[KICSV Special AI Lecture]
Core AI Concepts and Modern Architectures

Sunghee Yun

Co-Founder & CTO @ Erudio Bio, Inc. / Advisor @ CryptoLab, Inc.
Adjunct Professor @ Sogang Univ / Advisory Professor @ DGIST

About Speaker

- *Co-Founder & CTO @ Erudio Bio, San Jose & Novato, CA, USA*
- *Advisor & Evangelist @ CryptoLab, Inc., San Jose, CA, USA*
- Chief Business Development Officer @ WeStory.ai, Cupertino, CA, USA
- Advisory Professor, Electrical Engineering and Computer Science @ DGIST, Korea
- Adjunct Professor, Electronic Engineering Department @ Sogang University, Korea
- Global Advisory Board Member @ Innovative Future Brain-Inspired Intelligence System Semiconductor of Sogang University, Korea
- *KFAS-Salzburg Global Leadership Initiative Fellow @ Salzburg Global Seminar*, Salzburg, Austria
- Technology Consultant @ Gerson Lehrman Gruop (GLG), NY, USA
- *Co-Founder & CTO / Head of Global R&D & Chief Applied Scientist / Senior Fellow @ Gauss Labs, Inc., Palo Alto, CA, USA* *2020 ∼ 2023*

- Senior Applied Scientist @ Amazon.com, Inc., Vancouver, BC, Canada $\sim$ 2020
- Principal Engineer @ Software R&D Center, DS Division, Samsung, Korea $\sim$ 2017
- Principal Engineer @ Strategic Marketing & Sales Team, Samsung, Korea $\sim$ 2016
- Principal Engineer @ DT Team, DRAM Development Lab, Samsung, Korea $\sim$ 2015
- Senior Engineer @ CAE Team, Samsung, Korea $\sim$ 2012
- PhD - Electrical Engineering @ Stanford University, CA, USA $\sim$ 2004
- Development Engineer @ Voyan, Santa Clara, CA, USA $\sim$ 2001
- MS - Electrical Engineering @ Stanford University, CA, USA $\sim$ 1999
- BS - Electrical & Computer Engineering @ Seoul National University 1994 $\sim$ 1998

Highlight of Career Journey

- BS in EE @ SNU, MS & PhD in EE @ Stanford University
  - *Convex Optimization - Theory, Algorithms & Software*
  - advised by *Prof. Stephen P. Boyd*
- Principal Engineer @ Samsung Semiconductor, Inc.
  - AI & Convex Optimization
  - collaboration with *DRAM/NAND Design/Manufacturing/Test Teams*
- Senior Applied Scientist @ Amazon.com, Inc.
  - e-Commerce AIs - anomaly detection, deep RL, and recommender system
  - Bezos's project - drove *$200M* in additional sales via Amazon Mobile Shopping App
- *Co-Founder & CTO / Global R&D Head & Chief Applied Scientist @ Gauss Labs, Inc.*
- *Co-Founder & CTO* - AI Technology & Business Development @ Erudio Bio, Inc.

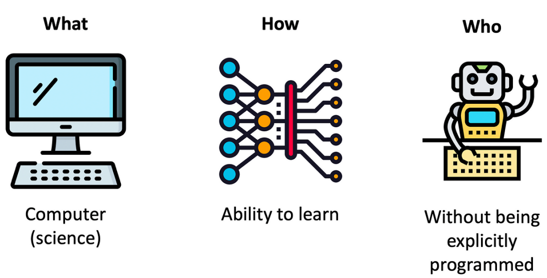
Today

ML Basics

Machine learning

Machine Learning

- ML
  - subfield of computer science that
    "gives computers the ability to learn without being explicitly programmed."
    - Arthur Samuel (1959)
  - *not* magic, still less intelligent than humans for many cases
  - *numerically minimizes* certain (mathematical) loss function to (indirectly) solve *some statistically meaningful* problems



Two famous quotes

- Albert Einstein

  The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest possible number of hypotheses or axioms.

- Alfred North Whitehead

  Civilization advances by extending the number of important operations which we can perform without thinking about them. - Operations of thought are like cavalry charges in a battle – they are strictly limited in number, they require fresh horses, and must only be made at decisive moments.

Statistical problem formulation

- assume data set $X_m = \{x^{(1)}, \ldots, x^{(m)}\}$
  - drawn independently from (true, but unknown) data generating distribution $p_{\mathrm{data}}(x)$
- maximum likelihood estimation (MLE) is to solve

$$\text{maximize} \quad p_{\mathrm{model}}(X;\theta) = \textstyle\prod_{i=1}^m p_{\mathrm{model}}(x^{(i)};\theta)$$

- equivalent (but numerically friendly) formulation

$$\text{maximize} \quad \log p_{\mathrm{model}}(X;\theta) = \textstyle\sum_{i=1}^m \log p_{\mathrm{model}}(x^{(i)};\theta)$$

MLE & KL divergence

- in (context of) information theory, Kullback-Leibler (KL) divergence defines distance between two probability distributions p & q

$$D_{\mathrm{KL}}(p\|q) = \mathop{\mathbf{E}}_{X\sim p} \log p(X)/q(X) = \int_{x\in\Omega} p(x) \log \frac{p(x)}{q(x)}\, dx$$

- KL divergence between data distribution p_{data} & model distribution p_{model} can be approximated by Monte Carlo method as

$$D_{\mathrm{KL}}(p_{\mathrm{data}}\|p_{\mathrm{model}}) \simeq \frac{1}{m} \sum_{i=1}^{m} (\log p_{\mathrm{data}}(x^{(i)}) - \log p_{\mathrm{model}}(x^{(i)};\theta))$$

- *minimizing KL divergence is equivalent to solving MLE problem!*

Equivalence of MLE to MSE

- assume model is Gaussian, $i.e.$, $y \sim \mathcal{N}(g_\theta(x), \Sigma)$ ($g_\theta(x) \in \mathbf{R}^p$, $\Sigma \in \mathbf{S}^p_{++}$)

$$p(y|x;\theta) = \frac{1}{\sqrt{2\pi}^p |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}\left(y - g_\theta(x)\right)^T \Sigma^{-1} \left(y - g_\theta(x)\right)\right)$$

- assuming that $\Sigma = \alpha I_p$, log-likelihood becomes

$$\begin{aligned}\sum_{i=1}^m \log p(x^{(i)}, y^{(i)}; \theta) &= \sum_{i=1}^m \log p(y^{(i)}|x^{(i)};\theta) p(x^{(i)}) \\ &= -\sum_{i=1}^m \|y^{(i)} - g_\theta(x^{(i)})\|_2^2 / 2\alpha - \frac{pm}{2}\log(2\pi\alpha) + \sum_{i=1}^m \log p(x^{(i)})\end{aligned}$$

- *minimizing mean-square-error (MSE) is equivalent to solving MLE problem!*

Numerical optimization problem formulation

- (true) problem to solve

$$\text{minimize} \quad \mathbf{E}\, l(g_\theta(X), Y)$$

  - *impossible* to solve

- basic formulation - surrogate problem to solve

$$\text{minimize} \quad f(\theta) = \frac{1}{m} \sum_{i=1}^{m} l(g_\theta(x^{(i)}), y^{(i)})$$

- formulation with regularization

$$\text{minimize} \quad f(\theta) = \frac{1}{m} \sum_{i=1}^{m} l(g_\theta(x^{(i)}), y^{(i)}) + \gamma r(\theta)$$

- stochastic gradient descent (SGD)

$$\theta^{(k+1)} = \theta^{(k)} - \alpha_k \nabla f(\theta^{(k)})$$

Deep Learning

Deep learning (DL)

- machine learning using artificial neural networks with multiple layers for
  - automatically learning hierarchical representations of data
- key components
  - deep neural networks, hidden layers, backpropagation, activation functions
  - hierarchical feature learning, representation learning, end-to-end learning
- key breakthroughs enabling DL
  - massively available data, GPU computing, algorithmic advances

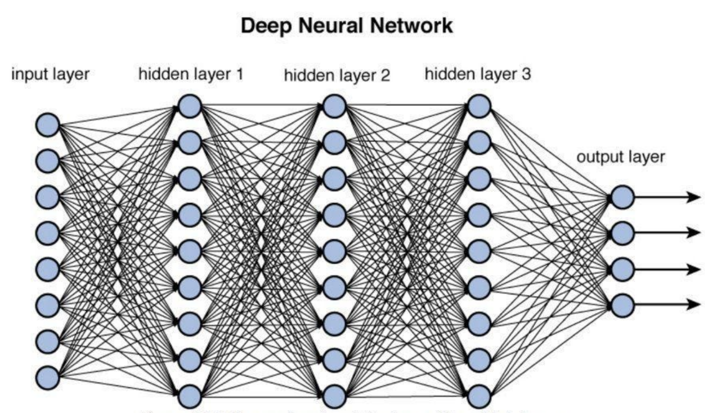
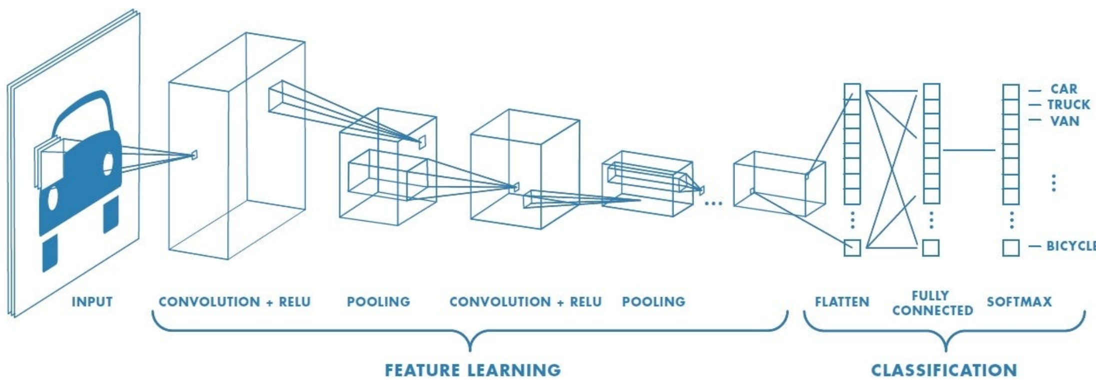


Figure 12.2 Deep network architecture with multiple layers.

