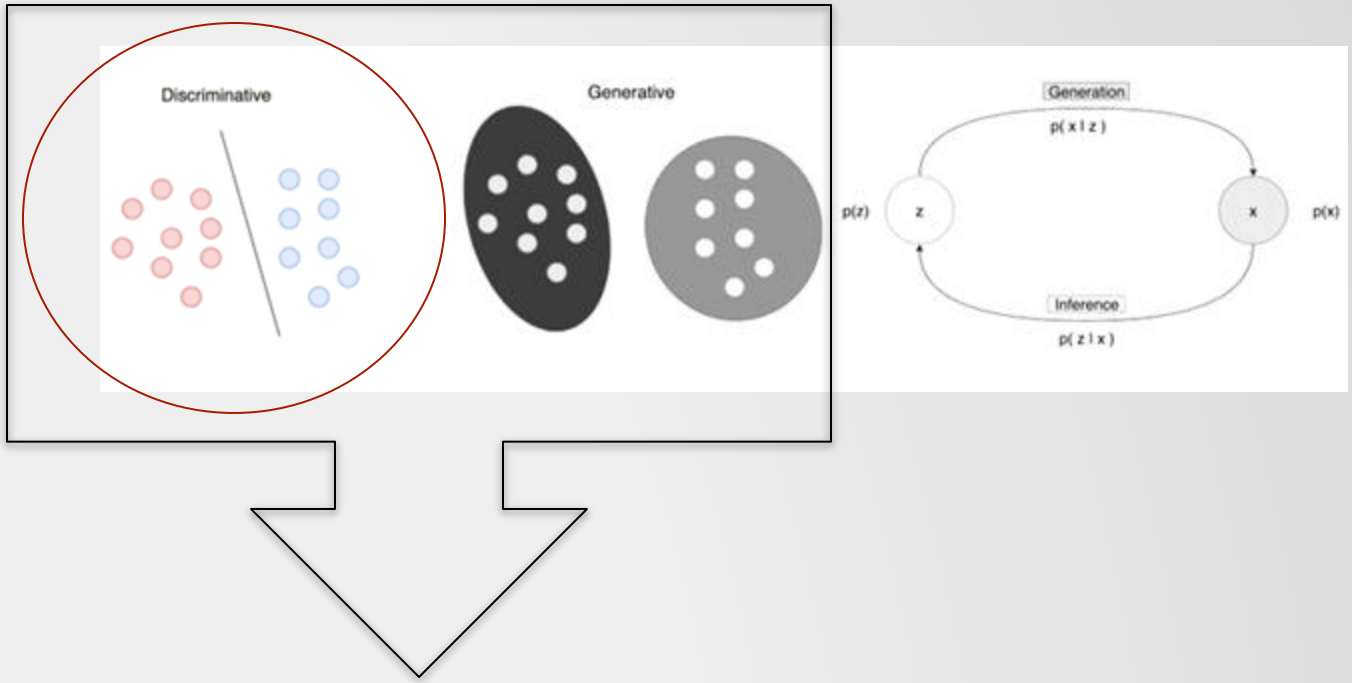


figure by coursera.org

Classification vs Generation

Discriminative models (or classifiers) are machine learning models that learn to classify input data into different categories based on a set of features.



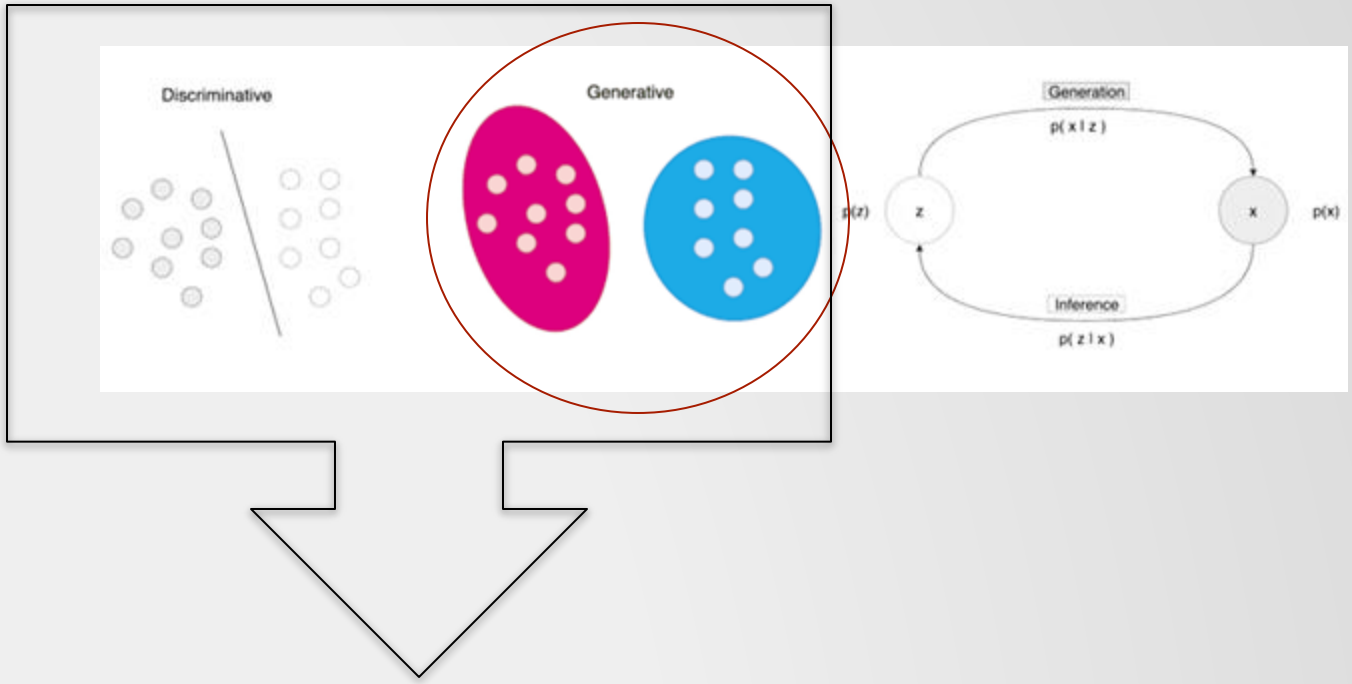
classifiers

These models learn to distinguish between different classes of data **without necessarily modeling the underlying probability distribution of the data**.

figure by coursera.org

Classification vs Generation

Generative classifiers are models that **learn to model the probability distribution of the input data for each class separately, and then use Bayes' rule to calculate the posterior probability of each class given the input**.



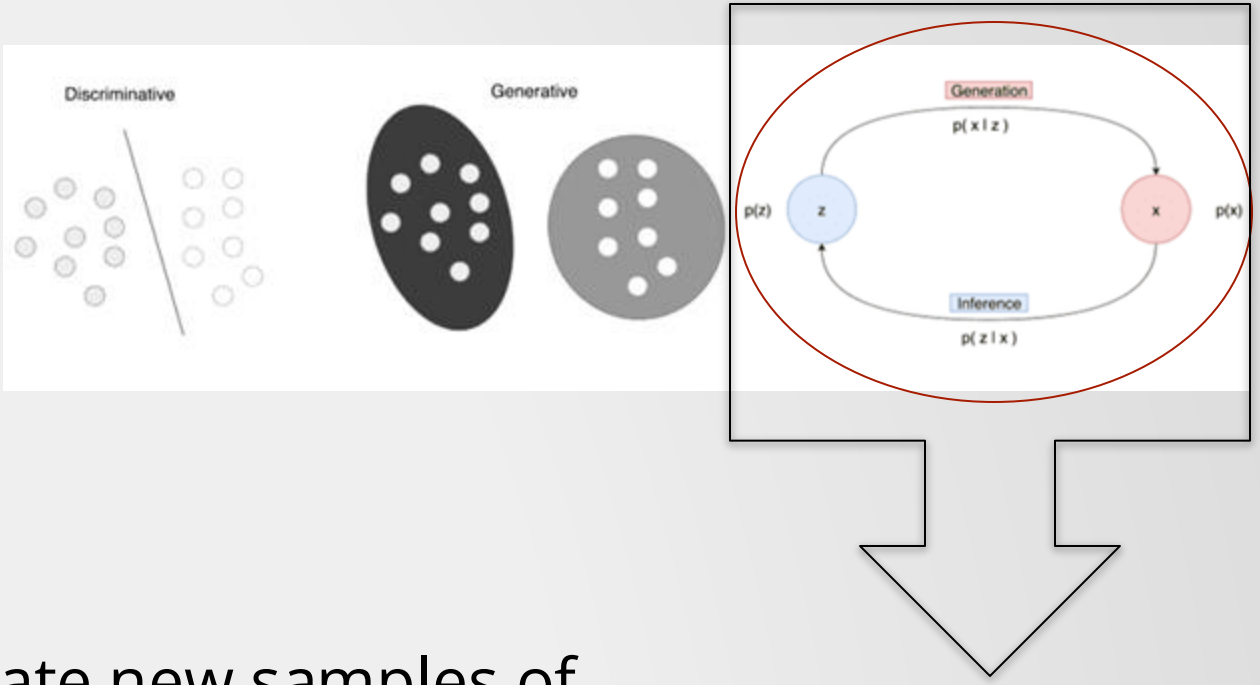
classifiers

These models can be used to classify new data points into different categories based on their likelihood under each class distribution.

figure by coursera.org

Classification vs Generation

Generative (or generation) models are **machine learning models that learn to model the probability distribution of the input data.**



These models can be used to generate new samples of data that resemble the original data distribution.

generators

figure by coursera.org

Generation

- $p(z)$: a noise distribution (i.e. latent space)
- $p(x|z)$: is the generation process, i.e. the generation of a new sample x, given a noise vector z
- $p(x)$: is the data/observation distribution.
- $p(z|x)$: is the inference process, i.e. extracting the noise vector z that would generate the sample x.

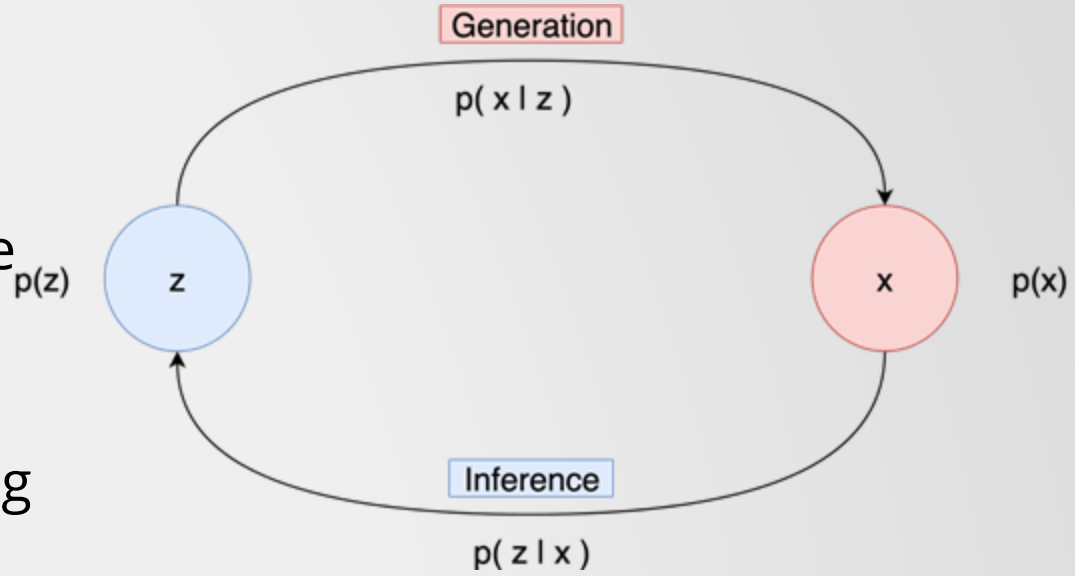


figure by coursera.org

Next lecture:

Part I: “Mathematical Background”

- Generation vs. Discrimination in Machine Learning
- **Data Distribution, Sampling, Inference and Generation**
- Expectation and Likelihood
- Evaluation for Generative Models, Distribution Distances, Divergence and Entropy

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

Lesson 1: Mathematical Background

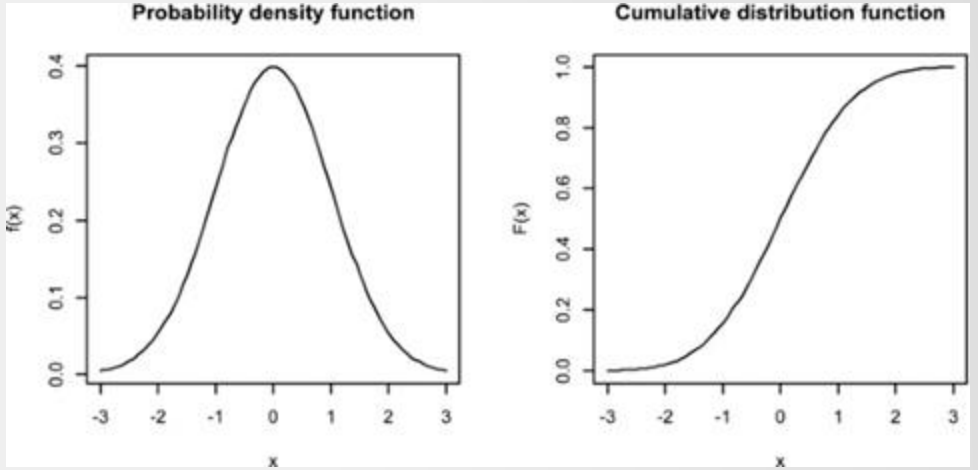
Welcome to Part I: “Mathematical Background”

This part includes four subsections:

- Generation vs. Discrimination in Machine Learning
- **Data Distribution, Sampling, Inference and Generation**
- Expectation and Likelihood
- Evaluation for Generative Models, Distribution Distances, Divergence and Entropy

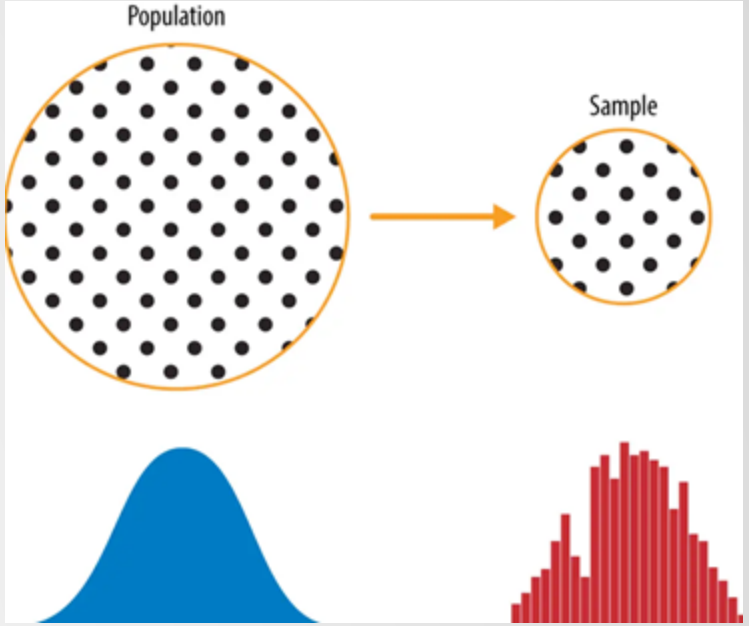
Cumulative Distribution vs Density

- Even though these words can be used interchangeably, in probability theory, they mean different things.
- A cumulative distribution describes the probability of a random variable taking on certain values, while a density function describes the probability of the random variable taking on values in a small interval around a particular point.
- For a continuous distribution, a density function, if it exists, is the derivative of the cumulative distribution.



Sampling

- Sampling is a process of generating random variables from a given distribution
- In generative modelling, sampling is <u>used to generate new samples from a learned distribution</u>
- There are several methods for sampling from probability distributions, including analytical (i.e. GMMs*) and non-analytical methods (i.e. GANs*).



Practice: write a Normal (Gaussian) sampler $g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right).$

Sampling

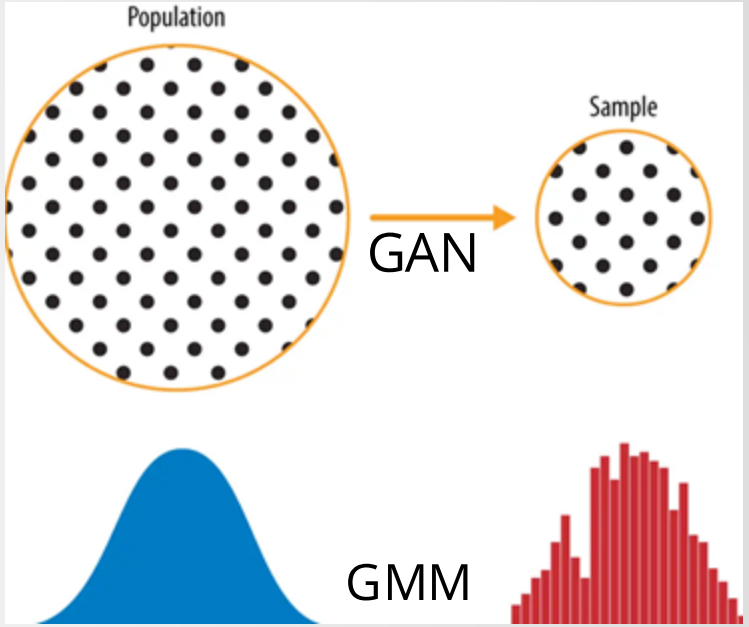
- In generative modeling, sampling refers to the process of generating new data points from a learned distribution.
- One popular approach to generative modeling is through the use of Generative Adversarial Networks (GANs)* .
- Once the GAN has been trained*, we can use it to sample new data points from the learned distribution.

z is a random variable, so that GAN creates a different sample each time

*(which we will learn later)

Sampling

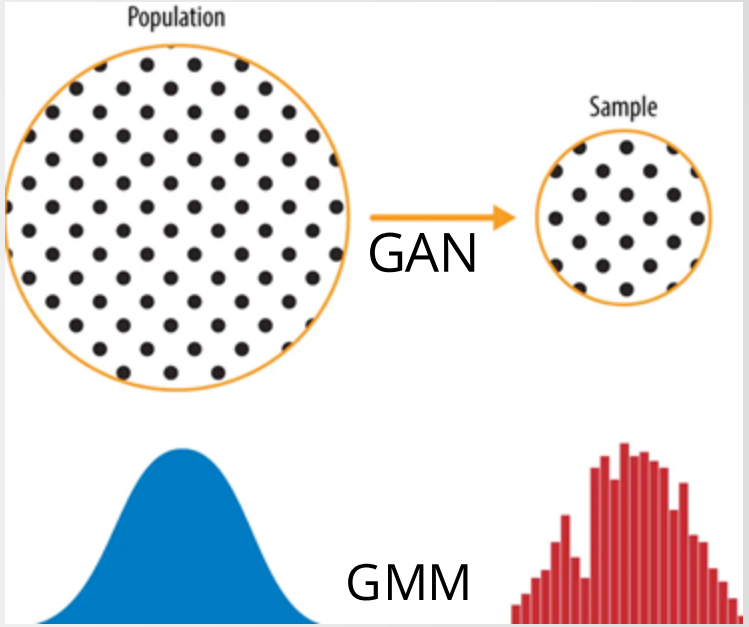
- In some cases, we want to draw samples from a probability distribution that we may not know analytically (like in GANs).
- Or in some other cases, we may know the functional form of the distribution and can use it to generate samples analytically (like in GMMs*)
- The random variable "***z***" is a mathematical construct that captures the randomness in the sampling process.



*(which we will learn later)

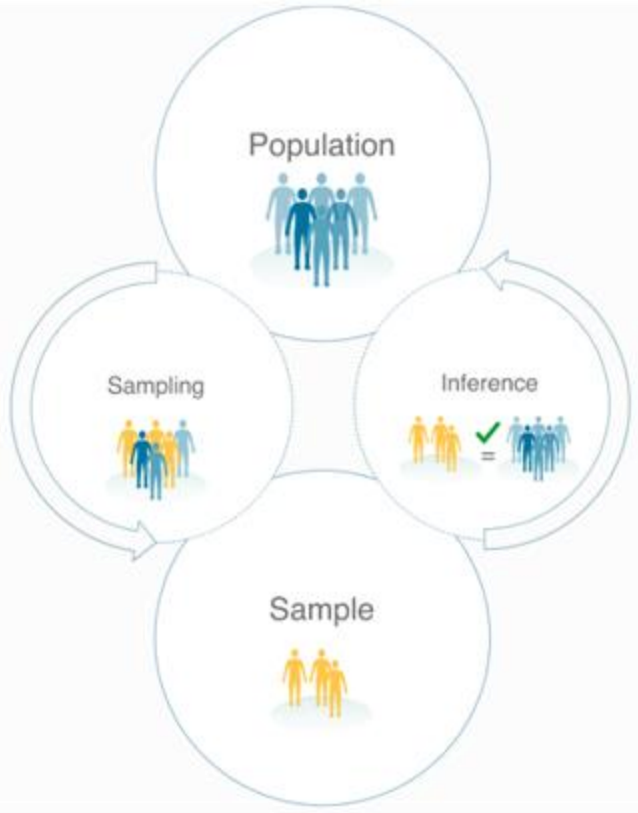
Sampling

- The selection of the input random variable depends on the specific generative model used. In some cases, the input random variable may be uniformly distributed in a specific range, while in other cases, a more complex distribution may be used.
- In Gaussian Mixture Models (GMMs), for example, the input random variable is often chosen from a mixture of Gaussian distributions that approximate the target distribution.
- In Generative Adversarial Networks (GANs), the input random variable is typically chosen from a simple distribution, such as a uniform or normal distribution, and then



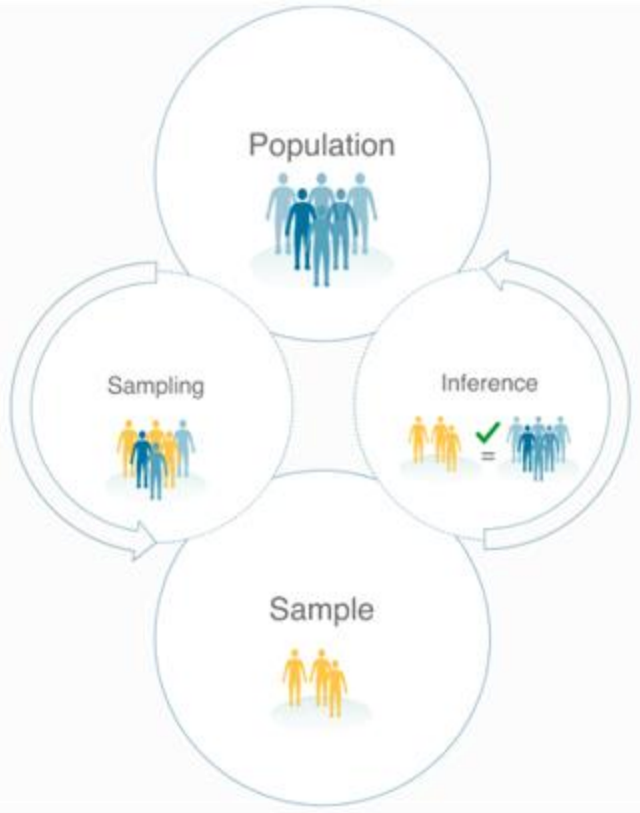
Inference

- So sampling is, when we draw random samples from the probability distribution defined by a model.
- However, in many real-world applications, we often want to do the opposite: given a new data point, we want to infer which model generated it.
- **This process is called inference, and it is the reverse process of sampling.**
- Inference involves using the observed data to update our beliefs about the parameters of the generative model.



