Deep Learning: advanced models

Generative models, Transformers, GNNs...

Nicolas Keriven
CNRS, IRISA, Rennes

ENSTA 2024

Pros and cons of deep learning

Deep Neural Nets are :

- **Data-hungry** : they *will not* work with only a few data

Pros and cons of deep learning

Deep Neural Nets are :

- **Data-hungry** : they *will not* work with only a few data
- **Power-hungry** : computationally expensive to train

Pros and cons of deep learning

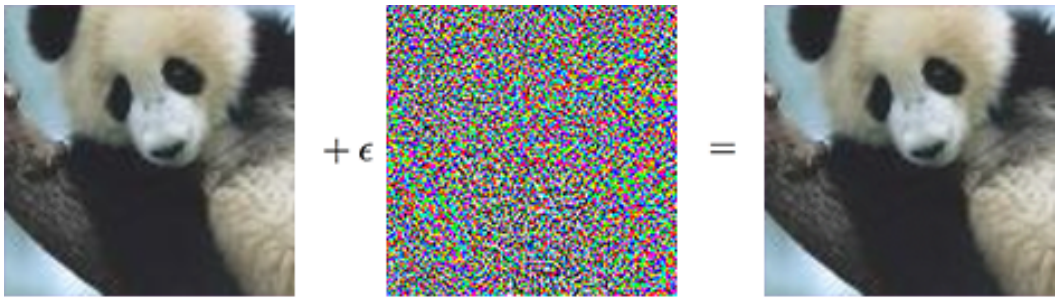
Deep Neural Nets are :

- **Data-hungry** : they *will not* work with only a few data
- **Power-hungry** : computationally expensive to train
- **Difficult to interpret** : hard to get confidence intervals, generalization guarantees...

Pros and cons of deep learning

Deep Neural Nets are :

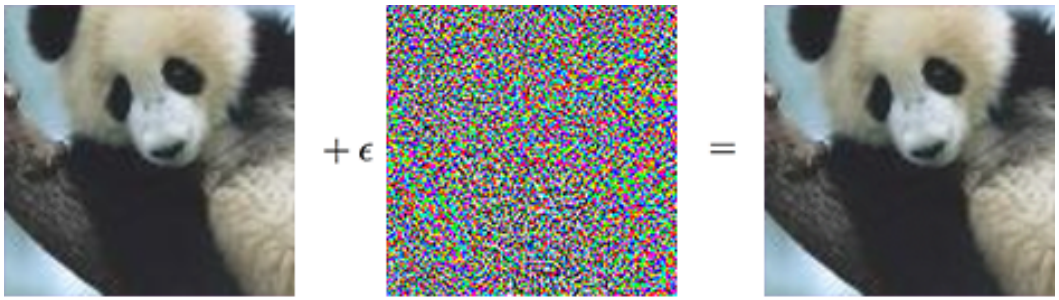
- **Data-hungry** : they *will not* work with only a few data
- **Power-hungry** : computationally expensive to train
- **Difficult to interpret** : hard to get confidence intervals, generalization guarantees...
- **Unstable** : a micro-change in the data can fool them. Such change almost never happens in practice, but can be engineered by a malicious party !



Pros and cons of deep learning

Deep Neural Nets are :

- **Data-hungry** : they *will not* work with only a few data
- **Power-hungry** : computationally expensive to train
- **Difficult to interpret** : hard to get confidence intervals, generalization guarantees...
- **Unstable** : a micro-change in the data can fool them. Such change almost never happens in practice, but can be engineered by a malicious party !



- A **bazooka**, when sometimes all you need to do is kill a fly ! In many real-life scenarii, linear models, PCA, k-means... work amazingly well

Pros and cons of deep learning

But ! Deep Neural Nets :

- are easy to implement with modern libraries, with automatic differenciation and GPU compatibility completely transparent but there are many tricks to train them ! Can be frustrating...

Pros and cons of deep learning

But ! Deep Neural Nets :

- are easy to implement with modern libraries, with automatic differenciation and GPU compatibility completely transparent but there are many tricks to train them ! Can be frustrating...

- work mysteriously well for problems we thought out-of-reach for decades !

Pros and cons of deep learning

But ! Deep Neural Nets :

- are easy to implement with modern libraries, with automatic differenciation and GPU compatibility completely transparent but there are many tricks to train them ! Can be frustrating...
- work mysteriously well for problems we thought out-of-reach for decades !
- are amazingly flexible ! You can plug "anything anywhere", and it will almost always "work". Huge playground ! This can be dangerous ! Always be aware of what you do, or bugs will become impossible to fix.

Pros and cons of deep learning

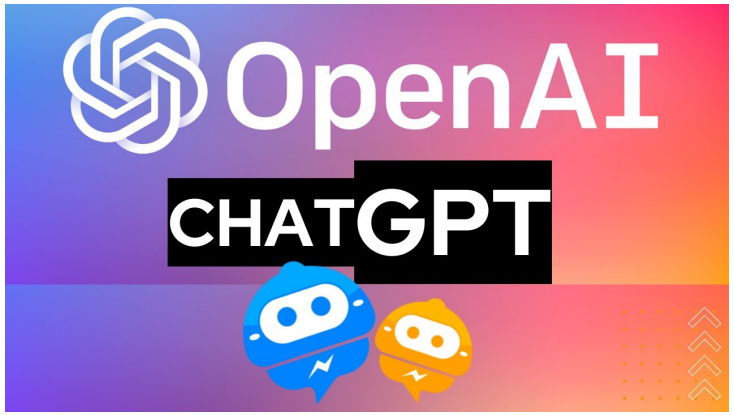
But ! Deep Neural Nets :

- are easy to implement with modern libraries, with automatic differenciation and GPU compatibility completely transparent but there are many tricks to train them ! Can be frustrating...
- work mysteriously well for problems we thought out-of-reach for decades !
- are amazingly flexible ! You can plug "anything anywhere", and it will almost always "work". Huge playground ! This can be dangerous ! Always be aware of what you do, or bugs will become impossible to fix.
- Today, we are going to see a few variants of DNNs, without going into details. Many ressources are available online.

Table of Contents

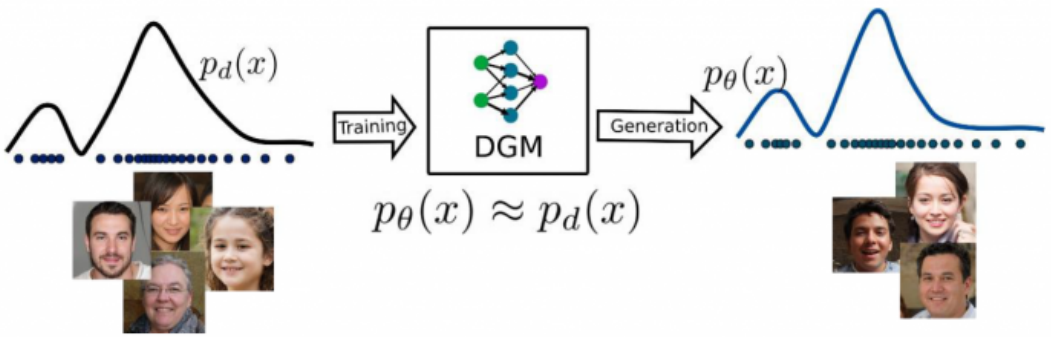
Generative model

- Goal : generate data ressembling other data.
- Extremely hot topic right now ! Text generation, image generation...



Basic generative model : density estimation !

- Most basic problem : observe samples from a distribution, generate new samples not observed before
- Just a *density estimation* problem ! But the probability distribution is far too complicated to be modelled explicitely (eg images)



- Basic idea : take a very simple distribution in a latent space, eg $z \sim \mathcal{N}(0, Id)$, train a complicated function f_θ such that $f_\theta(z)$ are good samples.

Table of Contents

Generative Adversarial Networks (GANs)

- Q : how to train a Neural Network f_θ such that $f_\theta(z)$ produces "realistic images" different from the training set ? Images are already hard to understand !

Generative Adversarial Networks (GANs)