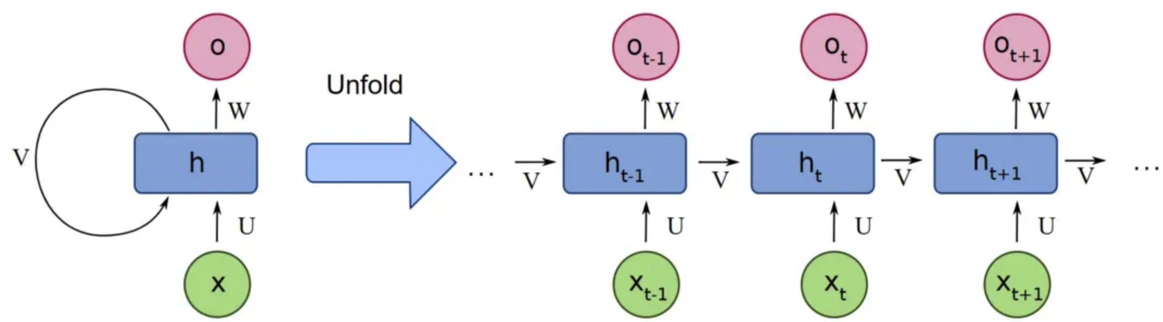
Convolutional neural network (CNN)

- specialized DL learning architecture designed for
  - processing grid-like data such as images
  - where spatial relationships between pixels matter
- key components
  - convolutional layers, pooling layers, activation functions, fully connected layers
- how it works
  - feature extraction, translation invariance, parameter sharing
- why it excels
  - local connectivity, hierarchical learning



Recurrent neural network (RNN)

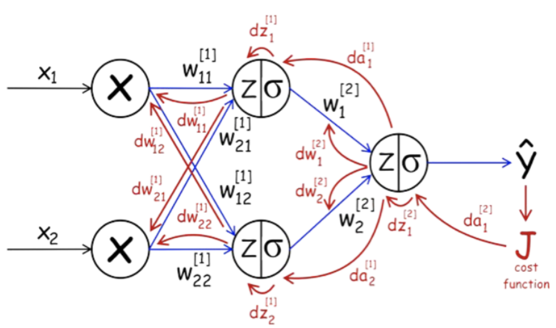
- neural network designed for
  - processing sequential data by maintaining memory of previous inputs
- key components
  - hidden states, recurrent connections, input/output layers, weight sharing
- how it works
  - sequential processing, memory mechanism, temporal dependencies
- why it excels
  - variable length input, context awareness, flexible architecture
- variants - long short-term memory (LSTM), gated recurrent unit (GRU)

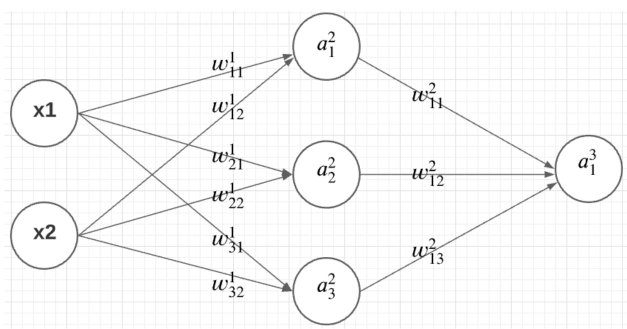


Training DNN using SGD

Notations

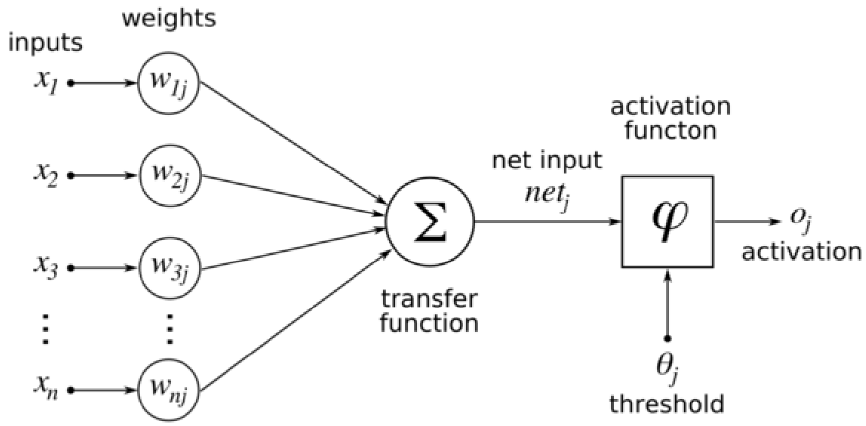
- p / q - dimension of input / output spaces
- $l : \mathbf{R}^q \times \mathbf{R}^q \to \mathbf{R}_+$ - loss function
- d - depth of neural network
- n_i $(1 \leq i \leq d)$ - number of perceptrons in ith layer
- $z^{[i]} \in \mathbf{R}^{n_i}$ - input to ith layer
- $o^{[i]} \in \mathbf{R}^{n_i}$ - output of ith layer
- $W^{[i]} \in \mathbf{R}^{n_i \times n_{i-1}}$ - weights of connections between $(i-1)$th and ith layer
- $w^{[i]} \in \mathbf{R}^{n_i \times n_{i-1}}$ - bias weights of ith layer
- $\phi^{[i]} : \mathbf{R}^{n_i} \to \mathbf{R}^{n_i}$ - activation functions of ith layer



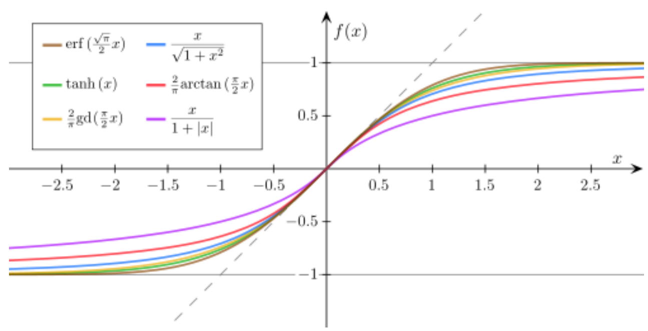


Basic unit & activation function

- basic unit



- activation function



Neural net equations

- modeling function for the (deep) neural network $g_\theta : \mathbf{R}^p \to \mathbf{R}^q$

$$g_\theta = \phi_\theta^{[d]} \circ \psi_\theta^{[d]} \circ \cdots \circ \phi_\theta^{[1]} \circ \psi_\theta^{[1]}$$

  or equivalently

$$g_\theta(x) = \phi_\theta^{[d]}(\psi_\theta^{[d]}(\cdots(\phi_\theta^{[1]}(\psi_\theta^{[1]}(x)))))$$

- for ith layer
  - output via (componentwise) activation function

$$o^{[i]} = \phi^{[i]}(z^{[i]}) \Leftrightarrow o_j^{[i]} = \phi_j^{[i]}(z_j^{[i]}) \quad (1 \leq j \leq n_i)$$

  - input via affine transformation $\psi^{[i]} : \mathbf{R}^{n_{i-1}} \to \mathbf{R}^{n_i}$

$$z^{[i]} = \psi^{[i]}(o^{[i-1]}) = W^{[i]}o^{[i-1]} + w^{[i]}$$

Stochastic gradient descent

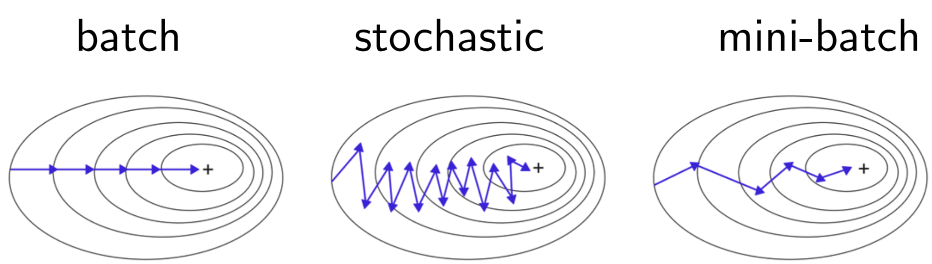
- ML training tries to minimize some loss function - $f(\theta)$ depends on (not only θ, but also) batch of data $(x^{(1)}, y^{(1)}), \ldots (x^{(m)}, y^{(m)})$

$$\text{minimize} \quad f(\theta)$$

- while exist hundreds of optimization methods solving this problem
  - *the only method used widely* is stochastic *gradient descent*!
- (stochastic) gradient descent

$$f(\theta^{k+1}) = f(\theta^k) - \alpha^k \nabla f(\theta^k)$$

- *backpropagation* is used to evaluate this (stochastic) *gradient* using *chain rule*



Chain rule

- suppose
  - two functions $f : \mathbf{R}^n \to \mathbf{R}^m$ & $g : \mathbf{R}^m \to \mathbf{R}$
  - Jacobian of f - $Df : \mathbf{R}^n \to \mathbf{R}^{m \times n}$
  - gradient of g - $\nabla g : \mathbf{R}^m \to \mathbf{R}^m$
- gradient of composite function $h = g \circ f$

$$\nabla h(\theta) = Df(\theta)^T \nabla g(f(\theta)) \in \mathbf{R}^n \quad \text{(using matrix-vector multiplication)}$$

in other words

$$\frac{\partial}{\partial \theta_i} h(\theta) = \sum_{j=1}^{m} \frac{\partial}{\partial \theta_i} f_j(\theta) \nabla_j g(f(\theta)) \quad \text{(scalar version)}$$

Loss function & its gradient

- assume cost function of deep neural network is

$$f(\theta) = \frac{1}{m}\sum_{k=1}^{m} l(g_\theta(x^{(k)}), y^{(k)}) = \frac{1}{m}\sum_{k=1}^{m} f_k(\theta)$$

where

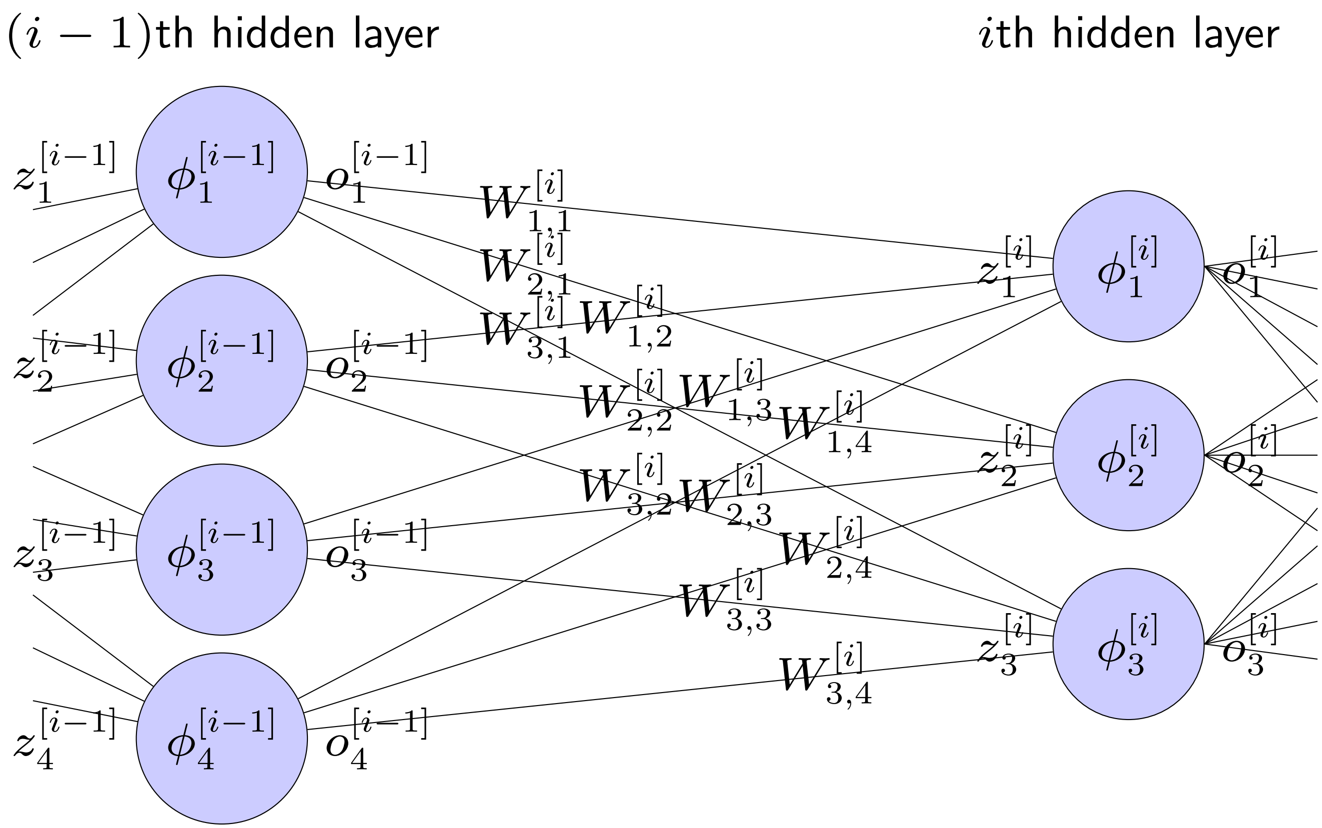
$$f_k(\theta) = l(g_\theta(x^{(k)}), y^{(k)})$$