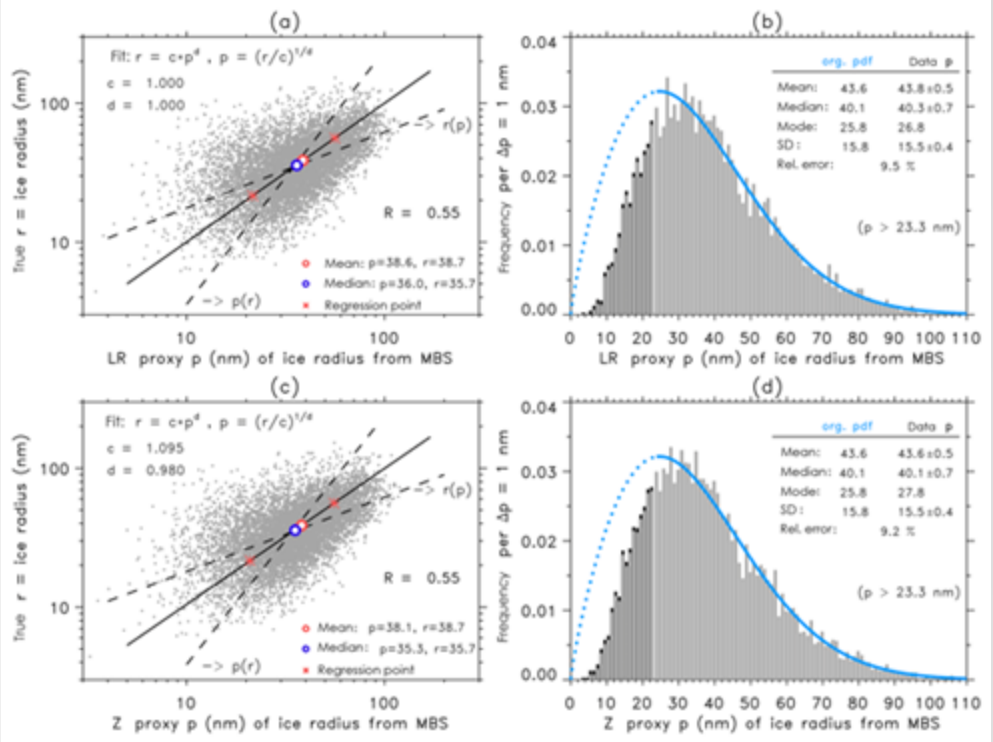
Inference

- In discriminative deep learning models like AlexNet, inference is simply the forward run of the model, where we input a data point and obtain a prediction.
- However, in generative models like GANs, inference is (kind of like, but not necessarily) the reverse run of the model, where we use the observed data to update our belief about the generative process.
  - And also **an integral part of the training process.**



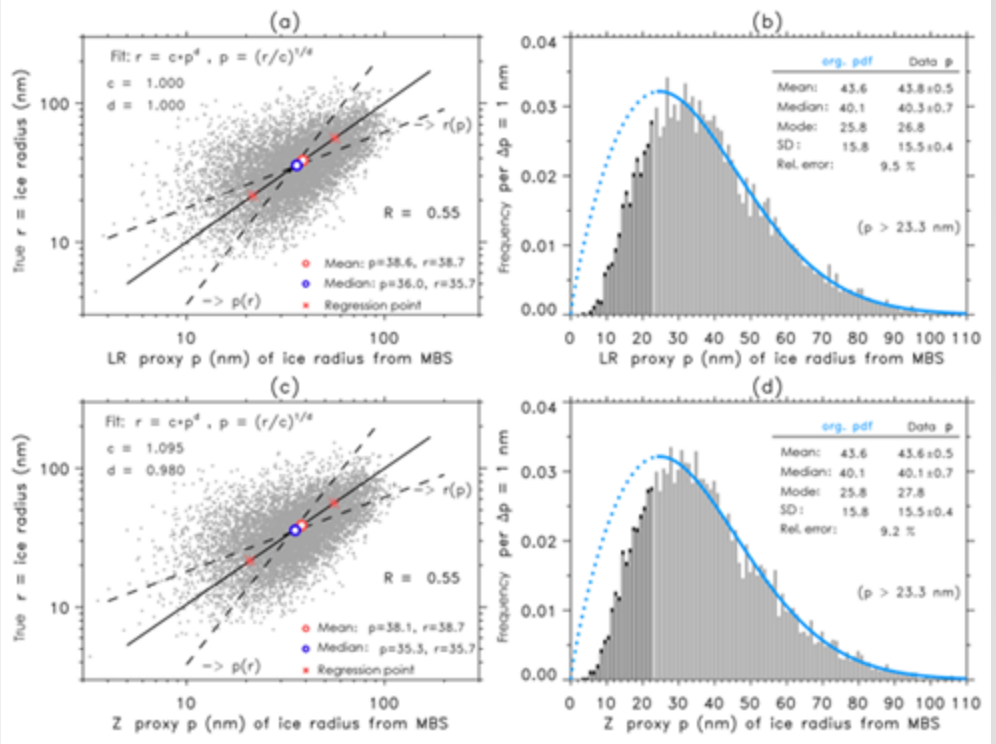
Distribution

- Distribution is a fundamental concept in statistics and probability theory.
- In the context of generative modeling, a distribution is a mathematical function that describes the probability of occurrence of each possible outcome in a given set of outcomes.

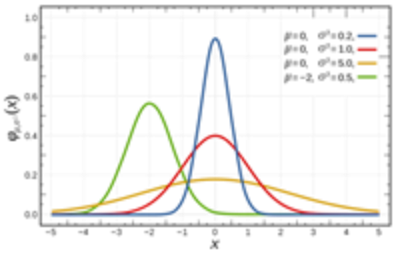


Distribution

- In generative modeling, the aim is to learn the probability distribution of a set of data, so that new data points can be generated from this learned distribution.
- The distribution can be either explicitly defined, as in the case of parametric models such as Gaussian Mixture Models (GMMs); or implicitly defined, as in the case of GANs. (remember previous week, but it was called density ?!)



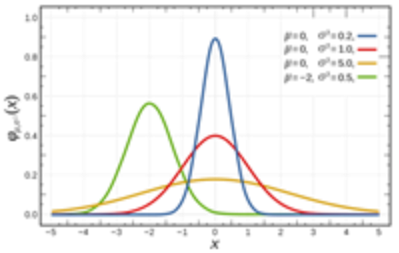
Gaussian Distribution



- The Gaussian distribution, also known as the normal distribution, is a probability distribution that describes how a continuous variable is likely to be distributed.
- It is characterized by two parameters: the mean (μ) and the standard deviation (σ).
- The Gaussian function has a bell-shaped curve, with the peak at the mean.
- The Gaussian distribution is widely used in statistics, machine learning, and other fields because of its mathematical properties and applicability to real-world phenomena.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

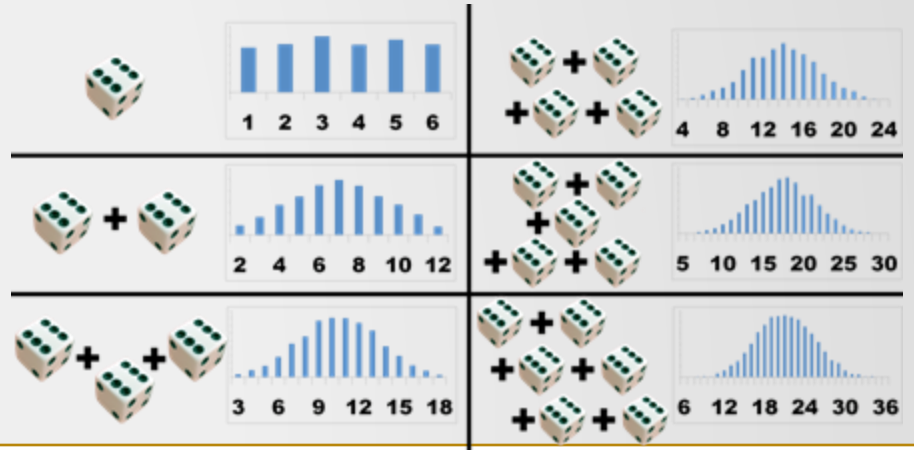
Gaussian Distribution



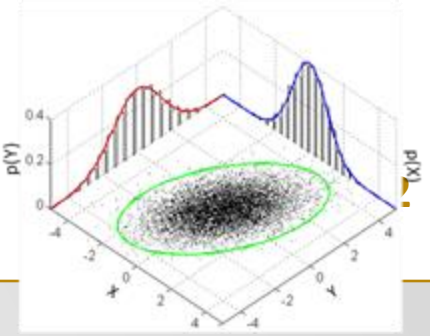
- The central limit theorem states that the sum of independent and identically distributed random variables approaches a Gaussian distribution as the number of variables increases.
- In practice, many real-world phenomena can be modeled as a sum of multiple small contributions, which leads to a Gaussian distribution according to the central limit theorem.

The central limit theorem was first discovered by the French mathematician Abraham de Moivre in the early 18th century.

However, the modern formulation of the theorem is attributed to the French mathematician Pierre-Simon Laplace in the late 18th and early 19th centuries.



Multivariate Gaussian Distribution



- In many real-world applications, we need to model data that has more than one dimension or feature.
- The multivariate Gaussian distribution is a generalization of the univariate Gaussian distribution to multiple dimensions.
- It is characterized by a mean vector (μ) and a covariance matrix (Σ) that describe the location and spread of the distribution in each dimension.
- The covariance matrix contains information about the correlations between the different features.

$$p(\boldsymbol{x}) = \frac{1}{\left|\sqrt{2\pi\boldsymbol{\Sigma}}\right|} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

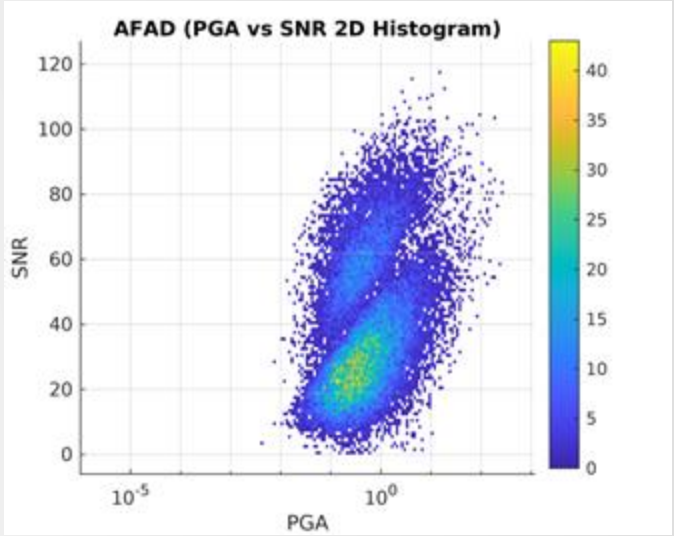
- Mean: $\boldsymbol{\mu}$ (vector 2x1)
- Covariance: $\boldsymbol{\Sigma}$ (matrix 2x2)

Modality

- Modality refers to the number of modes or peaks in a distribution.
- Unimodal distributions have a single mode or peak, while multimodal distributions have multiple modes or peaks.
- The number of modes in a distribution can be an important characteristic for understanding the underlying data.

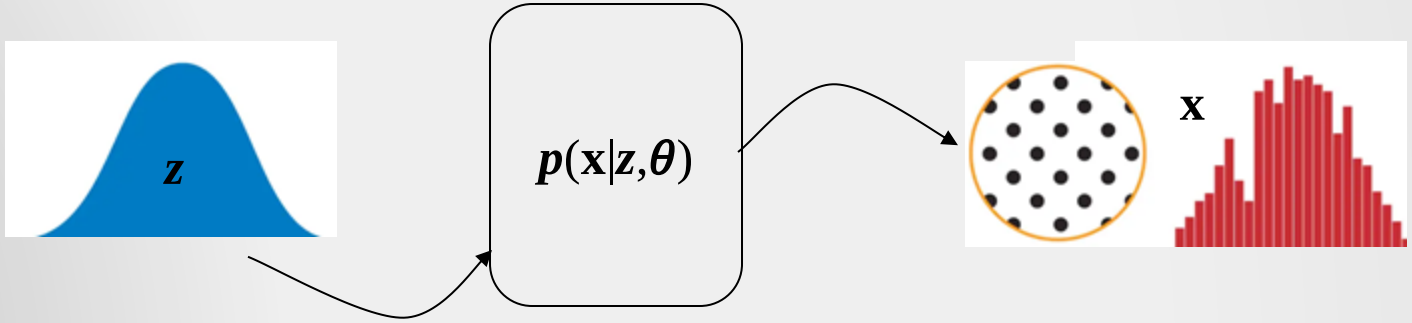
Peak ground acceleration (**PGA**) is equal to the maximum ground acceleration that occurred during earthquake shaking at a location

SNR is defined as the ratio of signal power to the noise power, often expressed in decibels



Generation and Distributions

- So, what is “generation” and why is it related to distribution?
- Is generation a stochastic process?
- If so, is the output of a generative function always a distribution?
- Are generative models always probability distributions?
- Crazy questions in my head…



Next lecture:

Part I: “Mathematical Background”

- Generation vs. Discrimination in Machine Learning
- Data Distribution, Sampling, Inference and Generation
- **Expectation and Likelihood**
- Evaluation for Generative Models, Distribution Distances, Divergence and Entropy

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

Lesson 1: Mathematical Background

Welcome to Part I: “Mathematical Background”

This part includes four subsections:

- Generation vs. Discrimination in Machine Learning
- Data Distribution, Sampling, Inference and Generation
- **Expectation and Likelihood**
- Evaluation for Generative Models, Distribution Distances, Divergence and Entropy

The theorem is named after Reverend Thomas Bayes, an 18th-century British statistician and theologian.

Bayes Theorem

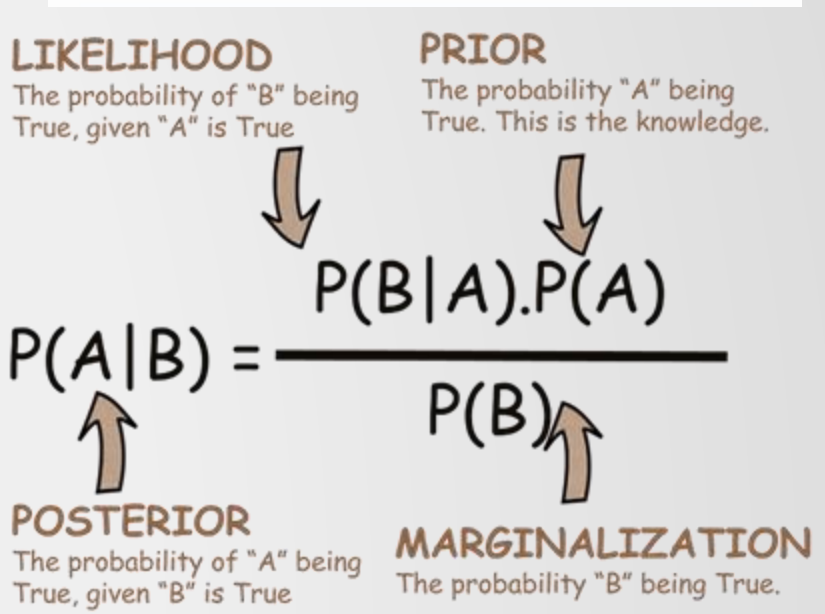
- Bayes' Theorem is a fundamental concept in probability theory that describes the probability of an event based on prior knowledge of related events.
- It provides a way to update our beliefs about the probability of an event as new evidence is obtained.

Independent Events

$$P(X \cap Y) = P(X) \bullet P(Y)$$

Dependent Events

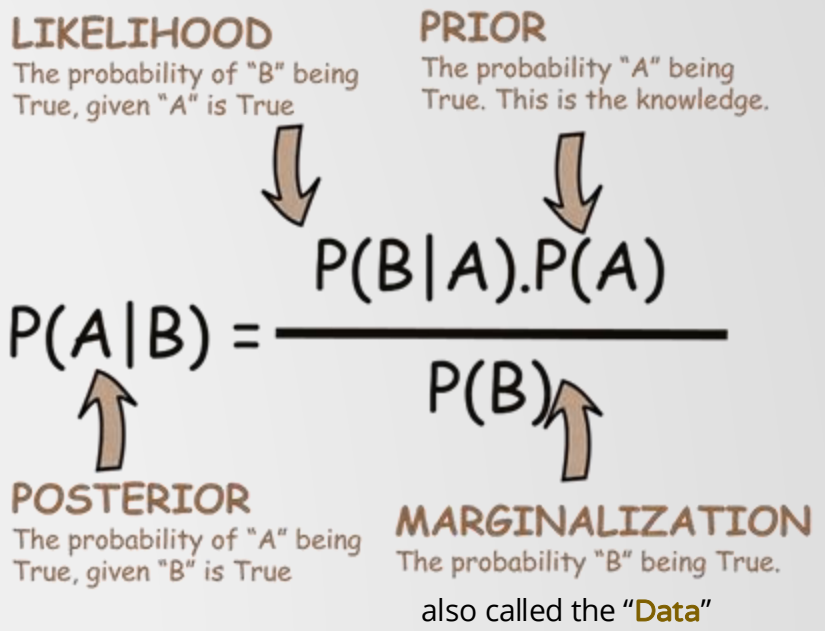
$$P(X \cap Y) = P(Y) \bullet P(X \mid Y)$$



The theorem is named after Reverend Thomas Bayes, an 18th-century British statistician and theologian.

Bayes Theorem

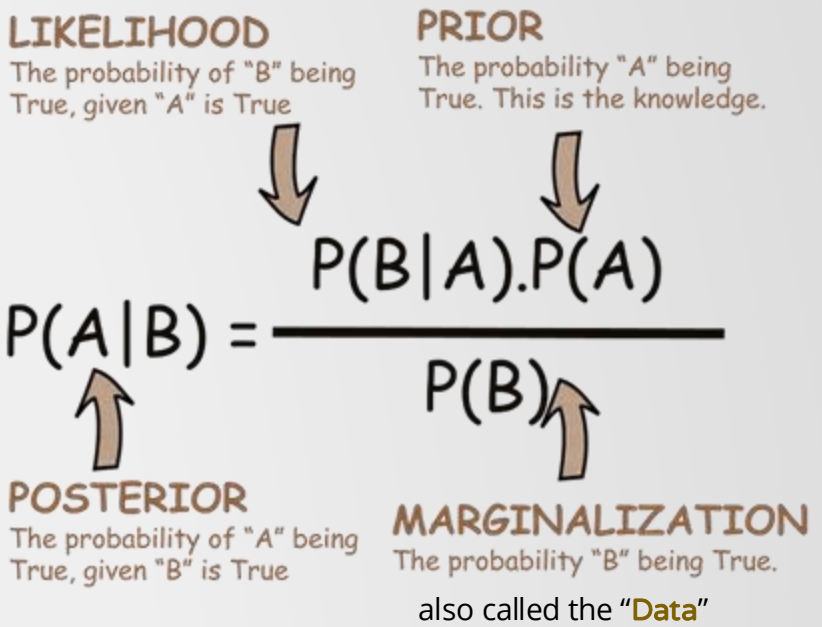
- Bayes Theorem plays a crucial role in generative models, which are used to learn the underlying probability distributions of a given dataset.
- Bayes Theorem provides a way to update our prior beliefs about the parameters of a probability distribution in light of new evidence (i.e., data).
- In the context of generative models, the theorem is used to estimate the parameters of the underlying distribution that generated the data.
- Specifically, it helps us to update our prior beliefs about the parameters of the distribution based on the observed data.



The theorem is named after Reverend Thomas Bayes, an 18th-century British statistician and theologian.

Bayes Theorem

- Bayes Theorem plays a crucial role in generative models, which are used to learn the underlying probability distributions of a given dataset.
- Bayes' Theorem is widely used in deep learning-based generative models, such as GANs, VAEs, and Bayesian neural networks.
- In GANs, for example, the discriminator network can be seen as an approximate likelihood function, and the generator network is used to generate samples from the learned posterior distribution over the latent variables.

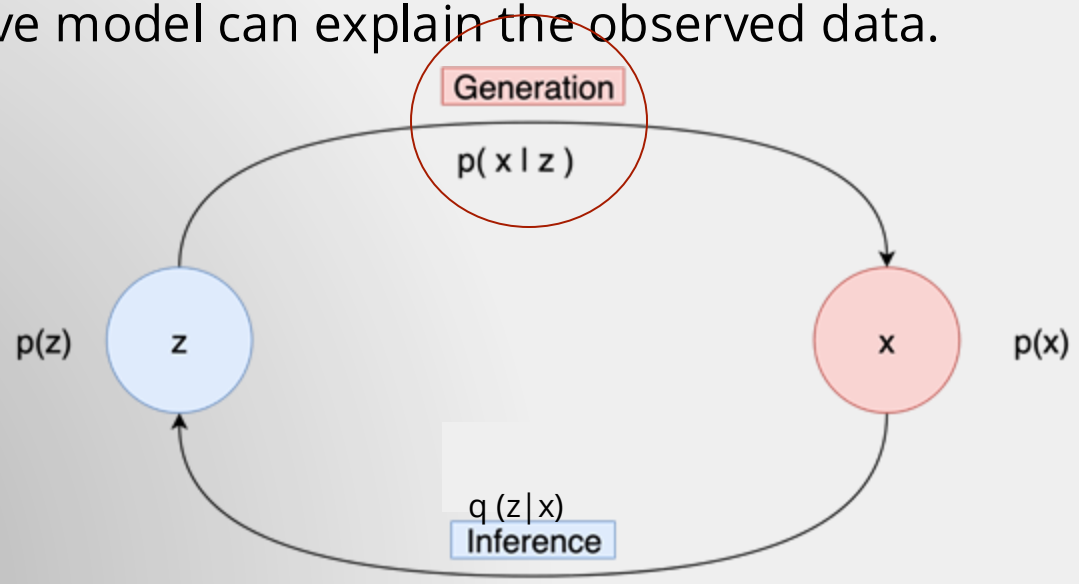


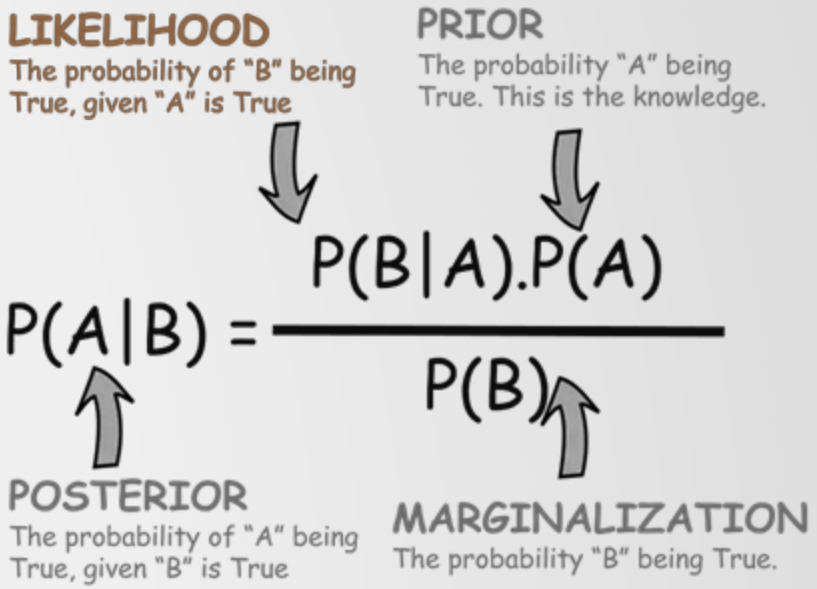
also called the "Data"

Likelihood?

In the context of generative models, the likelihood refers to **the probability of observing a given set of data points under the assumed probability distribution of the generative model.**

In other words, the likelihood measures how well the generative model can explain the observed data.





Maximizing the Likelihood

- Maximizing the likelihood involves finding the set of model parameters θ that maximize the likelihood function (i.e. the Generator)

  For example, by using the log-likelihood loss function in a deep generative model to maximize the likelihood of the model parameters given the data.

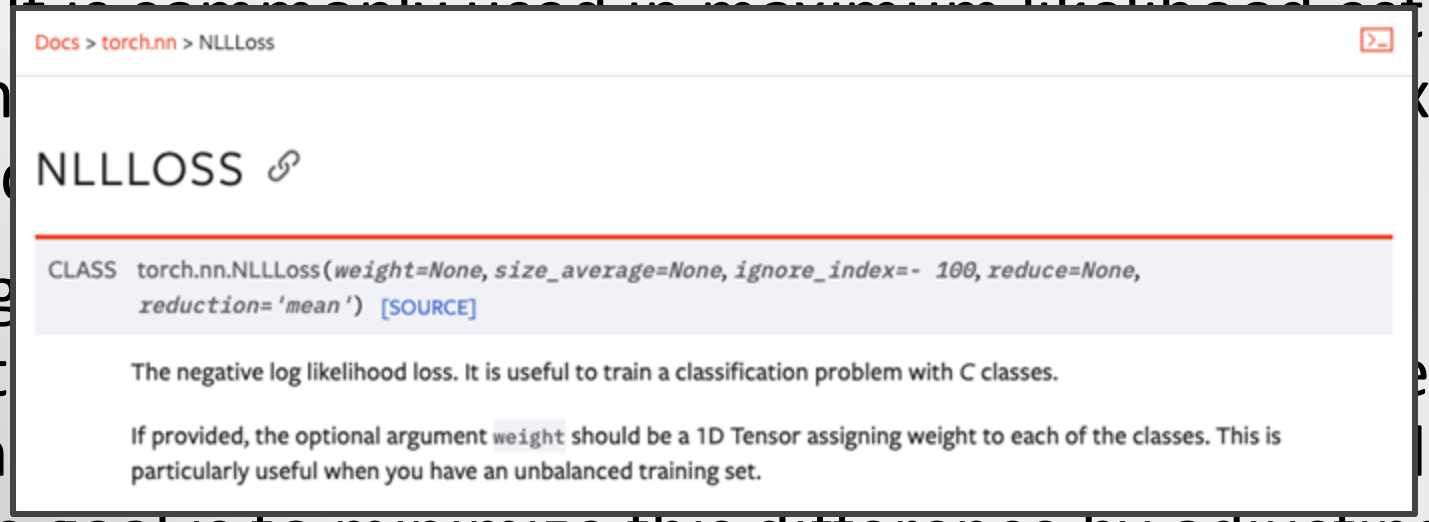
- In deep generative models, the likelihood function is often intractable or difficult to optimize directly.

  For example, in VAEs, a lower bound on the likelihood is optimized instead, while in GANs, a game-theoretic objective is used to implicitly maximize the likelihood.

- Maximizing likelihood is a common objective in deep generative models.

Log-likelihood Loss function

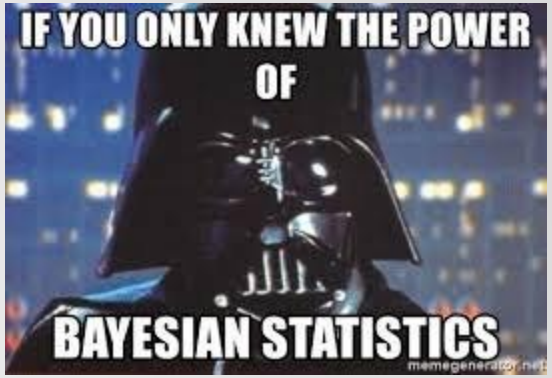
- Log-likelihood is a measure of how well a statistical model fits the data it is given. It is commonly used in maximum likelihood estimation (MLE), where th ximize the likelihoo
- In deep g a loss function to train t e difference between listribution of the data. The goal is to minimize this difference by adjusting the model's parameters.



