

PHM24 Conference Tutorial

From Perceptrons to Predictive Maintenance: An LLM Tutorial for PHM

Neil Eklund, Ph.D., FPHMS

Oak Grove Analytics, LLC

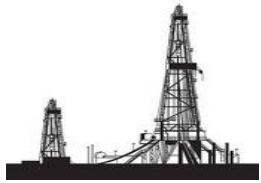
Who is this guy?



Neil Eklund, Ph.D., FPHMS

- *Principal, Oak Grove Analytics, LLC*
 - *Principal Scientist, Xerox PARC*
 - *Chief Data Scientist, Schlumberger*
 - *Senior Data Scientist, GE Digital/GE Research*
- 30 years of deep technical experience across multiple industry segments – Aerospace, Energy, Healthcare, Oil & Gas, Financial, and Rail
- External customers include DARPA, NASA, Lockheed Martin, ExxonMobil, and Boeing
- Co-founder of the *Prognostics & Health Management Society*
- Former adjunct faculty of Electrical Engineering and Computer Science, Union Graduate College:
 - System Modeling & Optimization
 - Selected Topics in Data Mining
- 100+ publications, patents, and book chapters

Deployed Applications



Drilling

- Predict success of next downhole run
- Data transmission to the cloud
- First deployed deep learning application for downhole tools
- \$10MM+ annual return



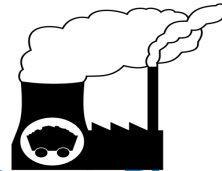
Surface Equipment

- Predict failures a week in advance
- Wellsite data transmission to the cloud
- Automated model updating
- \$50MM+ annual return



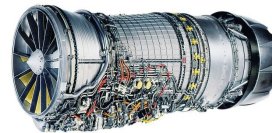
Production

- Predict failure 30 days in advance
- Structured and unstructured data
- Automated model updating
- \$20MM+ annual return



Power Plant Optimization & Control

- Multiobjective optimization of power plant
- Meet load constraint, maximize efficiency, minimize pollution
- Automated learning as the plant operates
- \$14MM+ annual return per 400MW plant



Military Aviation

- Defense Advanced Research Projects Agency (DARPA) project
- Fusion of data types (vibration, chips)
- Zero false alarms
- 5x increase in critical engine bearing spall detection capability



Commercial Aviation

- Fault detection for GENx and GE90 aircraft engines
- First deployed analytics application on GE Predix platform
- \$10MM+ annual return

Tutorial Roadmap

- Some awesome AI history
- LSTM
- Transformers
- LLMs
- Some sly subversive cultural references sprinkled throughout
- Demos
- Bloviating about LLMs

The background is a solid dark blue color. Overlaid on this are several translucent, wavy vertical lines that create a sense of depth and movement. A faint, light blue grid pattern is also visible, intersecting with the wavy lines. The text "Some Background..." is centered in the middle of the image in a white, sans-serif font.

Some Background...

13.7 Billion years ago: the Big Bang

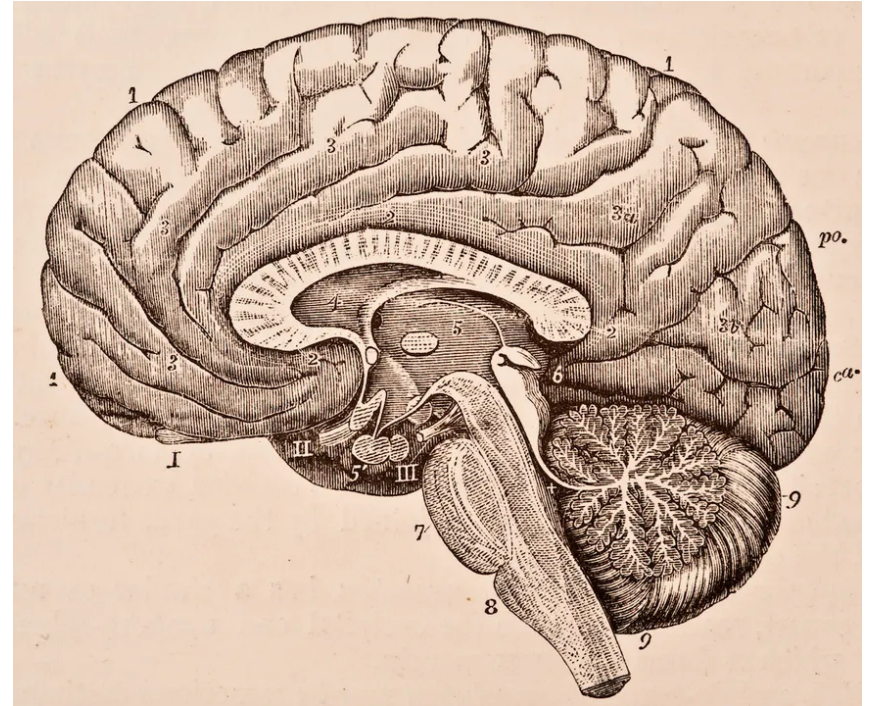


The background is a solid dark blue color. Overlaid on this are several translucent, wavy, vertical lines that create a sense of depth and movement, resembling a stylized representation of a path or a series of overlapping planes.

Tracing the Path from Perceptron to LLM

Your brain is amazing!

- **Superior Nonlinear Control and Adaptability:**
The human brain excels at controlling highly nonlinear and dynamic systems with incredible adaptability. It can seamlessly handle the complexities of nonlinear dynamics, such as those involved in balance and spatial orientation, by continuously integrating multisensory feedback to execute smooth and coordinated motor actions.
- **Efficient Real-Time Processing and Parallelism:**
The brain's ability to process vast amounts of information in real-time while effectively managing multiple control tasks simultaneously is unparalleled. For example, during activities like driving, the brain handles visual, auditory, and tactile inputs while also planning movements and anticipating future scenarios.



Why not try to emulate the brain to solve hard problems?