- gradient is

$$m\nabla_\theta f(\theta) = \sum_{k=1}^{m} \nabla_\theta l(g_\theta(x^{(k)}), y^{(k)}) = \sum_{k=1}^{m} \nabla_\theta f_k(\theta)$$

  - $i.e.$, evaluate gradient $\nabla_\theta f_k(\theta)$ for each data point $(x^{(k)}, y^{(k)})$

Hidden layers



Backpropagation formula using chain rule

- for each data $(x^{(k)}, y^{(k)})$
  - via activation function

$$\frac{\partial}{\partial z_j^{[i]}} f_k(\theta) = \frac{\partial}{\partial o_j^{[i]}} f_k(\theta) {\phi_j^{[i]}}'(o_j^{[i]}) \quad \text{for } 1 \leq j \leq n_i \tag{1}$$

    where ${\phi_j^{[i]}}'(o_j^{[i]})$ is derivative of activation function $\phi_j^{[i]}$ evaluated at $o_j^{[i]}$

  - via affine transformation

$$\begin{aligned} \frac{\partial}{W_{j,l}^{[i]}} f_k(\theta) &= o_l^{[i-1]} \frac{\partial}{\partial z_j^{[i]}} f_k(\theta) \quad \text{for } 1 \leq j \leq n_i \ \&\ 1 \leq l \leq n_{i-1} && (2) \\ \frac{\partial}{\partial w_j^{[i]}} f_k(\theta) &= \frac{\partial}{\partial z_j^{[i]}} f_k(\theta) \quad \text{for } 1 \leq j \leq n_i && (3) \\ \frac{\partial}{\partial o_l^{[i-1]}} f_k(\theta) &= \sum_{j=1}^{n_i} W_{j,l}^{[i]} \frac{\partial}{\partial z_j^{[i]}} f_k(\theta) \quad \text{for } 1 \leq l \leq n_{i-1} && (4) \end{aligned}$$

Backpropagation formula using matrix-vector multiplication

- for each data $(x^{(k)}, y^{(k)})$
  - via activation function

$$\nabla_{z^{[i]}} f_k(\theta) = D\phi^{[i]} \nabla_{o^{[i]}} f_k(\theta) \tag{5}$$

where $D\phi^{[i]} = \mathbf{diag}({\phi_1^{[i]}}'(o_1^{[i]}), \ldots, {\phi_{n_i}^{[i]}}'(o_{n_i}^{[i]}))$ is Jacobian of $\phi^{[i]}$ evaluated at $o^{[i]}$

  - via affine transformation

$$\begin{aligned} \nabla_{W^{[i]}} f_k(\theta) &= \nabla_{z^{[i]}} f_k(\theta) {o^{[i-1]}}^T \in \mathbf{R}^{n_i \times n_{i-1}} \quad (6) \\ \nabla_{w^{[i]}} f_k(\theta) &= \nabla_{z^{[i]}} f_k(\theta) \in \mathbf{R}^{n_i} \quad (7) \\ \nabla_{o^{[i-1]}} f_k(\theta) &= {W^{[i]}}^T \nabla_{z^{[i]}} f_k(\theta) \in \mathbf{R}^{n_{i-1}} \quad (8) \end{aligned}$$

Backpropagation formula using `Python` `numpy` package

- for each data $(x^{(k)}, y^{(k)})$
  - via activation function

$$\texttt{grad\_z = phi\_dir * grad\_o} \tag{9}$$

    where `grad_z`, `phi_dir`, `grad_o` are 1d `numpy.ndarray` of size n_i
  - via affine transformation

$$\texttt{grad\_W = numpy.dot(grad\_z, val\_o.T)} \tag{10}$$

$$\texttt{grad\_w = grad\_z.copy()} \tag{11}$$

$$\texttt{grad\_o\_prev = numpy.dot(grad\_z, W)} \tag{12}$$

    where `val_o`, `grad_w` are 1d `numpy.ndarray` of size n_i, `grad_o_prev` is 1d `numpy.ndarray` of size n_{i-1}, `grad_W` is 2d `numpy.ndarray` of shape (n_i, n_{i-1})

Gradient evaluation using backpropagation

- forward propagation - evaluate for each $(x^{(k)}, y^{(k)})$

$$g_\theta(x^{(k)}) = \phi_\theta^{[d]}(\psi_\theta^{[d]}(\cdots(\phi_\theta^{[1]}(\psi_\theta^{[1]}(x^{(k)})))))$$

- *backpropagation - evaluate partial derivatives backward*
  - evaluate gradient with respect to output of output layer $o^{[d]} = g_\theta(x^{(k)})$

$$\nabla_{o^{[d]}} f_k(\theta) = \nabla_{y_1} l(g_\theta(x^{(k)}), y^{(k)})$$

  - evaluate gradient with respect to input from that with respect to output using (1), or equivalently, using (5) $i.e.$, evaluate $\nabla_{z^{[i]}} f_k(\theta)$ from $\nabla_{o^{[i]}} f_k(\theta)$
  - evaluate gradient with respect to weights, bias, and intput of previous layer using (3), (4), & (2) or equivalently, using (7), (8), & (6) $i.e.$, evaluate $\nabla_{W^{[i]}} f_k(\theta)$, $\nabla_{w^{[i]}} f_k(\theta)$ & $\nabla_{o^{[i-1]}} f_k(\theta)$ from $\nabla_{z^{[i]}} f_k(\theta)$
  - repeat back to input layer to evaluate all

$$\nabla_{W^{[1]}} f_k(\theta), \nabla_{w^{[1]}} f_k(\theta), \ldots, \nabla_{W^{[d]}} f_k(\theta), \nabla_{w^{[d]}} f_k(\theta)$$

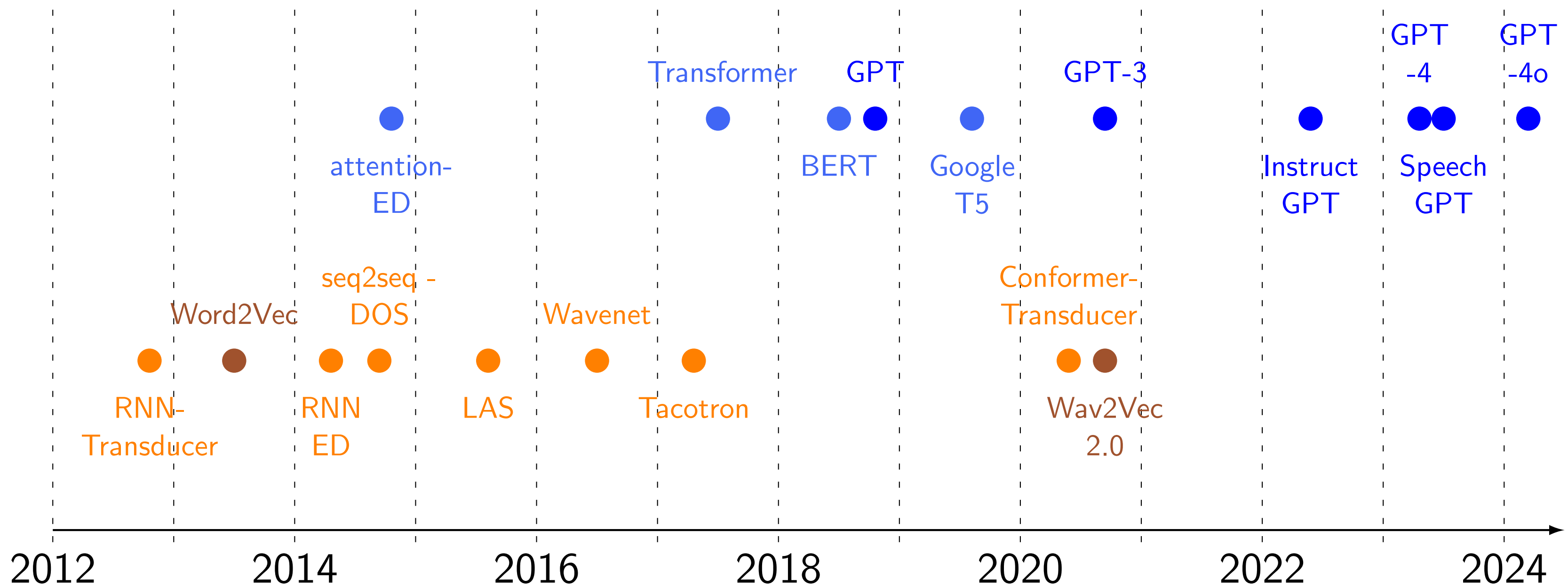
LLM

Language Models

History of language models

- bag of words - first introduced – 1954
- word embedding – 1980
- RNN based models - conceptualized by David Rumelhart – 1986
- LSTM (based on RNN) – 1997
- 380M-sized seq2seq model using LSTMs proposed – 2014
- 130M-sized seq2seq model using gated recurrent units (GRUs) – 2014
- Transformer - Attention is All You Need - A. Vaswani et al. @ Google – 2017
  - 100M-sized encoder-decoder multi-head attention model for machine translation
  - non-recurrent architecture, handle arbitrarily long dependencies
  - parallelizable, *simple* (linear-mapping-based) attention model

Recent advances in speech & language processing



- LAS: listen, attend, and spell, ED: encoder-decoder, DOS: decoder-only structure

Types of language models

- many of language models have common requirements - language representation learning
- can be learned via pre-tranining *high performing model* and fine-tuning/transfer learning/domain adaptation
- this *high performing model* learning essential language representation *is* (lanauge) foundation model

– actually, same for other types of learning, $e.g.$, CV

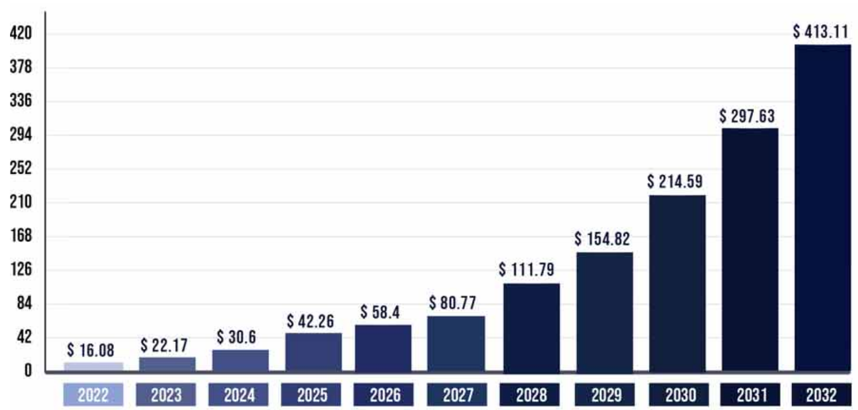
voice recognition	NL generation	summarization	spelling correction	action planning
voice synthesis	translation	topic model	topic model	
	Q&A	NLU		

NLP Market

NLP market size

- global NLP market size estimated at USD 16.08B in 2022, is expected to hit USD 413.11B by 2032 - *CAGR of 38.4%*
- in 2022
  - north america NLP market size valued at USD 8.2B
  - high tech and telecom segment accounted revenue share of over 23.1%
  - healthcare segment held a 10% market share
  - (by component) solution segment hit 76% revenue share
  - (deployment mode) on-premise segment generated 56% revenue share
  - (organizational size) large-scale segment contributed highest market share

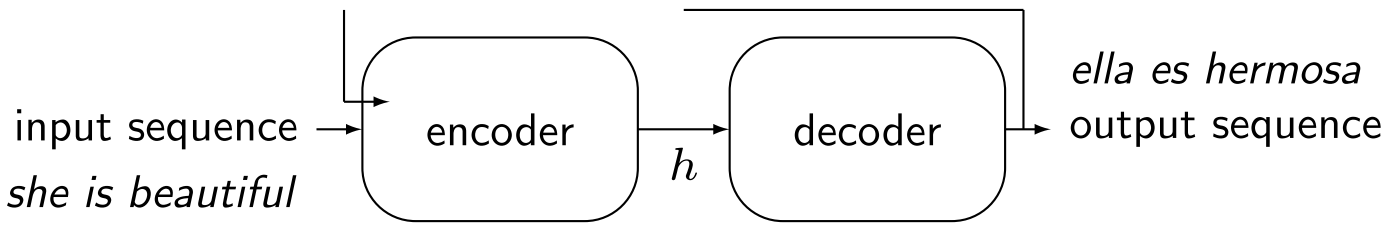
\- source - Precedence Research



Sequence-to-Sequence Models

Sequence-to-sequence (seq2seq) model

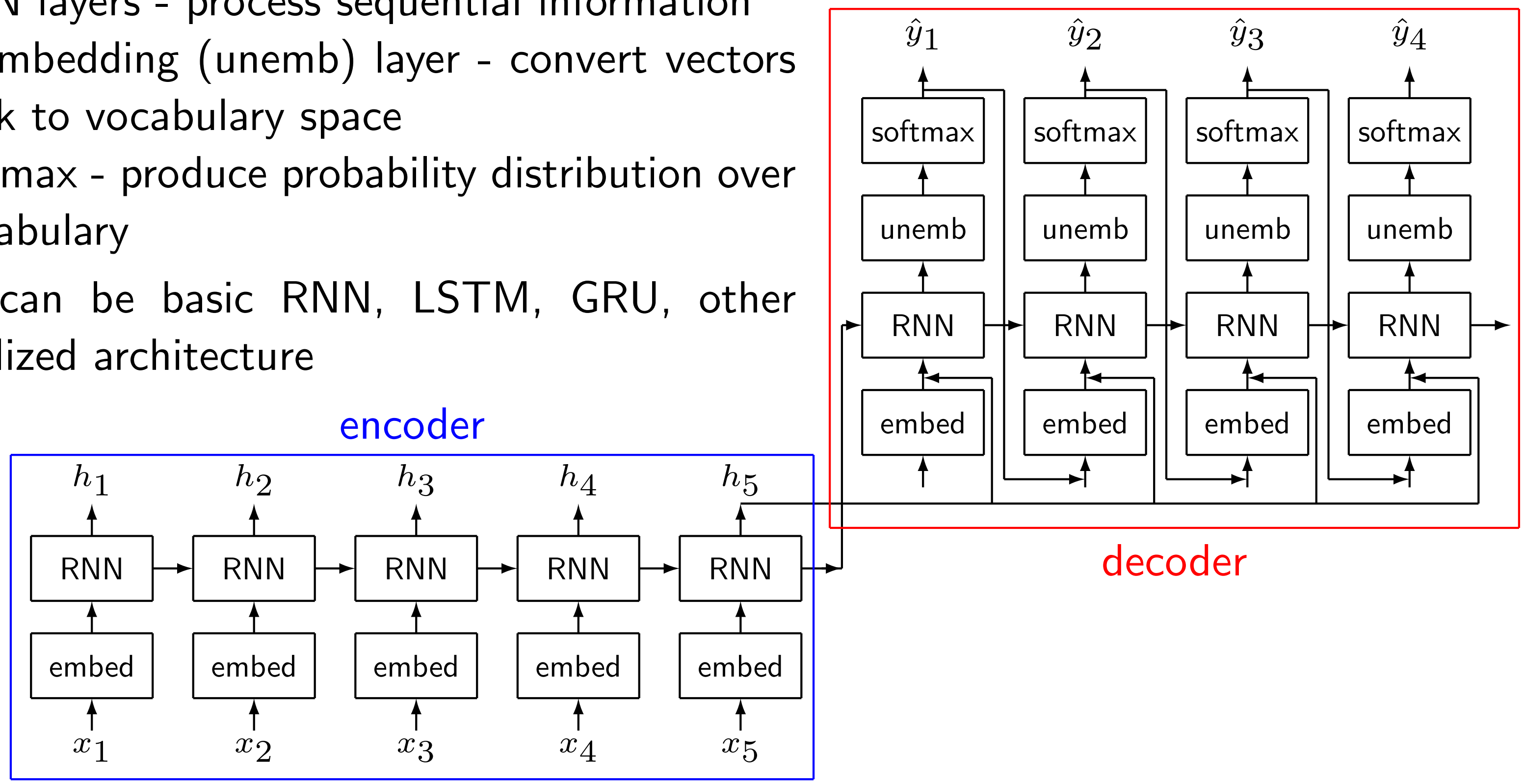
- seq2seq - take sequences as inputs and spit out sequences
- encoder-decoder architecture



  - encoder & decoder can be RNN-type models
  - $h \in \mathbf{R}^n$ - hidden state - *fixed length* vector
- (try to) condense and store information of input sequence (losslessly) in (fixed-length) hidden states
  - finite hidden state - not flexible enough, $i.e.$, cannot handle arbitrarily large information
  - memory loss for long sequences
  - LSTM was promising fix, but with (inevitable) limits

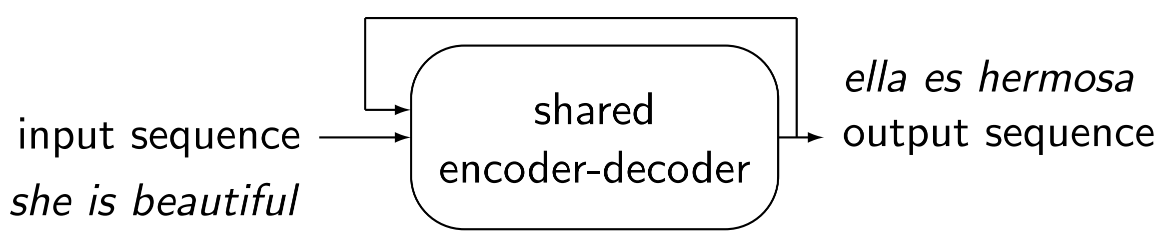
RNN-type encoder-decoder architecture

- components
  - embedding layer - convert input tokens to vector representations
  - RNN layers - process sequential information
  - unembedding (unemb) layer - convert vectors back to vocabulary space
  - softmax - produce probability distribution over vocabulary
- RNN can be basic RNN, LSTM, GRU, other specialized architecture



Shared encoder-decoder model

- single neural network structure can handle both encoding & decoding tasks
  - efficient architecture reducing model complexity
  - allow for better parameter sharing across tasks
- widely used in modern LLMs to process & generate text sequences
  - applications - machine translation, text summarization, question answering
- advantages
  - efficient use of parameters, versatile for multiple NLP tasks



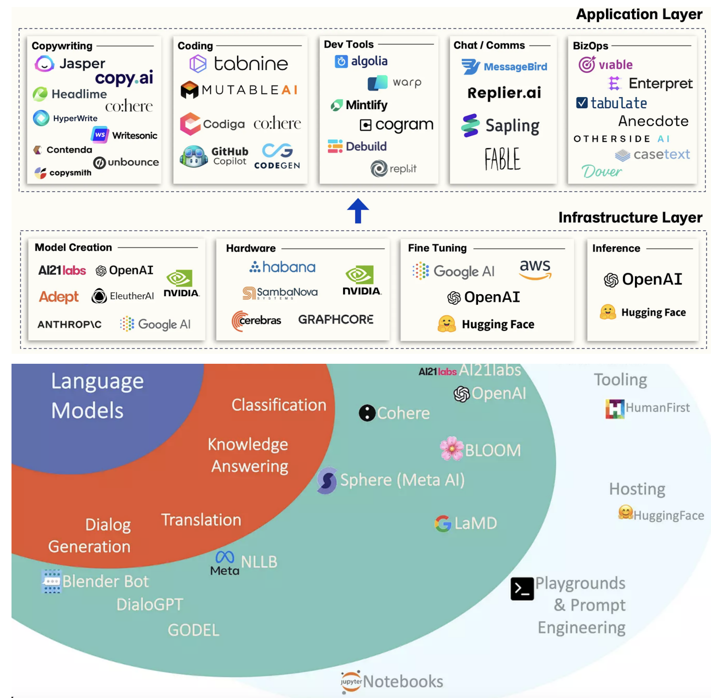
Large Language Models

LLM

- LLM
  - type of AI aimed for NLP trained on massive corpus of texts & programming code
  - allow learn statistical relationships between words & phrases, $i.e.$, conditional probabilities
  - *amazing performance shocked everyone - unreasonable effectiveness of data (Halevry et al., 2009)*
- applications
  - conversational AI agent / virtual assistant
  - machine translation / text summarization / content creation / sentiment analysis / question answering
  - code generation
  - market research / legal service / insurance policy / triange hiring candidates
  - \+ virtually infinite # of applications

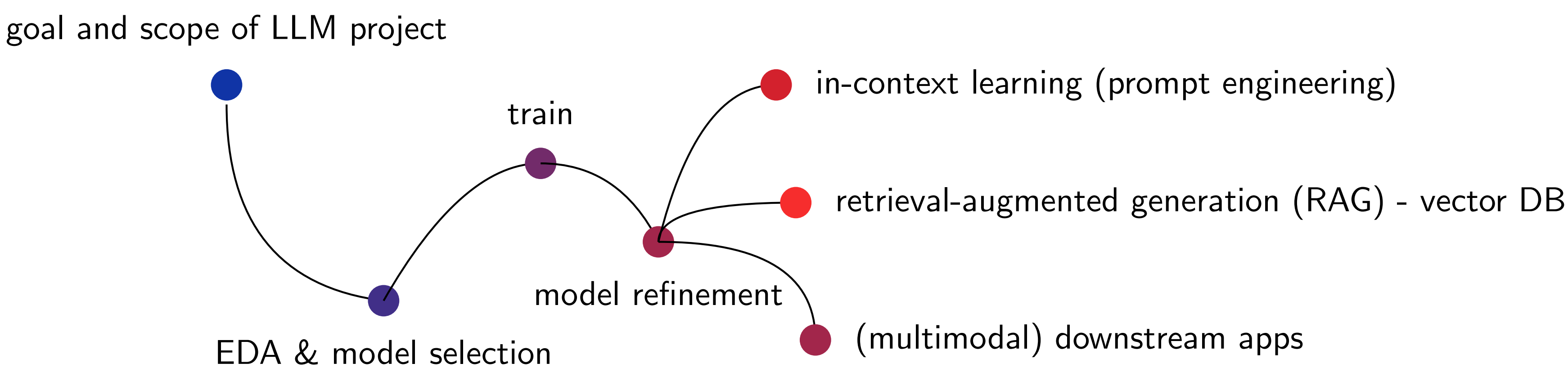
LLMs

- Foundation Models
  - GPT-x/Chat-GPT - OpenAI, Llama-x - Meta, PaLM-x (Bard) - Google
- # parameters
  - generative pre-trained transfomer (GPT) - GPT-1: 117M, GPT-2: 1.5B, GPT-3: 175B, GPT-4: 100T, GPT-4o: 200B
  - large language model Meta AI (Llama) - Llama1: 65B, Llama2: 70B, Llama3: 70B
  - scaling language modeling with pathways (PaLM) - 540B
- burns lots of cash on GPUs!
- applicable to many NLP & genAI applications



LLM building blocks

- data - trained on massive datasets of text & code
  - quality & size critical on performance
- architecture - GPT/Llama/Mistral
  - can make huge difference
- training - self-supervised/supervised learning
- inference - generates outputs
  - in-context learning, prompt engineering



Transformer

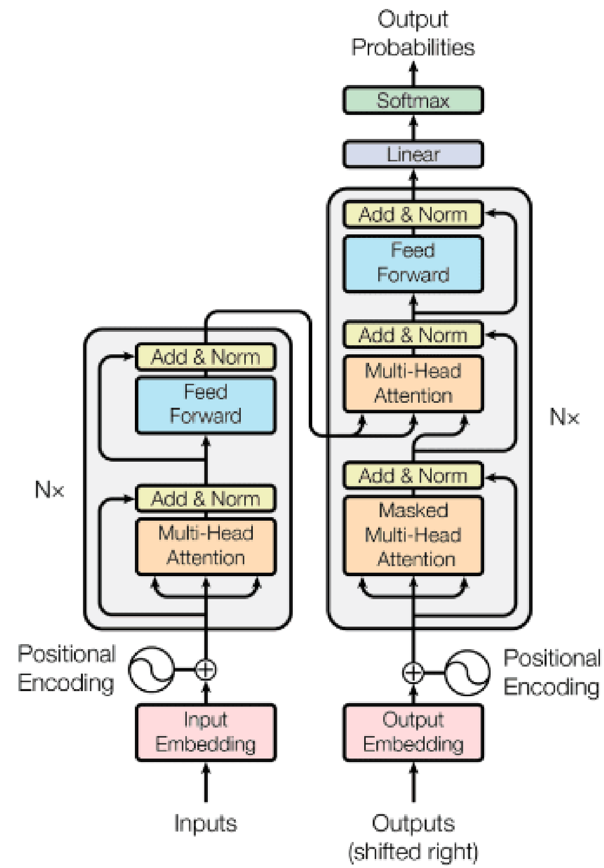
LLM architectural secret (or known) sauce

Transformer - simple parallelizable attention mechanism

A. Vaswani, et al. Attention is All You Need, 2017

Transformer architecture

- encoding-decoding architecture
  - input embedding space → multi-head & mult-layer representation space → output embedding space
- additive positional encoding - information regarding order of words @ input embedding
- multi-layer and multi-head attention followed by addition / normalization & feed forward (FF) layers
- *(relatively simple) attentions*
  - single-head (scaled dot-product) / multi-head attention
  - self attention / encoder-decoder attention
  - masked attention
- benefits
  - *evaluate dependencies between arbitrarily distant words*
  - has recurrent nature w/o recurrent architecture → parallelizable → fast w/ additional cost in computation



Single-head scaled dot-product attention

- values/keys/queries denote value/key/query *vectors*, d_k & d_v are lengths of keys/queries & vectors
- we use *standard* notions for matrices and vectors - not transposed version that (almost) all ML scientists (wrongly) use

- output: weighted-average of values where weights are attentions among tokens
- assume n queries and m key-value pairs

$$Q \in \mathbf{R}^{d_k \times n}, K \in \mathbf{R}^{d_k \times m}, V \in \mathbf{R}^{d_v \times m}$$

- attention! outputs n values (since we have n queries)

$$\mathrm{Attention}(Q, K, V) = V \mathsf{softmax}\left(K^T Q / \sqrt{d_k}\right) \in \mathbf{R}^{d_v \times n}$$

- *much simpler attention mechanism than previous work*
  - attention weights were output of complicated non-linear NN

Single-head - close look at equations

- focus on ith query, $q_i \in \mathbf{R}^{d_k}$, $Q = \left[\begin{array}{ccc} - & q_i & - \end{array}\right] \in \mathbf{R}^{d_k \times n}$
- assume m keys and m values, $k_1, \ldots, k_m \in \mathbf{R}^{d_k}$ & $v_1, \ldots, v_m \in \mathbf{R}^{d_v}$

$$K = \left[\begin{array}{ccc} k_1 & \cdots & k_m \end{array}\right] \in \mathbf{R}^{d_k \times m}, V = \left[\begin{array}{ccc} v_1 & \cdots & v_m \end{array}\right] \in \mathbf{R}^{d_v \times m}$$

- then

$$K^T Q / \sqrt{d_k} = \left[\begin{array}{ccc} & \vdots & \\ - & k_j^T q_i / \sqrt{d_k} & - \\ & \vdots & \end{array}\right]$$

$e.g.$, dependency between ith output token and jth input token is

$$a_{ij} = \exp\left(k_j^T q_i / \sqrt{d_k}\right) / \sum_{j=1}^{m} \exp\left(k_j^T q_i / \sqrt{d_k}\right)$$

- value obtained by ith query, q_i in $\mathrm{Attention}(Q, K, V)$

$$a_{i,1} v_1 + \cdots + a_{i,m} v_m$$

Multi-head attention

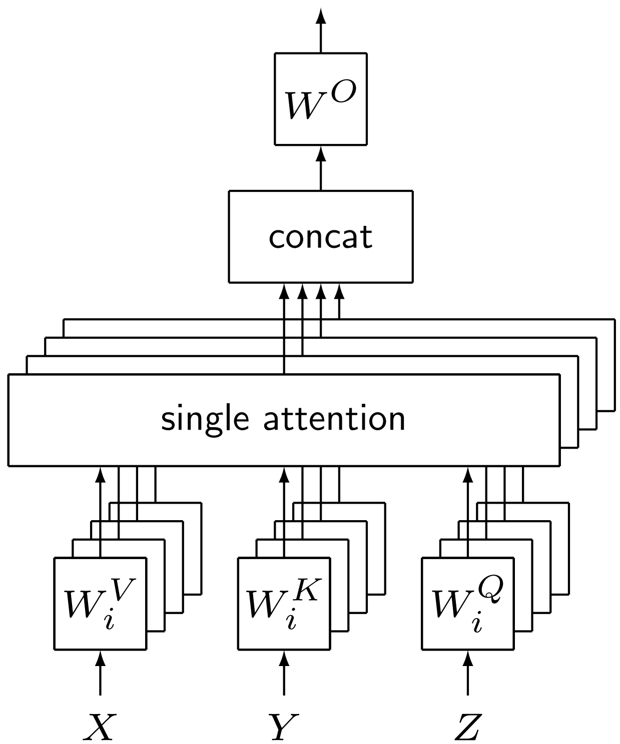
- evaluate h single-head attentions (in parallel)
- d_e: dimension for embeddings
- embeddings

$$X \in \mathbf{R}^{de \times m},\ Y \in \mathbf{R}^{de \times m},\ Z \in \mathbf{R}^{de \times n}$$

  e.g., n: input sequence length & m: output sequence length in machine translation
- h key/query/value weight matrices: $W_i^K, W_i^Q \in \mathbf{R}^{d_k \times de}$, $W_i^V \in \mathbf{R}^{dv \times de}$ $(i = 1, \ldots, h)$
- linear output layers: $W^O \in \mathbf{R}^{de \times hdv}$
- *multi-head attention!*

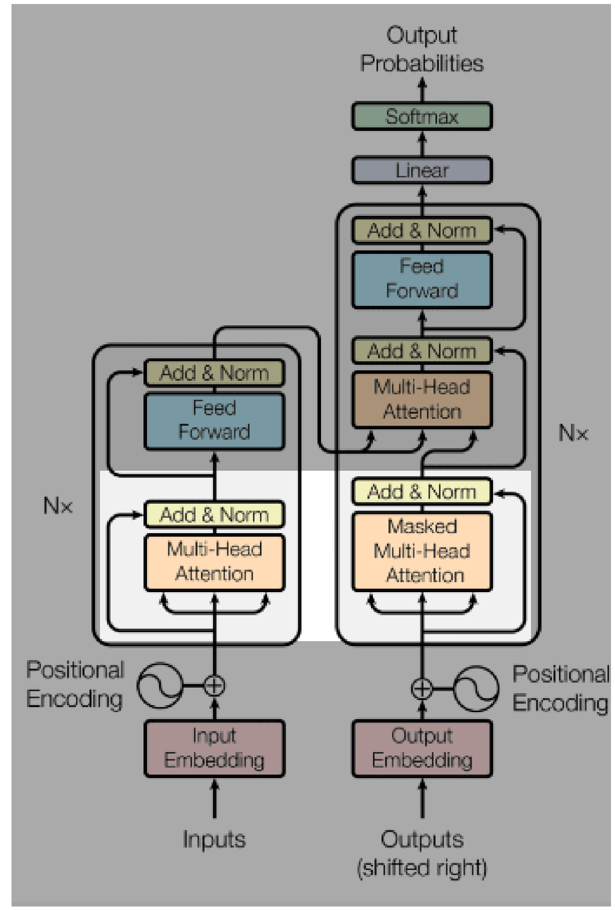
$$W^O \begin{bmatrix} A_1 \\ \vdots \\ A_h \end{bmatrix} \in \mathbf{R}^{de \times n},$$

$$A_i = \mathrm{Attention}(W_i^Q Z, W_i^K Y, W_i^V X) \in \mathbf{R}^{dv \times n}$$



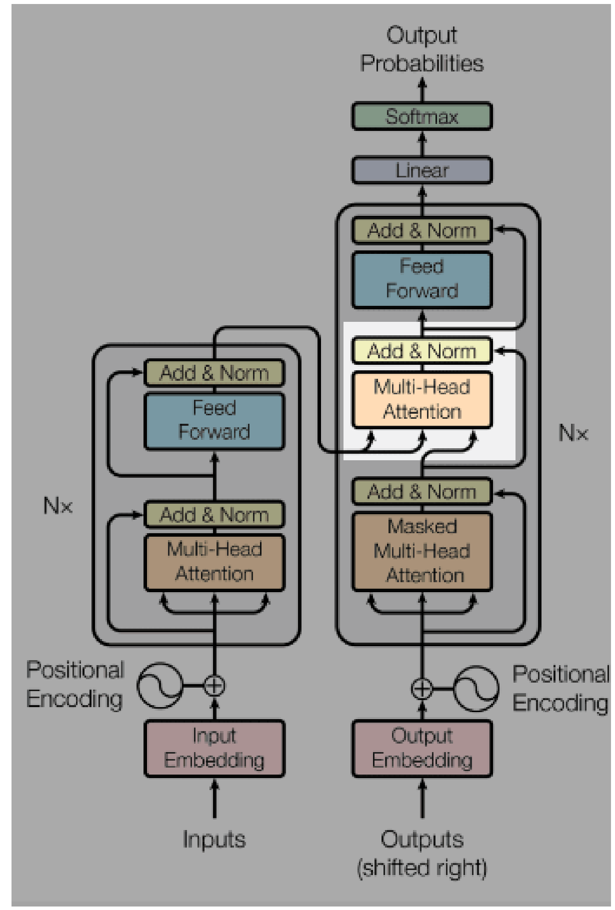
Self attention

- $m = n$
- encoder
  - keys & values & queries (K, V, Q) come from same place (from previous layer)
  - every token attends to every other token in input sequence
- decoder
  - keys & values & queries (K, V, Q) come from same place (from previous layer)
  - every token attends to other tokens up to that position
  - prevent leftward information flow to right to preserve causality
  - assign $-\infty$ for illegal connections in softmax (masking)



Encoder-decoder attention

- m: length of input sequence
- n: length of output sequence
- n queries (Q) come from previous decoder layer
- m keys / m values (K, V) come from output of encoder
- every token in output sequence attends to every token in input sequence

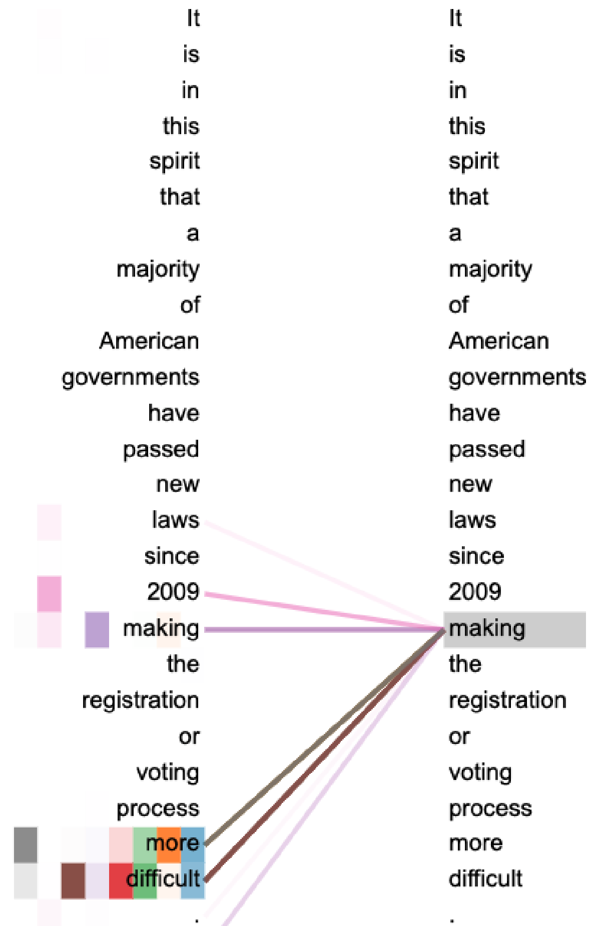


Visualization of self attentions

example sentence

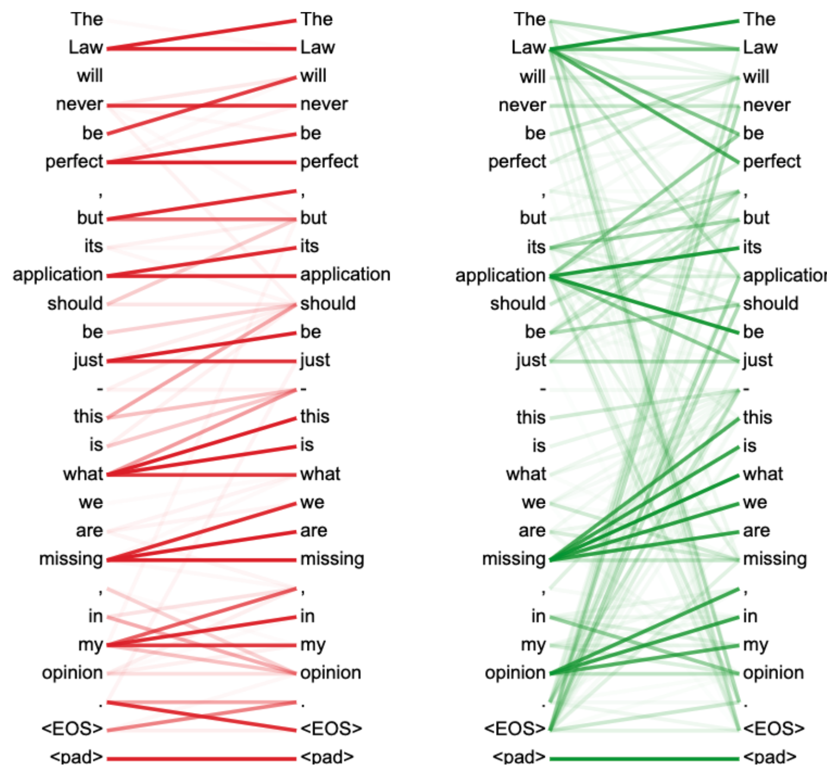
"It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult."

- self attention of encoder (of a layer)
  - right figure
    - show dependencies between "making" and other words
    - different columns of colors represent different heads
  - "making" has strong dependency to "2009", "more", and "difficult"

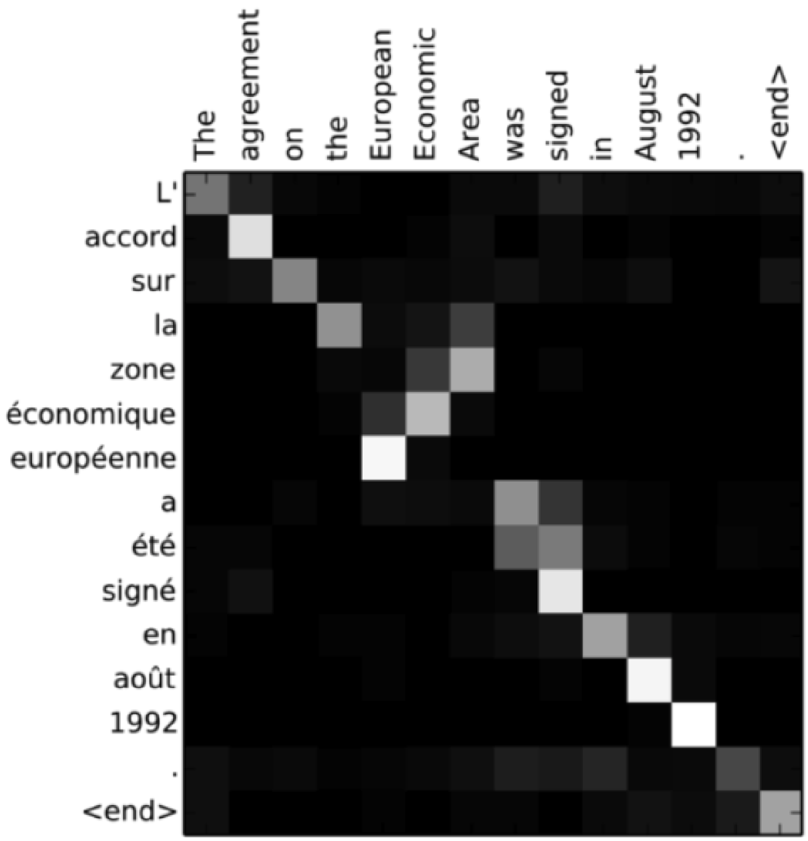


Visualization of multi-head self attentions

- self attentions of encoder for two heads (of a layer)
  - different heads represent different structures
    $\rightarrow$ advantages of multiple heads
  - multiple heads work together to colletively yield good results
  - dependencies *not* have absolute meanings (like embeddings in collaborative filtering)
  - randomness in resulting dependencies exists due to stochastic nature of ML training



Visualization of encoder-decoder attentions



- machine translation: English $\rightarrow$ French
  - input sentence: "The agreement on the European Economic Area was signed in August 1992."
  - output sentence: "L' accord sur la zone économique européenne a été signé en août 1992."
- encoder-decoder attention reveals relevance between
  - European $\leftrightarrow$ européenne
  - Economic $\leftrightarrow$ européconomique
  - Area $\leftrightarrow$ zone

Model complexity

- computational complexity
  - n: sequence length, d: embedding dimension
  - complexity per layer - self-attention: $\mathcal{O}(n^2 d)$, recurrent: $\mathcal{O}(1)$
  - sequential operations - self-attention: $\mathcal{O}(1)$, recurrent: $\mathcal{O}(n)$
  - maximum path length - self-attention: $\mathcal{O}(1)$, recurrent: $\mathcal{O}(n)$

- *massive parallel processing, long context windows*

  $\longrightarrow$ *makes NVidia more competitive, hence profitable!*

  $\longrightarrow$ *makes SK Hynix prevail HBM market!*

Variants of Transformer

Bidirectional encoder representations from transformers (BERT)

- Bidirectional Encoder Representations from Transformers [DCLT19]
- pre-train deep bidirectional representations from unlabeled text
- fine-tunable for multiple purposes

input	[cls]	my	dog	is	cute	[sep]	she	likes	playing	[sep]
token embeddings	$E_{\text{[cls]}}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{\text{[sep]}}$	E_{she}	E_{likes}	E_{playing}	$E_{\text{[sep]}}$
	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
segment embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B
	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
poistional embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9

Challenges in LLMs

- *hallucination - can give entirely plausible outcome that is false*
- data poison attack
- unethical or illegal content generation
- huge resource necessary for both training & inference
- model size - need compact models
- outdated knowledge - can be couple of years old
- lack of reproducibility
- *biases - more on this later . . .*

do not, though, focus on downsides but on *infinite possibilities!*

- it evolves like internet / mobile / electricity
- only "tip of the iceburg" found & releaved

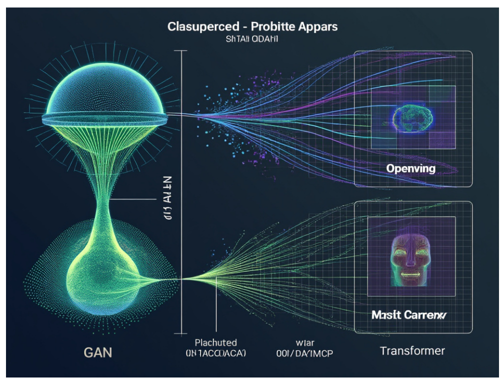
genAI

Definition of genAI

Generative AI

- genAI refers to systems capable of producing new (& original) contents based on patterns learned from training data (representation learning)
  - as opposed to discriminative models for, *e.g.*, classification, prediction & regression
  - here content can be text, images, audio, video, *etc.* - what about smell & taste?
- genAI model examples
  - generative adversarial networks (GANs), variational autoencoders (VAEs), diffusion models, Transformers


by Midjourney



by Grok 2 mini


by Generative AI Lab

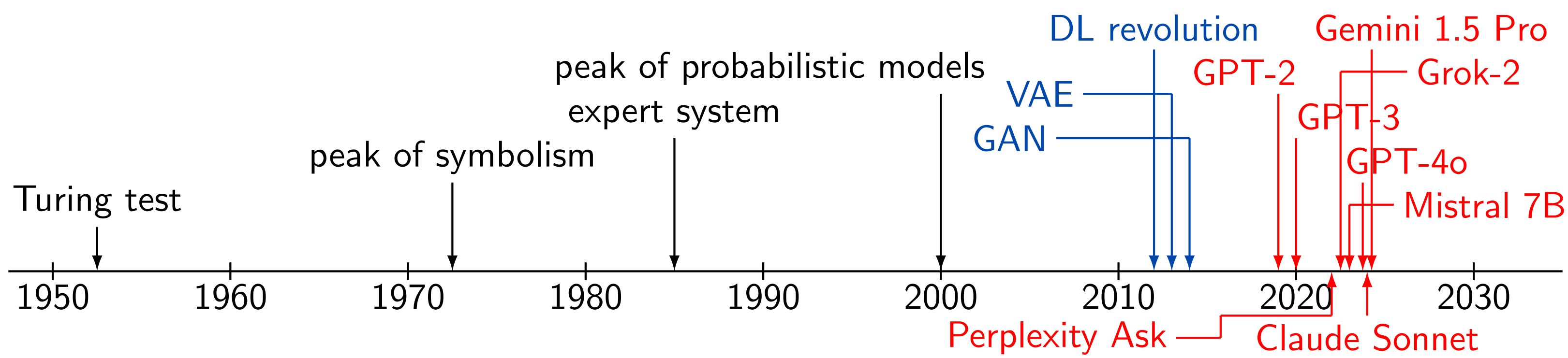
Examples of genAI in action

- text generation
  - Claude, ChatGPT, Mistral, Perplexity, Gemini, Grok
  - conversational agent writing articles, code & even poetry
- image generation
  - DALL-E - creates images based on textual descriptions
  - Stable Diffusion - uses diffusion process to generate high-quality images from text prompts (by denoising random noise)
  - MidJourney - art and visual designs generated through deep learning
- music generation
  - Amper Music - generates unique music compositions
- code generation
  - GitHub Copilot - generates code snippets based on natural language prompts

History of genAI

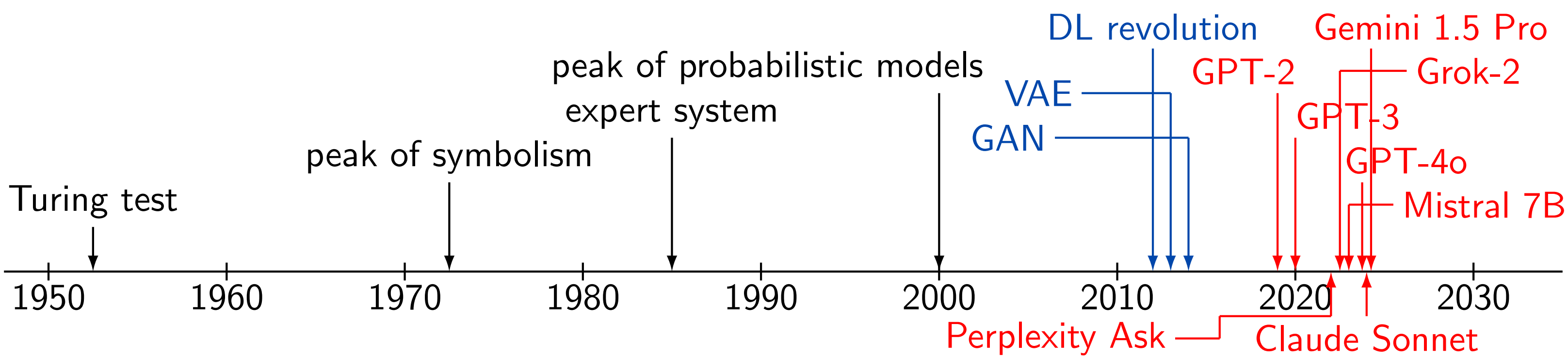
Birth of AI - early foundations & precursor technologies

- 1950s $\sim$ 1970s
  - Alan Turing - concept of *"thinking machine" & Turing test* to evaluate machine intelligence (1950s)
  - *symbolists* (as opposed to connectionists) - early AI focused on symbolic reasoning, logic & problem-solving - Dartmouth Conference in 1956 by *John McCarthy, Marvin Minsky, Allen Newell & Herbert A. Simon*
  - precursor technologies - genetic algorithms (GAs), Markov chains & *hidden Markov models (HMMs)* - laying foundation for generative processes (1970s $\sim$)



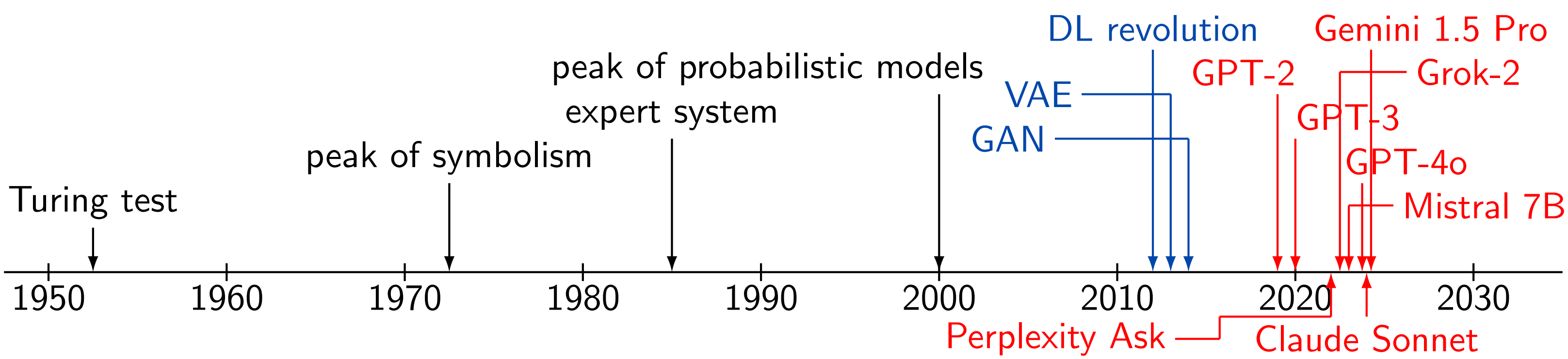
Rule-based systems & probabilistic models

- 1980s $\sim$ early 2000s
  - *expert systems* (1980s) - AI systems designed to mimic human decision-making in specific domains
  - development of neural networks (NN) w/ backpropagation *training multi-layered networks* - setting stage for way more complex generative models
  - *probabilistic models* (including network models, $i.e.$, Bayesian networks) & Markov models - laying groundwork for data generation & pattern prediction



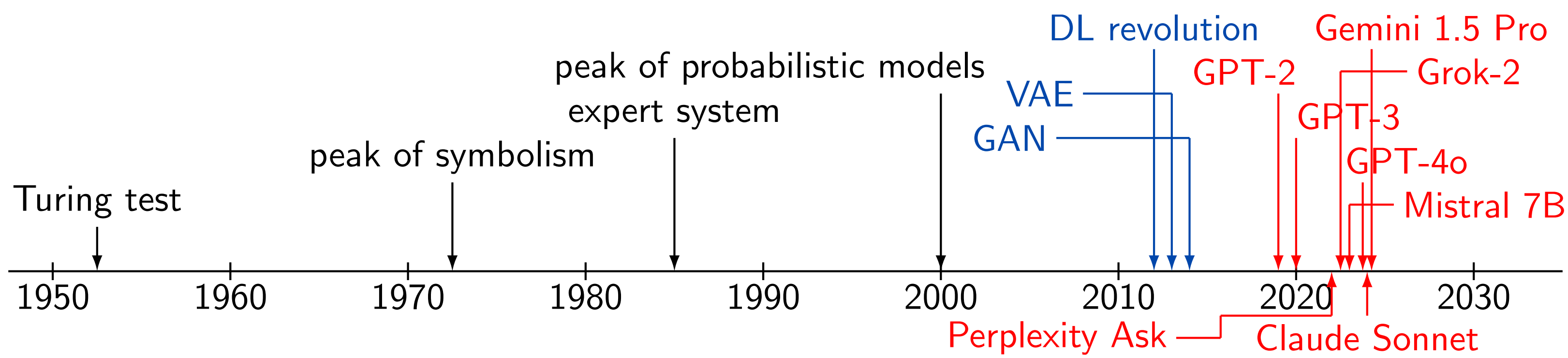
Rise of deep learning & generative models

- 2010s - breakthrough in genAI
  - *deep learning (DL) revolution* - advances in GPU computing and data availability led to the rapid development of deep neural networks.
  - *variational autoencoder (VAE)* (2013) - by Kingma and Welling - learns mappings between input and latent spaces
  - *generative adversarial network (GAN)* (2014) - by Ian Goodfellow - game-changer in generative modeling where two NNs compete each other to create realistic data
    - widely used in image generation & creative tasks



Transformer models & multimodal AI

- late 2010s $\sim$ Present
  - Transformer architecture (2017) - by Vaswani et al.
    - *revolutionized NLP,* $e.g.$, LLM & various genAI models
  - GPT series - generative pre-trained transformer
    - GPT-2 (2019) - generating human-like texts - *marking leap in language models*
    - GPT-3 (2020) - 175B params - set *new standards for LLM*
  - multimodal systems - DALL-E & CLIP (2021) - *linking text and visual data*
  - emergence of diffusion models (2020s) - new approach for generating high-quality images - progressively "denoising" random noise (DALL-E 2 & Stable Diffusion)



Mathy Views on genAI

genAI models

- definition of generative model

$$\boxed{\mathcal{Z}} \xrightarrow{g_\theta(z)} \boxed{\mathcal{X}}$$

- *generate samples in original space, $\mathcal{X}$, from samples in latent space, $\mathcal{Z}$*
- g_θ is parameterized model $e.g.$, CNN / RNN / Transformer / diffuction-based model
- training
  - finding θ that minimizes/maximizes some (statistical) loss/merit function so that $\{g_\theta(z)\}_{z \in \mathcal{Z}}$ generates plausiable point in $\mathcal{X}$
- inference
  - random samples z to generated target samples $x = g_\theta(z)$
  - $e.g.$, image, text, voice, music, video

VAE - early genAI model

- variational auto-encoder (VAE) [KW19]

$$\boxed{\mathcal{X}} \xrightarrow{q_\phi(z|x)} \boxed{\mathcal{Z}\circ} \xrightarrow{p_\theta(x|z)} \boxed{\mathcal{X}}$$

- log-likelihood & ELBO - for any $q_\phi(z|x)$

$$\begin{aligned}\log p_\theta(x) &= \mathop{\mathbf{E}}_{z\sim q_\phi(z|x)} \log p_\theta(x) = \mathop{\mathbf{E}}_{z\sim q_\phi(z|x)} \log \frac{p_\theta(x,z)}{q_\phi(z|x)} \cdot \frac{q_\phi(z|x)}{p_\theta(z|x)} \\ &= \mathcal{L}(\theta,\phi;x) + D_{KL}(q_\phi(z|x) \| p_\theta(z|x)) \geq \mathcal{L}(\theta,\phi;x)\end{aligned}$$

- (indirectly) maximize likelihood by maximizing evidence lower bound (ELBO)

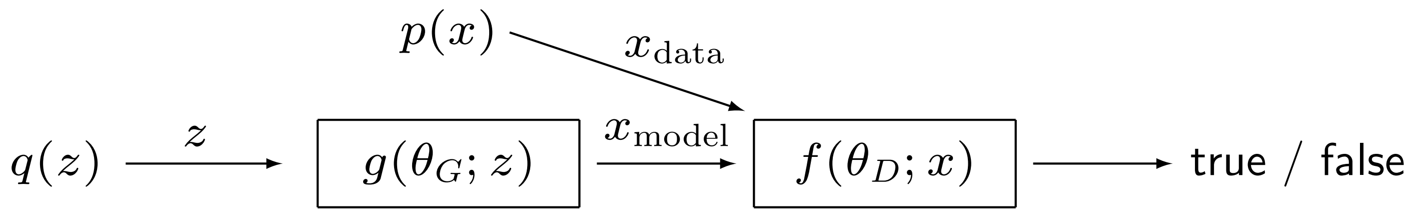
$$\mathcal{L}(\theta,\phi;x) = \mathop{\mathbf{E}}_{z\sim q_\phi(z|x)} \log \frac{p_\theta(x,z)}{q_\phi(z|x)}$$

- generative model

$$p_\theta(x|z)$$

GAN - early genAI model

- generative adversarial networks (GAN) [GPAM+14]



  - value function

$$V(\theta_D, \theta_G) = \mathop{\mathbf{E}}_{x \sim p(x)} \log f(\theta_D; x)) + \mathop{\mathbf{E}}_{z \sim q(z)} \log(1 - f(\theta_D; g(\theta_G; z)))$$

  - modeling via playing min-max game

$$\min_{\theta_G} \max_{\theta_D} V(\theta_D, \theta_G)$$

  - generative model

$$g(\theta_G; z)$$

  - variants: conditional / cycle / style / Wasserstein GAN

genAI - LLM

- *maximize conditional probability*

$$\underset{\theta}{\text{maximize}} \;\; d\left(p_\theta(x_t|x_{t-1}, x_{t-2}, \ldots), p_{\text{data}}(x_t|x_{t-1}, x_{t-2}, \ldots)\right)$$

  where $d(\cdot, \cdot)$ distance measure between probability distributions

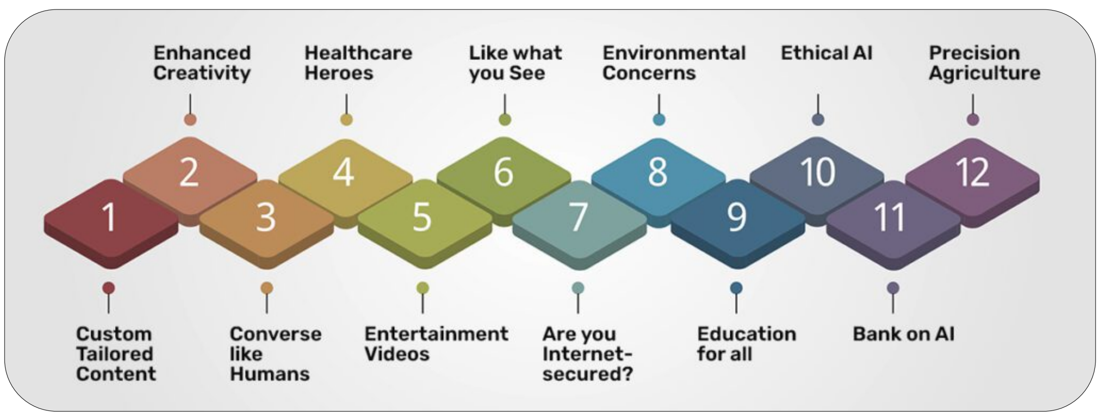
  - previous sequence: $x_{t-1}, x_{t-2}, \ldots$
  - next token: x_t
- p_θ represented by (extremely) complicated model
  - $e.g.$, containing multi-head & multi-layer Transformer architecture inside
- model parameters, $e.g.$, for Llama2

$$\theta \in \mathbf{R}^{70,000,000,000}$$

Current Trend & Future Perspectives

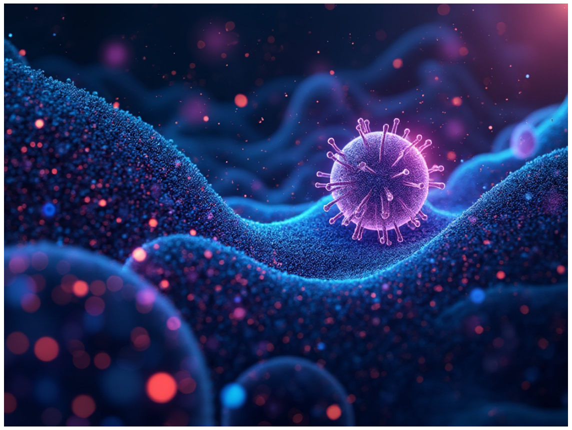
Current trend of genAI

- rapid advancement in language models & multimodal AI capabilities
- rise of AI-assisted creativity & productivity tools
- growing adoption across industries
  - creative industries - design, entertainment, marketing, software development
  - life sciences - healthcare, medical, biotech
- infrastructure & accessibility, *e.g.*, Hugging Face democratizes AI development
- integration with cloud platforms & enterprise-level tools
- increased focus on AI ethics & responsible development



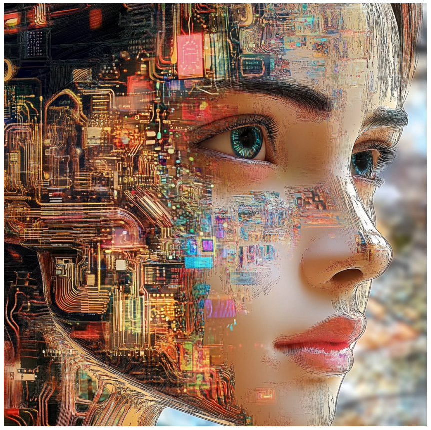
Industry & business impacts

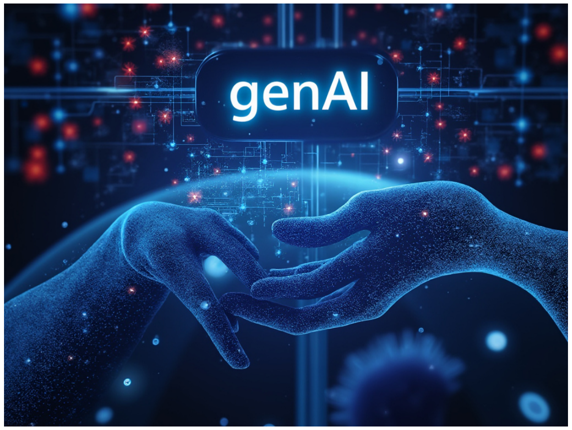
- how genAI is transforming industries
  - creative industries - content creation - advertising, gaming, film
  - life science - enhance research, drug discovery & personalized treatments
  - finance - automating document generation, risk modeling & fraud detection
  - manufacturing & Design - rapid prototyping, 3D modeling & optimization
  - business operations - automate routine tasks to boost productivity



Future perspectives of genAI

- hyper-personalization - highly personalized content for individual users - music, products & services
- AI ethics & governance - concerns over deepfakes, misinformation & bias
- interdisciplinary synergies - integration with other fields such as quantum computing, neuroscience & robotics
- human-AI collaboration - augment human creativity rather than replace it
- energy efficiency - have to figure out how to dramatically reduce power consumption





Selected References & Sources

Selected references & sources

- Robert H. Kane "Quest for Meaning: Values, Ethics, and the Modern Experience" 2013
- Michael J. Sandel "Justice: What's the Right Thing to Do?" 2009
- Daniel Kahneman "Thinking, Fast and Slow" 2011
- Yuval Noah Harari "Sapiens: A Brief History of Humankind" 2014
- M. Shanahan "Talking About Large Language Models" 2022
- A.Y. Halevry, P. Norvig, and F. Pereira "Unreasonable Effectiveness of Data" 2009
- A. Vaswani, et al. "Attention is all you need" @ NeurIPS 2017
- S. Yin, et. al. "A Survey on Multimodal LLMs" 2023
- Chris Miller "Chip War: The Fight for the World's Most Critical Technology" 2022

- CEOs, CTOs, CFOs, COOs, CMOs & CCOs @ startup companies in Silicon Valley
- VCs on Sand Hill Road - Palo Alto, Menlo Park, Woodside in California, USA

References

References

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[KW19] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of 31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

Thank You