Next lecture:

Part I: “Mathematical Background”

- Generation vs. Discrimination in Machine Learning
- Data Distribution, Sampling, Inference and Generation
- Expectation and Likelihood
- **Evaluation for Generative Models, Distribution Distances, Divergence and Entropy**



Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

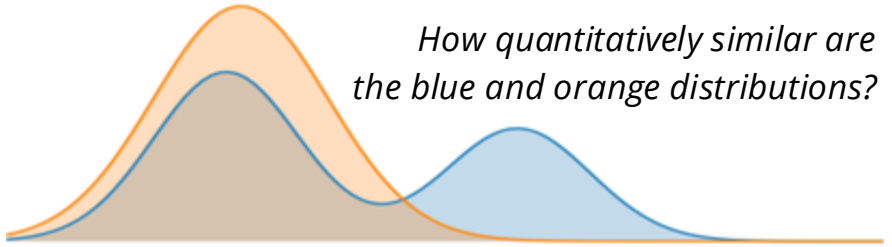
Lesson 1: Mathematical Background

Welcome to Part I: “Mathematical Background”

This part includes four subsections:

- Generation vs. Discrimination in Machine Learning
- Data Distribution, Sampling, Inference and Generation
- Expectation and Likelihood
- **Evaluation for Generative Models, Distribution Distances, Divergence and Entropy**

Distribution Distances



EURO 2

- Why do we need to compare distributions?
  - In generative modeling, it is important to compare different probability distributions to determine how well our model is performing.
  - For instance, we need to be able to evaluate the similarity between the true data distribution and the distribution learned by our generative model.
- “Divergence”
  - is a way to measure the distance between two probability distributions.
  - measures quantify how much one distribution differs from another in terms of their shapes, locations, or other characteristics.

Divergence (first)

- Not all distance measures between two distributions are “divergence” measures, but we will start with them first.
- Some of which, we may benefit from in this course, are:
  - Kullback-Leibler (KL) divergence
  - Jensen-Shannon (JS) divergence
  - Total Variation (TV) distance
  - Hellinger distance

Kullback-Leibler (KL) Divergence

- The KL divergence is a measure of the difference between two probability distributions, P and Q.
- It is defined as the expected value of the logarithmic difference between P and Q, where the expectation is taken with respect to P. The KL divergence is denoted as D(P||Q).

$$D_{KL}(P||Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)}$$

Kullback-Leibler (KL) Divergence

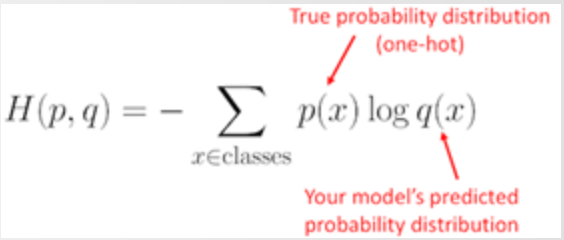
- The KL divergence is always non-negative, and it is zero if and only if the two distributions P and Q are identical.
- The KL divergence is not symmetric, meaning that D(P||Q) is not necessarily equal to D(Q||P).

$$D_{KL}(P||Q) \neq D_{KL}(P||Q)$$

- The KL divergence can be interpreted **as the amount of information lost when using Q to approximate P** (or vice versa). It measures the additional number of bits of information needed to specify P instead of Q.

Kullback-Leibler (KL) Divergence

- The KL divergence is commonly used in generative modeling to measure the similarity between the true data distribution and the distribution learned by a generative model.
- It is often used as a loss function to train generative models, such as Variational Autoencoders (VAEs), which aim to learn a lower-dimensional representation of the data that can be used to generate new samples.
- Did KL remind you of something?
  - Cross-entropy maybe?



KL vs CE

- Cross-entropy is a measure of the dissimilarity between two probability distributions, typically between a true distribution and an estimated distribution.

$$H(P,Q) = -\sum_{x} P(x) \log Q(x)$$

- KL divergence measures the divergence between two probability distributions, P and Q, by measuring the additional number of bits of information needed to specify P instead of Q.

$$KL(P|Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

KL vs CE

- Both KL divergence and cross-entropy are used to measure the similarity or dissimilarity between two probability distributions.
- Both measures are commonly used in generative modeling to evaluate the performance of generative models and to optimize their parameters.
- Both measures are non-negative and minimize to zero if and only if the two distributions are identical.
- Both measures are asymmetrical.

$$H(P,Q) = -\sum_{x} P(x) \log Q(x)$$

$$KL(P|Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

KL vs CE

- KL Divergence focuses on the additional information needed to be accurate about the true distribution when starting with an approximation. It emphasizes the "gap" between the true distribution and the approximation.
- Cross Entropy is more about the efficiency of encoding events from the true distribution when using the code optimized for an approximation. It looks at the average number of bits needed and emphasizes the cost of using the code optimized for

$$H(P,Q) = -\sum_{x} P(x) \log Q(x)$$

$$KL(P|Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

Jensen-Shannon (JS) Divergence

- The JS divergence is a symmetric measure of the difference between two probability distributions.
- It is a smoothed version of the KL divergence, which can be used to compare two probability distributions that may have disjoint support.

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$$

Jensen-Shannon (JS) Divergence

- The JS divergence is a symmetric measure of the difference between two probability distributions.
- It is a smoothed version of the KL divergence, which can be used to compare two probability distributions that may have disjoint support.
- The JS divergence is bounded by 0 and 1, and is equal to 0 if and only if the two distributions are identical.

$$0 \leq \mathrm{JSD}(P \parallel Q) \leq 1$$

- The JS divergence are used as a loss function to train generative models, such as Generative Adversarial Networks (GANs).

Information Theory

- Information theory was initially developed to understand and improve communication systems, especially in the context of telegraphy and radio.
- However, its scope has expanded to various other areas such as data compression, cryptography, and more recently, deep learning and generative models.
- In the context of deep generative models, information theory provides a theoretical framework for understanding and designing models that can generate high-quality and diverse samples from complex distributions.

Entropy & Information

- Entropy is a measure of uncertainty or disorder in a random variable, while information is the reduction of uncertainty or surprise gained from an event.
- In deep generative models,
  - the entropy of the output distribution can be used to measure the complexity of the generated samples,
  - while information can be used to measure the amount of information captured in the learned representation.
- **<u>They are formulated measures!</u>**

Shannon's Entropy

- Introduced by Claude Shannon in 1948 as a measure of uncertainty or information content in a random variable or probability distribution, defined as the expected value of the information contained in each possible outcome, given the probability distribution:

$$H(X) = -\sum_{i} P(x_i) \log P(x_i)$$

- maximized when all outcomes are equally likely (i.e., maximum uncertainty)
- minimized when there is only one possible outcome (i.e., no uncertainty)

Conditional Entropy

- is the amount of uncertainty remaining in a random variable given that another random variable has been observed or known.

$$H(X) = -\sum_i P(x_i) \log P(x_i)$$

$$\longrightarrow$$

$$\mathrm{H}(Y|X) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)}$$

- Conditional entropy tells us how much information we gain about Y by observing X. If conditional entropy is high, it means that observing X gives us little information about Y, and vice versa.

Entropy and Generative Models

- Entropy measures such as Shannon entropy and differential entropy are used to quantify the uncertainty or randomness in the generated samples.
- In a well-trained generative model, the generated samples should have high entropy, indicating that the model is able to produce a diverse set of samples that capture the variability in the training data.
    - Example: In a generative adversarial network (GAN), the generator tries to generate samples that fool the discriminator. The entropy of the generated samples can be used to measure the diversity and quality of the samples generated by the GAN.

Shannon's "Self-Information"

- The amount of information gained by an event with probability p is defined as:

$$\mathrm{I}(x) := -\log_b [\Pr(x)] = -\log_b (P).$$

- Shannon's definition of self-information meets several axioms:
  - An event with probability 100% is perfectly unsurprising and yields no information.
  - The less probable an event is, the more surprising it is and the more information it yields.
  - If two independent events are measured separately, the total amount of information is the sum of the self-informations of the individual events

Mutual Information

- is a measure of the amount of information that two variables share.
- MI quantifies the reduction in uncertainty about one variable given knowledge of the other variable, and can be represented using the Entropy:

$$\begin{aligned} \mathrm{I}(X;Y) &\equiv \mathrm{H}(X) - \mathrm{H}(X \mid Y) \\ &\equiv \mathrm{H}(Y) - \mathrm{H}(Y \mid X) \\ &\equiv \mathrm{H}(X) + \mathrm{H}(Y) - \mathrm{H}(X, Y) \\ &\equiv \mathrm{H}(X, Y) - \mathrm{H}(X \mid Y) - \mathrm{H}(Y \mid X) \end{aligned}$$

  - Example: MI can be used as a regularizer to encourage disentanglement of the latent variables in the learned representation.

Information and Generative Models

- Information measures such as mutual information and conditional entropy are used to evaluate the ability of the generative model to capture the underlying structure of the data.
- In a well-trained generative model, the mutual information between the generated samples and the training data should be low, indicating that the generated samples are not duplicating the training data.
    - Example: In a variational autoencoder (VAE), the encoder tries to compress the input data into a low-dimensional latent space. The mutual information between the latent space and the input data can be used to measure the amount of information that is preserved in the latent space (like a regularizer to encourage disentanglement of the latent variables in the learned representation)

Next lecture:

- **PART II: “Latent Spaces”**

- **(The Curse of) Dimensionality, Deep Features vs. Latent Spaces**
- Latent Space properties, Continuity, Entanglement, etc

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

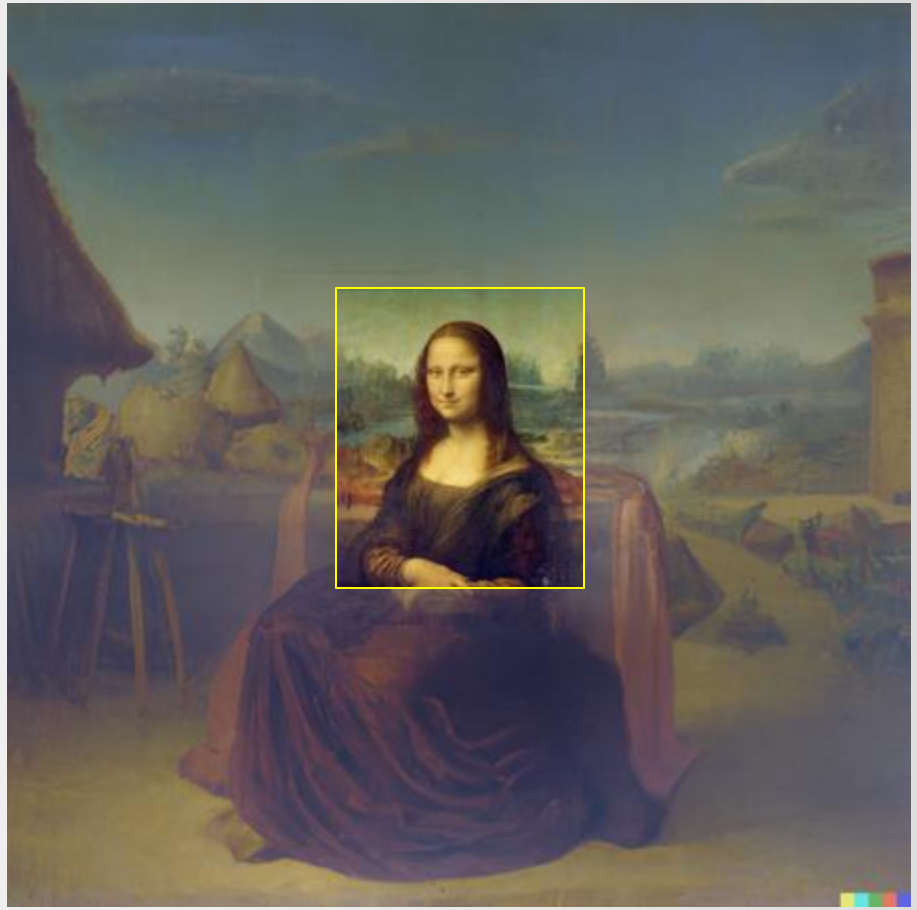
Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

Lesson 2: Latent Spaces

Welcome to Part II: “Latent Spaces”

This part includes two subsections:

- **(The Curse of) Dimensionality, Deep Features vs. Latent Spaces**
- Latent Space properties: Continuity, Entanglement, etc



Dimensionality

In the context of generative models (and deep learning), dimensionality refers to the number of input features or variables that describe the input.

High-dimensional data poses challenges such as increased computational complexity and difficulty in understanding and visualizing the data.

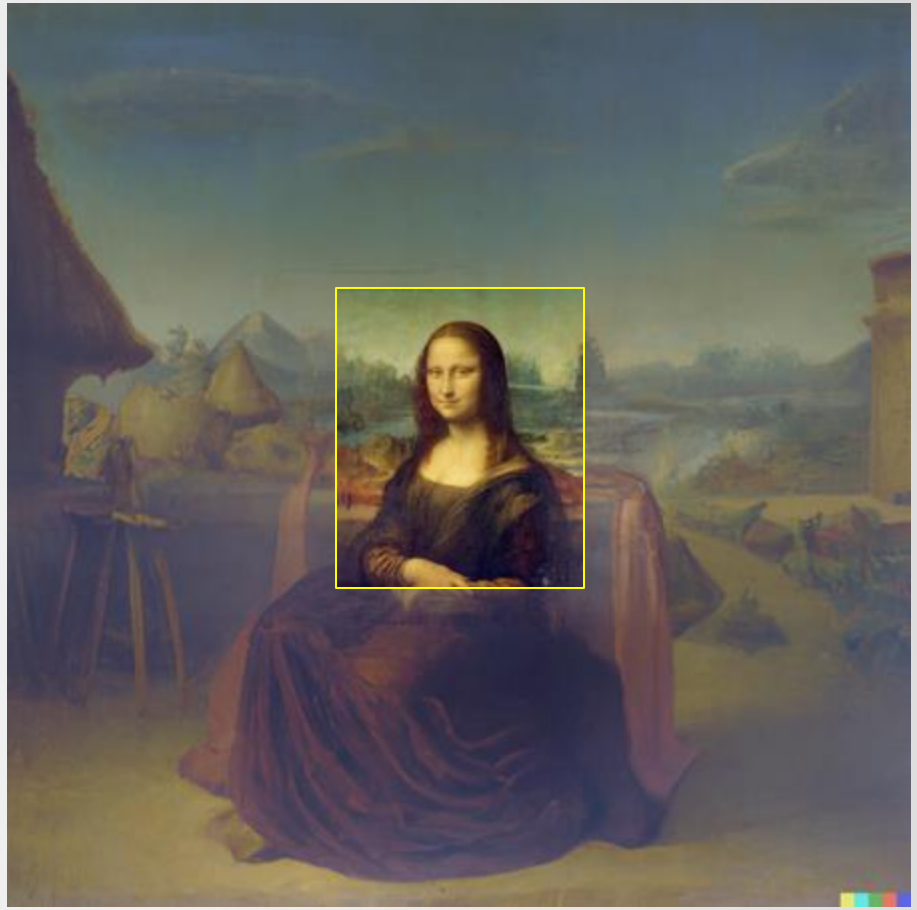


Image by openai.org

Dimensionality

Dimensionality reduction techniques play a vital role in mitigating these challenges by reducing the number of dimensions while preserving important information.

We'll dive into the motivations, techniques, and benefits of dimensionality reduction in generative modeling. This will lead us to the idea of a "latent space".

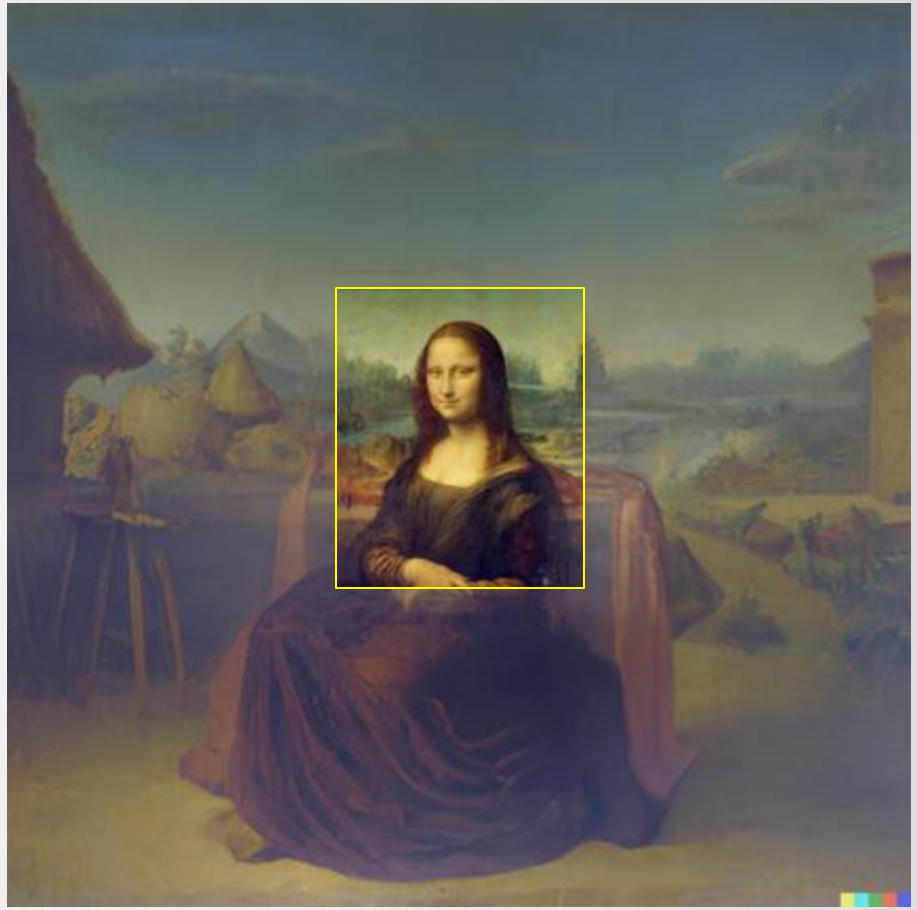
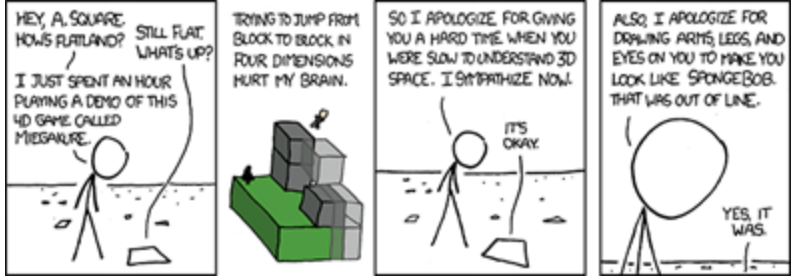


Image by openai.org

Curse of Dimensionality



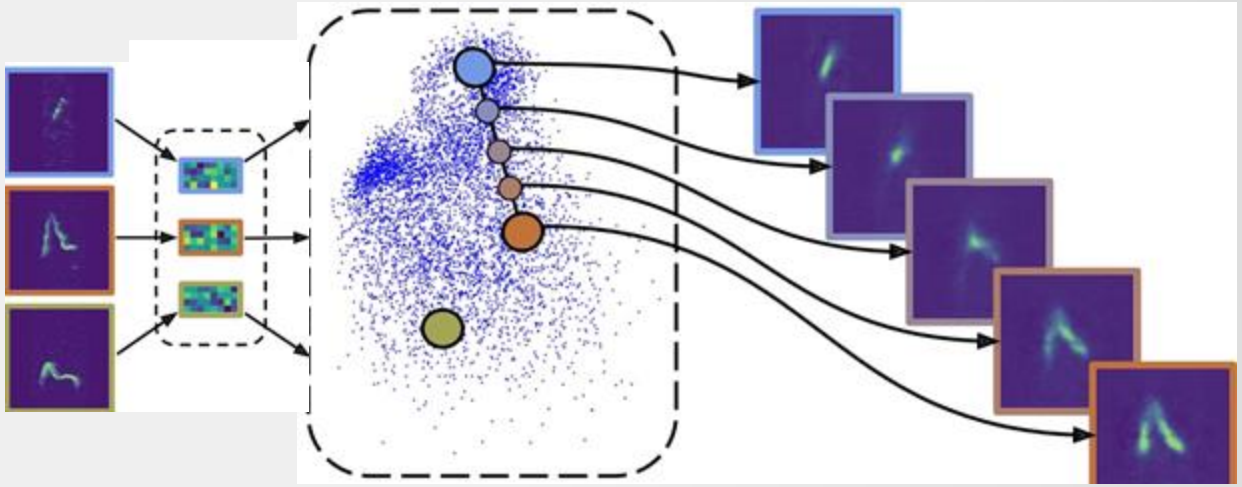
- In generative modeling (and in machine learning general), the curse of dimensionality refers to the challenges that arise when working with high-dimensional data.
- High-dimensional data refers to datasets with a large number of features or variables that describe each data point.
- CoD causes many problems in generative modelling such as “mode collapse” in GANs (which we’ll learn later)
- The curse of dimensionality necessitates the need for dimensionality reduction techniques to address these challenges.

figure by miegakure.com

Sampling from a Vector Space

- In generative modeling, we require a intermediate vector space that is both **continuous** and **representative** of the underlying data distribution for effective sampling from a generator.
- A continuous vector space enables smooth interpolation and exploration of the data, allowing for **seamless transitions** and generation of new samples.

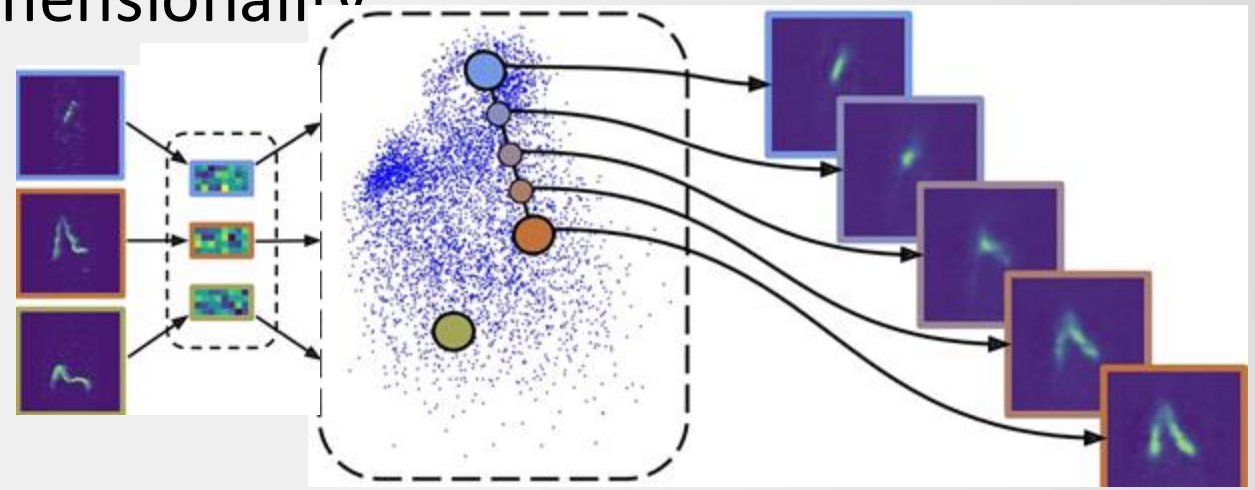
figure from https://doi.org/10.7554/eLife.67855



Sampling from a Vector Space

- Representativeness implies that similar samples or concepts in the original data space should be close to each other in the vector space, facilitating accurate modeling and sampling.
- However, directly working with the high-dimensional data space poses challenges in achieving a continuous and representative vector space due to the curse of dimensionality.
- We need techniques that maps high-dimensional data to a lower-dimensional vector space.

figure from https://doi.org/10.7554/eLife.67855

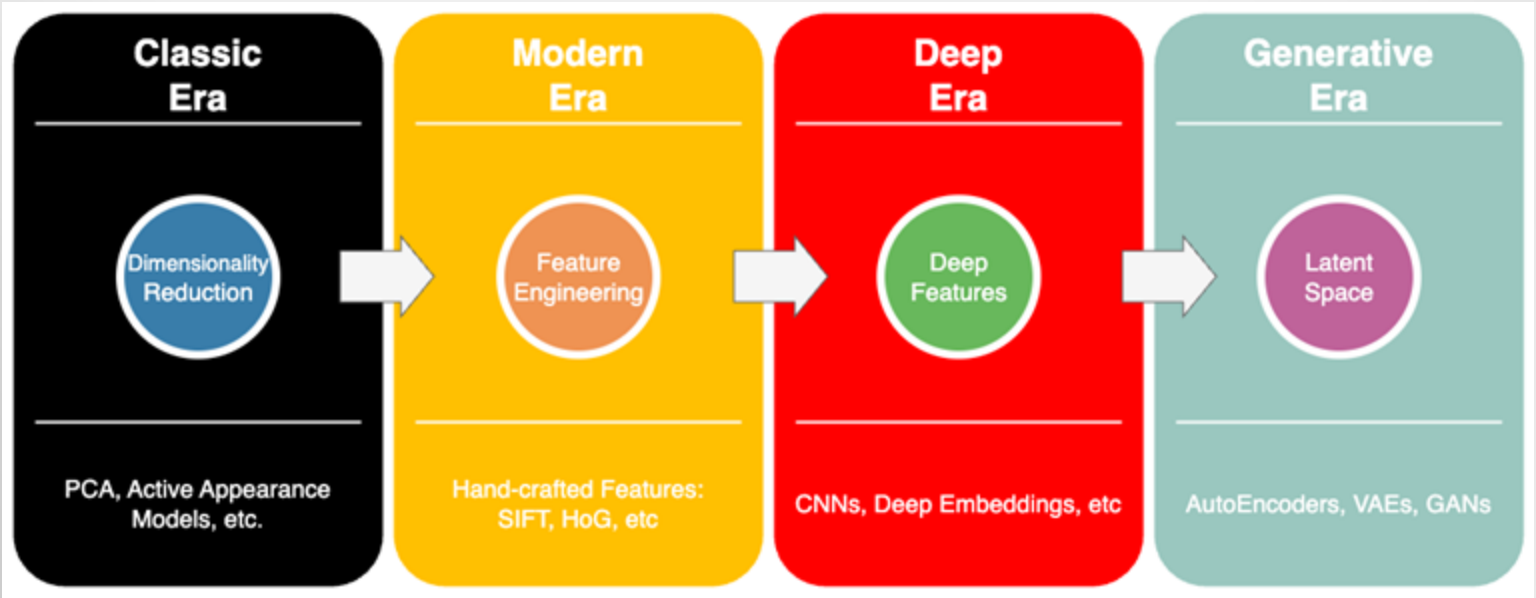


Evolution

This is a categorisation of eras I came up with myself. I am open to ideas, discussions and improvement.