Cellular Neuroscience in 60 Seconds

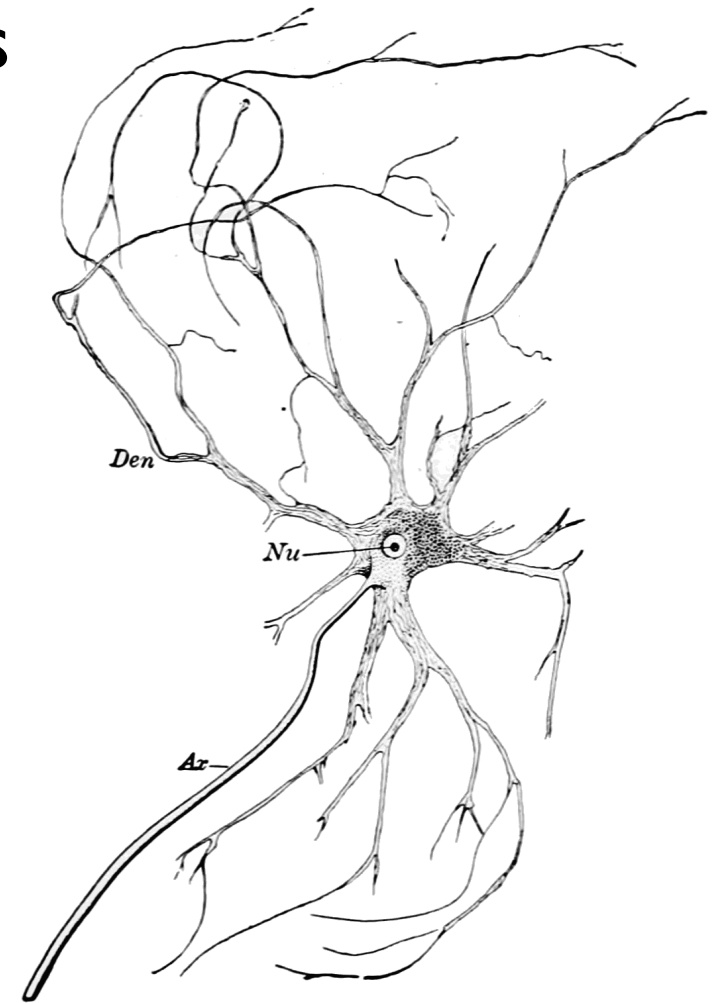
Neurons communicate through specialized structure comprising:

- **dendrites**, which receive incoming signals
- the **cell body**, which processes these signals
- and the **axon**, which transmits the electrical impulse away from the neuron.

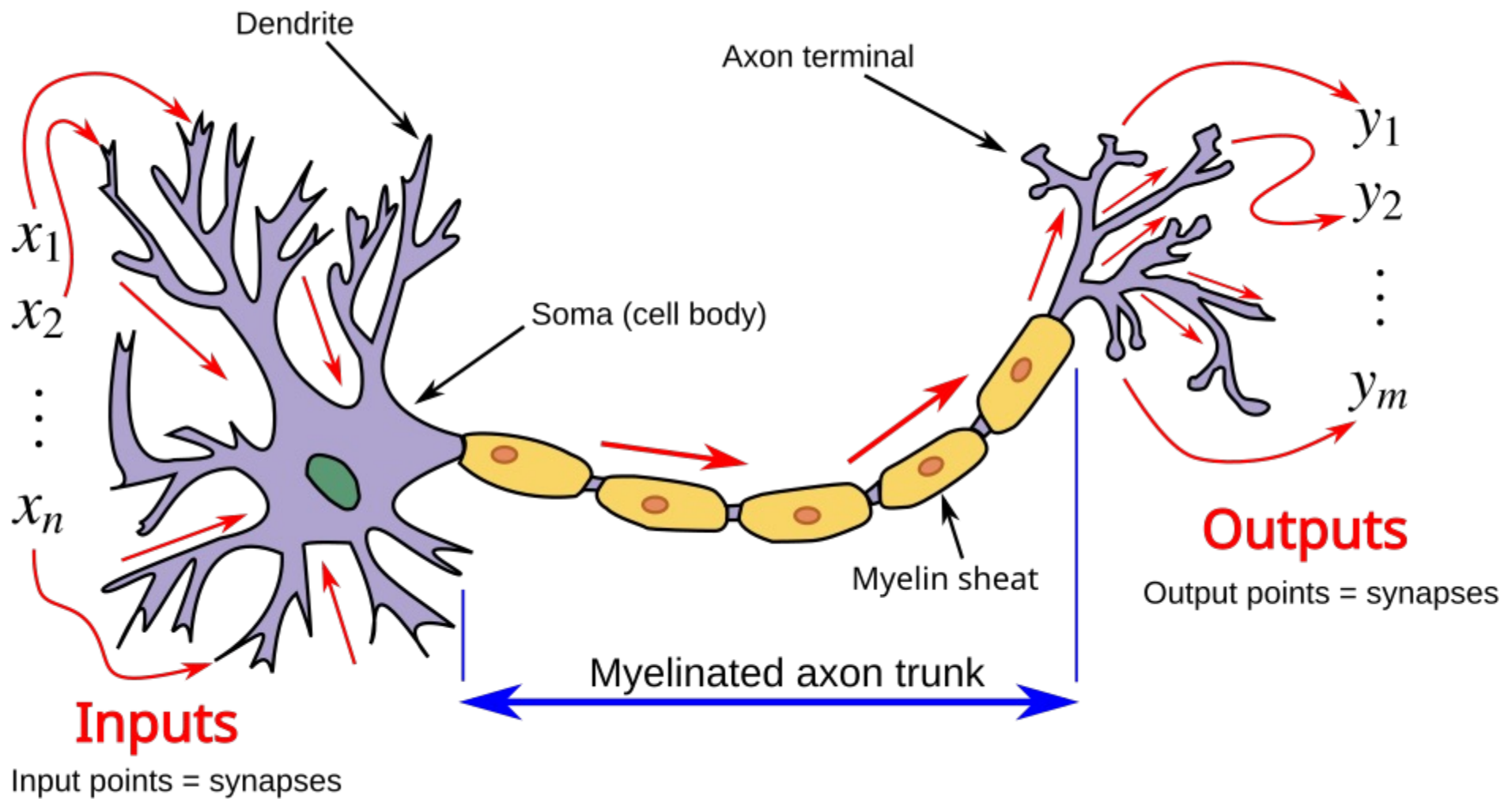
The axon terminals release neurotransmitters across a **synapse** to pass the signal to the next neuron, muscle, or gland, effectively forming a complex network for rapid information processing and response throughout the nervous system.