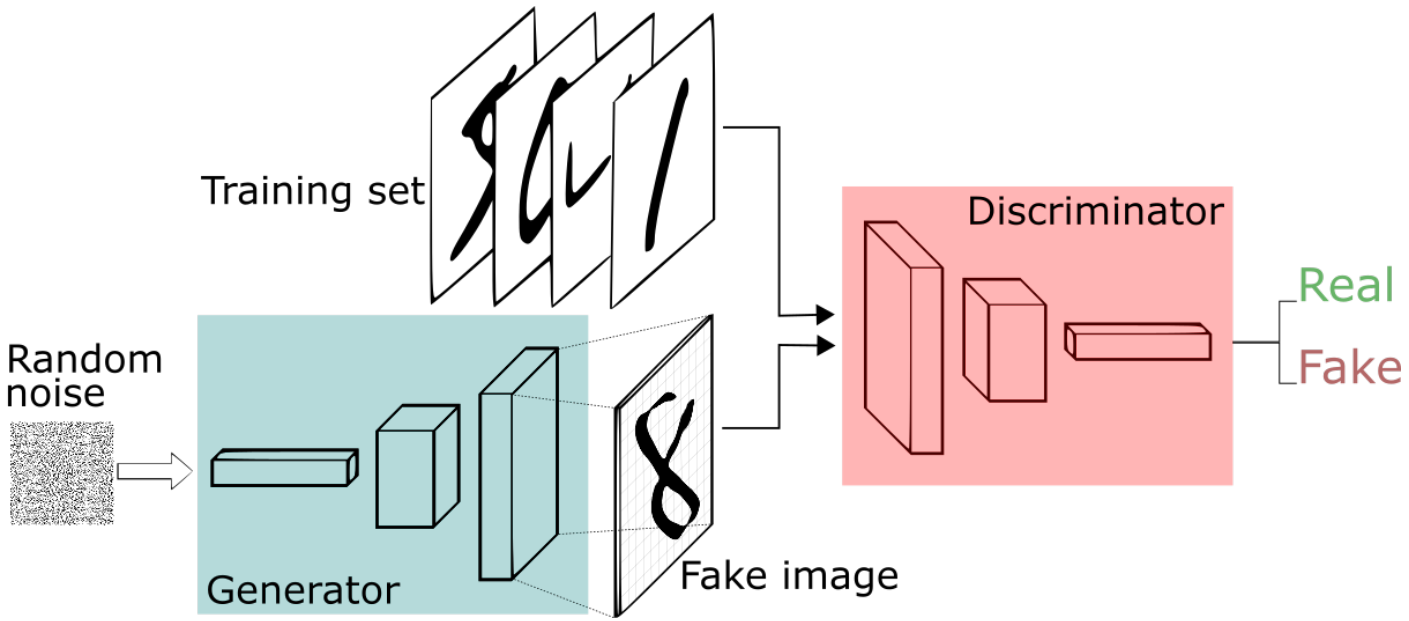
- Q : how to train a Neural Network f_θ such that $f_\theta(z)$ produces "realistic images" different from the training set ? Images are already hard to understand !
- Breakthrough idea : only Neural Networks are capable of automatically understanding images... well, let's train two CNNs ! One to produce the images, one to criticize them, and make them compete against each other ! Generative Adversarial Networks (Goodfellow et al. 2014)

Generative Adversarial Networks (GANs)

- Q : how to train a Neural Network f_θ such that $f_\theta(z)$ produces "realistic images" different from the training set ? Images are already hard to understand !
- Breakthrough idea : only Neural Networks are capable of automatically understanding images... well, let's train two CNNs ! One to produce the images, one to criticize them, and make them compete against each other ! Generative Adversarial Networks (Goodfellow et al. 2014)

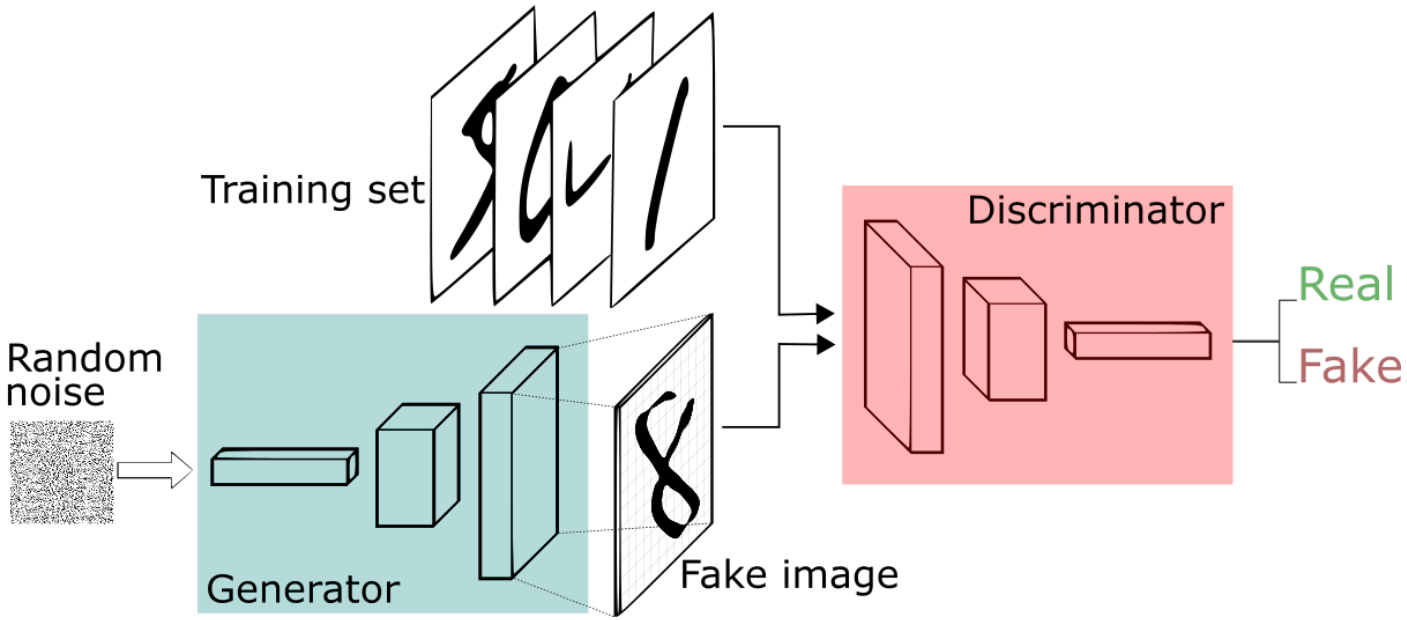
GAN description

- Train a generator network G_{θ_g} such that $G_{\theta_g}(z)$, with z Gaussian, are good samples
- Train a discriminator network D_{θ_d} to classify the samples as being from the training set or from the generator



GAN description

- Train a generator network G_{θ_g} such that $G_{\theta_g}(z)$, with z Gaussian, are good samples
- Train a discriminator network D_{θ_d} to classify the samples as being from the training set or from the generator
- Make them "compete" : the discriminator will become better at classifying the fake samples, the generator will become better at fooling the discriminator



GAN training

- The original GAN paper presents the problem as a two-players minimax game, based on the logistic regression loss :

$$\min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x \sim p_{data}} [\log D_{\theta_d}(x)] + \mathbb{E}_{z \sim \mathcal{N}} [\log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

GAN training

- The original GAN paper presents the problem as a two-players minimax game, based on the logistic regression loss :

$$\min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x \sim p_{data}} [\log D_{\theta_d}(x)] + \mathbb{E}_{z \sim \mathcal{N}} [\log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

  - $D(x)$ represent the probability that x is a true sample. D wants to maximize the log-likelihood, ie, outputting 1 for x_i and 0 for $G(z)$.

GAN training

- The original GAN paper presents the problem as a two-players minimax game, based on the logistic regression loss :

$$\min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x \sim p_{data}} [\log D_{\theta_d}(x)] + \mathbb{E}_{z \sim \mathcal{N}} [\log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

  - $D(x)$ represent the probability that x is a true sample. D wants to maximize the log-likelihood, ie, outputting 1 for x_i and 0 for $G(z)$.
  - G want to minimize the log-likelihood, ie, make $G(z)$ ressemble the distribution of the data

GAN training

- The original GAN paper presents the problem as a two-players minimax game, based on the logistic regression loss :

$$\min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x \sim p_{data}}[\log D_{\theta_d}(x)] + \mathbb{E}_{z \sim \mathcal{N}}[\log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

  - $D(x)$ represent the probability that x is a true sample. D wants to maximize the log-likelihood, ie, outputting 1 for x_i and 0 for $G(z)$.
  - G want to minimize the log-likelihood, ie, make $G(z)$ ressemble the distribution of the data

Original algo : alternate k steps of gradient ascent for D with one step of gradient descent for G

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

GAN issues

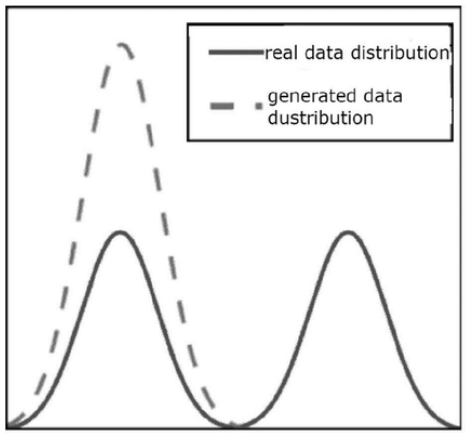
- Hard to train : minimax problems are typically hard !
  - they can loop
  - saddle points are unstable
  - convergence is (very) slow, etc.

GAN issues

- Hard to train : minimax problems are typically hard !
  - they can loop
  - saddle points are unstable
  - convergence is (very) slow, etc.
- Data-hungry : GANs require a **lot** of data to be trained properly

GAN issues

- Hard to train : minimax problems are typically hard !
  - they can loop
  - saddle points are unstable
  - convergence is (very) slow, etc.
- Data-hungry : GANs require a **lot** of data to be trained properly
- Mode collapse : the generator produces limited diversity (eg, for a dataset with cats and dogs, produces only dogs)



(a)


(b)

Sota GANs

There are many (many!) tricks and variants to improve GANs. Mostly, bigger models with more data seem to be always better! (BigGAN below). Seemed a bit depressing for a time...

Sota GANs

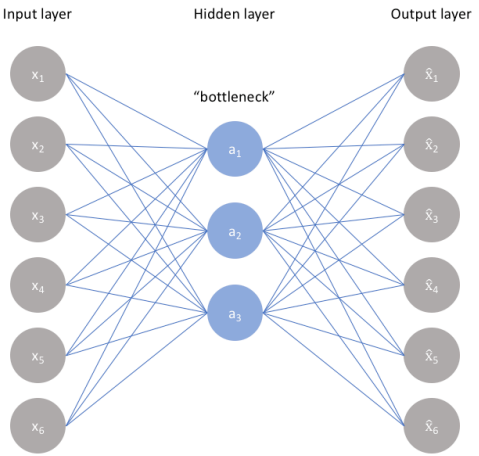
There are many (many !) tricks and variants to improve GANs. Mostly, bigger models with more data seem to be always better ! (BigGAN below). Seemed a bit depressing for a time...