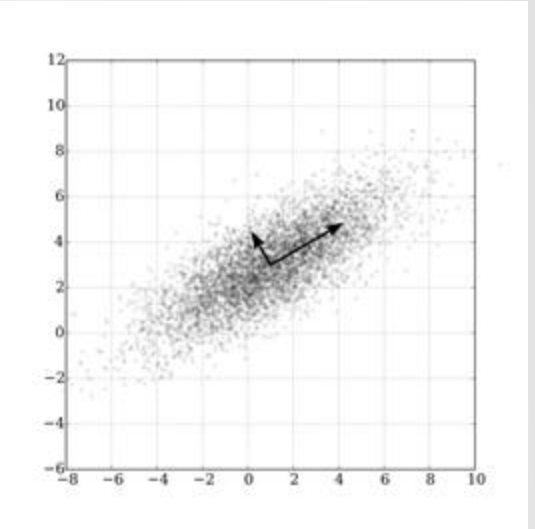
EURO 2

- In the field of generative modeling, the quest to find a suitable vector space evolved through distinct eras, each characterized by different techniques ar



Classical Methods

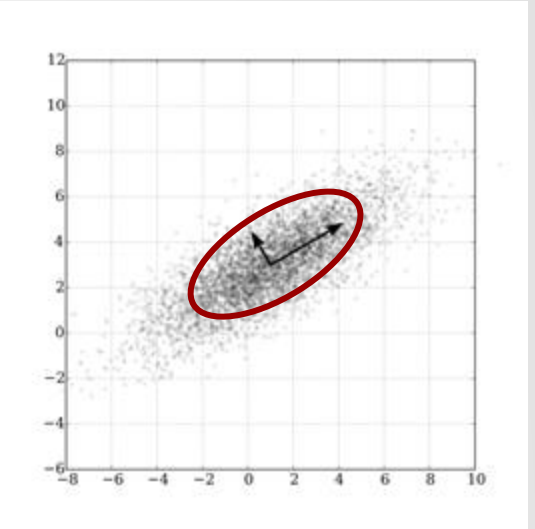
- Classical methods like Principal Component Analysis (PCA) were primarily developed for dimensionality reduction.
- The goal was to capture the most important components of the data, reducing the dimensionality while preserving as much information as possible.
- The resulting principal components capture the directions of maximum variance in the **original data**.



Classical Methods

- Classical methods like Principal Component Analysis (PCA) were primarily developed for dimensionality reduction.
- The goal was to capture the most important components of the data, reducing the dimensionality while preserving as much information as possible.
- To create a distribution representing the dataset, you can model the principal components as a multivariate normal distribution.

$$p(\boldsymbol{x}) = \frac{1}{\left|\sqrt{2\pi\boldsymbol{\Sigma}}\right|} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

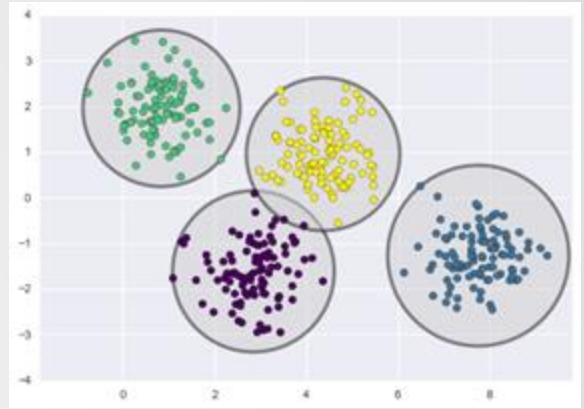


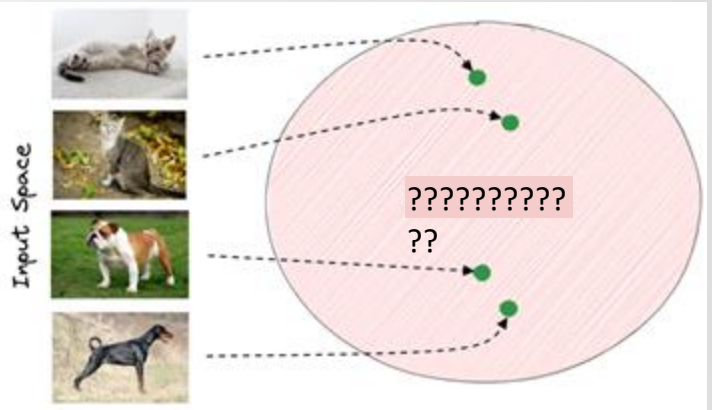
Generation with Classical Methods

- For complex data (like images etc) this idea failed miserably because:

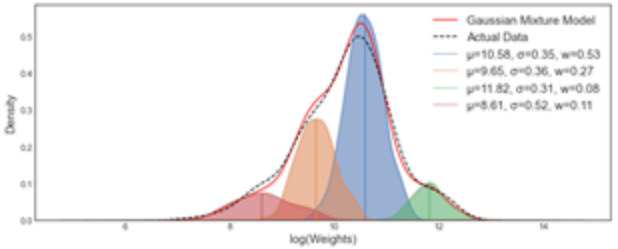
What if the data is multi-modal? PCA assumes that the data follows a unimodal Gaussian distribution along the principal components

Classical methods may not capture all the complex and high-level semantics of the data. They focus on capturing statistical variations rather than semantic meaning. The generated samples may resemble the statistical properties of the original data, but they may not capture the full complexity or exhibit higher-level semantics.





Gaussian Mixture Models (GMM)



EURO 2

A Gaussian mixture model (GMM) is a generative probabilistic model that consists of multiple Gaussian distributions.

- It is used to model complex data that cannot be represented by a single Gaussian distribution.
- GMMs are commonly used in clustering and density estimation tasks.
- The parameters of a GMM include the number of mixture components, the mean and standard deviation of each component, and the mixing coefficients that determine the weight of each component.

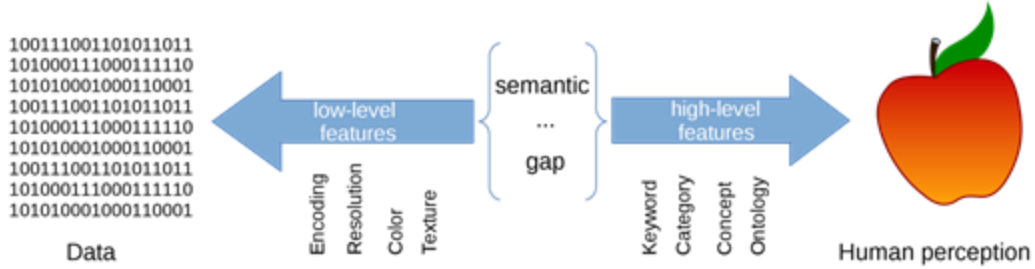
GMM for Generation?

What kinds of problems focus on utilizing simple generation techniques like GMMs for data-related tasks ?

(rather than explicitly incorporating semantic or higher-level feature spaces)

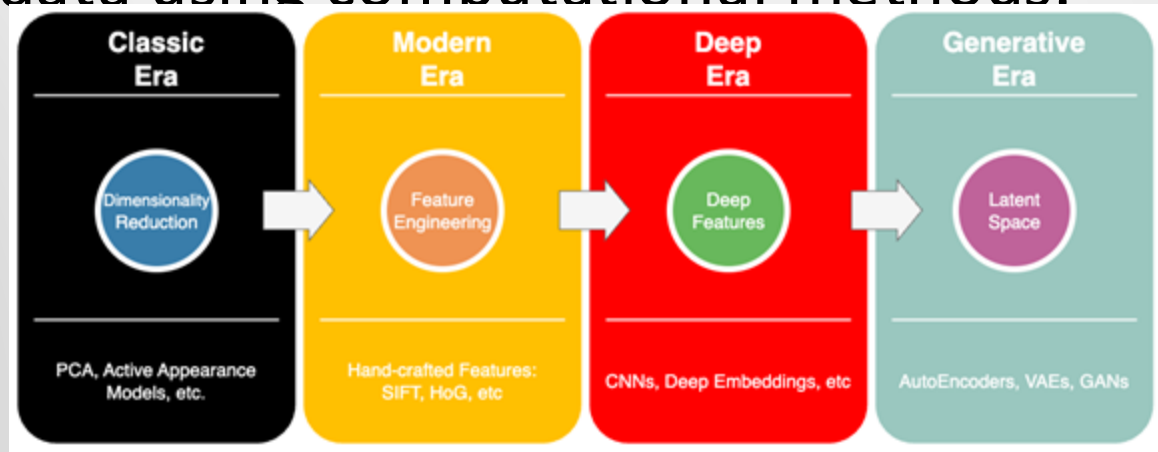
- Simulations
- Games/Animation
- Outlier Detection/Statistical Analysis
- Compression

Semantic Gap

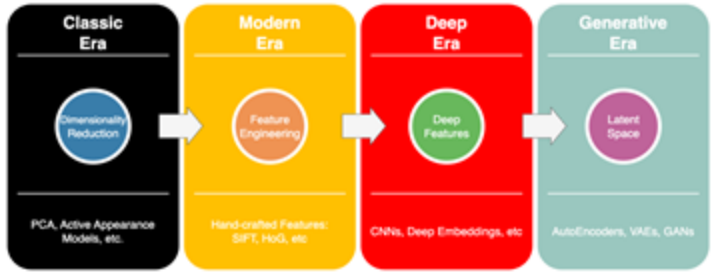


EURO 2

- The "semantic gap" is a well-known problem in the field of AI that refers to the disparity between the low-level, perceptual representation of data and the high-level, semantic understanding and interpretation of that data.
- In simpler terms, the semantic gap problem highlights the difficulty in capturing and representing the rich and complex semantics, meaning, and context of data using computational methods.



Feature Space Evaluation



- Modern Era: Pattern Recognition and Handcrafted Features
  - Focus on the use of handcrafted features, such as SIFT and HOG,
  - Dimensionality Reduction → Feature Engineering:
    - (Active Appearance Models)
- Deep Era: Deep Feature Learning
  - By training a deep encoder on a large-scale dataset, automatically extracting hierarchical and abstract features from the input data.
  - Feature Engineering → Feature Learning:
    - (Alexnet)
- Generative Era: The Latent Space
  - Deep features shift into latent spaces for generative purposes
  - Feature Learning → Latent Space:
    - (VAEs)

Next lecture:

- PART II: “Latent Spaces”

- (The Curse of) Dimensionality, Deep Features vs. Latent Spaces
- **Latent Space properties: Continuity, Entanglement, etc**

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

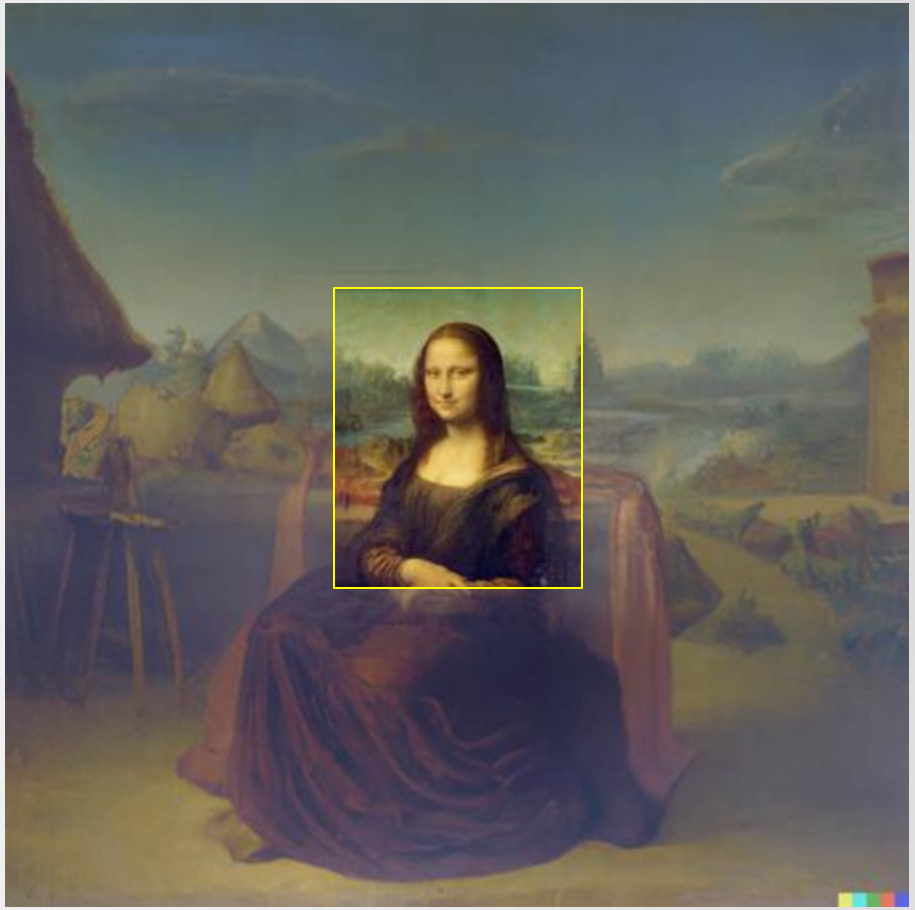
Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

Lesson 2: Latent Spaces

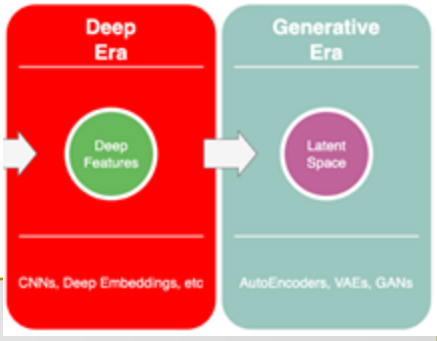
Welcome to Part II: “Latent Spaces”

This part includes two subsections:

- (The Curse of) Dimensionality, Deep Features vs. Latent Spaces
- **Latent Space properties: Continuity, Entanglement, etc**

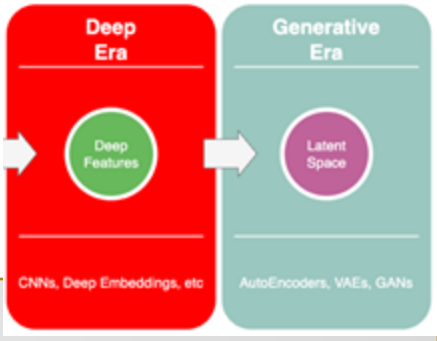


Deep Features vs Latent Space



- The difference between deep features learned by a deep encoder and a latent space of a generative model lies in their origin, representation, and usage within generative modeling.
- Let’s delve into a comparative overview of deep features learned by an encoder and the latent space of a generative model, highlighting their differences and the significance of these distinctions in the context of generative modeling:

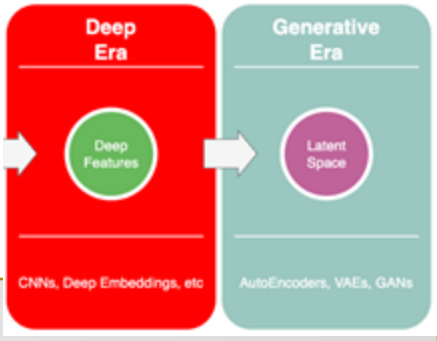
Deep Features vs Latent Space



- Representation vs. Distribution Modelling
  - **Deep Features:** Deep features learned by an encoder are representations of data that capture high-level patterns and discriminative information. **They focus on encoding the salient features necessary for classification or other downstream tasks.**
  - **Latent Space**: The latent space of a generative model is a lower-dimensional representation of the data that aims to capture the underlying structure and variations. **It focuses on modeling the probability distribution of the data in the latent space for generation purposes.**

- While deep features focus on capturing discriminative information for specific tasks, the latent space in generative models aims to learn a compressed, structured representation that can be used for data synthesis and generation.

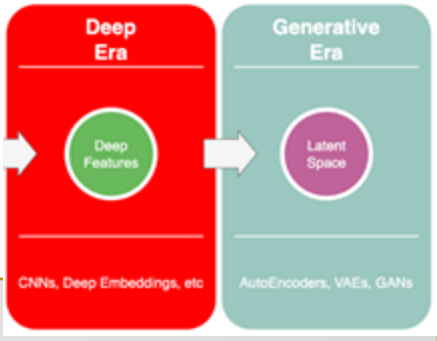
Deep Features vs Latent Space



- Supervised vs. Unsupervised Learning:
  - **Deep Features:** Learning deep features typically requires labeled data for supervision. **Models are trained using labeled examples to optimize the features for the specific task at hand, such as image classification.**
  - **Latent Space:** Generative models learn the latent space in an unsupervised manner, often using unlabeled data. **The models capture the underlying patterns and variations present in the data distribution without relying on explicit labels.**

- Unsupervised learning of the latent space allows generative models to capture the inherent structure of the data without the need for labeled examples.

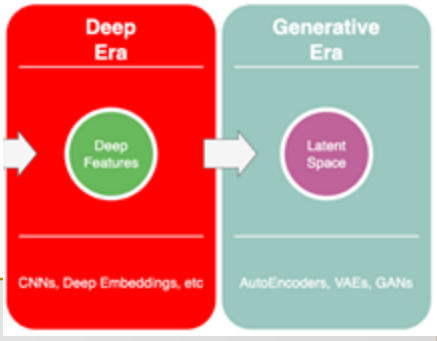
Deep Features vs Latent Space



- Discrimination vs. Generation:
  - **Deep Features**: Deep features focus on capturing discriminative information to distinguish between different classes or categories in the data. **They emphasize what differentiates one data point from another.**
  - **Latent Space**: The latent space of a generative model emphasizes the generation of new, realistic samples that resemble the original data distribution. **It encapsulates the factors of variation present in the data, allowing for the synthesis of diverse and novel instances.**

- The focus on generation in the latent space enables generative models to go beyond discrimination and create new data samples. This is crucial for tasks like image synthesis, data augmentation, and generating novel instances in various creative applications.

Deep Features vs Latent Space



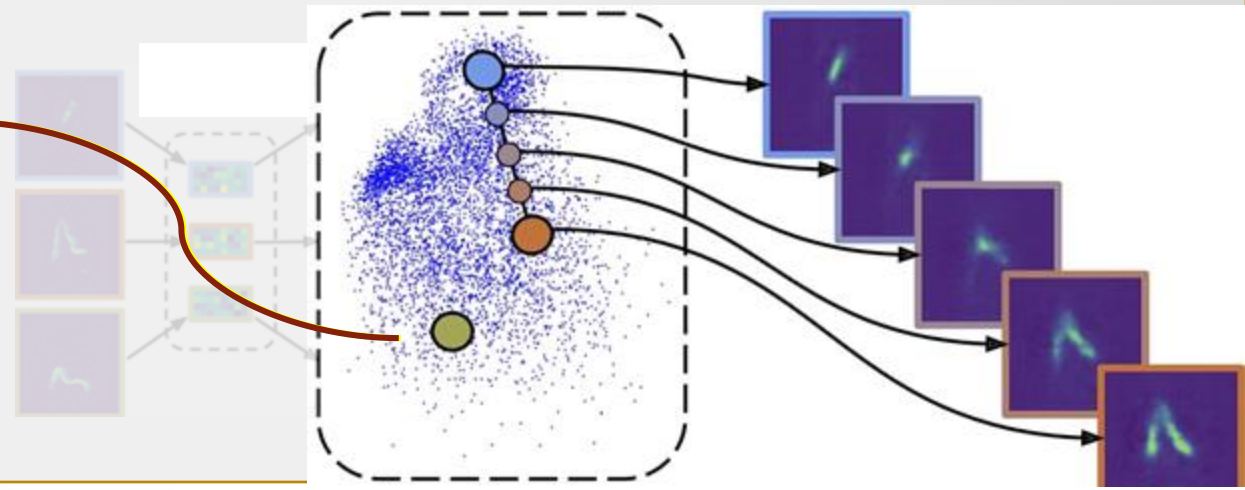
- Fixed vs. Controllable Representations:
  - **Deep Features**: Deep features learned by an encoder are fixed representations that do not offer explicit control over specific attributes or characteristics of the data. **They are optimized for the specific task and lack explicit manipulability.**
  - **Latent Space:** The latent space of a generative model offers the potential for explicit control and manipulation of specific attributes or factors of variation. **By exploring different regions or interpolating between latent vectors, specific attributes can be modified or combined to generate desired outputs.**

- The controllability of the latent space provides flexibility in generative modeling, allowing users to manipulate specific attributes or create variations in the generated outputs, particularly useful in applications like image editing, style transfer, and interactive content creation.

Sampling from a Vector Space

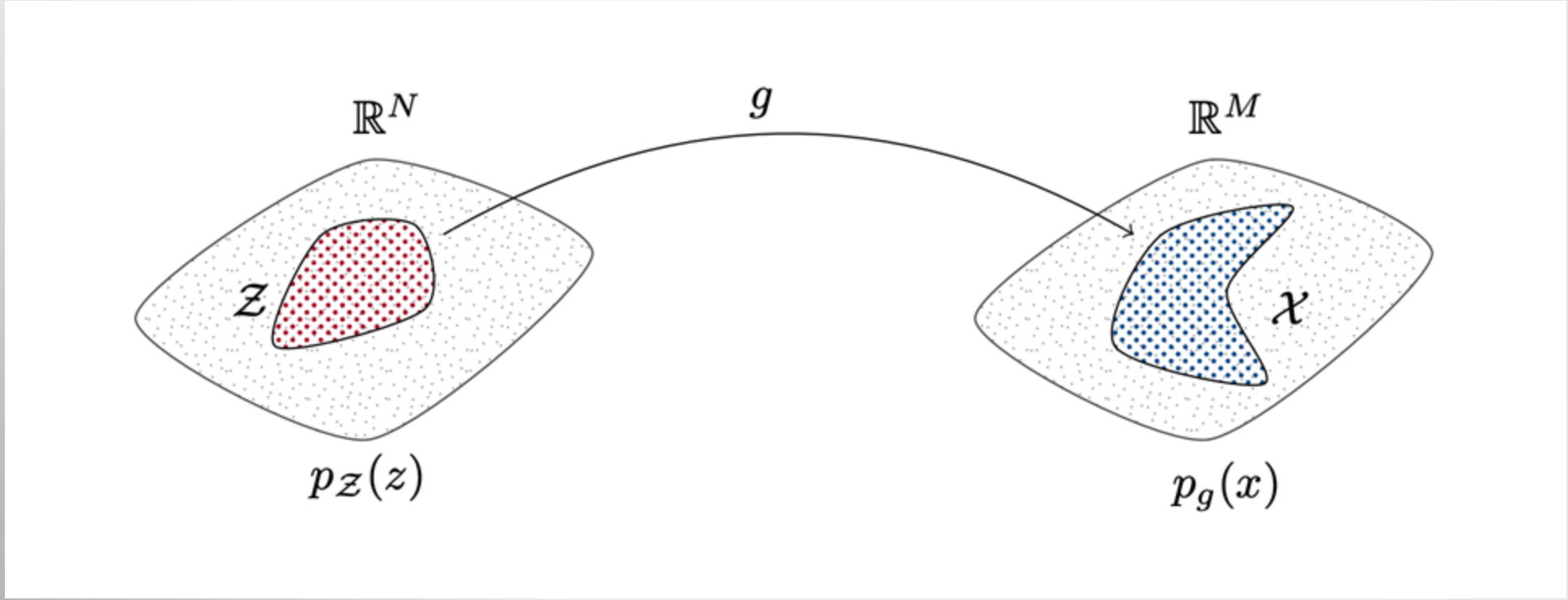
- In generative modeling, we require a intermediate vector space that is both **continuous** and **representative of the underlying data distribution** for effective sampling from a generator.
- A continuous vector space enables smooth interpolation and exploration of the data, allowing for **seamless transitions** and generation of new samples.