Excitatory connections increase the likelihood that the receiving neuron will generate an action potential (a nerve impulse). They are like an “on” switch, promoting neural activity.

In contrast, **inhibitory connections** decrease the likelihood that the receiving neuron will fire an action potential. They act as an “off” switch, regulating and restraining neural activity to prevent excessive firing.



“A single isolated motor nerve cell of an ox spinal cord” *Popular Science Monthly* volume 71, 1907.



Biologically Inspired Artificial Neuron, 1943

Bulletin of Mathematical Biophysics, Vol. 5, 1943, p. 115-133

Warren McCulloch and Walter Pitts published a seminal paper proposing a model of artificial neurons, inspired by the structure and function of the human brain.

They built a simple neural network that simulated the operation of neurons using threshold logic.

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

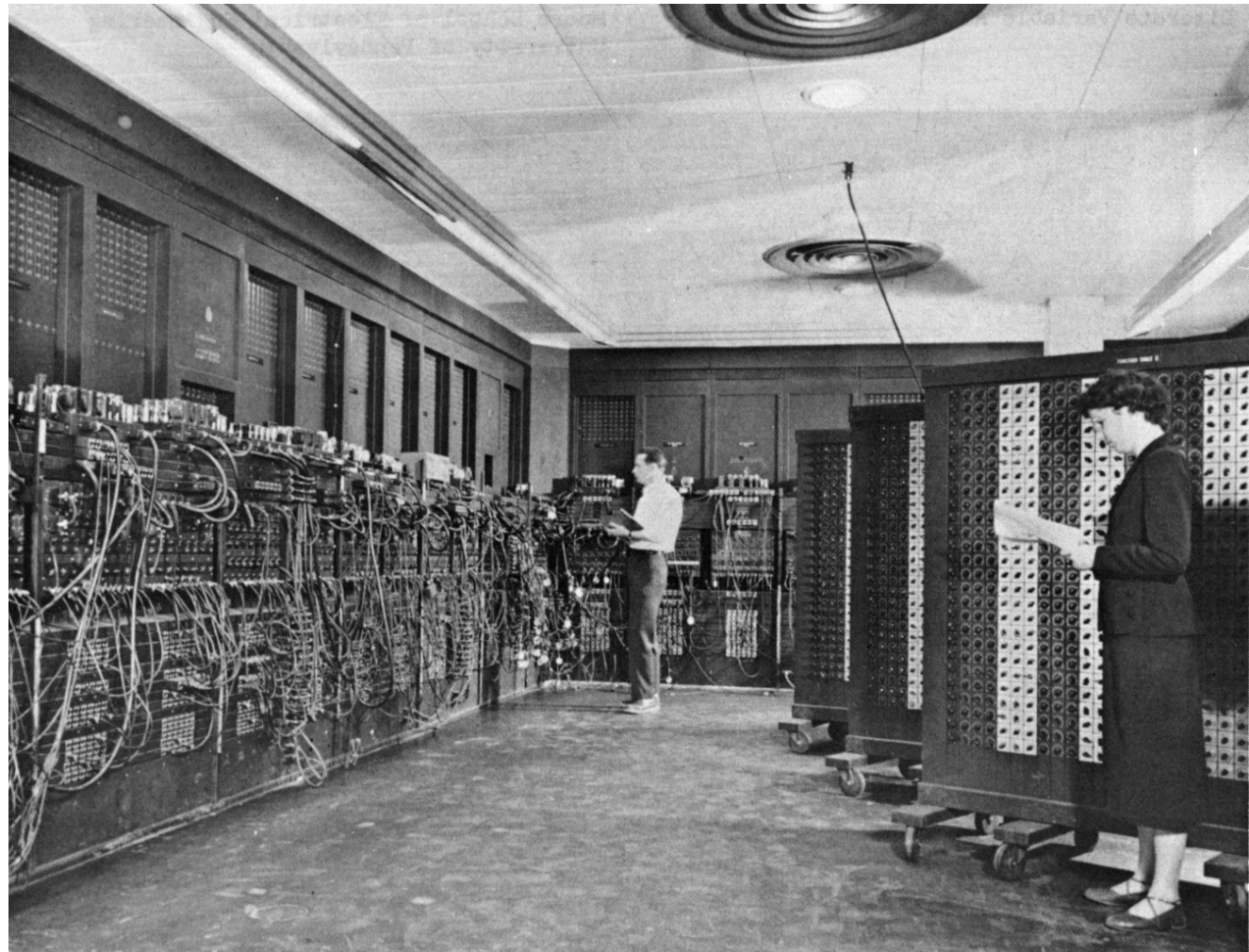
WARREN S. MCCULLOCH and WALTER H. PITTS

Because of the “all-or-none” character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

Electronic Numerical Integrator and Computer (ENIAC), 1946

First reprogrammable electronic digital computer. ENIAC was groundbreaking for its use of vacuum tubes to perform calculations, making it significantly faster than electromechanical machines of the time. It occupied a large room, consisting of approximately 18,000 vacuum tubes, 7,200 crystal diodes, 1,500 relays, and hundreds of thousands of resistors, capacitors, and inductors.

ENIAC marked a major step forward in the development of computing technology, laying the groundwork for future computers.



“cells that fire together, wire together” 14

In 1949, Donald Olding [Hebb](#) wrote *The Organization of Behaviour*. He proposes [neural flexibility](#) and that [neural pathways are reinforced each time they are used](#).

1.Synaptic Plasticity: The theory highlights the brain's ability to change and adapt by modifying the strength of connections between neurons, a concept known as synaptic plasticity.

2.Associative Learning: Hebbian learning explains how [associations between different stimuli or ideas can form](#).

3.Memory and Learning: Hebbian theory provides a cellular and molecular basis for understanding how learning and memory occur in the brain. It suggests that [repeated exposure to experiences can lead to the strengthening of certain neural pathways](#), encoding memories.

4.Hebbian Modification: This concept refers to the changes in the synaptic strength between two neurons based on their activity patterns.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*.