But nowadays, they have been mostly replaced with more modern approaches like diffusion models.

Table of Contents

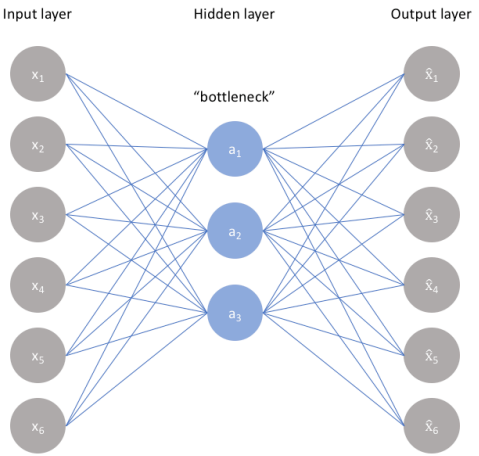
Auto-Encoder

- We have seen Auto-Encoder :
  - Encoder : images $\rightarrow$ compressed latent space
  - Decoder : latent space $\rightarrow$ image



Auto-Encoder

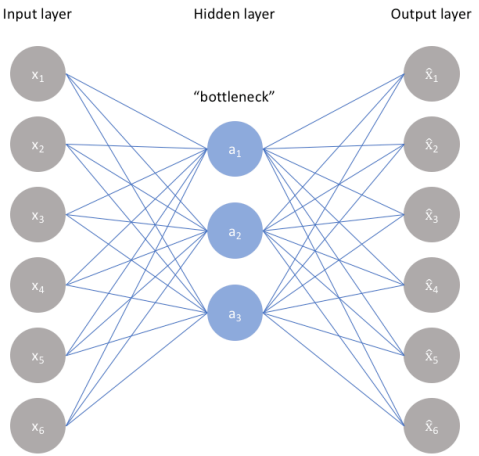
- We have seen Auto-Encoder :
  - Encoder : images $\rightarrow$ compressed latent space
  - Decoder : latent space $\rightarrow$ image



- Idea : by working directly in the latent space, the decoder is a generative model !

Auto-Encoder

- We have seen Auto-Encoder :
  - Encoder : images $\rightarrow$ compressed latent space
  - Decoder : latent space $\rightarrow$ image



- Idea : by working directly in the latent space, the decoder is a generative model !
- Difficulty : sampling a random vector in the latent space might produce garbage. The latent space is highly irregular !

Variation Auto-Encoder (VAE) Kingma and Welling, 2014

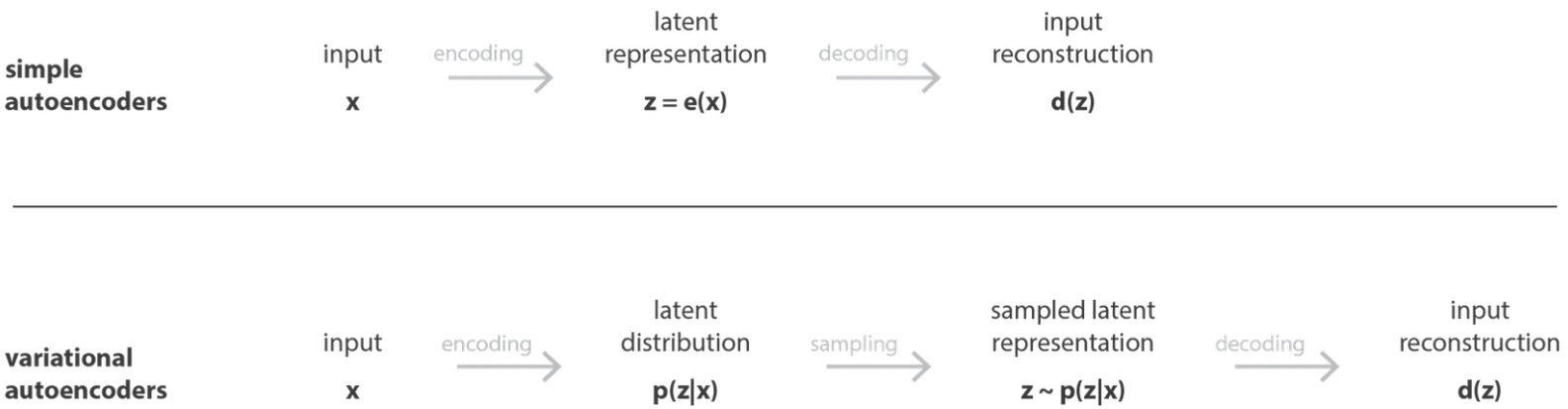
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

- Idea : regularize the distribution of the encodings, such that the latent space is smoother, and new sample can be produced

Variation Auto-Encoder (VAE) Kingma and Welling, 2014

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

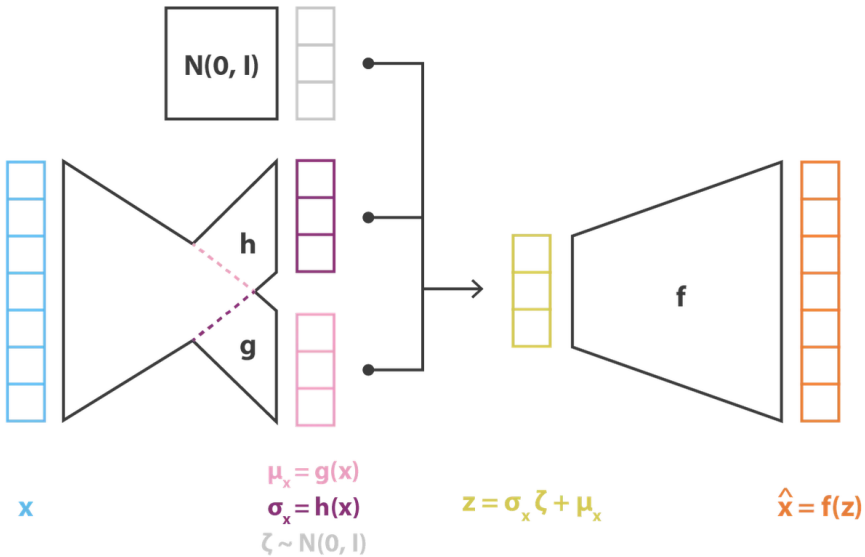
- Idea : regularize the distribution of the encodings, such that the latent space is smoother, and new sample can be produced
- To do so :
  - encode inputs as distributions, and feed sample from these distributions to the decoder.



Variation Auto-Encoder (VAE) Kingma and Welling, 2014

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

- Idea : regularize the distribution of the encodings, such that the latent space is smoother, and new sample can be produced
- To do so :
  - encode inputs as distributions, and feed sample from these distributions to the decoder.
  - Add a regularizing term (Kullback-Leibler divergence) on these distributions.



$$\text{loss} = C \,||\, x - \hat{x} \,||^2 + KL[\, N(\mu_x, \sigma_x), N(0, I)\,] = C \,||\, x - f(z) \,||^2 + KL[\, N(g(x), h(x)), N(0, I)\,]$$

Variation Auto-Encoder (VAE) Kingma and Welling, 2014

- Desired properties : continuity (two close points in the latent space should give close outputs) and completeness (sampling any point should give a reasonable result)

Variation Auto-Encoder (VAE) Kingma and Welling, 2014

- Desired properties : continuity (two close points in the latent space should give close outputs) and completeness (sampling any point should give a reasonable result)
- Connection with Variational Inference (hence the name)

Variation Auto-Encoder (VAE) Kingma and Welling, 2014

- Desired properties : continuity (two close points in the latent space should give close outputs) and completeness (sampling any point should give a reasonable result)
- Connection with Variational Inference (hence the name)
- Easier to train than GANs, but output might have less diversity

Table of Contents

Diffusion models

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

- Diffusion models have take the generative world by storm and start to replace GANs and VAEs nowadays (Stable Diffusion, Midjourney, DALL-E, Imagen...), due to their performance and flexibility

Diffusion models

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

- Diffusion models have take the generative world by storm and start to replace GANs and VAEs nowadays (Stable Diffusion, Midjourney, DALL-E, Imagen...), due to their performance and flexibility
- Diffusion model were *not* invented as generative models, and the idea seems a bit ludicrous at first glance !

Diffusion models

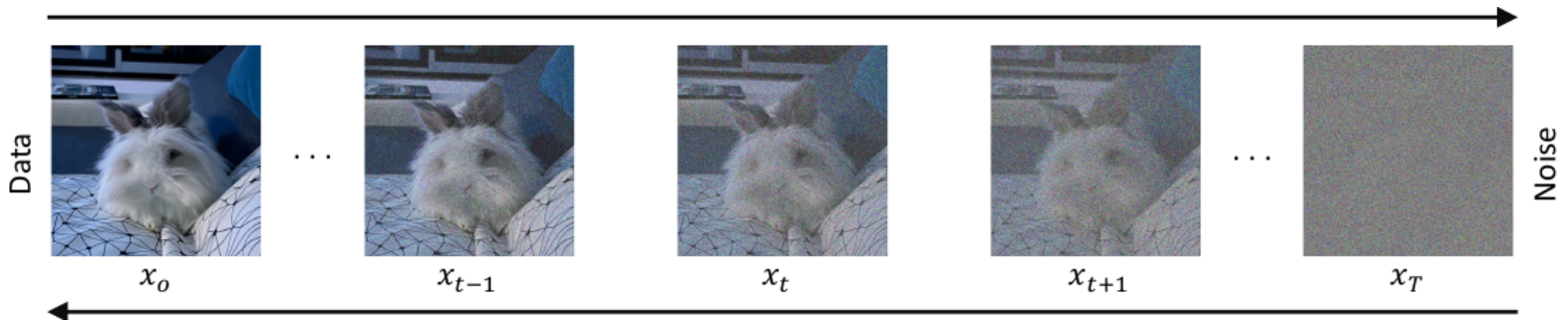
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

- Diffusion models have take the generative world by storm and start to replace GANs and VAEs nowadays (Stable Diffusion, Midjourney, DALL-E, Imagen...), due to their performance and flexibility
- Diffusion model were *not* invented as generative models, and the idea seems a bit ludicrous at first glance !
- Idea (roughly) : train a model to denoise an image, at many different noise levels. Then, start with random noise (!), and denoise it until an image is produced !