The Latent Space



The Latent Space

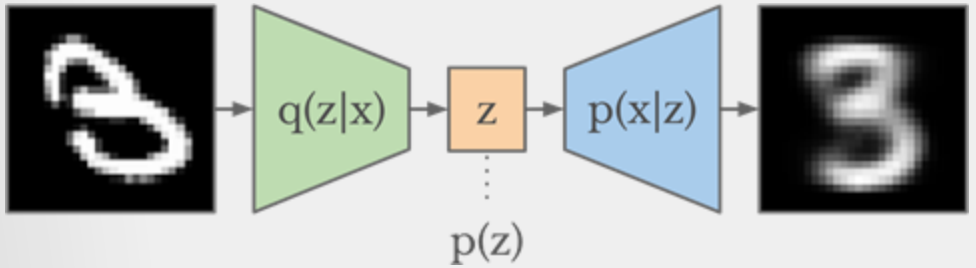
- A generative model g is a function that maps
  - a latent space $\mathcal{Z} \subseteq \mathbb{R}^N$ → on a target space $\mathcal{X} \subseteq \mathbb{R}^M$



a latent space $\boldsymbol{z} \subseteq \mathbb{R}^N \rightarrow$ on a target space $\boldsymbol{x} \subseteq \mathbb{R}^M$

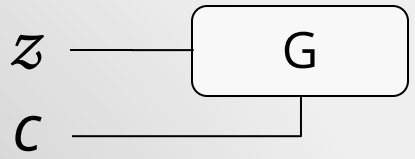
The Latent Space

- Early generative models primarily focused on explicit feature engineering and handcrafted representations.
- Latent space was not explicitly defined, and generation relied on manually designed algorithms or statistical models.
- Autoencoders, (not generative models but will evolve in to generative VAEs), played a significant role in developing the concept of the latent space.

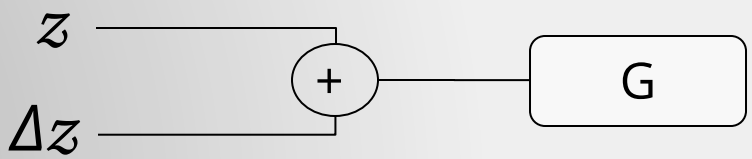


Controllable Generation

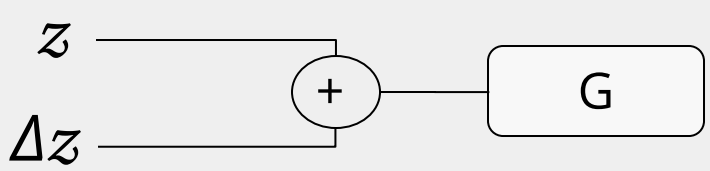
- Controllability vs Conditioning
  - Conditioning involves providing explicit information or constraints during generation, guiding the model to produce output aligned with specified conditions.

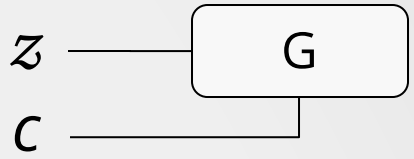


  - Controllability emphasizes user-driven customization, allowing manipulation of specific features in the generated output.



Controllable Generation



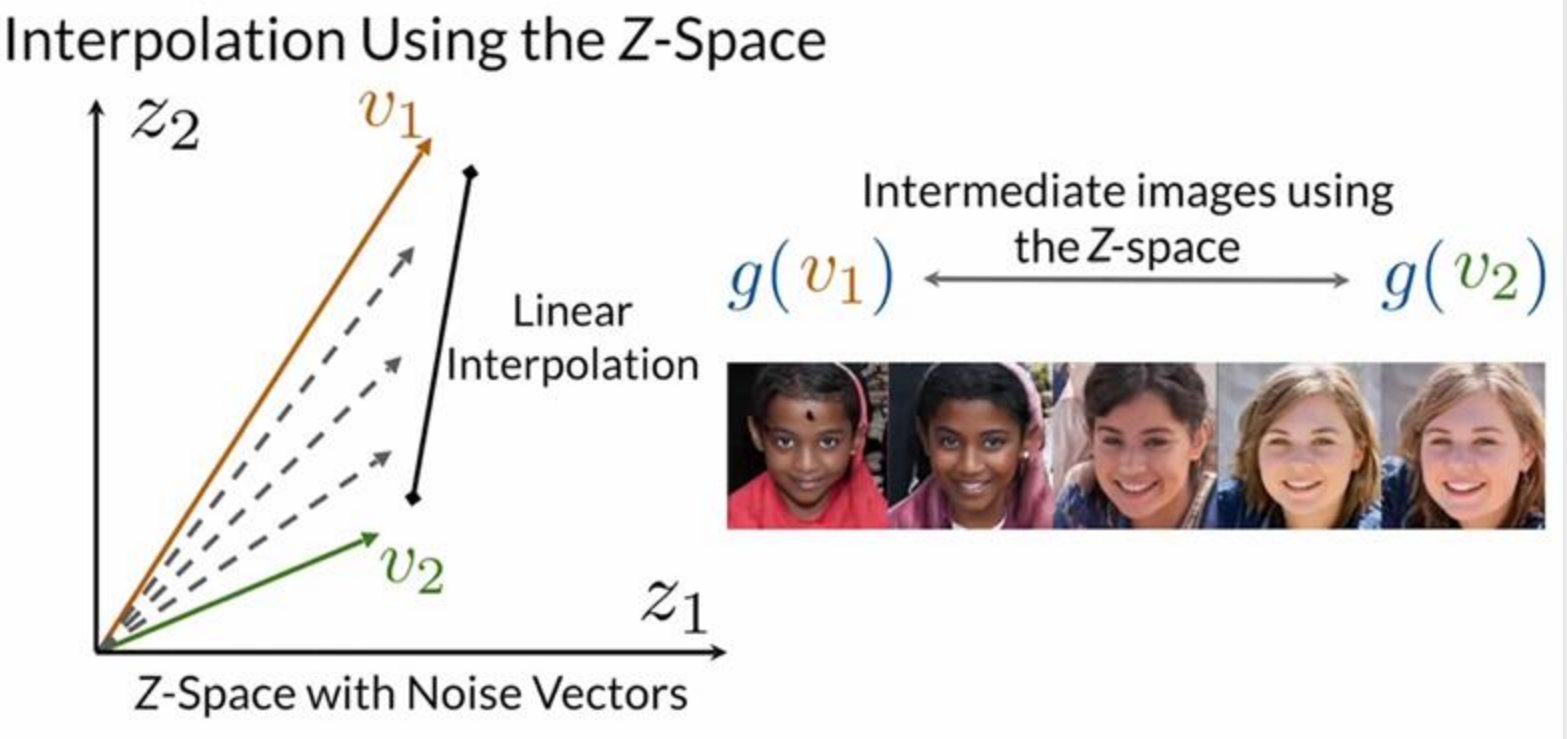


Controllable	Conditional
Examples with the **features that you want**	Examples from *the classes you want*
Training dataset **doesn't need to be labeled**	Training dataset *needs to be labeled*
Manipulate the z vector input	*Append a class vector* to the input

slide by coursera

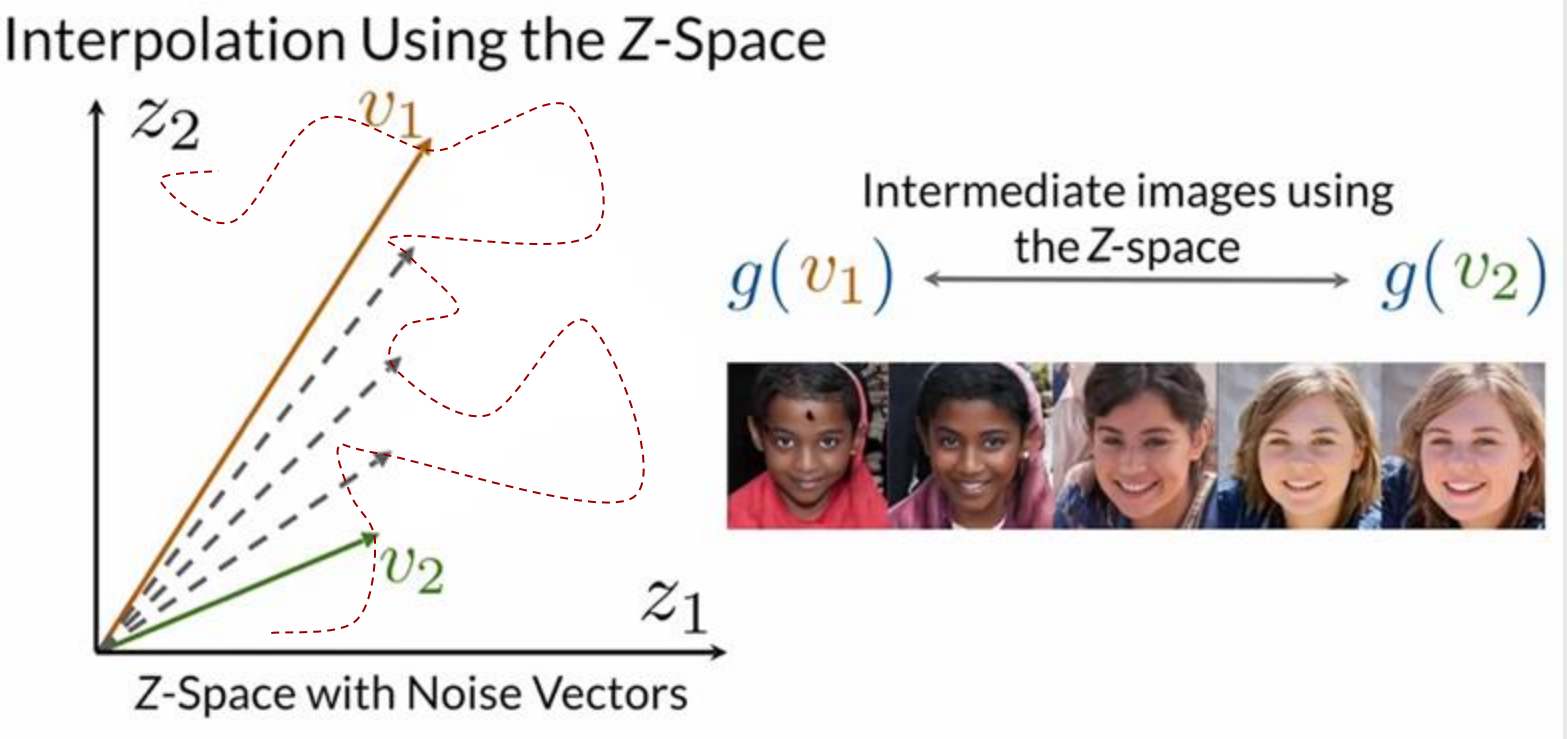
Controllable Generation

- Desired case is we can interpolate the z-space with perceptually smooth changes in the target space.



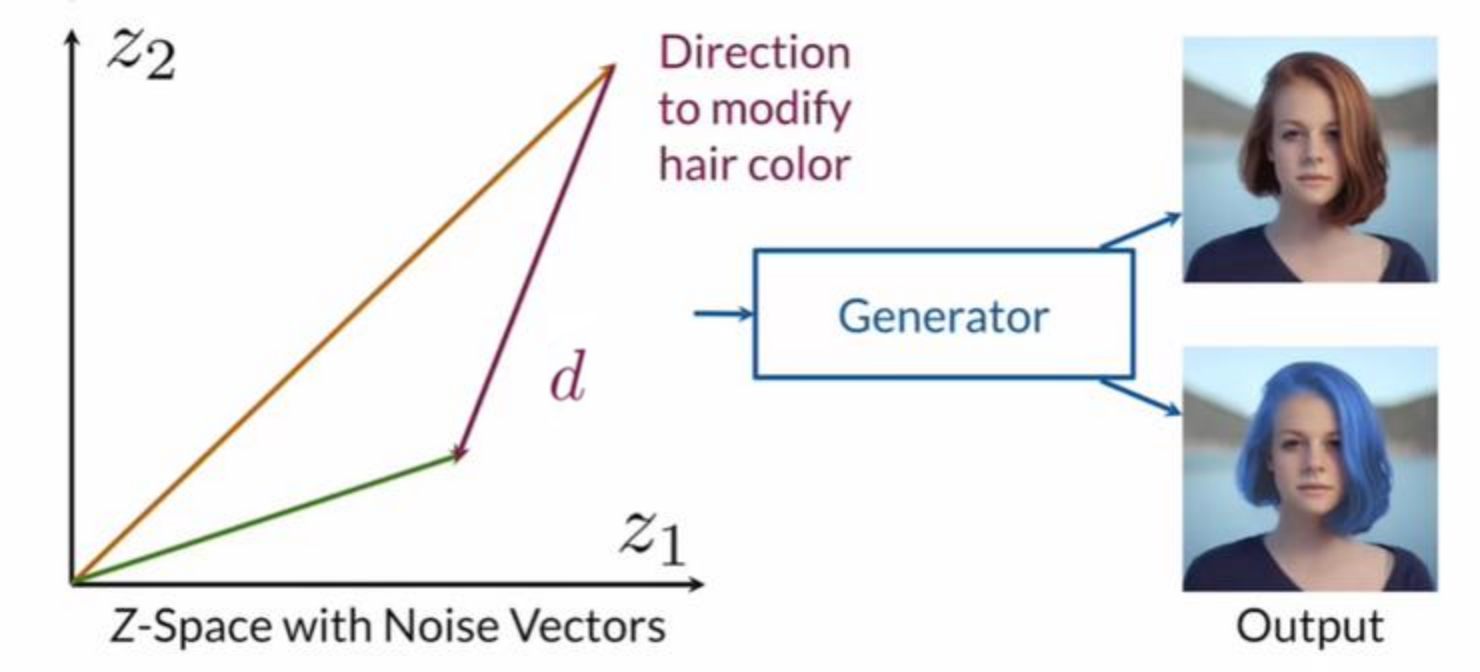
Controllable Generation

- However the reality is more like (if you do not specifically design it) this:



Controllable Generation (desired case)

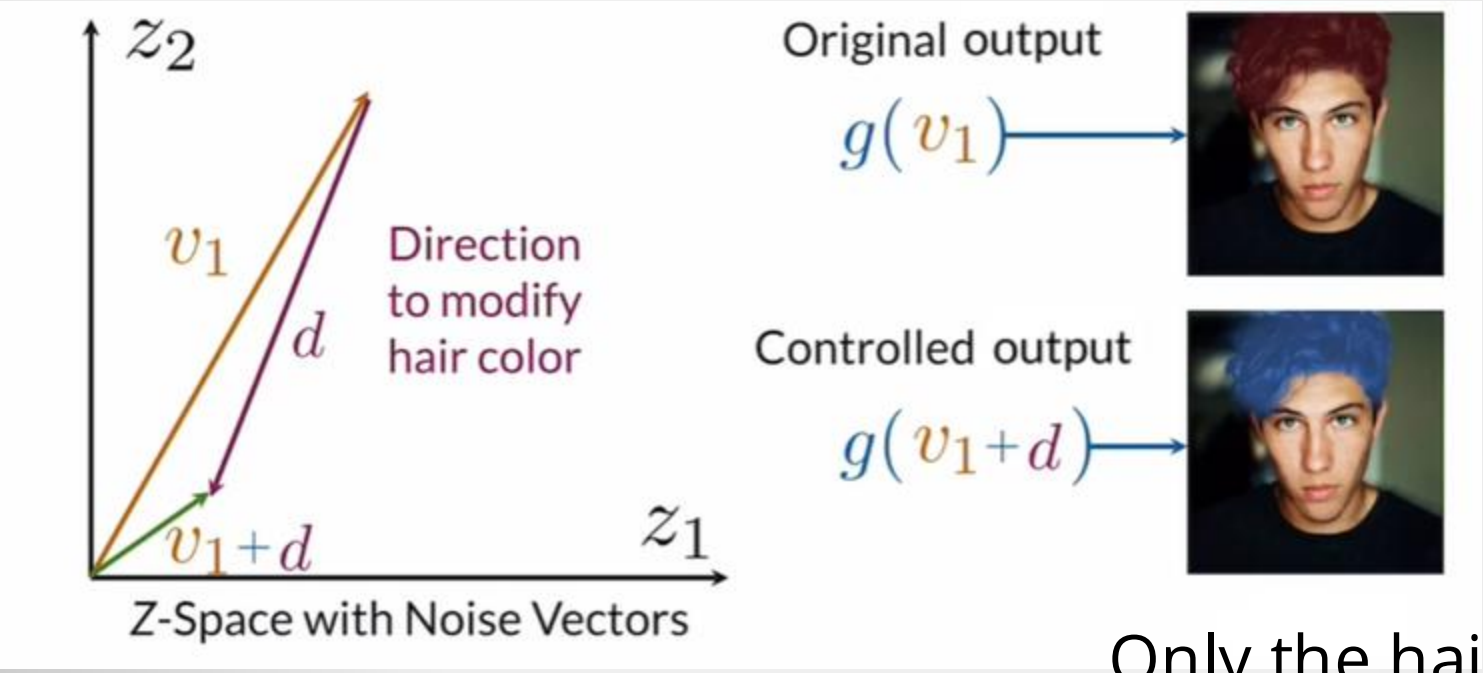
- In controlling the generation process, our goal is to find “desired” directions.



slide by coursera

Controllable Generation (desired case)

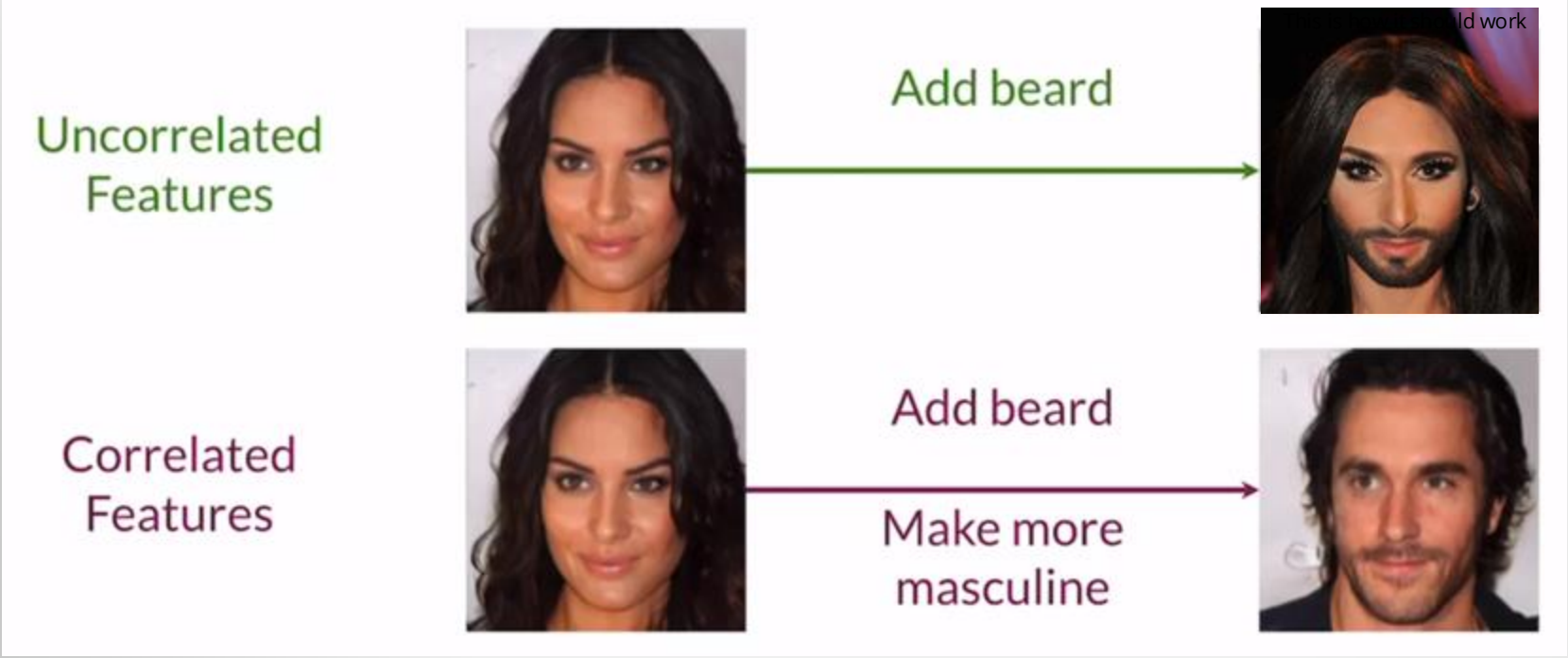
- In controlling the generation process, our goal is to find “desired” directions.



Only the hair!

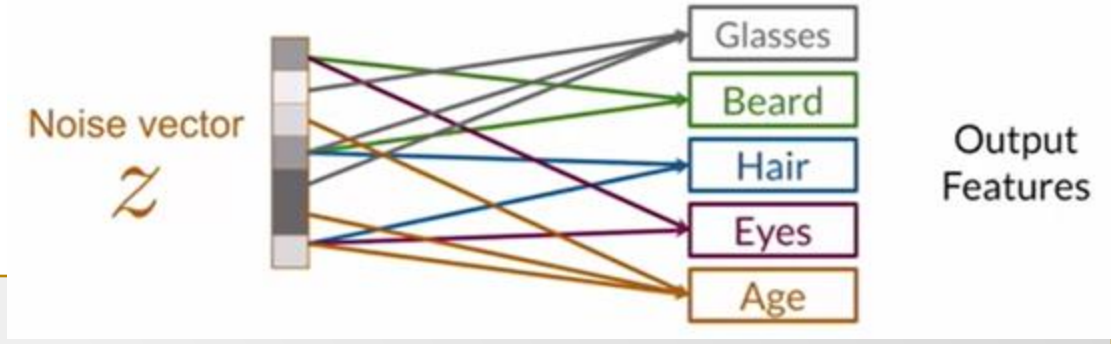
Feature Correlation

- The fundamental reason linear interpolation in the latent space is not feasible is because the features are correlated (maybe for the best)



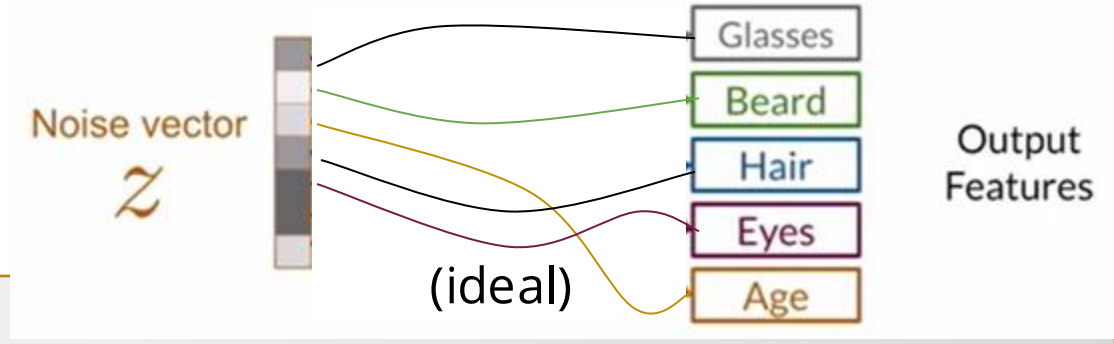
slide by coursera

Entanglement



- Entanglement refers to the intricate interdependence of different dimensions or features within the latent space.
- In simpler terms, changes in one dimension impact and influence other dimensions, creating complex relationships.
- The degree of entanglement directly affects the model's capacity to capture and represent diverse and complex patterns in the data.
- Understanding entanglement is crucial for improving the interpretability and controllability of generated outputs.

Entanglement



- Entanglement refers to the intricate interdependence of different dimensions or features within the latent space.
- In simpler terms, changes in one dimension impact and influence other dimensions, creating complex relationships.
- The degree of entanglement directly affects the model's capacity to capture and represent diverse and complex patterns in the data.
- Understanding entanglement is crucial for improving the interpretability and controllability of generated outputs.

Next lecture:

- **PART III: Auto-Encoding**

- **Autoencoders and Dimensionality Reduction**
- Variational Inference and VAEs
- Conclusions

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

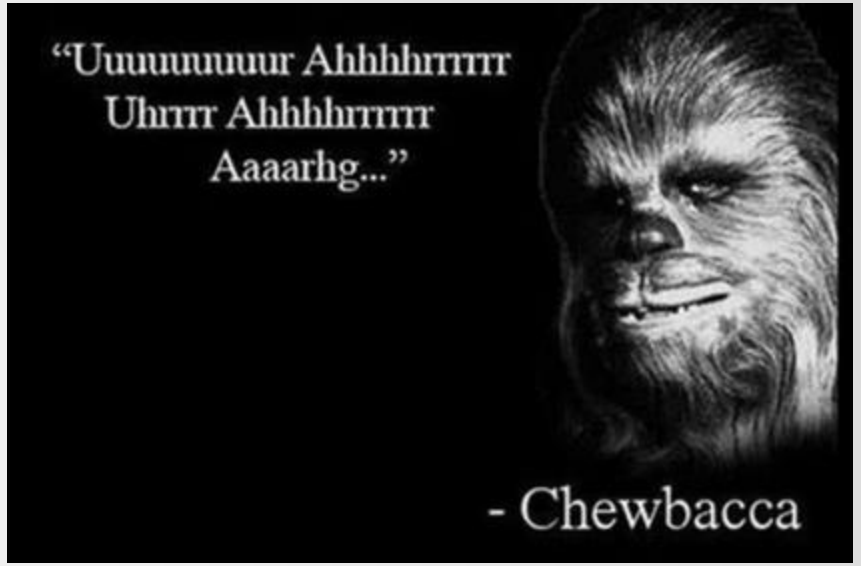
Conclusions

- Generative AI is the next phase of AI.
- It is based on concepts derived from Probability Theory, Information Theory and Machine Learning.
- The concept of a Latent Space is an invention that changed the realm of Generative AI.

What to learn next?