The Problem and the Line of Attack

Thus the only barrier to assuming that a structural change in specific neural cells is the basis of memory lies in the generalization of the perception of patterns. Man sees a square as a square, whatever its size, and in almost any setting. A rat trained to look for food behind a horizontal rectangle will there-

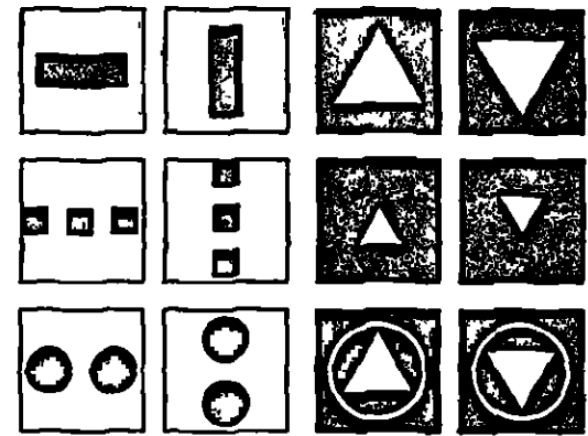


FIGURE 1 Illustrating generalization in perception. A rat is first *trained* with the two diagrams at the top, in the left half of the figure, until he always avoids the vertical bar and chooses the horizontal. He is then *tested* with the next pair of diagrams, the horizontally and vertically arranged squares, and with the next pair, circles side by side versus circles one above the other. Transfer of response occurs, the rat in each case preferring the horizontal diagram, showing that perception of horizontal and vertical is generalized beyond any particular pattern. Similarly, after training with the large plain triangles, top right, the animal is tested with smaller triangles and with circumscribed triangles. If the erect triangle, the one to the left, is positive in training, the rat will choose the left-hand diagram when presented with either of the two lower pairs, again demonstrating perceptual generalization.

Hebbian Network First Simulated “by digital computer”, 1954

Belmont Farley and Wesley Clark simulating learning and adaptive behavior in machines using networks of artificial neurons. They explored how such networks could be used to recognize patterns and learn from examples, concepts that are foundational to modern machine learning and neural networks.

Although technology at the time was limited, their pioneering efforts helped lay the groundwork for the machine learning breakthroughs that would follow in subsequent decades, particularly with the advent of more powerful computers and sophisticated algorithms.

SIMULATION OF SELF-ORGANIZING SYSTEMS BY DIGITAL COMPUTER *

B. G. Farley and W. A. Clark
Lincoln Laboratory, Massachusetts Institute of Technology
Cambridge, Massachusetts

ABSTRACT

A general discussion of ideas and definitions relating to self-organizing systems and their synthesis is given, together with remarks concerning their simulation by digital computer. Synthesis and simulation of an actual system is then described. This system, initially randomly organized within wide limits, organizes itself to perform a simple prescribed task.

Farley, B., & Clark, W. (1954). *Simulation of self-organizing systems by digital computer*. Transactions of the IRE Professional Group on Information Theory, 4(4), 76–84.

The term *Artificial Intelligence* coined, 1956

The *Dartmouth Summer Research Project on Artificial Intelligence*, held in 1956, is considered the founding event of artificial intelligence as a field of study.

- Oliver Selfridge: AI pioneer known for work in machine perception and pattern recognition.
- Nathaniel Rochester: IBM computer scientist instrumental in AI development and *designer of IBM's first scientific computer*.
- Marvin Minsky: *Co-founder of the MIT AI Lab*, contributed to AI and robotics; author of "The Society of Mind."
- John McCarthy: Coined "artificial intelligence," developed LISP, and organized the founding AI Dartmouth Conference.
- Ray Solomonoff: Pioneer in AI and algorithmic information theory; contributed to inductive inference concepts.
- Trenchard More: Contributed to early AI development, theorem proving, and formal systems at IBM.
- Claude Shannon: *Father of information theory, introduced entropy in communication, foundational in digital communication.*



In the back row from left to right are Oliver Selfridge, Nathaniel Rochester, Marvin Minsky, and John McCarthy. In front are Ray Solomonoff, Trenchard More, and Claude Shannon.

The Perceptron, 1958

Frank Rosenblatt developed a single-layer neural network designed to mimic the way the human brain processes visual information. The perceptron model consists of input nodes that feed data into a single output node through weighted connections. It learned to classify data through a process of adjusting the weights based on errors in its output, using an algorithm known as supervised learning.

It is effective at solving linearly separable problems...

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.