Diffusion models

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

- Diffusion models have take the generative world by storm and start to replace GANs and VAEs nowadays (Stable Diffusion, Midjourney, DALL-E, Imagen...), due to their performance and flexibility
- Diffusion model were *not* invented as generative models, and the idea seems a bit ludicrous at first glance !
- Idea (roughly) : train a model to denoise an image, at many different noise levels. Then, start with random noise (!), and denoise it until an image is produced !
- Interpretation : we learn to "reverse" the diffusion process that gradually transforms an image into pure noise

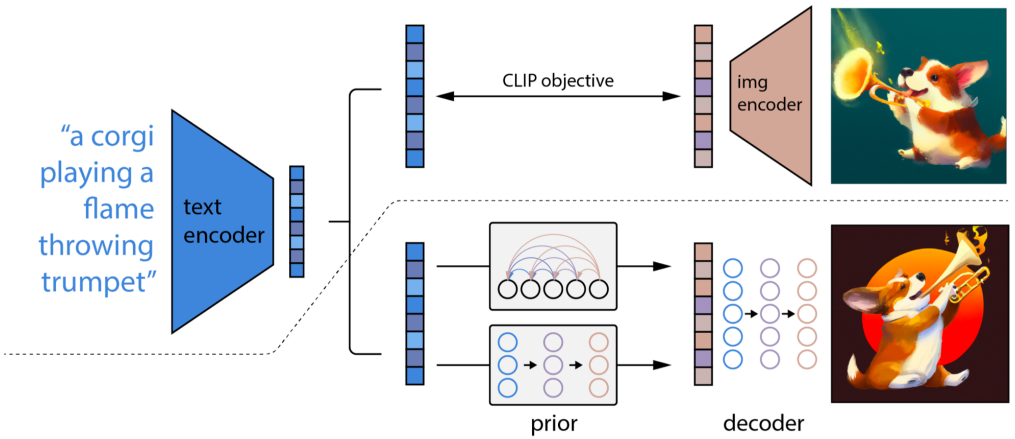


Diffusion models : pros

- Pro : Flexibility
  - denoising can happen in the latent space ; somehow, learn to "project" random noise to a point in the latent space that makes sense

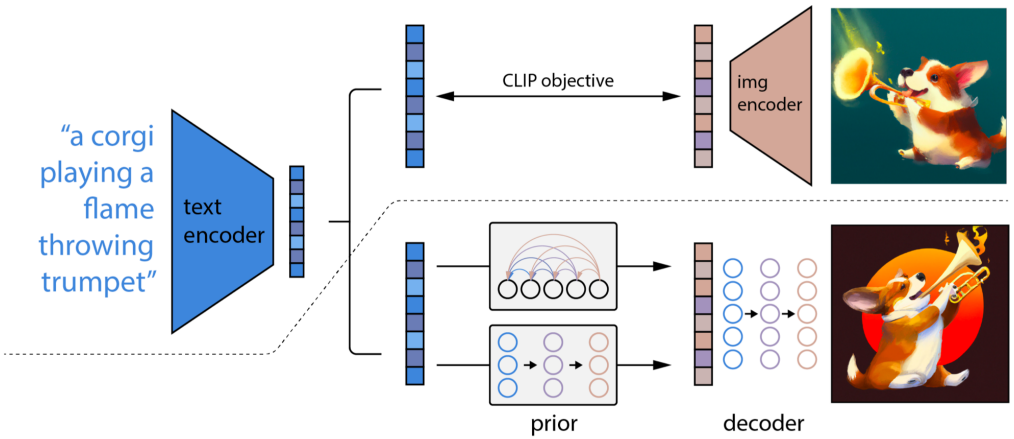
Diffusion models : pros

- Pro : Flexibility
  - denoising can happen in the latent space ; somehow, learn to "project" random noise to a point in the latent space that makes sense
  - possible to incorporate additional information/embedding at various stages before conditional diffusion happens : basis for all modern text-to-image systems ! (DALL-E, Stable Diffusion, Midjourney...)
  - see eg `https://www.assemblyai.com/blog/how-dall-e-2-actually-works/`



Diffusion models : pros

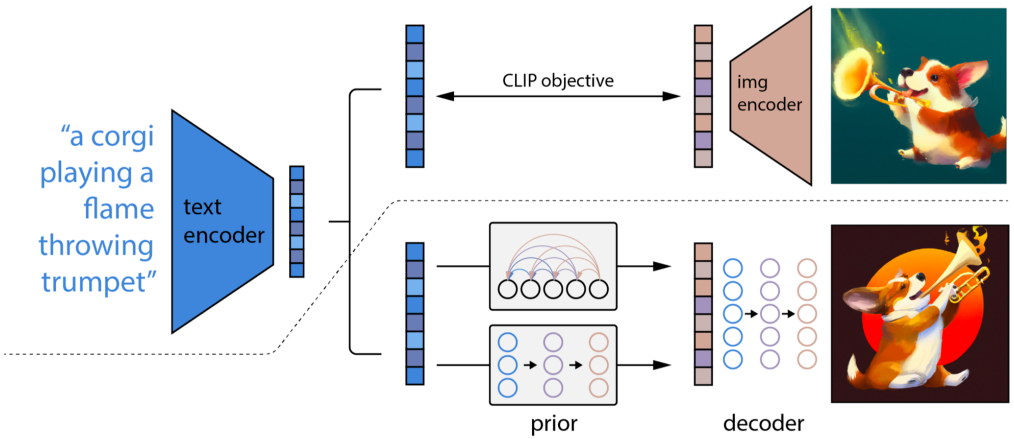
- Pro : Flexibility
  - denoising can happen in the latent space ; somehow, learn to "project" random noise to a point in the latent space that makes sense
  - possible to incorporate additional information/embedding at various stages before conditional diffusion happens : basis for all modern text-to-image systems ! (DALL-E, Stable Diffusion, Midjourney...)
  - see eg `https://www.assemblyai.com/blog/how-dall-e-2-actually-works/`



- Pro : Tractability and performance. Every step is analytically tractable, thus improving stability and interpretability. Images are generally more realistic than GANs or VAEs

Diffusion models : pros

- Pro : Flexibility
  - denoising can happen in the latent space ; somehow, learn to "project" random noise to a point in the latent space that makes sense
  - possible to incorporate additional information/embedding at various stages before conditional diffusion happens : basis for all modern text-to-image systems ! (DALL-E, Stable Diffusion, Midjourney...)
  - see eg `https://www.assemblyai.com/blog/how-dall-e-2-actually-works/`



- Pro : Tractability and performance. Every step is analytically tractable, thus improving stability and interpretability. Images are generally more realistic than GANs or VAEs
- Con : slow sampling. The sampling process is slower, we must perform many steps of denoising.

Table of Contents

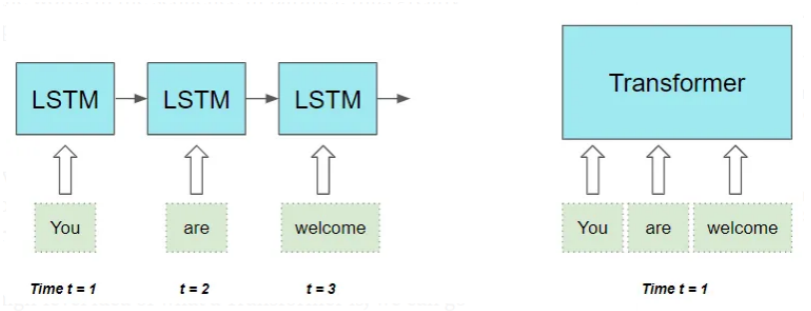
Transformers : what

https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

- Transformers (Vaswani et al.), introduced by Google in 2017, have revolutionized first Natural Language Processing (NLP), then many other domains
- Documents classification, translation, text generation, text-to-image, image-to-text, chatbots...
- Well-known models, mostly from two companies :
  - Google : BERT, and successors : RoBERTa, ALBERT... (for French language : CamemBERT model !). Also T5, PaLM, etc.
  - OpenAI : GPT, and successors : GPT-2, GPT-3, ChatGPT...

Transformers : how ?

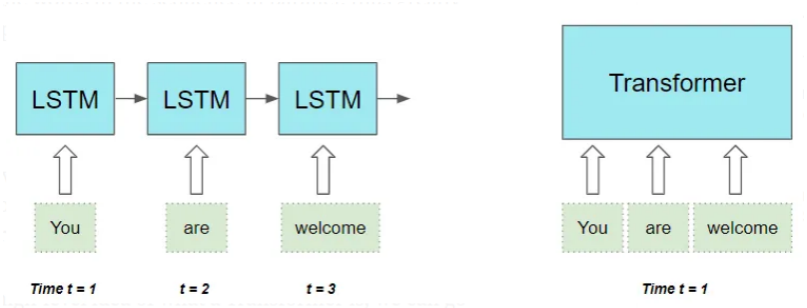
- Unlike RNNs, Transformers almost totally discard the "sequential" nature of texts : everything depends on everything !



- far, **far** more parameters ! (usually in the billions) : huge training data, pre-trained components. Modern Transformers can almost only be trained by companies with massive funding and resources. Ethical problems...

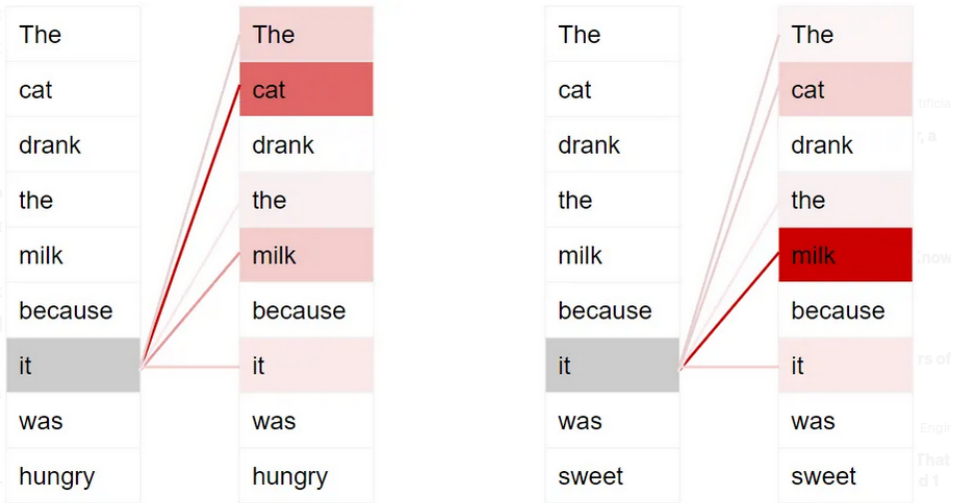
Transformers : how ?

- Unlike RNNs, Transformers almost totally discard the "sequential" nature of texts : everything depends on everything !



- far, **far** more parameters ! (usually in the billions) : huge training data, pre-trained components. Modern Transformers can almost only be trained by companies with massive funding and resources. Ethical problems...

The key to Transformers is **attention** : a (deceptively) simple mechanism to model dependencies between words/objects



Attention : equations

- In NLP, words (or subwords) are *token*, embedded using a tokenizer.

Algorithm 3: Basic single-query attention.

Input: $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token

Input: $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of context tokens $t \in [T]$.

Output: $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined.

Parameters: $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$, $\boldsymbol{b}_q, \boldsymbol{b}_k \in \mathbb{R}^{d_{\text{attn}}}$, the query and key linear projections.

Parameters: $\boldsymbol{W}_v \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\boldsymbol{b}_v \in \mathbb{R}^{d_{\text{out}}}$, the value linear projection.

$\boldsymbol{q} \leftarrow \boldsymbol{W}_q \boldsymbol{e} + \boldsymbol{b}_q$

$\forall t: \ \boldsymbol{k}_t \leftarrow \boldsymbol{W}_k \boldsymbol{e}_t + \boldsymbol{b}_k$

$\forall t: \ \boldsymbol{v}_t \leftarrow \boldsymbol{W}_v \boldsymbol{e}_t + \boldsymbol{b}_v$

$\forall t: \ \alpha_t = \frac{\exp\left(\boldsymbol{q}^\intercal \boldsymbol{k}_t / \sqrt{d_{\text{attn}}}\right)}{\sum_u \exp\left(\boldsymbol{q}^\intercal \boldsymbol{k}_u / \sqrt{d_{\text{attn}}}\right)}$

return $\tilde{\boldsymbol{v}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{v}_t$

Phuong and Hutter. Formal Algorithms for Transformers. 2022

Attention : equations

- In NLP, words (or subwords) are *token*, embedded using a tokenizer.
- Given a token e, and context tokens e_t, attention coefficients are computed to relate e to the context

Algorithm 3: Basic single-query attention.

Input: $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token

Input: $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of context tokens $t \in [T]$.

Output: $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined.

Parameters: $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$, $\boldsymbol{b}_q, \boldsymbol{b}_k \in \mathbb{R}^{d_{\text{attn}}}$, the query and key linear projections.

Parameters: $\boldsymbol{W}_v \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\boldsymbol{b}_v \in \mathbb{R}^{d_{\text{out}}}$, the value linear projection.

$\boldsymbol{q} \leftarrow \boldsymbol{W}_q \boldsymbol{e} + \boldsymbol{b}_q$

$\forall t: \ \boldsymbol{k}_t \leftarrow \boldsymbol{W}_k \boldsymbol{e}_t + \boldsymbol{b}_k$

$\forall t: \ \boldsymbol{v}_t \leftarrow \boldsymbol{W}_v \boldsymbol{e}_t + \boldsymbol{b}_v$

$\forall t: \ \alpha_t = \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_t / \sqrt{d_{\text{attn}}}\right)}{\sum_u \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_u / \sqrt{d_{\text{attn}}}\right)}$

return $\tilde{\boldsymbol{v}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{v}_t$

Phuong and Hutter. Formal Algorithms for Transformers. 2022

Attention : equations

- In NLP, words (or subwords) are *token*, embedded using a tokenizer.
- Given a token e, and context tokens e_t, attention coefficients are computed to relate e to the context
- It can be self-attention (a word within its sentence), or attention to a target context

Algorithm 3: Basic single-query attention.

Input: $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token

Input: $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of context tokens $t \in [T]$.

Output: $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined.

Parameters: $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$, $\boldsymbol{b}_q, \boldsymbol{b}_k \in \mathbb{R}^{d_{\text{attn}}}$, the query and key linear projections.

Parameters: $\boldsymbol{W}_v \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\boldsymbol{b}_v \in \mathbb{R}^{d_{\text{out}}}$, the value linear projection.

1 $\boldsymbol{q} \leftarrow \boldsymbol{W}_q \boldsymbol{e} + \boldsymbol{b}_q$

2 $\forall t: \boldsymbol{k}_t \leftarrow \boldsymbol{W}_k \boldsymbol{e}_t + \boldsymbol{b}_k$

3 $\forall t: \boldsymbol{v}_t \leftarrow \boldsymbol{W}_v \boldsymbol{e}_t + \boldsymbol{b}_v$

4 $\forall t: \alpha_t = \frac{\exp\left(\boldsymbol{q}^\intercal \boldsymbol{k}_t / \sqrt{d_{\text{attn}}}\right)}{\sum_u \exp\left(\boldsymbol{q}^\intercal \boldsymbol{k}_u / \sqrt{d_{\text{attn}}}\right)}$

5 **return** $\tilde{\boldsymbol{v}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{v}_t$

Phuong and Hutter. Formal Algorithms for Transformers. 2022

Attention : equations

- In NLP, words (or subwords) are *token*, embedded using a tokenizer.
- Given a token e, and context tokens e_t, attention coefficients are computed to relate e to the context
- It can be self-attention (a word within its sentence), or attention to a target context
- The key operation is softmax $a_k : x \mapsto \frac{e^{x_k}}{\sum_i e^{x_i}}$. Attention is nothing else that common linear operations with learned parameters, with proper normalization.

Algorithm 3: Basic single-query attention.

Input: $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token

Input: $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of context tokens $t \in [T]$.

Output: $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined.

Parameters: $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$, $\boldsymbol{b}_q, \boldsymbol{b}_k \in \mathbb{R}^{d_{\text{attn}}}$, the query and key linear projections.

Parameters: $\boldsymbol{W}_v \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\boldsymbol{b}_v \in \mathbb{R}^{d_{\text{out}}}$, the value linear projection.

1 $\boldsymbol{q} \leftarrow \boldsymbol{W}_q \boldsymbol{e} + \boldsymbol{b}_q$

2 $\forall t : \boldsymbol{k}_t \leftarrow \boldsymbol{W}_k \boldsymbol{e}_t + \boldsymbol{b}_k$

3 $\forall t : \boldsymbol{v}_t \leftarrow \boldsymbol{W}_v \boldsymbol{e}_t + \boldsymbol{b}_v$

4 $\forall t : \alpha_t = \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_t / \sqrt{d_{\text{attn}}}\right)}{\sum_u \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_u / \sqrt{d_{\text{attn}}}\right)}$

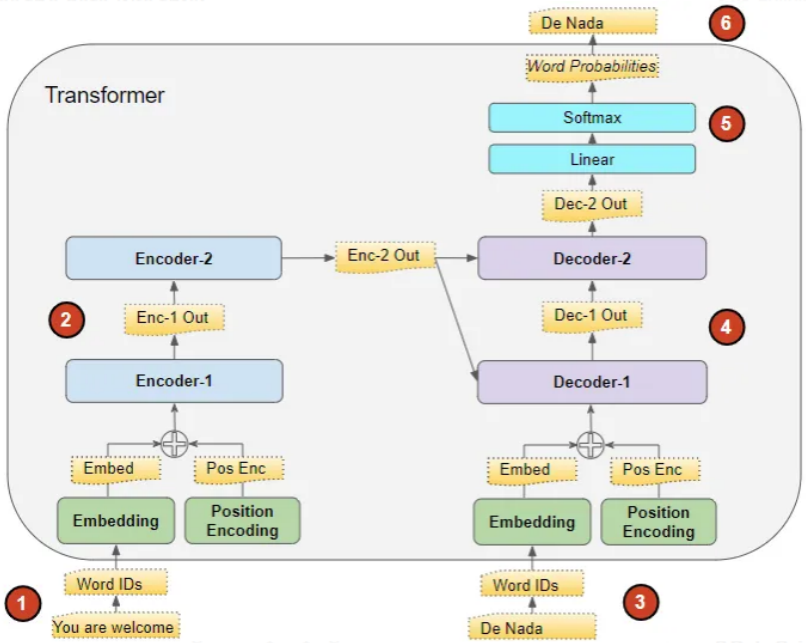
5 **return** $\tilde{\boldsymbol{v}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{v}_t$