1. Auto-encoding
2. Normalizing Flows
3. GANs
4. Diffusion Models

Thanks



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

EURO^2

Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

Lesson 3: Auto-Encoding

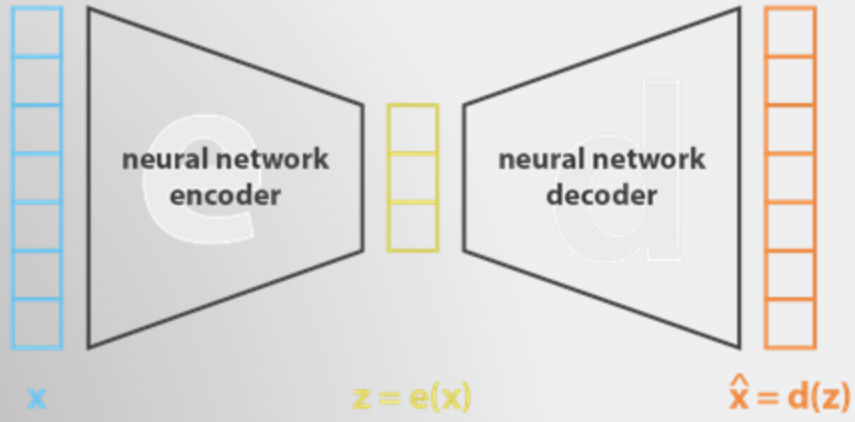
Welcome to **Part III: "Auto-Encoding"**

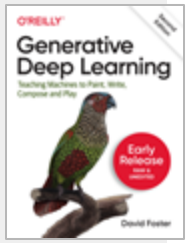
This part includes three subsections:

- **Autoencoders and Dimensionality Reduction**
- Variational Inference and VAEs
- Conclusions

Auto-Encoders

- Autoencoders comprise an encoder network that maps input data to a latent representation and a decoder network that reconstructs the input from the latent space.





Autoencoders

page 62

A diagram of the process described by the story is shown in Figure 3-2. You play the part of the **encoder**, moving each item of clothing to a location in the wardrobe. This process is called **encoding**. Brian plays the part of the **decoder**, taking a location in the wardrobe and attempting to re-stitch the item. This process is called **decoding**.

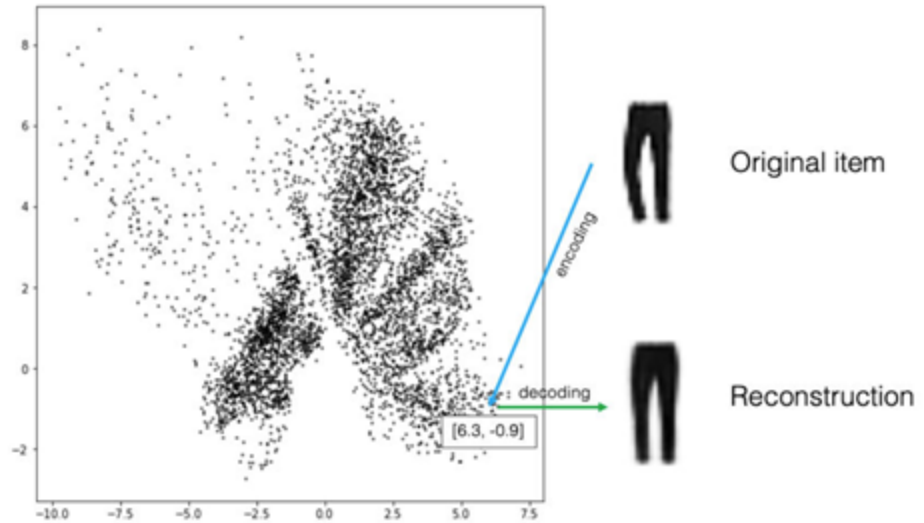


Figure 3-2. Items of clothing in the infinite wardrobe - each black dot represents an item of clothing.

Auto-Encoders

- **Encoder**: A module that compresses the train-validate-test set input data into an encoded representation that is typically several orders of magnitude smaller than the input data.
- **Bottleneck**: A module that contains the compressed knowledge representations and is therefore the most important part of the network.
- **Decoder**: A module that helps the network“decompress” the knowledge representations and reconstructs the data back from its encoded form. The output is then compared with a ground truth.

Autoencoders

A diagram of the process described by the story is shown in Figure 3-2. You play the part of the **encoder**, moving each item of clothing to a location in the wardrobe. This process is called **encoding**. Brian plays the part of the **decoder**, taking a location in the wardrobe and attempting to re-stitch the item. This process is called **decoding**.

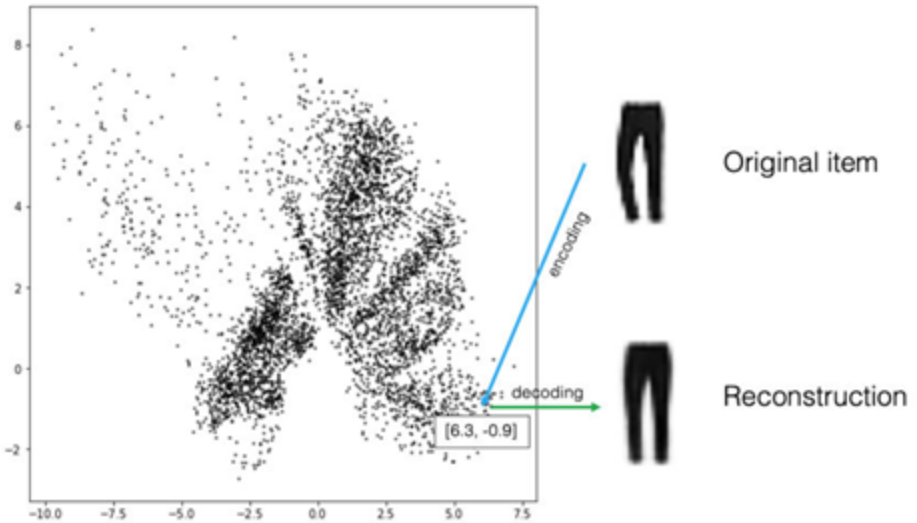
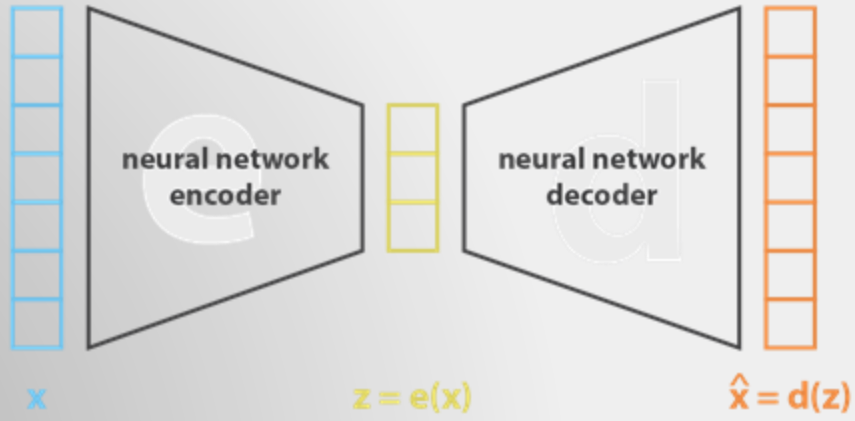


Figure 3-2. Items of clothing in the infinite wardrobe - each black dot represents an item of clothing.

Auto-Encoders

- The bottleneck layer in autoencoders serves as the latent space, capturing a compressed representation of the input data.



Autoencoders

A diagram of the process described by the story is shown in Figure 3-2. You play the part of the **encoder**, moving each item of clothing to a location in the wardrobe. This process is called **encoding**. Brian plays the part of the **decoder**, taking a location in the wardrobe and attempting to re-stitch the item. This process is called **decoding**.

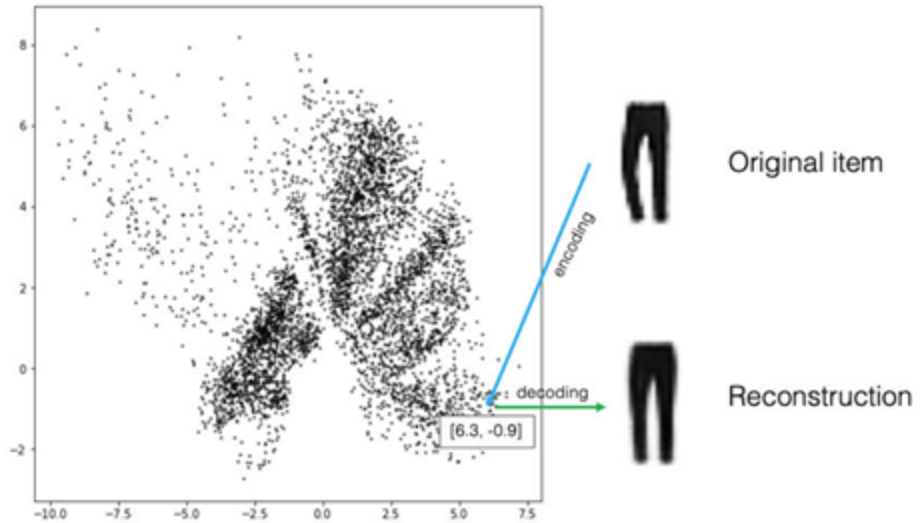
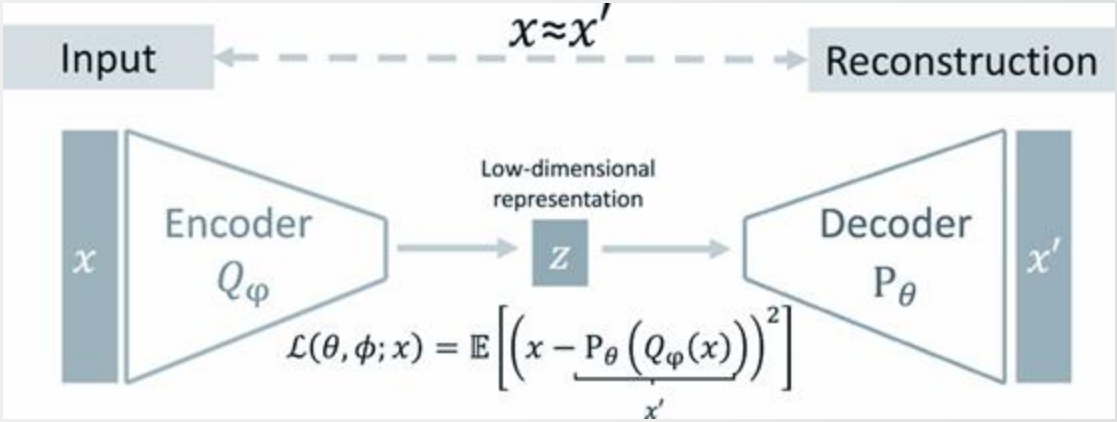


Figure 3-2. Items of clothing in the infinite wardrobe - each black dot represents an item of clothing.

Training Auto-Encoders

There are 4 fundamental hyperparameters for training an autoencoder:

- **Code/Latent Space Size**: The bottleneck size decides how much the data has to be compressed. This can also act as a regularisation term.
- **Number of layers/nodes**: While a higher depth/number of nodes increases model capacity/complexity, a lower depth is faster to process and avoid overfitting.
- **Reconstruction Loss**: This is the most important one. It defines the character of the AutoEncoder.



Auto-Encoder Loss function

- In an autoencoder, the loss function plays a crucial role in training the model and optimizing the reconstruction process.
- The choice of the loss function depends on the specific objectives and characteristics of the autoencoder.
- There can be many features that shape the loss character of an AE
  - Reconstruction (main)
  - Regularization (main)
  - Characterized loss (optional): Sparsity, Variational loss, other problem dependent

Reconstruction Loss

- The primary objective of an autoencoder is to reconstruct the input data from the compressed latent space.
- The reconstruction loss measures the discrepancy between the original input and the reconstructed output.
- Commonly used reconstruction loss functions depend on the input data and include:
  - mean squared error (continuous data), binary cross-entropy (binary data), Kullback-Leibler Divergence (distributions), Categorical Cross-Entropy (multiple classes), etc.

Regularization Loss

- Autoencoders can be prone to overfitting, especially when the model capacity is high or the dataset is limited.
- Regularization techniques, such as L1 or L2 regularization, drop-out etc, can be incorporated into the loss function to discourage complex or redundant representations.
- Regularization loss plays a vital role in controlling the behavior of an autoencoder and ensuring that the learned representations possess desirable properties.
- The specific choice and application of regularization techniques depend on the characteristics of the data, the desired properties of the representations, and the overall training objectives.

Characterized Loss

- A “”characterized loss” component can be introduced to go beyond traditional objectives like reconstruction or regularization, addressing specific requirements or goals of the task at hand.
- Such as:
  - Sparsity Loss: Only a small subset of the latent variables or activations should be active or non-zero.
    - $\lambda * \sum |z|$
  - Diversity Loss: Generation of diverse outputs by penalizing models that generate repetitive or similar samples.
    - $(1/(n*(n-1))) * \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} d(x_i, x_j)$
  - Consistency Loss: encourages the model to produce consistent predictions across augmented versions of the same sample, improving generalization.

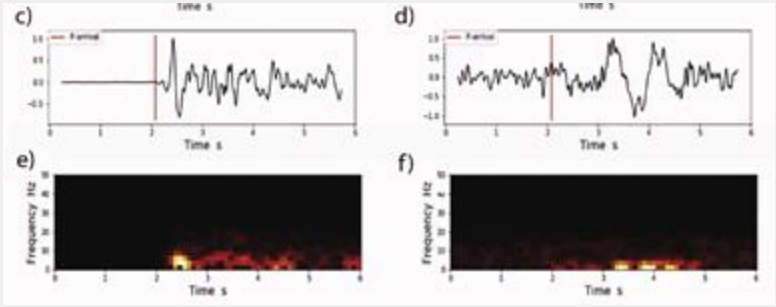
Characterized Loss: Clustering

Unsupervised Clustering of Seismic Signals Using Deep Convolutional Autoencoders

S. Mostafa Mousavi, Weiqiang Zhu, William Ellsworth, and Gregory Beroza

EURO²

- A clustering layer, connected to the AE's bottleneck, assigns the hidden features of each sample to a cluster.
- The clustering loss component L_c is specifically designed to enhance the clustering behavior of the learned latent representations.
- The input is data is seismic waveforms, that reach a seismic station.



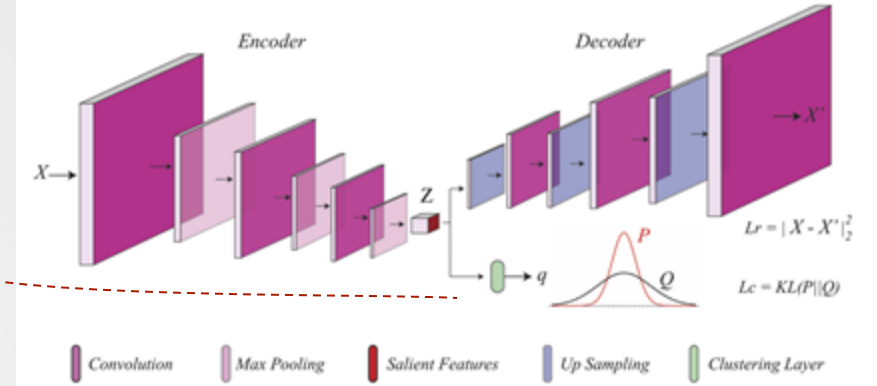


Fig. 1. Network architecture. The encoder and decoder are composed of fully convolutional layers followed by max-pooling and up-sampling layers, respectively. Reconstruction loss (L_r) and the clustering loss (L_c) are given in the figure.

$$L = (1 - \lambda)L_r + \lambda L_c \tag{1}$$

$$q_{ij} = \frac{(1+ \| z_i - \mu_j \|^2)^{-1}}{\sum_j (1+ \| z_i - \mu_j \|^2)^{-1}}. \tag{2}$$

The membership probabilities are used to compute an auxiliary target distribution, p_{ij}

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})} \tag{3}$$

and clustering is performed by minimizing the Kullback–Leibler (KL) divergence between the soft assignments, q_{ij}, and the target distribution, p_{ij}

$$L_c = KL(P \| Q) = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right). \tag{4}$$

Characterized Loss: Clustering

- The clustering is done in two steps.
  - In the initial step, they train by setting λ=0 (i.e. only reconstruction)
  - In the next step, the learned features are used to initialize the cluster centroids (μ_j) in the feature space using k -means. Following, cluster assignment and feature learning are jointly performed (λ=0.1)

Or, just for fun (and for a good project) regions can be manually labeled for **regions**, and the second step could be supervised! That would be a mix of unsupervised and supervised learning!

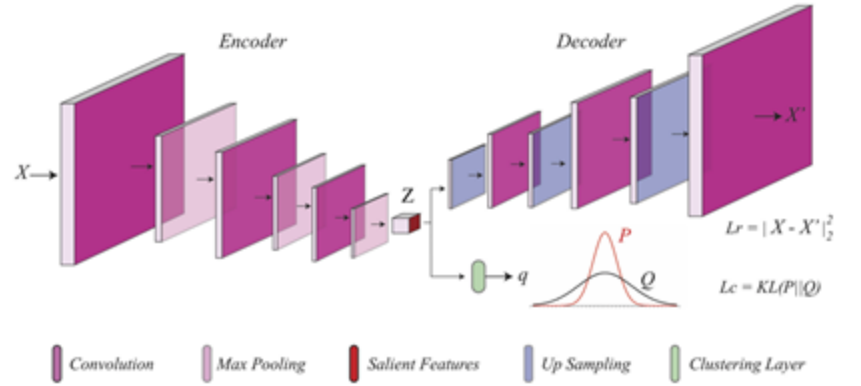


Fig. 1. Network architecture. The encoder and decoder are composed of fully convolutional layers followed by max-pooling and up-sampling layers, respectively. Reconstruction loss (L_r) and the clustering loss (L_c) are given in the figure.

$$L = (1 - \lambda)L_r + \lambda L_c \quad (1)$$

$$q_{ij} = \frac{(1+\| z_i - \mu_j \|^2)^{-1}}{\sum_j (1+\| z_i - \mu_j \|^2)^{-1}}. \quad (2)$$

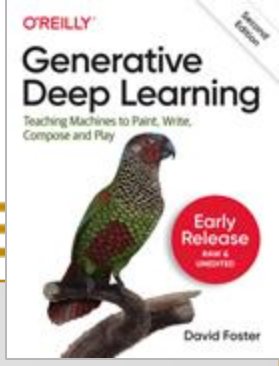
The membership probabilities are used to compute an auxiliary target distribution, p_{ij}

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})} \quad (3)$$

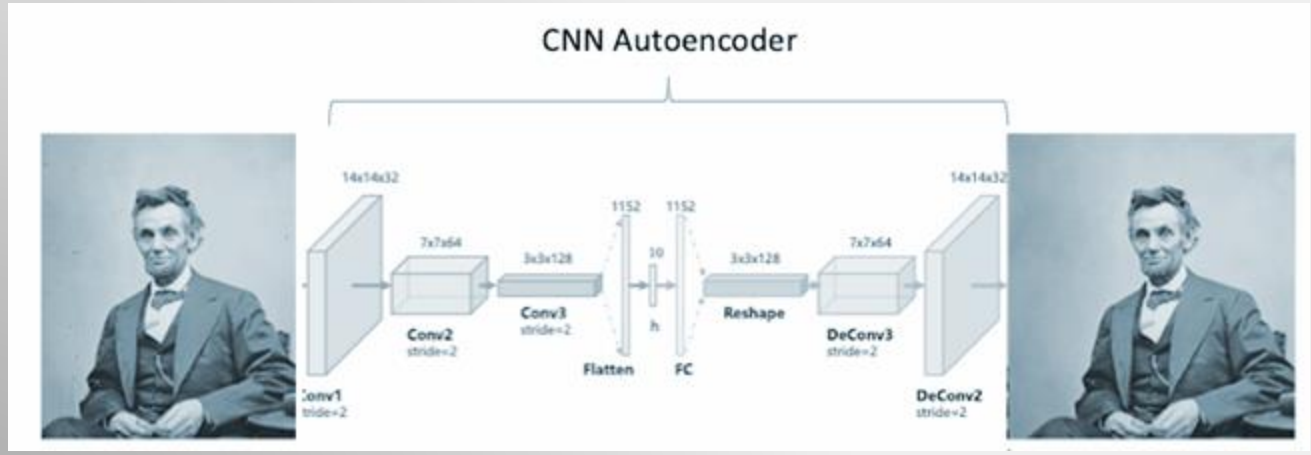
and clustering is performed by minimizing the Kullback–Leibler (KL) divergence between the soft assignments, q_{ij}, and the target distribution, p_{ij}

$$L_c = KL(P \| Q) = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right). \quad (4)$$

Auto-Encoders



- Autoencoders comes in all types
  - **Convolutional,** (even TCN)
  - LSTM
  - Fully-Connected…



Autoencoders

A diagram of the process described by the story is shown in Figure 3-2. You play the part of the **encoder**, moving each item of clothing to a location in the wardrobe. This process is called **encoding**. Brian plays the part of the **decoder**, taking a location in the wardrobe and attempting to re-stitch the item. This process is called **decoding**.

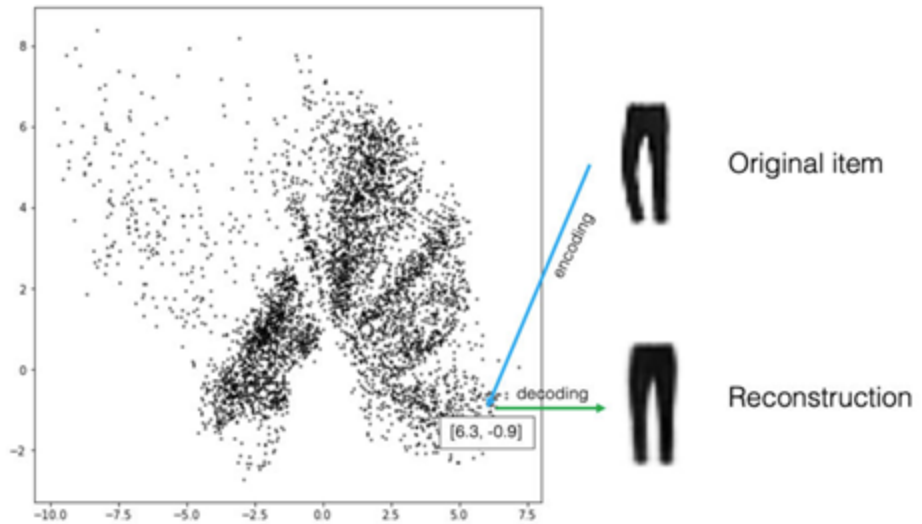
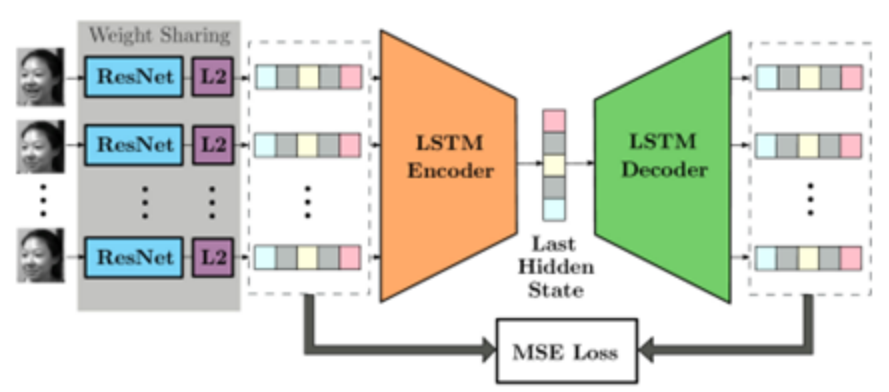


Figure 3-2. Items of clothing in the infinite wardrobe - each black dot represents an item of clothing.

Auto-Encoders

- Autoencoders comes in all types
  - Convolutional
  - **LSTM**
  - Fully-Connected



Autoencoders

A diagram of the process described by the story is shown in Figure 3-2. You play the part of the **encoder**, moving each item of clothing to a location in the wardrobe. This process is called **encoding**. Brian plays the part of the **decoder**, taking a location in the wardrobe and attempting to re-stitch the item. This process is called **decoding**.