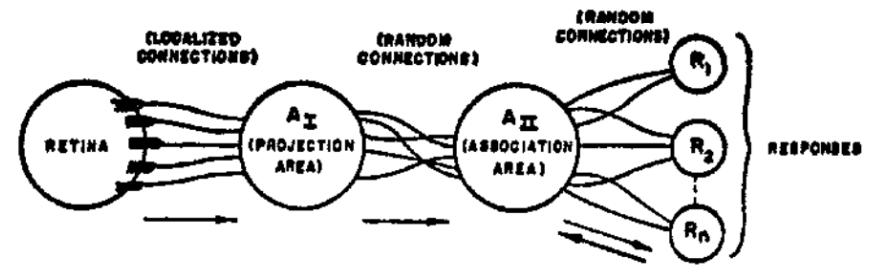
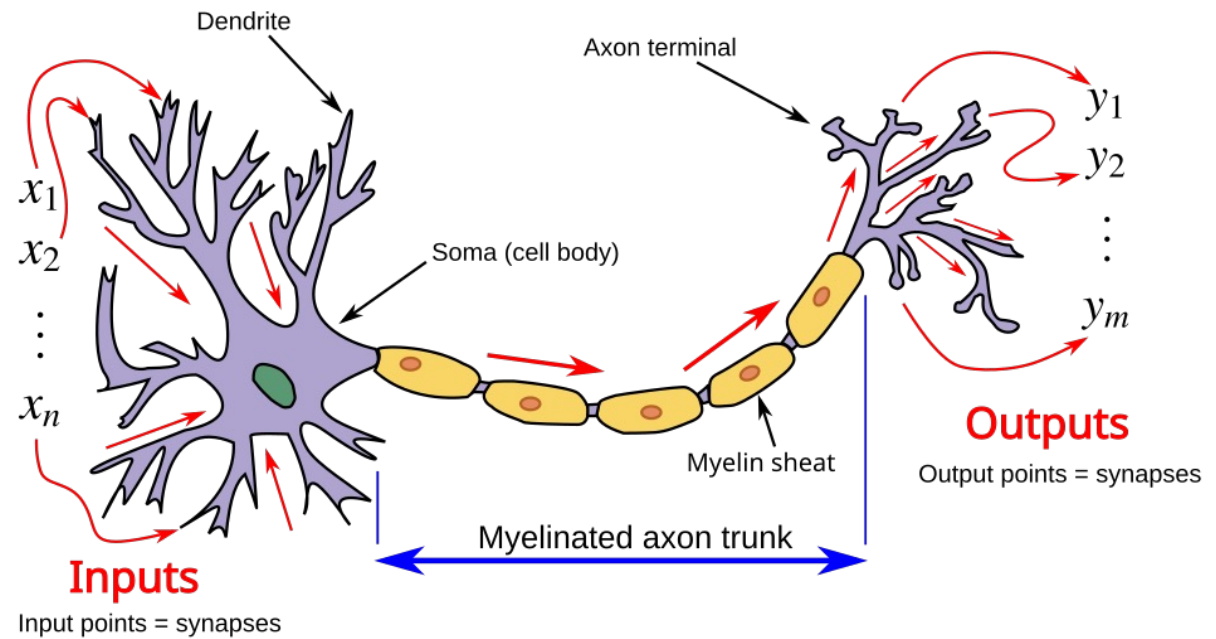
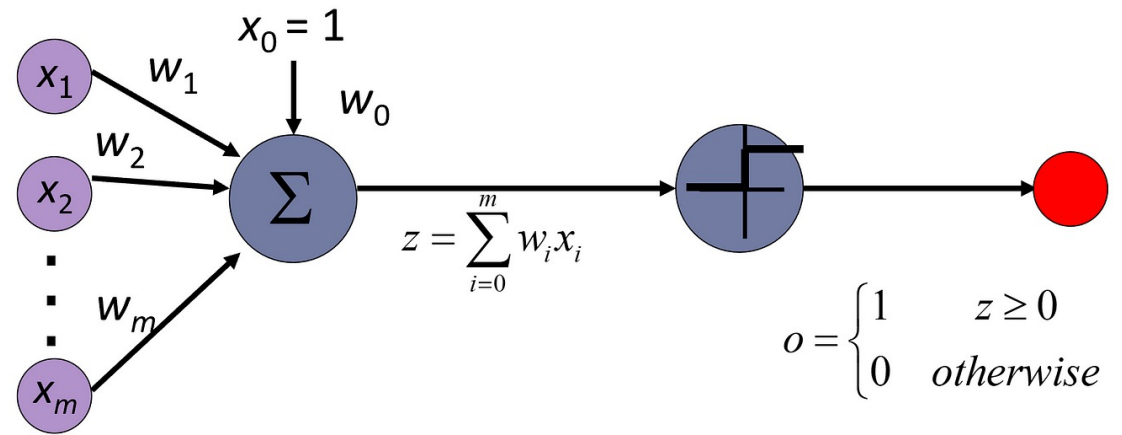


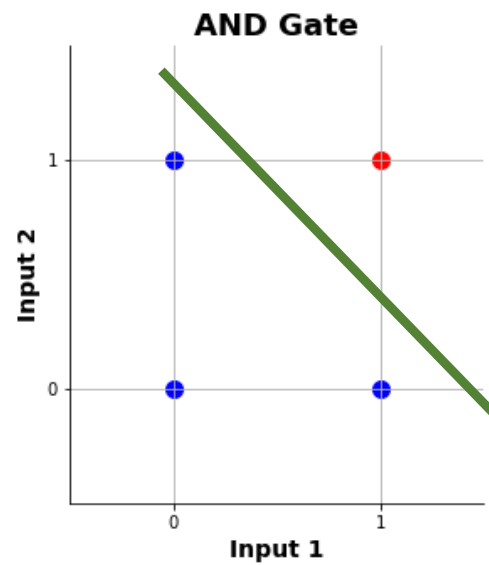
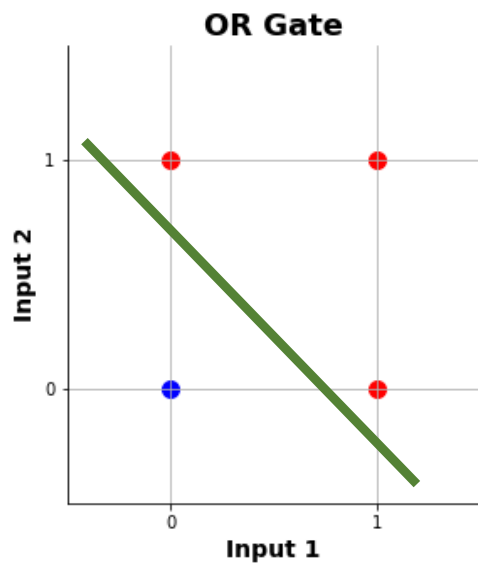
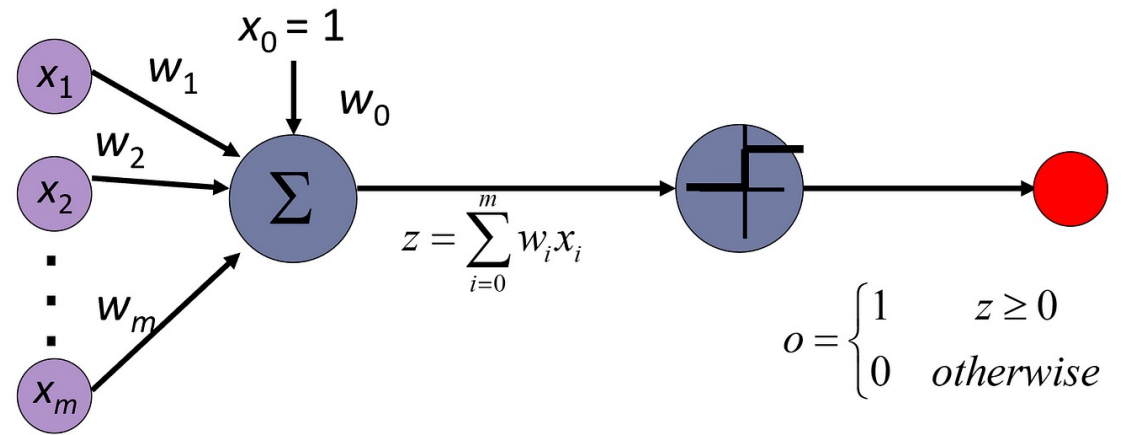
FIG. 1. Organization of a perceptron.