Phuong and Hutter. Formal Algorithms for Transformers. 2022

Transformer : example of a (high-level) seq2seq architecture

https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

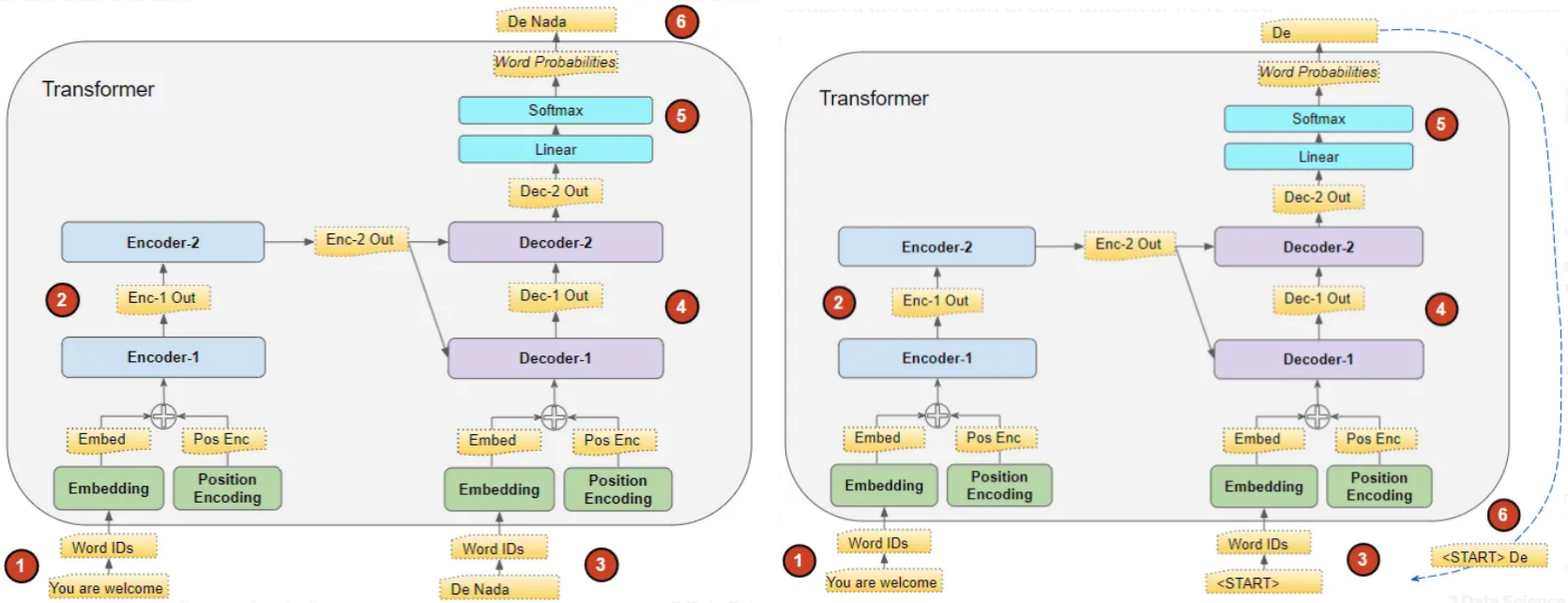
- Training : takes a query x, a target y, produces an output



Transformer : example of a (high-level) seq2seq architecture

https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

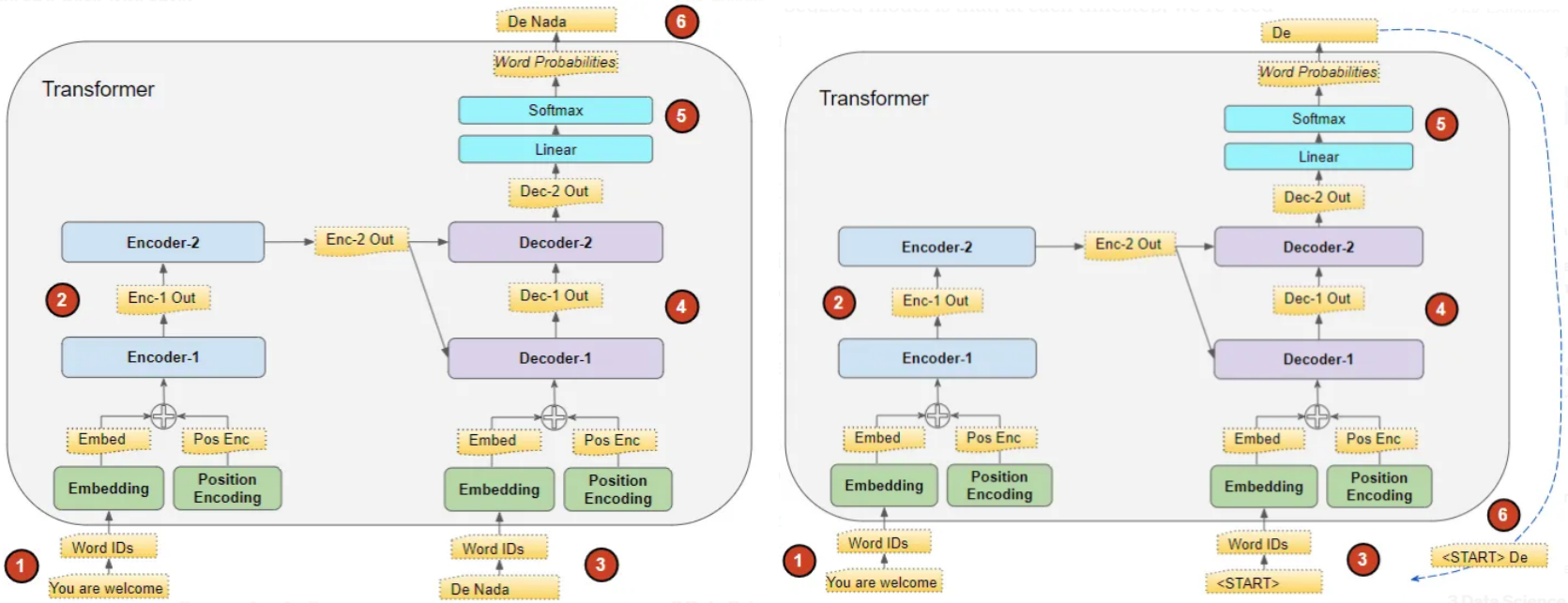
- Training : takes a query x, a target y, produces an output
- Inference : takes a query x, start with an empty target, generate words one by one, by feeding the sequence generated until now as the target ("next word prediction")



Transformer : example of a (high-level) seq2seq architecture

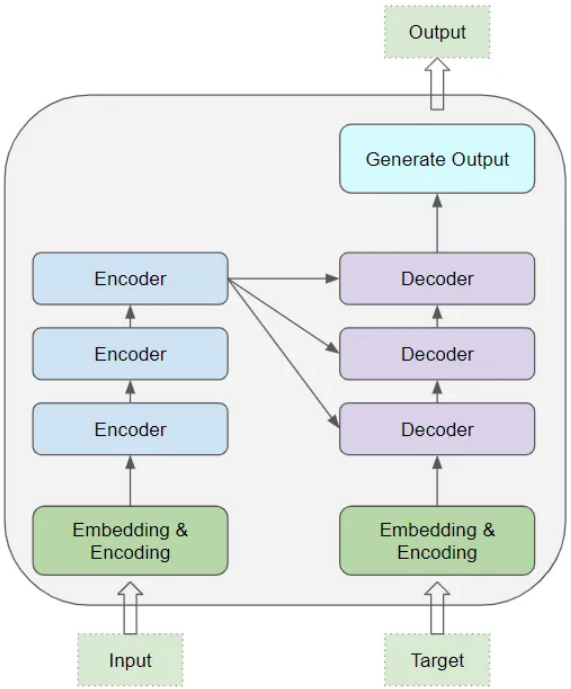
https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

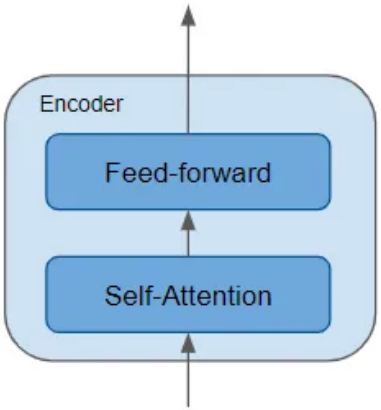
- Training : takes a query x, a target y, produces an output
- Inference : takes a query x, start with an empty target, generate words one by one, by feeding the sequence generated until now as the target ("next word prediction")
- Unlike RNNs/LSTMs, the next word generated directly depends on everything else (through attention and self-attention), not just by recurrent mechanisms

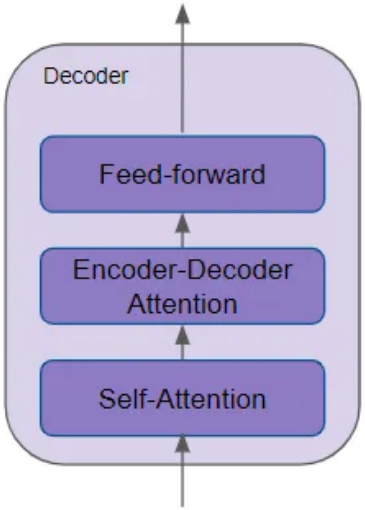


Transformer : example of a (high-level) seq2seq architecture

- Familiar concepts : encoding/decoding, latent space... but with attention layers !





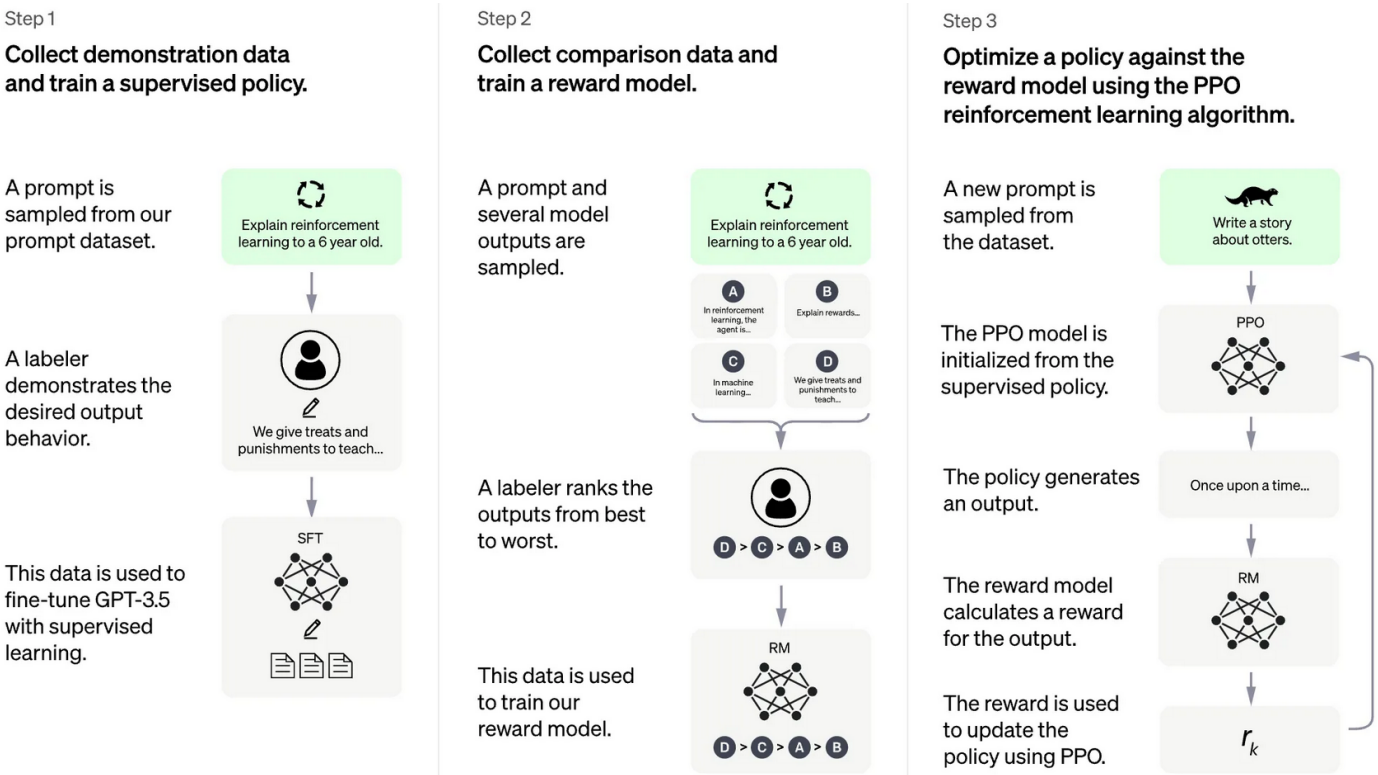


Transformers + RL

- Combined with RL strategies (action/rewards), Transformers become incredible chatbots.

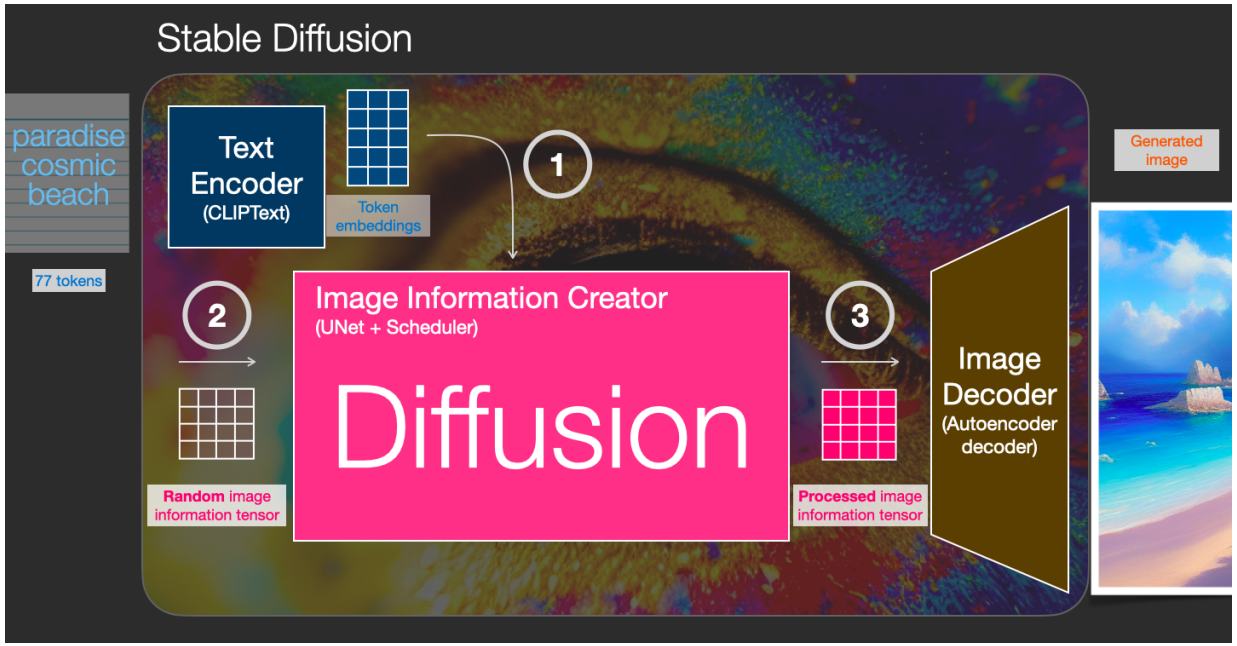
Transformers + RL

- Combined with RL strategies (action/rewards), Transformers become incredible chatbots.
- ChatGPT is based on the GPT3 model (Generative Pre-trained Transformer), trained with complex RL strategies and HUGE amount of data.



Transformers + diffusion models

- Transformers are great at computing embeddings for text, that can be used for other tasks. The BERT model is only an embedding model !
- These embeddings can be fed to other generative models. For instance, all the modern text-to-images models (Stable Diffusion, Midjourney, Imagen, DALL-E 2...) are based on Transformers + diffusion models.
- All these components are pretrained ! And then fine-tuned once put together.
- Many models/datasets are open-source ! See HuggingFace 🤗 and eg `https://www.assemblyai.com/blog/build-a-free-stable-diffusion-app-with-a-gpu-backend/`



Transformers + language models + diffusion models + time series analysis + CNN + massive training data + immense computing power + ...

Text-to-video : OpenAI's Sora (from last week)

https://cdn.openai.com/sora/videos/tokyo-walk.mp4

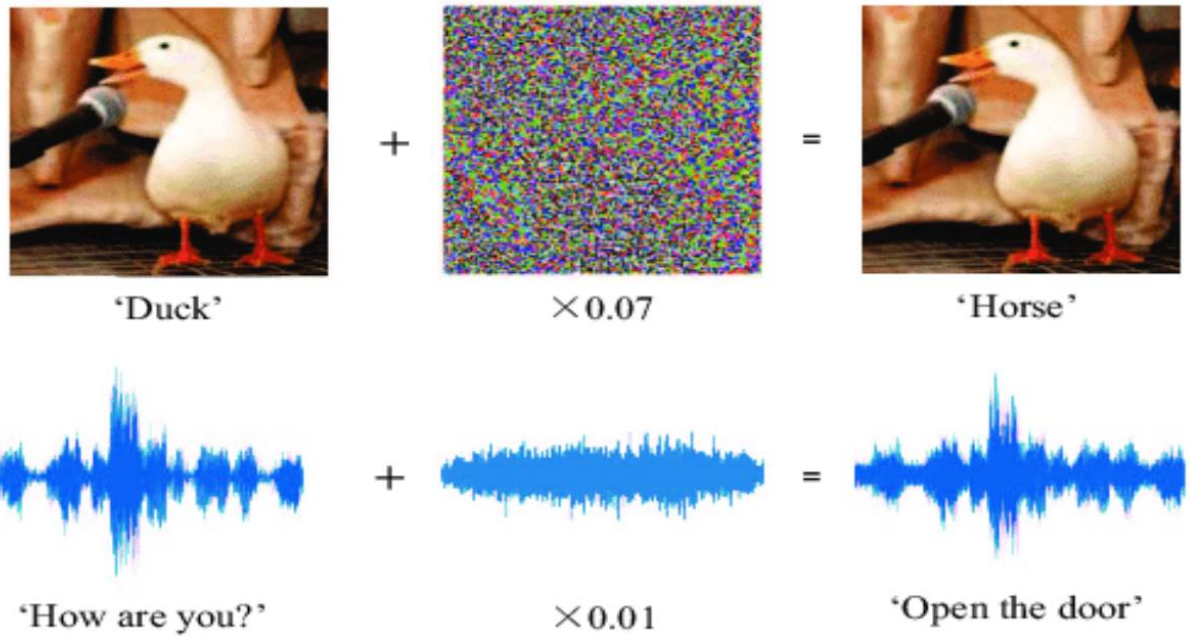
Table of Contents

Conclusion

- Deep learning is still mysterious and frustrating for many reasons
- They work incredibly well in some situations...
- ...but don't kill flies with bazookas, and always remember the basic principles ! (which are useful in deep learning too !)
- Modern librairies make them easy to build, easy to train...
- ... but they are useful for non-deep stuff too !