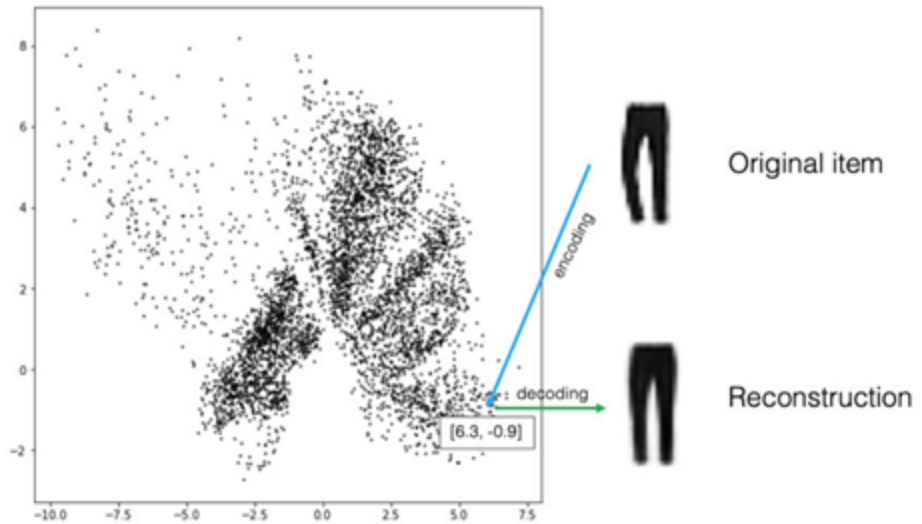
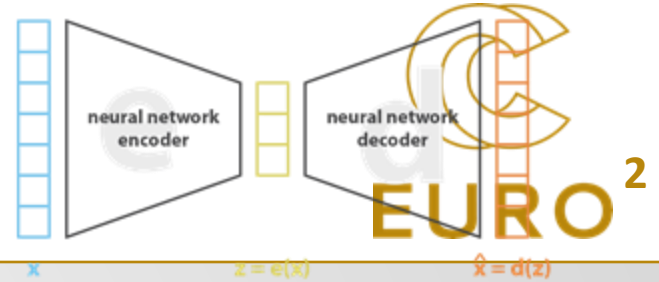
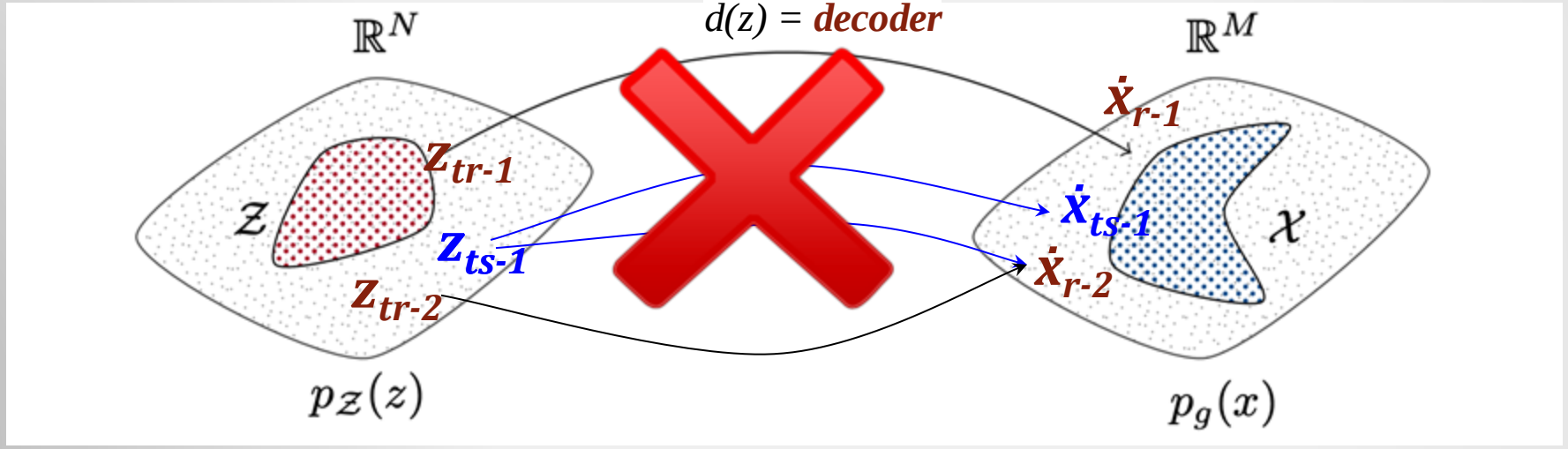


Figure 3-2. Items of clothing in the infinite wardrobe - each black dot represents an item of clothing.

AE problems

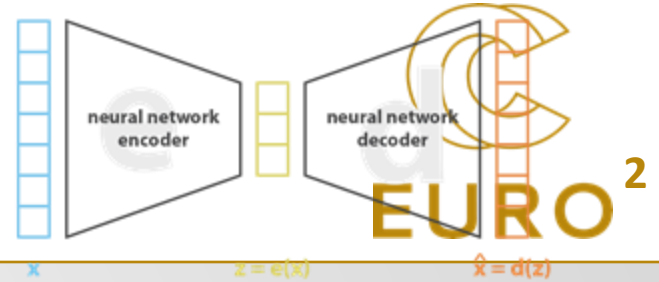


- One of the main limitations of traditional AEs is that they do not necessarily create a continuous latent space.
- This discontinuity makes it difficult for AEs to perform tasks like interpolation or extrapolation, where generating new samples in between or beyond the training data becomes challenging.

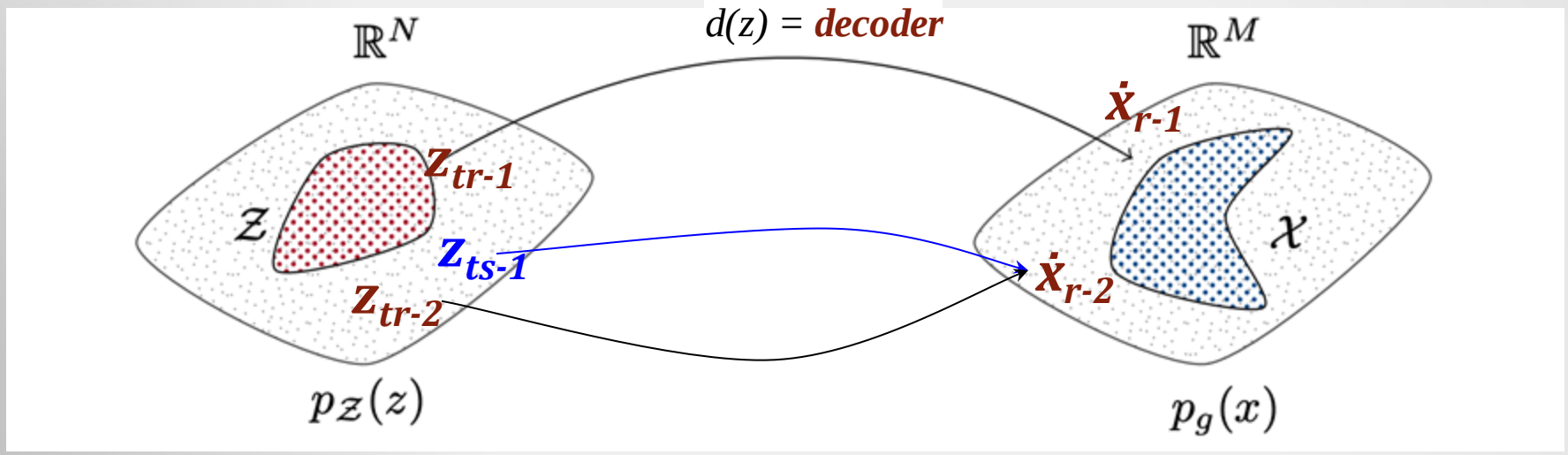


because it memorises the training samples

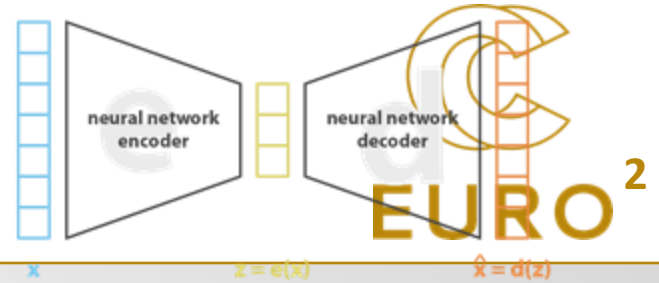
AE problems



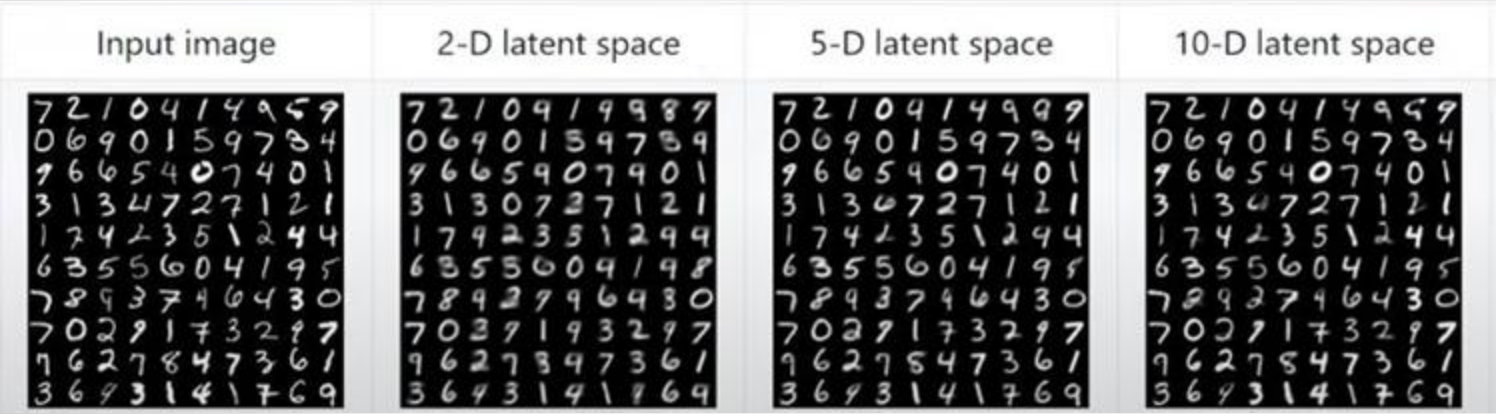
- The fundamental problem with autoencoders, for generation, is that the latent space they convert their inputs to and where their encoded vectors lie, may not be continuous, or allow easy interpolation.
- This is fine if you're just replicating the same images.



Latent Space size ?



- Increasing the latent space vector ***z*** dimension size (***n***), will improve the replication/reconstruction success, but will affect the continuity problem even worse.
- Increasing the latent space vector ***z*** dimension size (***n***), will degrade the replication/reconstruction performance.



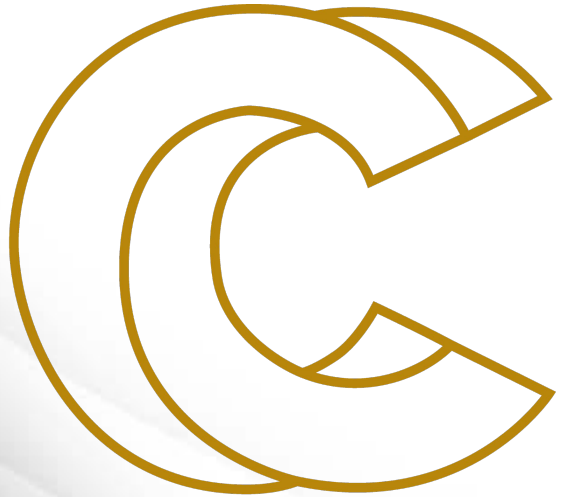
Next lecture:

- PART III: Auto-Encoding

- Autoencoders and Dimensionality Reduction
- **Variational Inference and VAEs**
- Conclusions

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia



EURO^2

Fundamental Concepts of Generative Machine Learning

Erdem Akagündüz, PhD
Graduate School of Informatics, METU, Türkiye
ncc@ulakbim.gov.tr

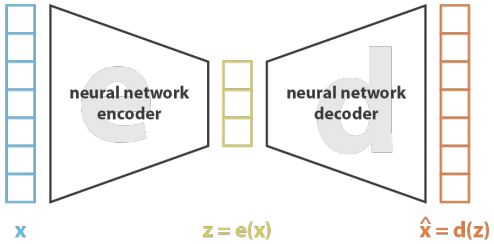
Lesson 3: Auto-Encoding

Welcome to **Part III: "Auto-Encoding"**

This part includes three subsections:

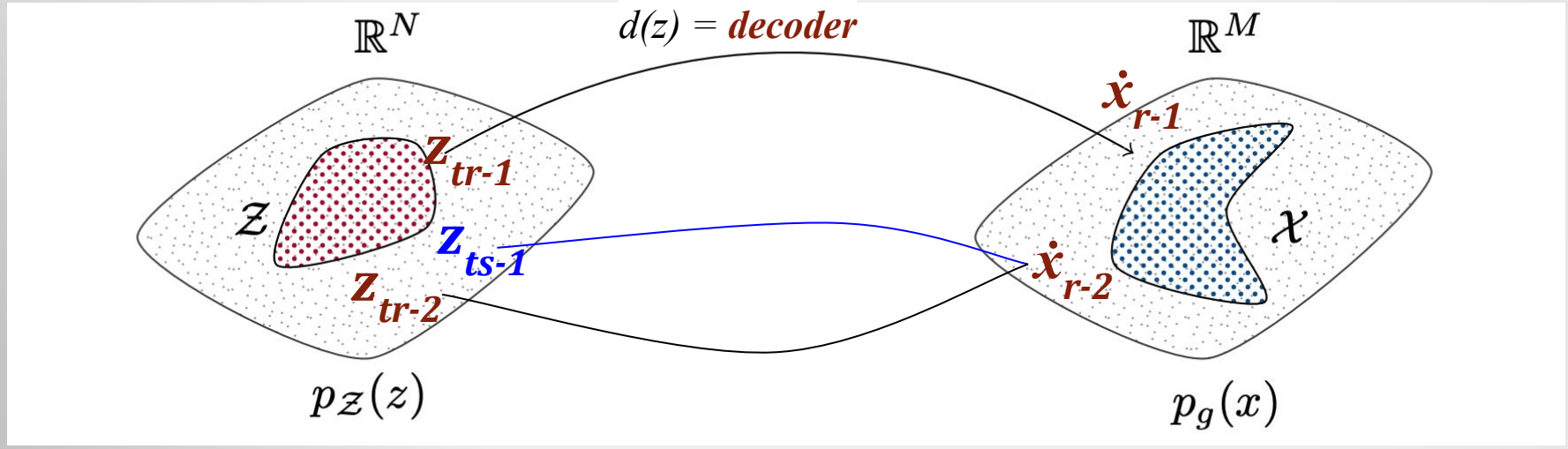
- Autoencoders and Dimensionality Reduction
- **Variational Inference and VAEs**
- Conclusions

AE problems



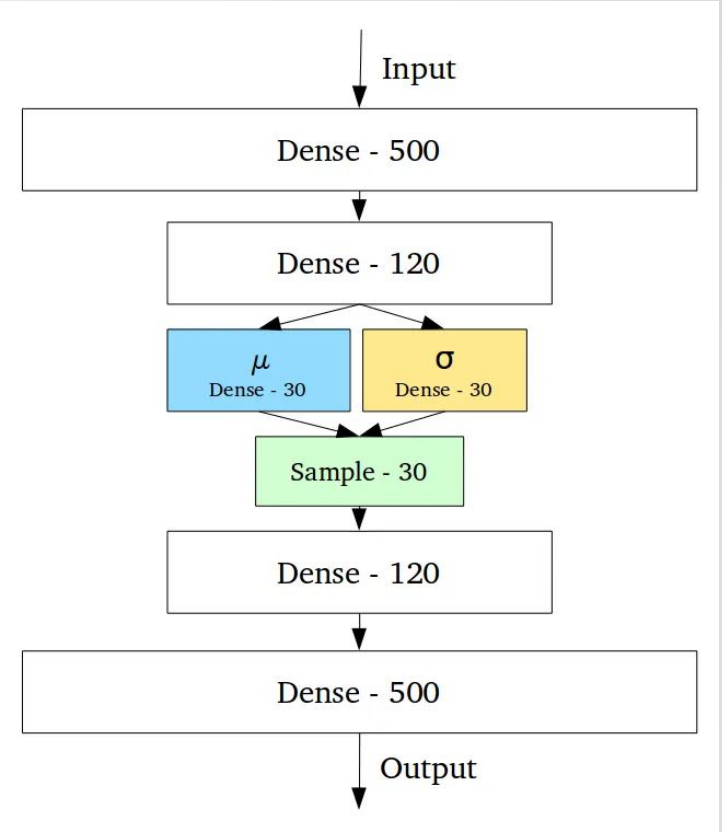
EURO²

- But when you're building a generative model, you don't want to replicate the same input.
- You want to randomly sample from the latent space, or generate variations on an input image, from a continuous latent space.



Variational AutoEncoders (VAEs)

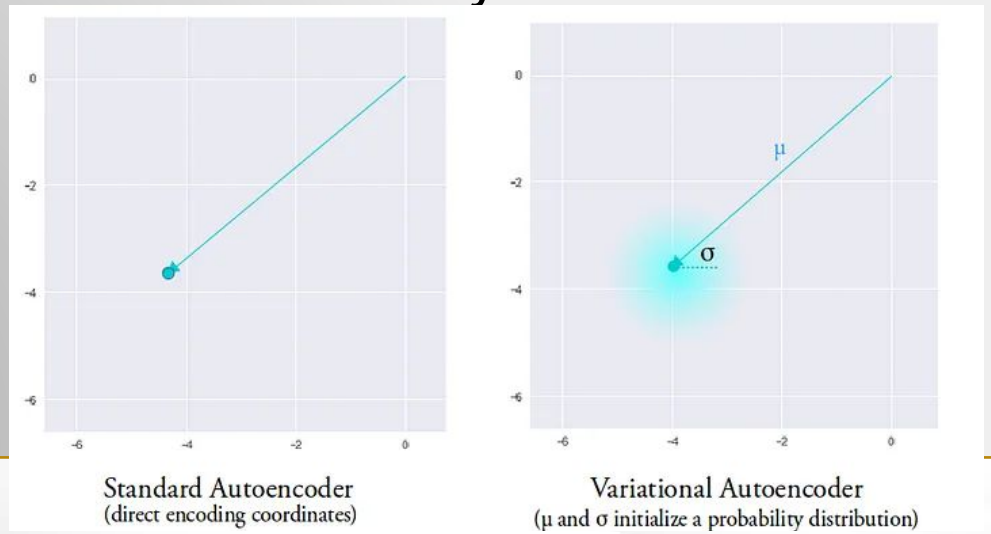
- Variational Autoencoders (VAEs) have one fundamentally unique property that separates them from vanilla autoencoders, and it is this property that makes them so useful for generative modeling:
- their latent spaces are, **by design**, continuous, allowing easy random sampling and interpolation.

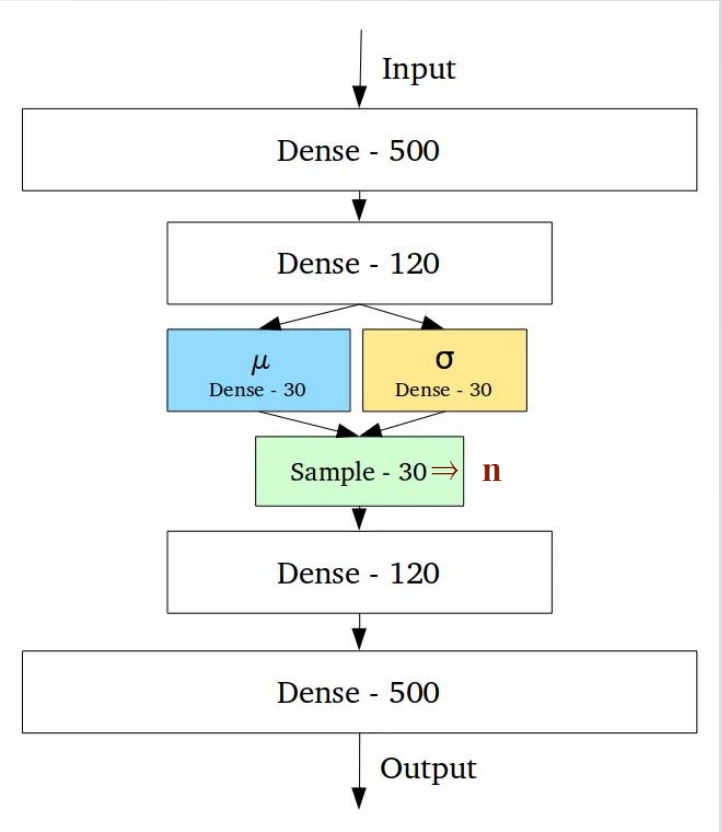


Variational AutoEncoders (VAEs)

- Instead of mapping the input to a fixes latent vector, they map it to a distribution!

- What is more, VAEs force the latent variables to be Normal distributed.
- It achieves this by making its encoder *:*
  *- not output an encoding vector of size **n** (like AEs),*
  *- rather, outputting two vectors of size **n**:*
  - *a vector of means, **μ**,*
  - *and another vector of standard deviations, **σ**.*





Variational AutoEncoders (VAEs)

- Instead of mapping the input to a fixes latent vector, they map it to a distribution!
- What is more, VAEs force the latent variables to be Normal distributed.
- It achieves this by making its encoder *:*
  *- not output an encoding vector of size **n** (like AEs),*
  *- rather, outputting two vectors of size **n**:*
  - *a vector of means, **μ**,*
  - *and another vector of standard deviations, **σ**.*

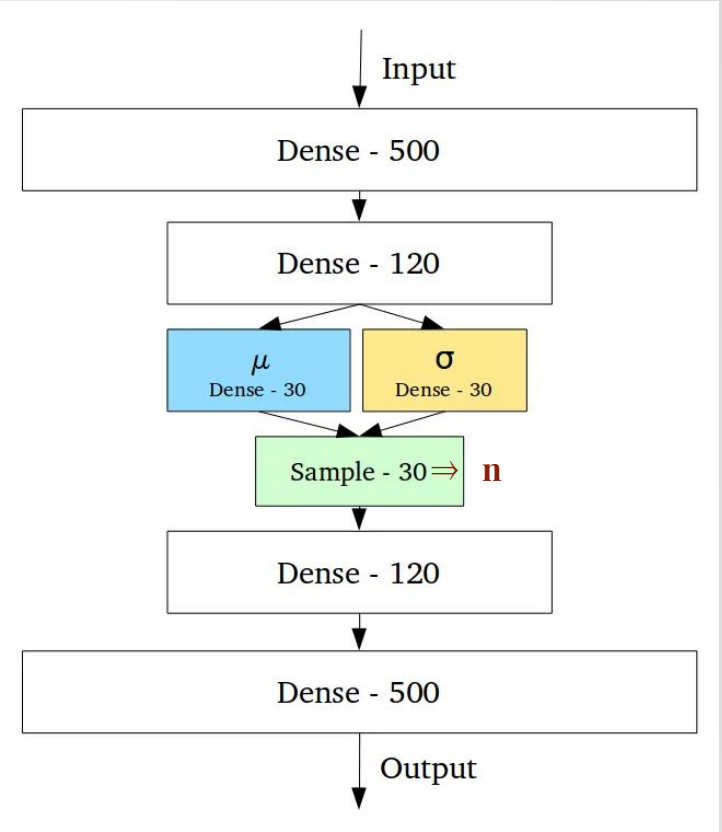
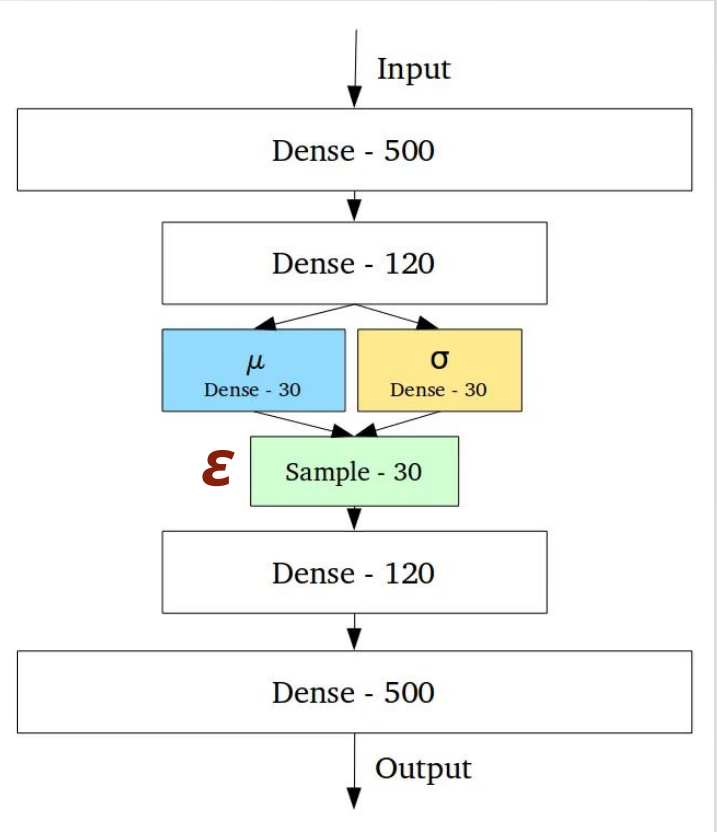


figure from https://towardsdatascien...