The cells in the projection area each receive a number of connections from the sensory points. The set of S-points transmitting impulses to a particular A-unit will be called the *origin points* of that A-unit. These origin points may be either *excitatory* or *inhibitory* in their effect on the A-unit. If the algebraic sum of excitatory and inhibitory impulse intensities is equal to or greater than the threshold (θ) of the A-unit, then the A-unit fires, again on an all-or-nothing basis (or, in some

Rosenblatt's Perceptron

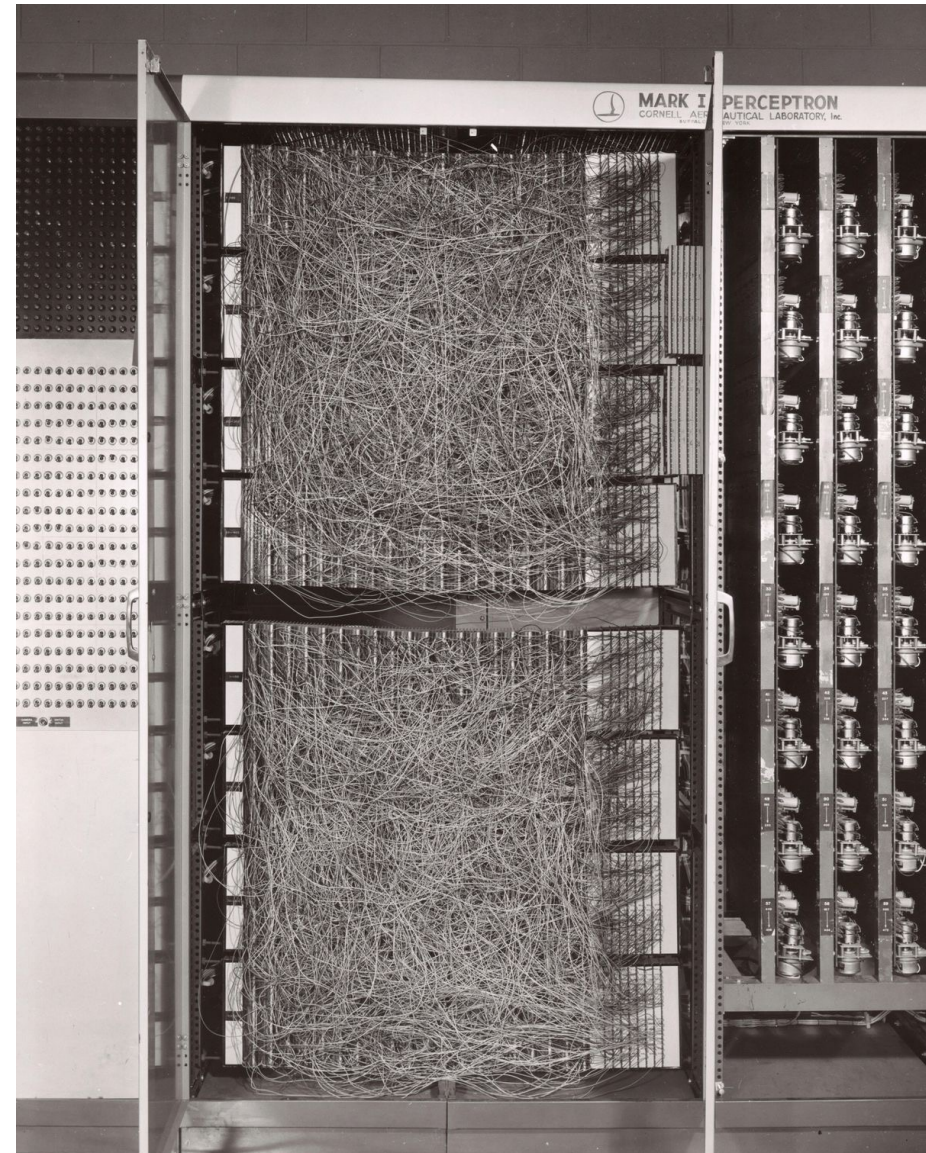


Rosenblatt's Perceptron



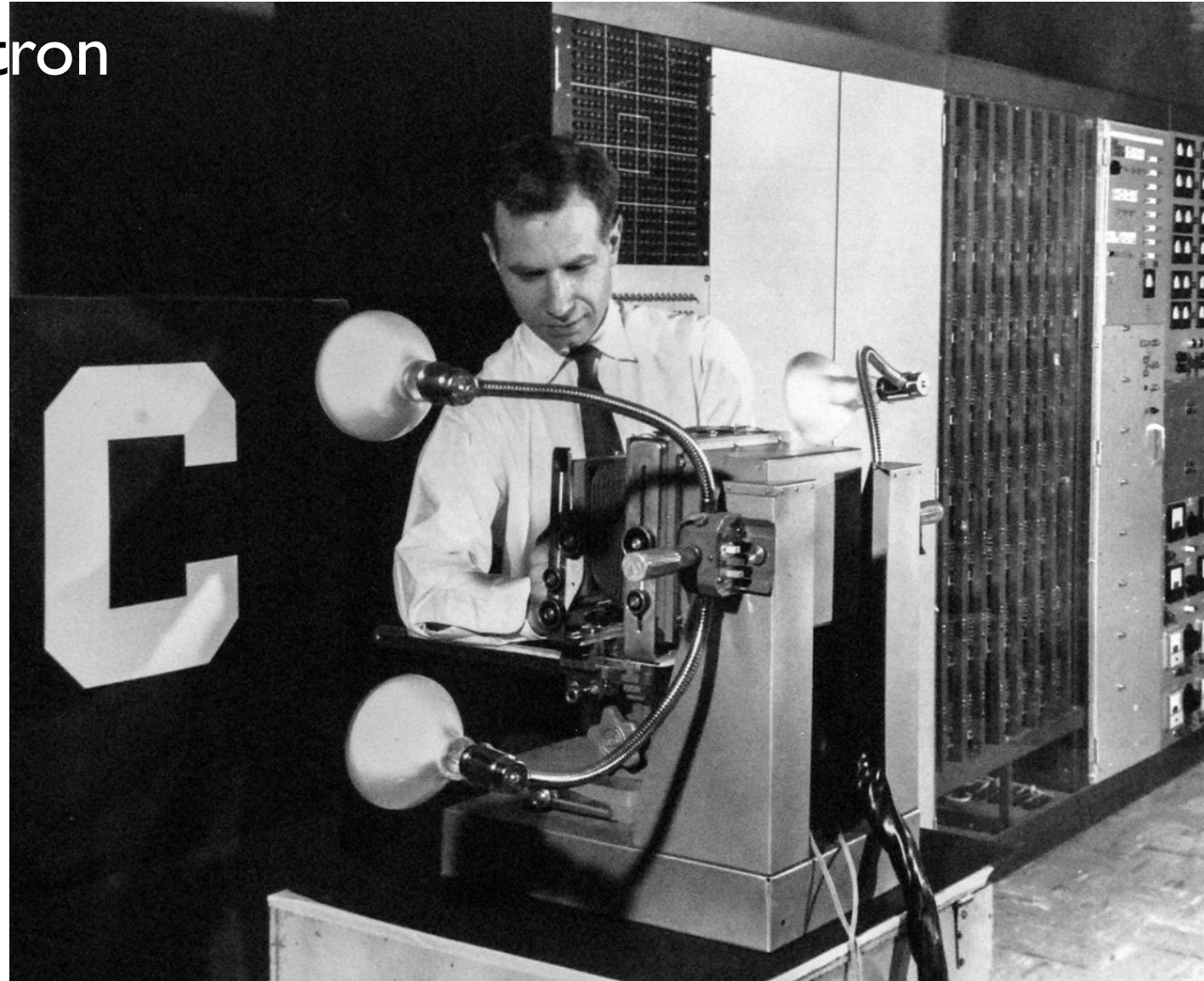
“Mark I” Perceptron

- The Perceptron was designed to recognize patterns and had the ability to learn through training.
- Two-layer architecture: an input layer and an output layer directly connected by a set of adjustable weights.
- Learning algorithm to adjust these weights based on the "error" between the desired and actual outputs.



“Mark I” Perceptron

Originally implemented as an **electronic device with a camera** to capture visual input, which it could associate with different categories.



Adaptive Linear Neuron, 1959

Adaptive switching circuits are systems that can adjust their parameters in response to changes in the environment or input signals.

ADALINE (Adaptive Linear Neuron) is a simple type of artificial neural network that consists of a single linear neuron. ADALINE adjusts its weights using a supervised learning process known as the Least Mean Squares (LMS) algorithm, which Widrow and Hoff developed. This algorithm modifies the weights and biases in an adaptive manner to minimize the error in signal processing tasks.

This network was used to implement adaptive filters, which are still actively used today. Their research found applications in various domains such as echo cancellation in telephony, seismic signal processing, noise cancellation, and many others where real-time adaptation to changing conditions is crucial.

Bernard Widrow and Marcian E. Hoff, "Adaptive switching circuits," 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104

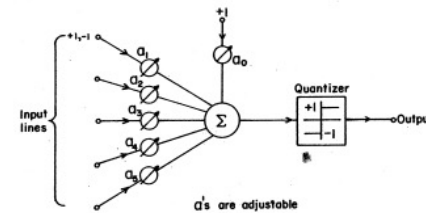


Fig. 1. An adjustable neuron.

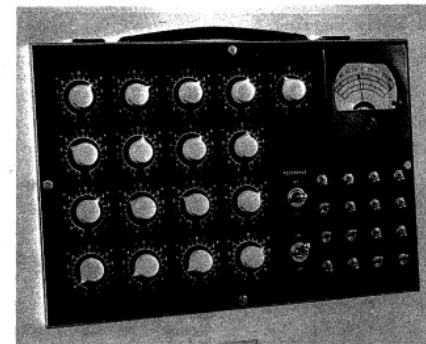


Fig. 2. Adaline.

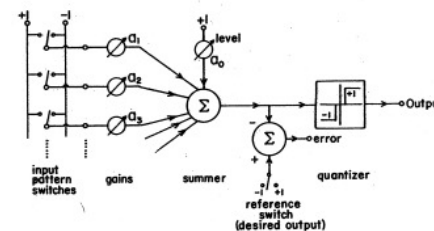


Fig. 3. Schematic of Adaline.

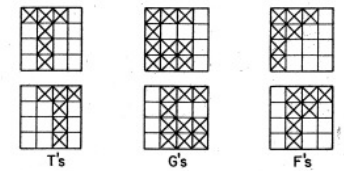


Fig. 4. Patterns for classification experiment.

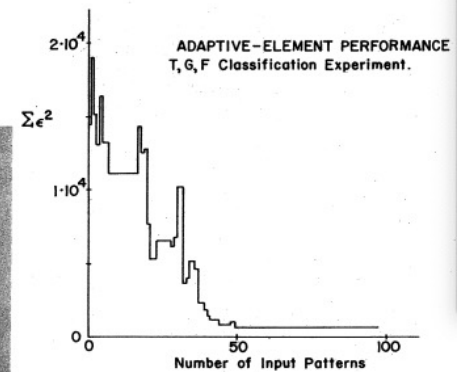


Fig. 5. Adaptive-element performance curve.

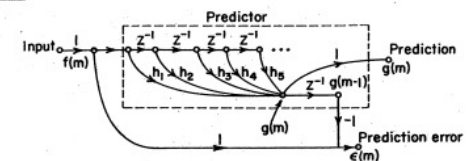


Fig. 6. An adjustable sampled-data predictor.

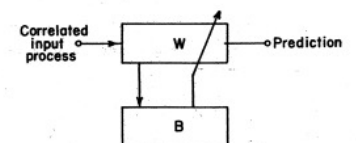
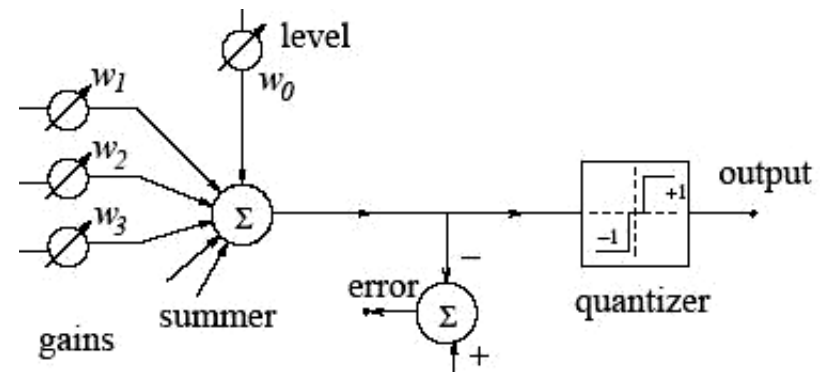


Fig. 7. An adaptive predictor.