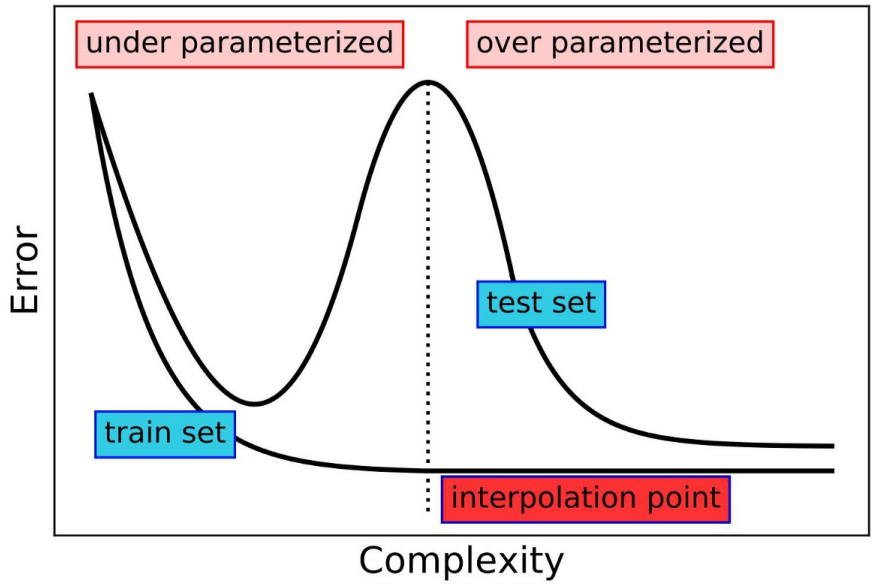
Hot question : adversarial attack

Once trained, a deep NN usually can be fooled easily, by designing an attack specifically for this NN. This unstability seems to be an integral part of high-performance NNs !



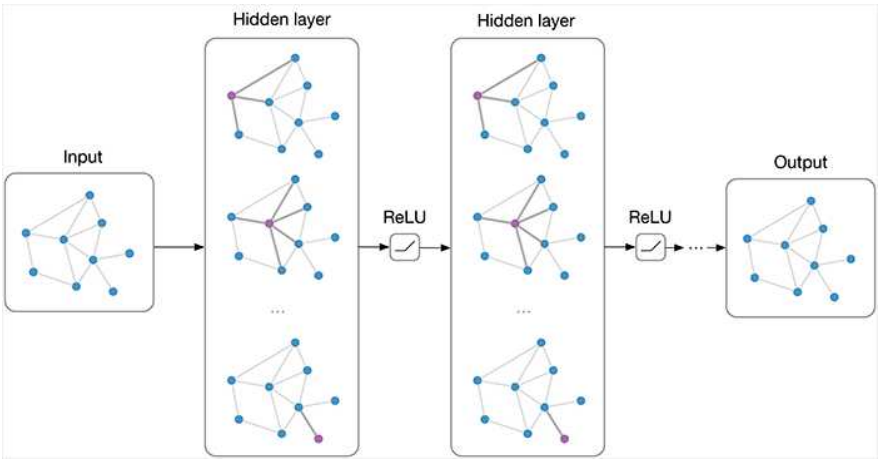
Hot question : generalization and double descent

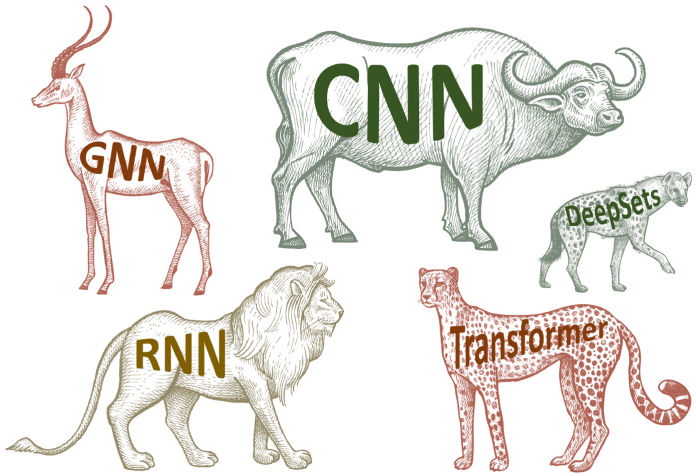
- Modern Deep NN are wildly overparametrized, and do interpolate the training data ! !
- But they still generatlize well !
- There is a double descent phenomenon : by increasing the complexity of the model past the interpolation point, among *all* interpolating models, it seems DNN+SGD favors models that generalize well. Still very much an open topic.



Hot topic : GNN, Geometric Deep Learning

- Graph Neural Networks (GNNs) work on irregular data that lives on network : molecules, proteins, social networks, computer networks, meshes, etc. They use "convolution" in this irregular space
- Most importantly, they are permutation-invariant : their output are left unchanged by a relabelling of nodes of the graph (graph isomorphism)
- They gave rise to geometric deep learning, which is the study of symmetries and invariances in DNNs : translation for CNN, permutation for GNN, but also rotation, gauge, etc. Much inspired by physics !





Hot(test) topic : ethics

- **Privacy** Since DNNs interpolate, and thus generally *memorize training data*, data privacy is very much a problem

Hot(test) topic : ethics

- **Privacy** Since DNNs interpolate, and thus generally *memorize training data*, data privacy is very much a problem
- **Environmental cost** : AI is very, very power-hungry. So far, each improvement in the efficacity of (any) technology leads to the *rebound effect*.

Hot(test) topic : ethics

- **Privacy** Since DNNs interpolate, and thus generally *memorize training data*, data privacy is very much a problem
- **Environmental cost** : AI is very, very power-hungry. So far, each improvement in the efficacity of (any) technology leads to the *rebound effect*.
- **Copyrights, job market...** : are Internet-scrapped data stolen from artists/writers ? Why would a company pay to hire artists ?

Hot(test) topic : ethics

- **Privacy** Since DNNs interpolate, and thus generally *memorize training data*, data privacy is very much a problem
- **Environmental cost** : AI is very, very power-hungry. So far, each improvement in the efficacity of (any) technology leads to the *rebound effect*.
- **Copyrights, job market...** : are Internet-scrapped data stolen from artists/writers ? Why would a company pay to hire artists ?
- **Art and innovation** : at its core, generative models combine/reproduce the training data. They cannot do otherwise, structurally. Doesn't this encourage stagnation ? Is true creativity gonna be hidden is a sea of AI-generated content ? Moreover, all these are controlled by a few megacorp ?

Hot(test) topic : ethics

- **Privacy** Since DNNs interpolate, and thus generally *memorize training data*, data privacy is very much a problem
- **Environmental cost** : AI is very, very power-hungry. So far, each improvement in the efficacity of (any) technology leads to the *rebound effect*.
- **Copyrights, job market...** : are Internet-scrapped data stolen from artists/writers ? Why would a company pay to hire artists ?
- **Art and innovation** : at its core, generative models combine/reproduce the training data. They cannot do otherwise, structurally. Doesn't this encourage stagnation ? Is true creativity gonna be hidden is a sea of AI-generated content ? Moreover, all these are controlled by a few megacorp ?
- **Misinformation** : fake news, deep fakes... is the Internet of the future gonna be a collection of not-to-be-trusted fake content ? Through fake news and recommendation algorithms, is the world gonna be more and more polarized ? Are video evidences still gonna be acceptable in court ? Is online bullying gonna take immense proportion ? ...

Hot(test) topic : ethics

- **Privacy** Since DNNs interpolate, and thus generally *memorize training data*, data privacy is very much a problem
- **Environmental cost** : AI is very, very power-hungry. So far, each improvement in the efficacity of (any) technology leads to the *rebound effect*.
- **Copyrights, job market...** : are Internet-scrapped data stolen from artists/writers ? Why would a company pay to hire artists ?
- **Art and innovation** : at its core, generative models combine/reproduce the training data. They cannot do otherwise, structurally. Doesn't this encourage stagnation ? Is true creativity gonna be hidden is a sea of AI-generated content ? Moreover, all these are controlled by a few megacorp ?
- **Misinformation** : fake news, deep fakes... is the Internet of the future gonna be a collection of not-to-be-trusted fake content ? Through fake news and recommendation algorithms, is the world gonna be more and more polarized ? Are video evidences still gonna be acceptable in court ? Is online bullying gonna take immense proportion ? ...

AI is *not* gonna "destroy" the world, at least not Terminator-style. But... have we just destroyed / altered fundamentally crucial common goods ? Like the Internet ? Are we the last generation to "mostly" trust what we see online ?