Variational AutoEncoders (VAEs)

- To achieve this, VAEs introduce two additional layers in the bottleneck: the mean layer and the standard deviation layer.
- These layers take the output of the previous bottleneck layer and output the mean vector (***μ***) and the standard deviation vector (***σ***) respectively.
- Specifically, we sample a vector ***ε*** from a standard normal distribution (zero-mean, uniti var), and sample **z,** accordingly.

$$z = \mu + \sigma * \varepsilon$$



Variational AutoEncoders (VAEs)

- Here, $\boldsymbol{\varepsilon}$ serves as a source of randomness and allows us to sample different points from the latent space during training or generation.
- Ok. Perfect. Forward run is stochastic! Great! But:

How is this random operation handled during training? Can you backpropagate a layer that creates a random variable?

$$z = \mu + \sigma * \varepsilon$$

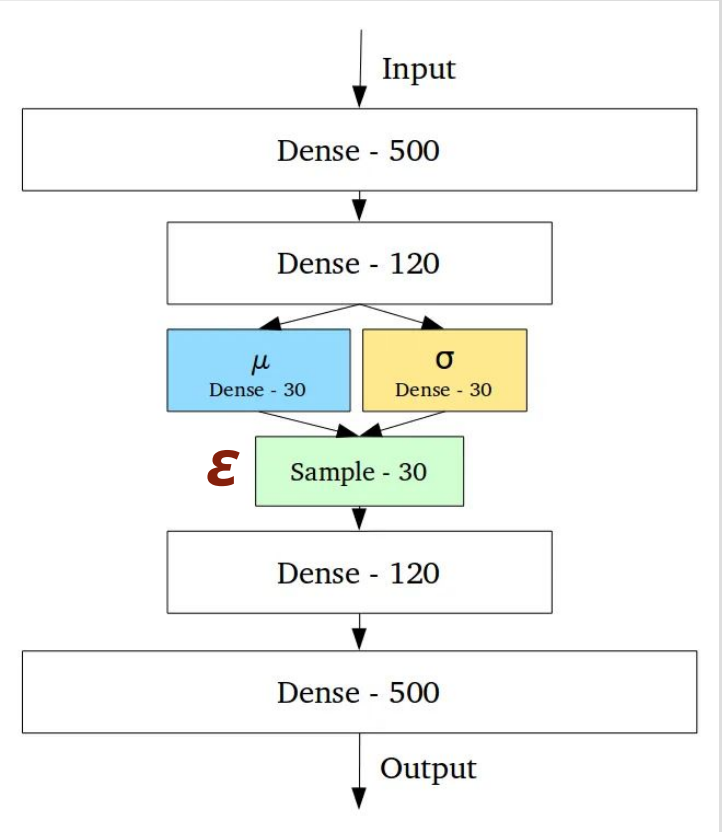
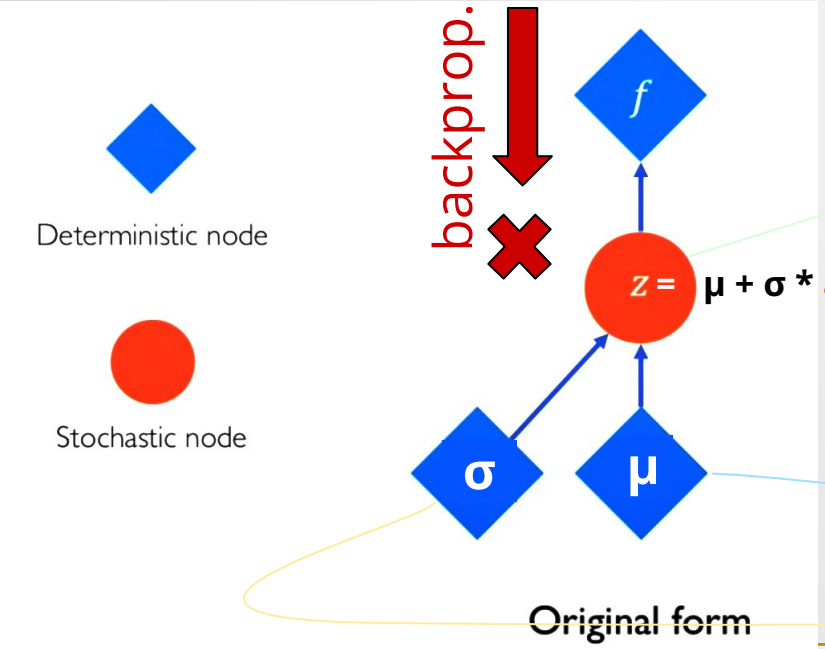


figure from https://towardsdatascien...

Reparameterization Trick

- The random sampling operation in VAEs, where **ε** is drawn from a standard normal distribution, introduces a challenge for backpropagation since it involves a non-differentiable operation.



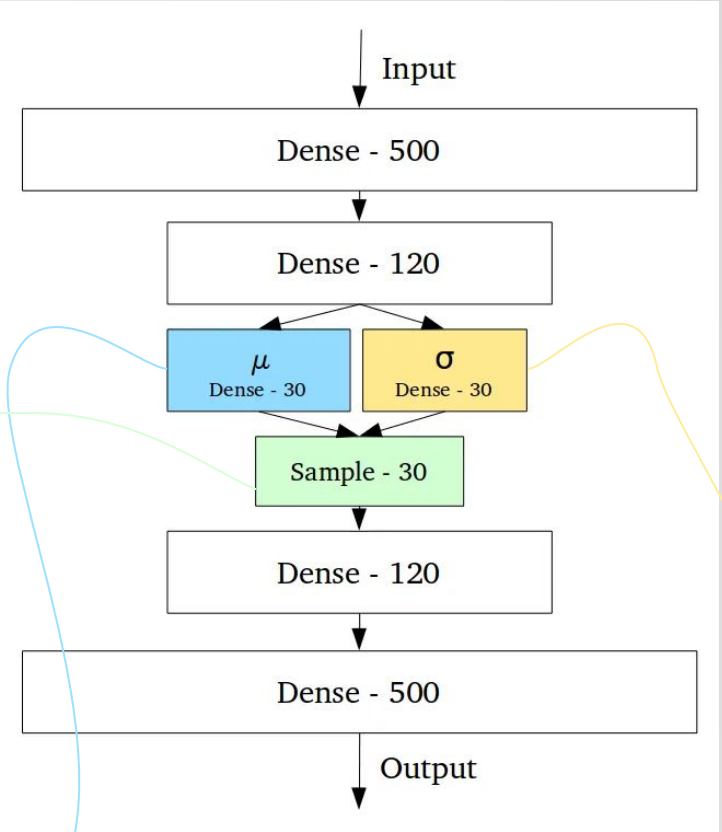
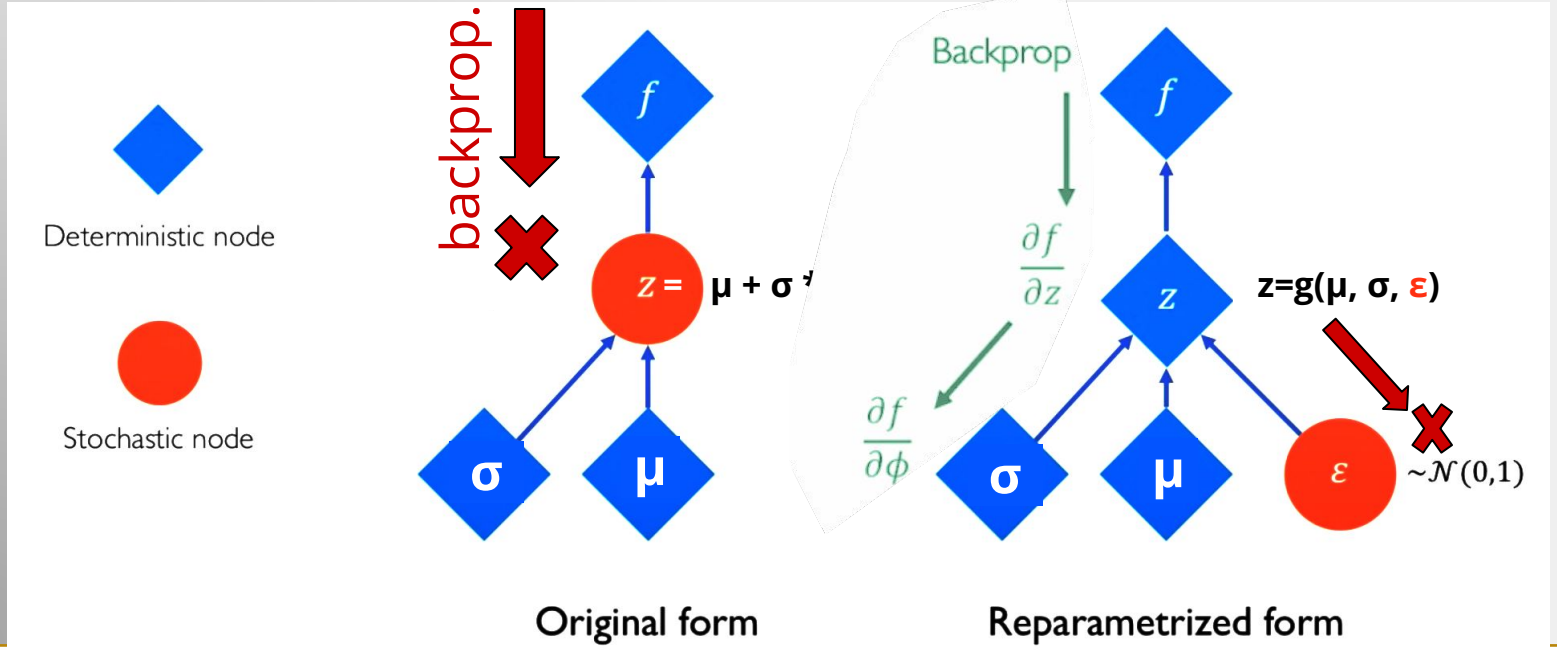


figure from http://introtodeeplearning....

Reparameterization Trick

- The random sampling operation in VAEs, where **ε** is drawn from a standard normal distribution, introduces a challenge for backpropagation since it involves a non-differentiable operation.



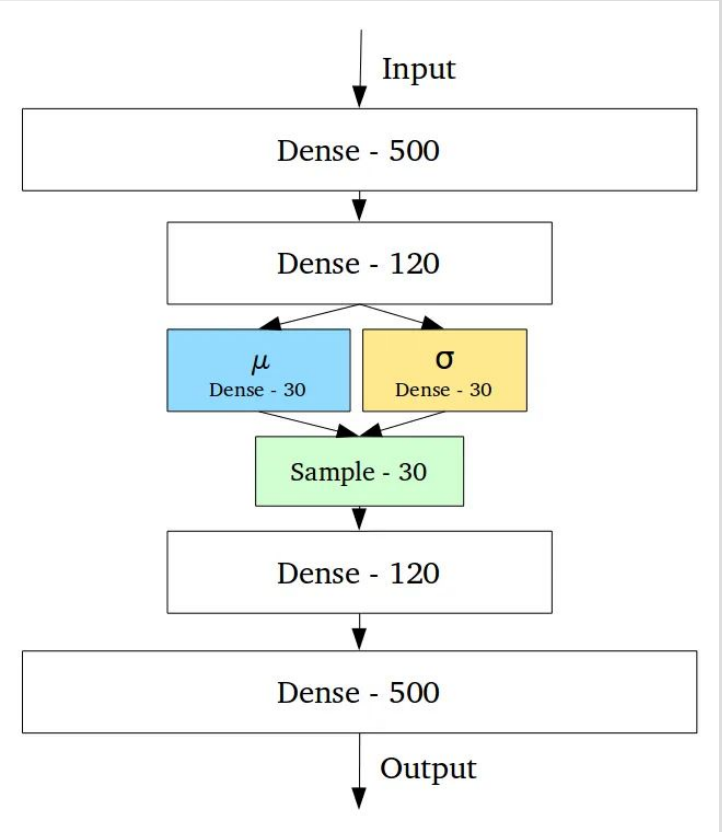


figure from http://introtodeeplearning....

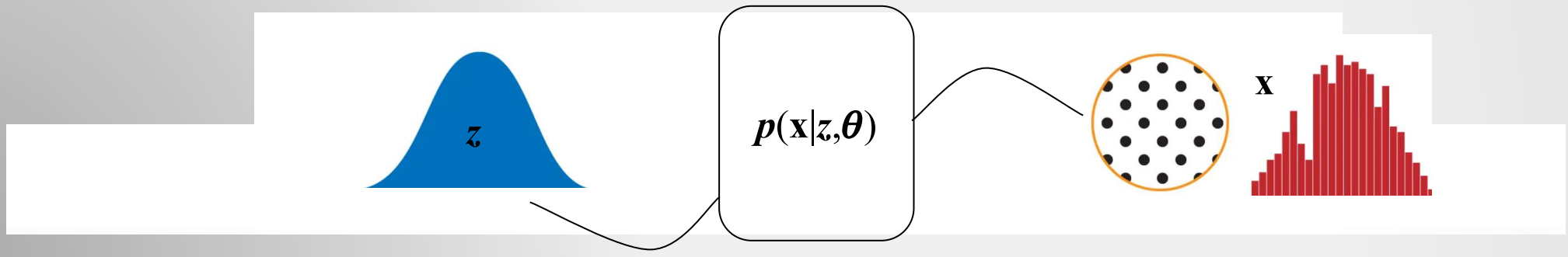
VAE Loss as ELBO

- To train a VAE, we need to maximize the likelihood of the observed data 'x' given the model parameters 'θ'. However, directly computing p(x | z, θ) is not feasible because we do not have access to the “true” latent variables 'z', and the encoding process is not perfectly reversible.
- Instead, VAEs use the Evidence Lower Bound (ELBO) as an approximation to the log-likelihood. The ELBO involves two terms: the expected log-likelihood of the data given the latent space (decoder part) and the negative KL divergence between the approximate posterior (encoded distribution) and the prior distribution over the latent space.

$$\mathcal{L}(\phi, \theta, x) = \boxed{\text{(reconstruction loss)}} + \text{(regularization term)}$$

VAE Loss as ELBO

- By maximizing the ELBO, the VAE effectively encourages the latent space to be structured and follow the prior distribution (usually a standard normal distribution), and it encourages the decoder to generate data that is similar to the observed data.
- The inability to directly calculate $p(x|z, \theta)$ due to the non-invertibility of the encoder is the reason why VAEs use ELBO for training instead of maximum likelihood.



VAE Loss

- So we know how to backprop a VAE. Ready for training then!
- During training, VAEs has two optimization goals:
  - minimize the reconstruction error (similar to traditional autoencoders)
  - and to ensure that the generated latent vectors follow the desired probability distribution.
- The first component is the reconstruction loss and it is trivial.

$$\mathcal{L}(\phi, \theta, x) = \boxed{\text{(reconstruction loss)}} + \text{(regularization term)}$$

VAE Reconstruction Loss

- Reconstruction loss is straightforward.

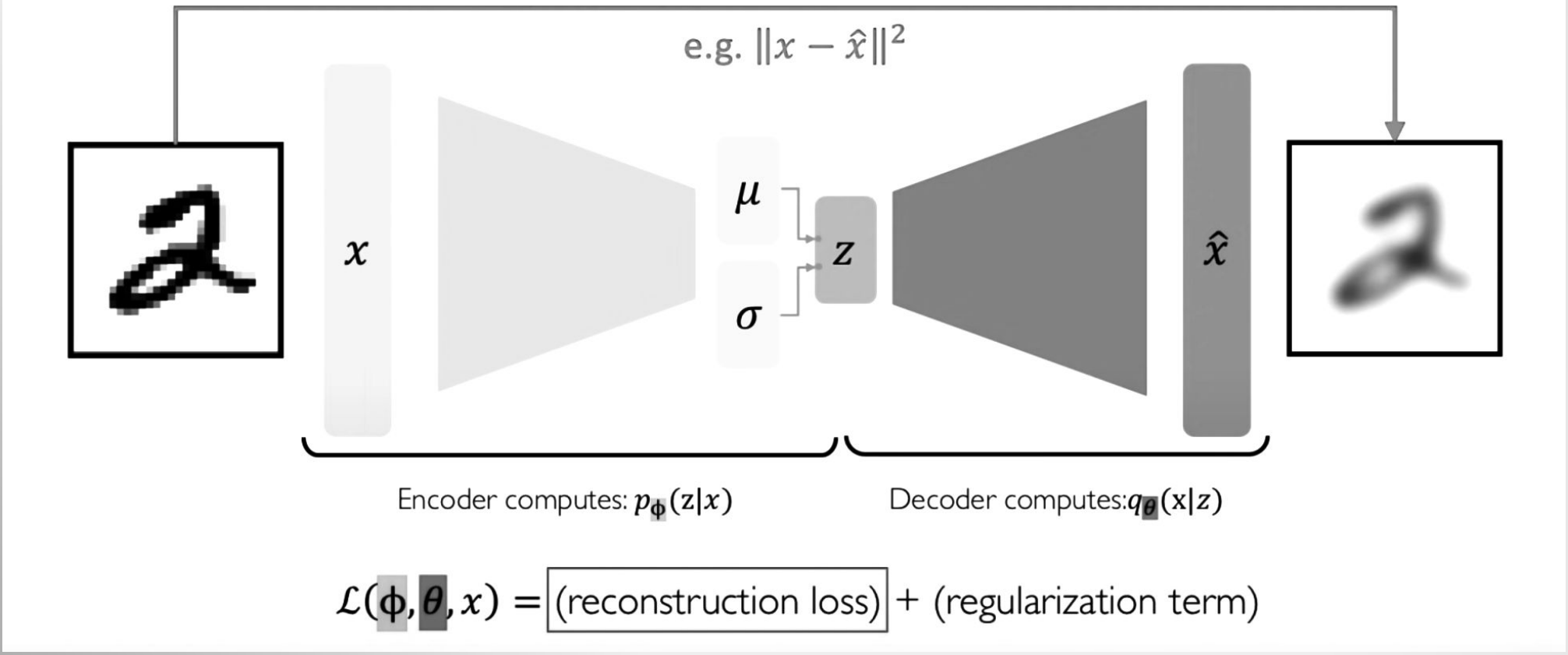


figure from http://introtodeeplearning....

VAE Regularization Loss

- So we know how to backprop a VAE. Ready for training then!
- During training, VAEs has two optimization goals:
  - minimize the reconstruction error (similar to traditional autoencoders)
  - and to ensure that the generated latent vectors follow the desired probability distribution.
- The second is done by including a regularization term in the loss function called the Kullback-Leibler (KL) divergence.
  - The KL divergence measures how different the distribution of the generated latent vectors is from the desired distribution (in this case, a standard normal distribution).

$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + \boxed{(\text{regularization term})}$$

Kullback-Leibler (KL) Divergence

- The KL divergence is a measure of the difference between two probability distributions, P and Q.
- It is defined as the expected value of the logarithmic difference between P and Q, where the expectation is taken with respect to P. The KL divergence is denoted as D(P||Q).

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

VAE Regularization Loss

- Regularization loss shapes the distributions.

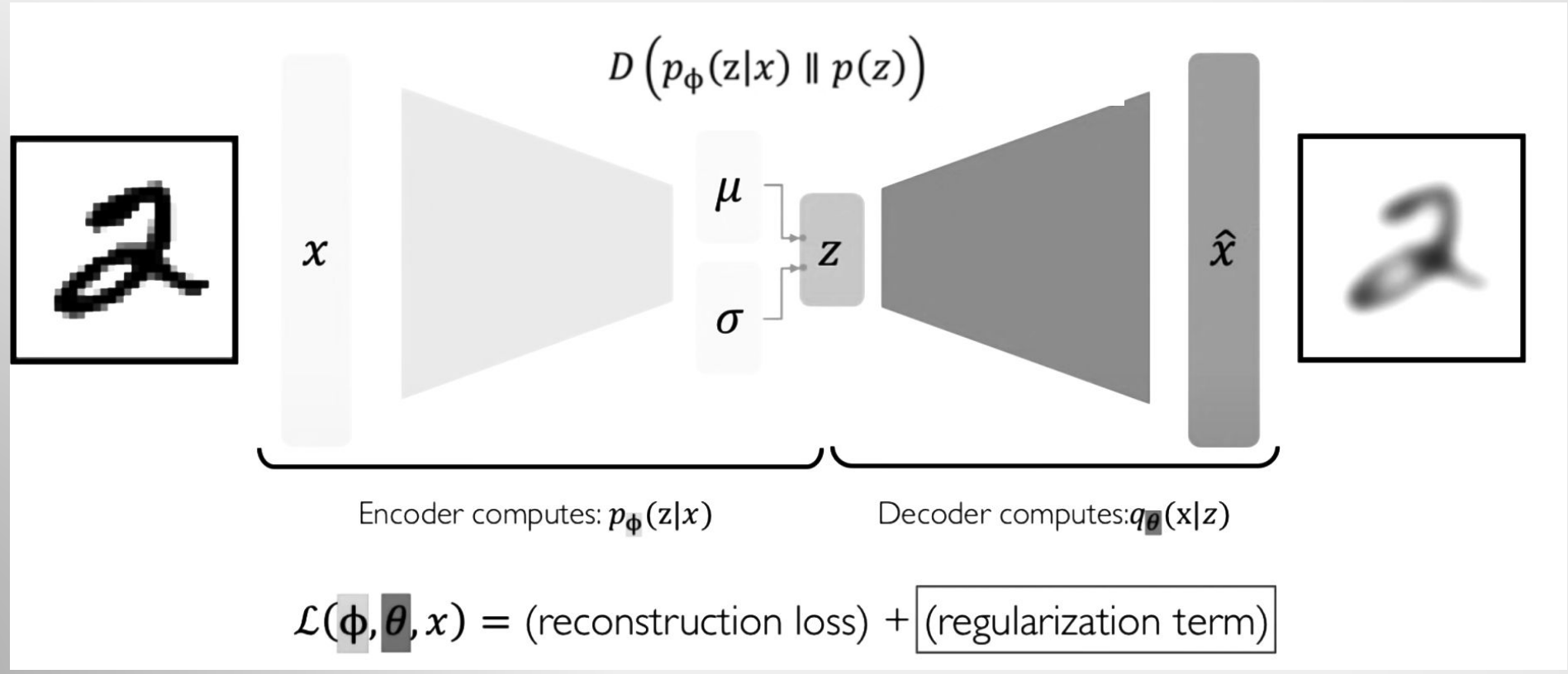
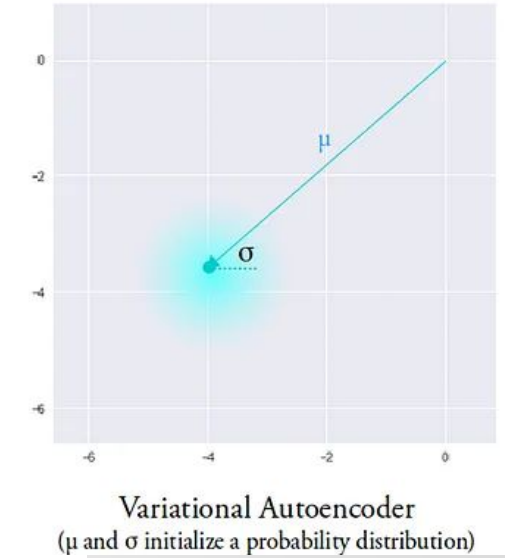


figure from http://introtodeeplearning....

VAE Regularization Loss

- Regularization loss shapes the distributions.



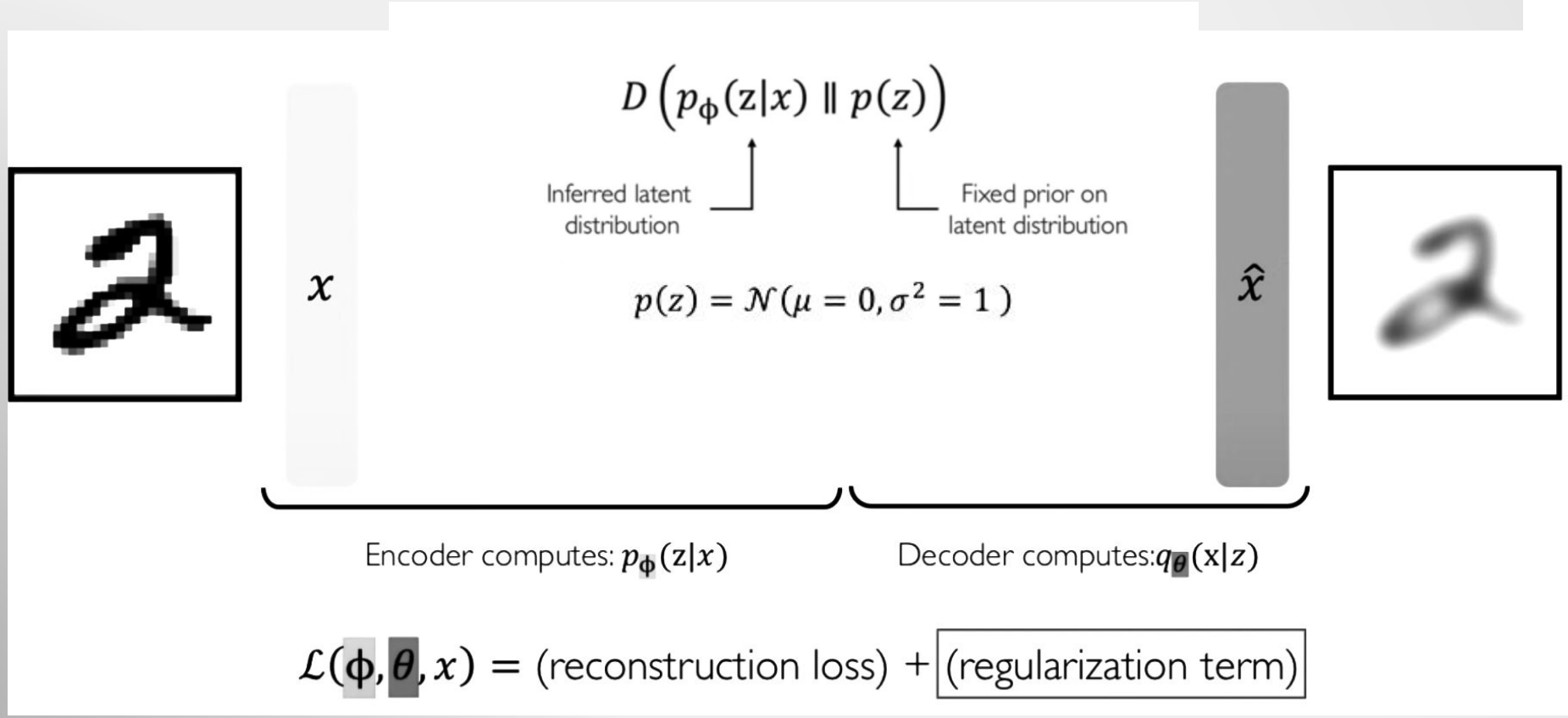


figure from http://introtodeeplearning....

VAE Regularization Loss

- Regularization loss shapes the distributions.

$$D\left(p_\phi(z|x) \parallel p(z)\right)$$

Inferred latent distribution

Fixed prior on latent distribution

- The KL divergence mea $p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$ ent the distribution of the generated latent vectors is from the desired distribution (in this case, a standard normal distribution).

Next lecture:

- PART III: Auto-Encoding

- Autoencoders and Dimensionality Reduction
- Variational Inference and VAEs
- **Conclusions**

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia