# Introduction to Deep learning

## Graduate School Orléans Numérique

Daniel Förster
UFR Sciences et Techniques Université d'Orléans

# (Variational) autoencoders

- Reminder: Simple autoencoder
- Problems of simple autoencoder as generative model
- Architecture of variational autoencoders
- Demo & example

# Artificial neural networks

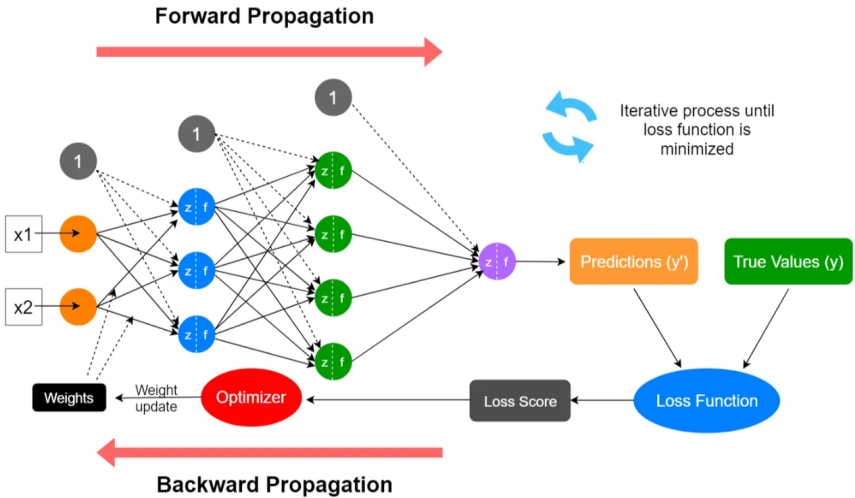



Image source [1]

# Autoencoder

- **Autoencoder Structure**
  - Composed of two main parts: encoder and decoder
  - Encoder compresses the input into a lower-dimensional code
  - Decoder reconstructs the input from the code
- **Learning Objective**
  - To minimize the difference between the input and its reconstruction
  - Achieved by using a loss function, such as Mean Squared Error (MSE)
- **Encoder and Decoder Functions**
  - Typically implemented as neural networks

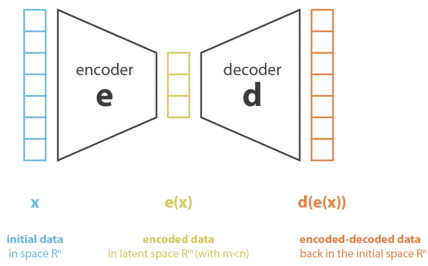



Image source [2]

# Idea for generative model

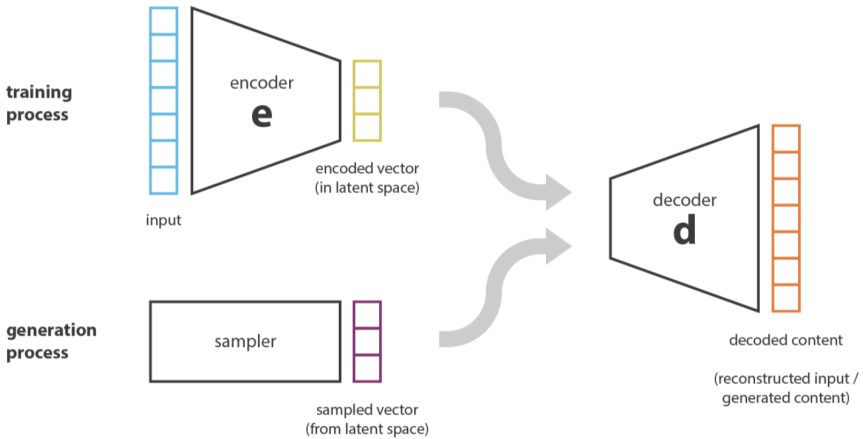
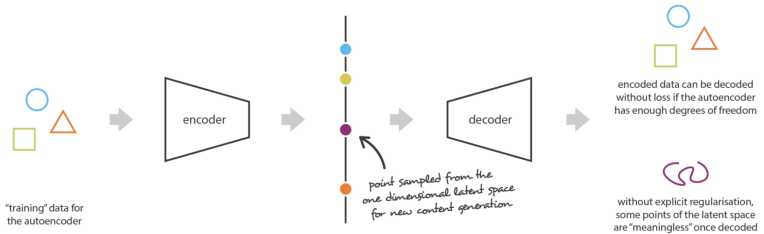



Image source [2]

# Problems with simple autoencoder




Regularity of the latent space is difficult, depending on:

- distribution of the data in the initial space
- the dimension of the latent space
- architecture of the encoder

$\rightarrow$ **Overfitting**, most of the latent space is meaningless no constraints on latent space during training

Image source [2]

# Variational autoencoders

Autoencoder, but training is **regularised**:

- avoiding overfitting
- latent space enables generative process


- **continuity:** proximity in latent space leads to similar decodings
- **completeness:** any point from the latent space gives "meaningful" decoding

Image source [2]

# How?

1. Input is encoded as a distribution over the latent space (and not as a single point)
2. Point from the latent space is sampled from that distribution
3. Sampled point is decoded and the reconstruction error computed (used for training)

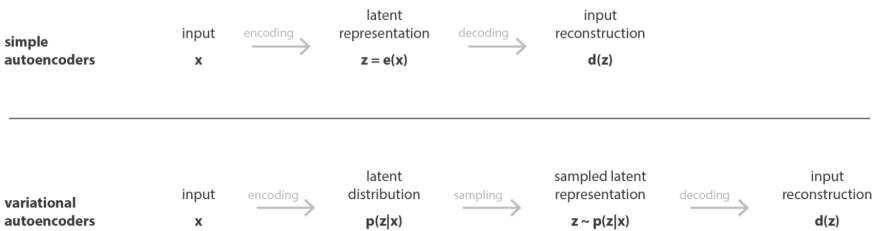



Image source [2]

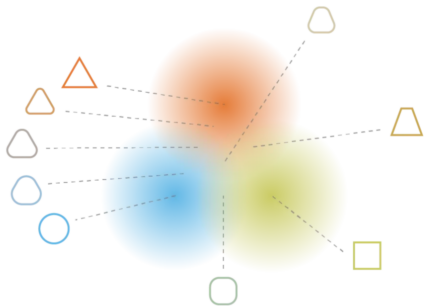
# Need for regularization


**Not enough:** Distributions may be far apart $\rightarrow$ similar to simple autoencoders
**We need instead:**

- Mean close to zero
- Covariance matrices to be close to the identity

Image source [2]

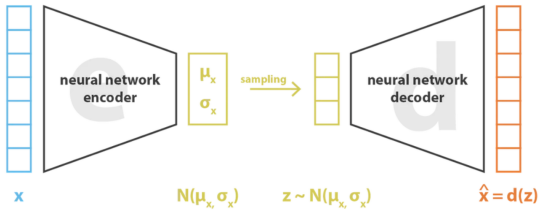
# Latent space



"Meaningful" latent space, with gradual transitions

Image source [2]

# Modified loss function




$$\text{loss} = C\,|x - \hat{x}|^2 + D_{\text{KL}}\left[N\left(\mu_x, \sigma_x\right) \parallel N\left(0, 1\right)\right]$$

with

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Kullback–Leibler divergence $D_{\text{KL}}(P \parallel Q)$: measure of similarity if $P$ and $Q$

$\rightarrow$ price to pay: higher reconstruction error

Image source [2]

# Reparametrization trick

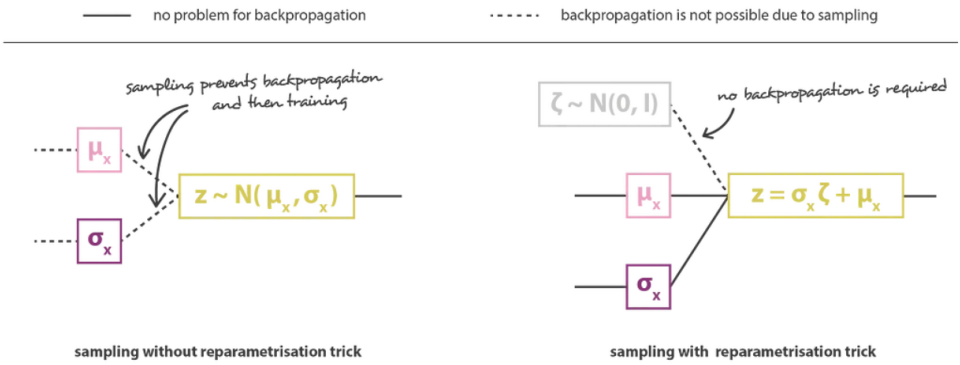
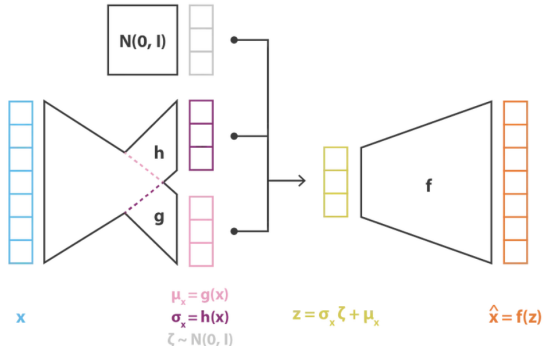



Image source [2]

# Architecture overview of the VAE




- $g(x)$ Mean of the Gaussian distribution
- $h(x)$ Covariance of the Gaussian distribution

Image source [2]

# Summary

**Simple autoencoders:**

- consists of encoder, latent space, and decoder
- loss: just linked to reconstruction (no regard to latent space)
- dimensionality reduction / compression
- anomaly detection

**Variational autoencoders:**

- distribution over the latent space instead of a single point
- loss function: a reconstruction term and a regularisation term
- useful for generative processes

# Natural language processing

- Tokenization and Byte Pair Encoding (BPE).
- Transformer architecture
- Attention mechanism
- “Scaling laws”
- Fine-tuning and reinforcement learning from human feedback
- Outlook

# Introduction: Example

The restaurant refused to serve me a ham sandwich because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambiance was just as good as the food and service.

**Questions:**

- Is the review positive?
- Is the restaurant serving ham sandwiches?

**Observation:**

- Meaning of words depends on context
- Long-range correlations
- Text can have any length

# Byte Pair Encoding (BPE)

**How it works**:

1. Start with characters as base vocabulary.
2. Iteratively merge the most frequent adjacent pairs to form subwords
3. Stop when the desired vocabulary size is reached

**Why use BPE?**

- Handles out-of-vocabulary words by breaking them into subwords
- Balances between word-level and character-level tokenization
- Ability to predefine vocabulary size

| Iteration | Sequence | Vocabulary |
|---|---|---|
| 0 | a b a b c a b c | {a, b, c} |
| 1 | ab ab c ab c | {a, b, c, ab} |
| 2 | ab abc abc | {a, b, c, ab, abc} |
| 3 | ababc abc | {a, b, c, ab, abc, ababc} |
| 4 | ababcabc | {a, b, c, ab, abc, ababc, ababcabc} |

Image source [1]

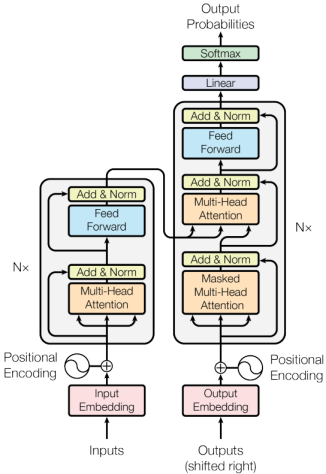
# Tokenization: Why it matters

**Tokenization is at the heart of much weirdness in LLMs**

The reason LLMs struggle with:

- counting letters in words
- simple string tasks like reversing a string
- non-English languages (e.g., Japanese)
- simple arithmetic
- coding in Python

# Transformer architecture




- Sequence-to-sequence learning, but no recurrence (during training) or internal state
- Attention mechanism
- Amenable to parallelization
- Possibilities: encoder-only, decoder-only, encoder-decoder

$\rightarrow$ Almost human-level translation

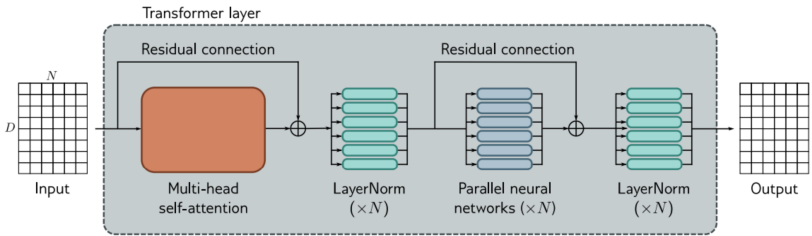
Image source [2]

# Decoder-only transformer

**Input**: Embedded tokenized sequence and positional embeddings

**Central part**:

- Masked Multi-Head Attention ("casual self-attention")
- Feedforward Network
- Layer norm
- Residual connections




**Outputs** Probability distribution over the vocabulary for predicting the next token

# Attention mechanism

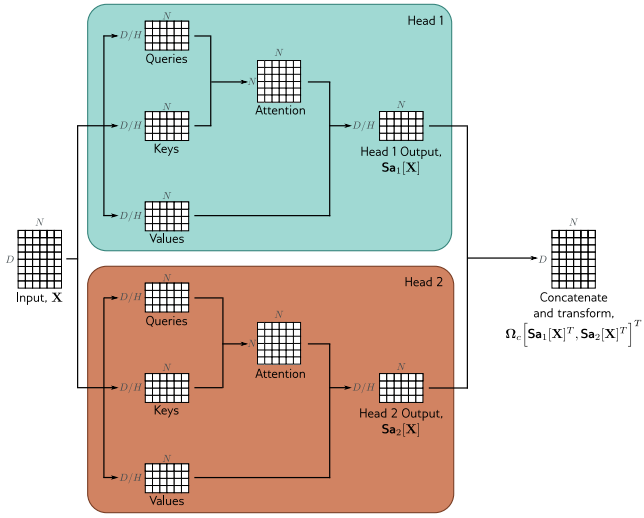
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- **Inputs**: Projected into three vectors: Query ($Q$), Key ($K$), and Value ($V$).
- **Attention score**: Relevance between $Q$ and $K$.
- **Weighted sum**: Combine $V$ values using scores.
- Softmax ensures the scores are normalized.
- Scaling factor $\sqrt{d_k}$ for normalization of activations

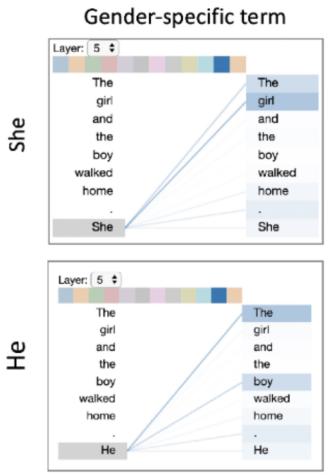
# Multi-head attention

- **Why use multiple attention heads?**
  - Each head focuses on different parts of the sequence simultaneously.
- **How it works:**
  - Split $Q$, $K$, and $V$ into smaller parts for each attention head.
  - Compute attention independently for each head.
  - Concatenate and project the results to the output space.

# Attention overview



# Attention example

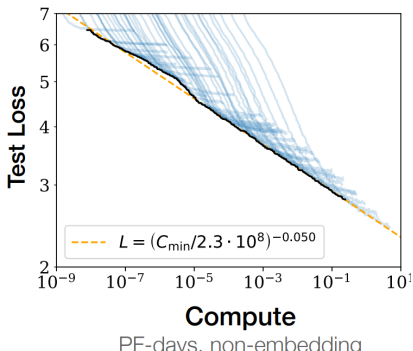


# GPT-2

- Next word prediction
- Can continue writing a text creatively, but not really answer questions or engage in conversation
- Can be “interviewed”
- However, tricks are possible: examples or “few shot” learning, summarization via “tl;dr”

$\rightarrow$ Transformers are multitask learners

# Scaling Laws




- From GPT-1 and GPT-2: larger models and more training data yields better results
- Scaling Laws: allows for evaluation and dimensioning of large models

Image source [4]

# Chat-GPT: Making GPT-3 more useful

- Next word prediction is not good for question answering
- Return useful information
- Avoid “hallucinations”, but retain creativity
- Refuse to answer certain questions (how to make a bomb, etc.)

# Solution: Reinforcement learning from human feedback

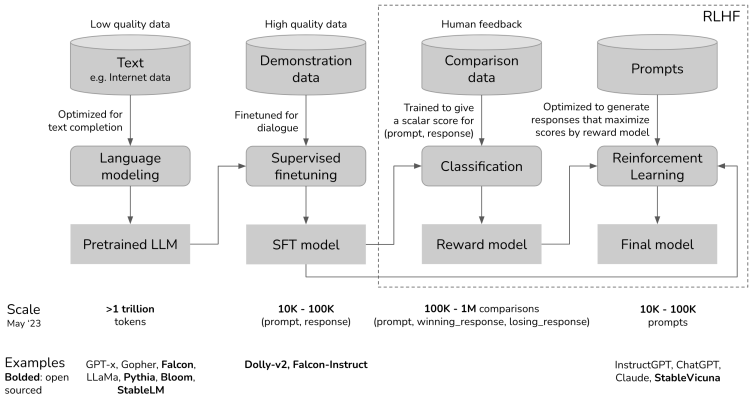



**Image source [5]**

# Current limitations

- Difficulty to distinguish fact from fiction
- Number of layers limits the number of steps
- Lack of explainability/interpretability

**Partial solutions:**

- Usage of "tools": e.g. web search (problem of trust) or python interpreter
- "Chain of thought" prompting and verification, GPT becomes Turing complete
- "Multi-modality": include images in the training data
- "Mechanistic interpretability": Identify features that correspond to certain concepts

# Fundamental Limitations

**Unpredictability of chaotic systems**

- *Weather Prediction:* long-term forecasting due to the chaotic nature of weather is *impossible*
- *Stock Markets:* Difficulty in predicting market movements accurately due to complex, multifactorial, and *unknowable* influences

**NP complete problems**

- Travelling salesman problem
- Elliptic curve cryptography