Widrow-Hoff learning rule, 1962

The Widrow-Hoff (or **Delta**) learning rule, also known as the Least Mean Squares (LMS) algorithm, is a fundamental method for training linear models and minimizing errors between predicted and actual outputs.

This algorithm has laid the groundwork for more complex learning techniques like stochastic gradient descent, and it is widely used in applications such as adaptive filtering, noise cancellation, and predictive modeling across various fields.



Error=Target Output-Predicted Output

$$\Delta w = \eta \times \text{Error} \times x$$

where:

- Δw is the change in weight,
- η (eta) is the learning rate,
- **Error** is the difference between target and predicted output, and
- x is the input value.

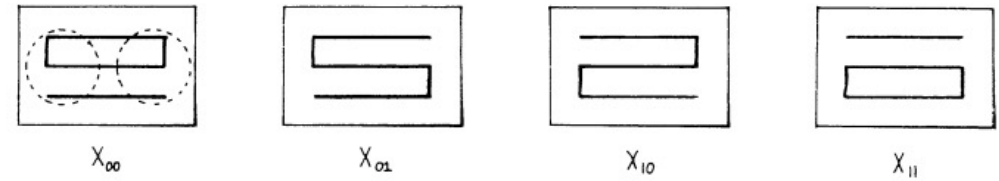
Widrow, B., & Hoff, M. E. (1962). Associative storage and retrieval of digital information in networks of adaptive “neurons”. In *Biological Prototypes and Synthetic Systems: Volume 1 Proceedings of the Second Annual Bionics Symposium sponsored by Cornell University and the General Electric Company, Advanced Electronics Center, held at Cornell University, August 30–September 1, 1961* (pp. 160-160). Springer US.

Limitations of Perceptrons, 1969

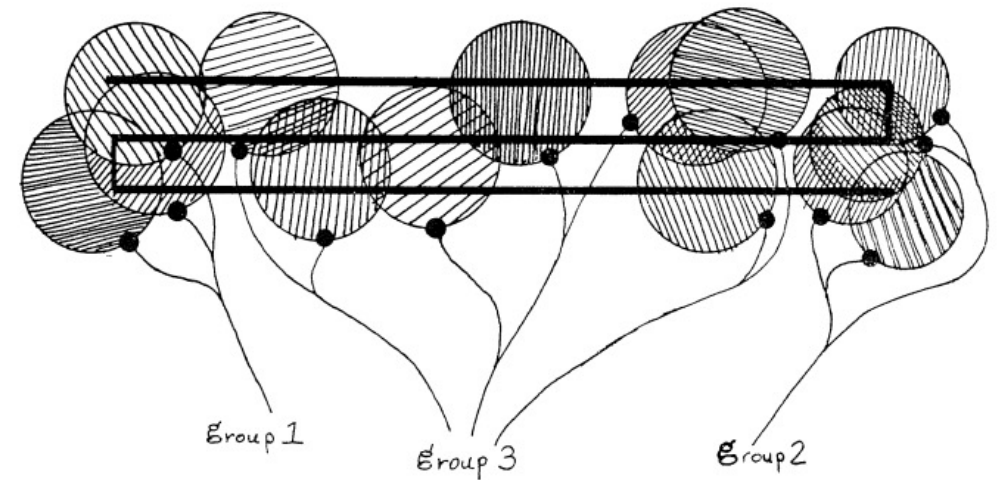
The 1969 book, *Perceptrons* by Marvin Minsky and Seymour Papert demonstrated that Rosenblatt's perceptrons could not solve problems that were not linearly separable, such as the XOR problem, without additional network layers.

At the time, multilayer perceptrons (now known as multi-layer neural networks) and effective training algorithms like backpropagation had not yet been developed or widely understood, making these limitations appear insurmountable.

The proof requires us to consider just four figures

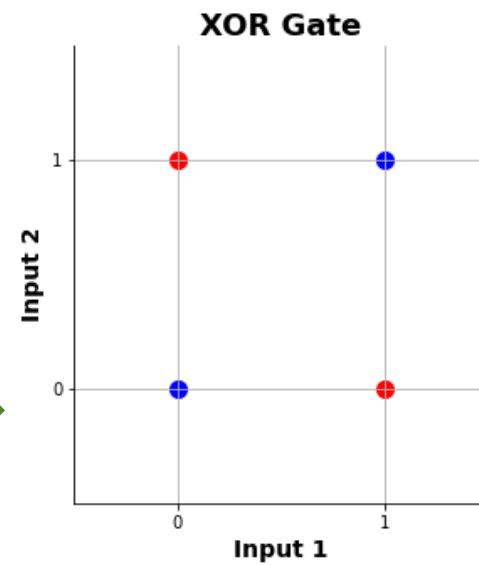
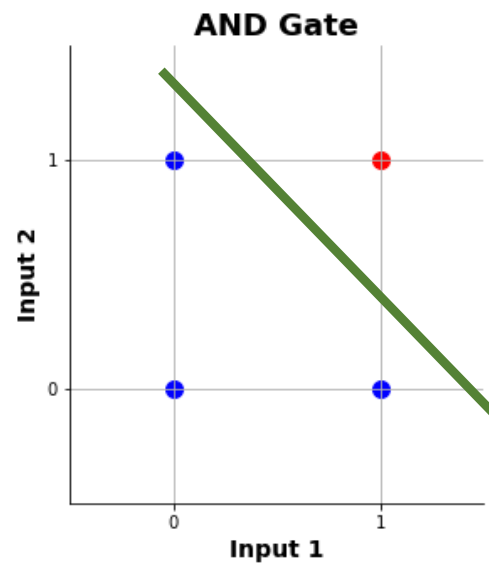
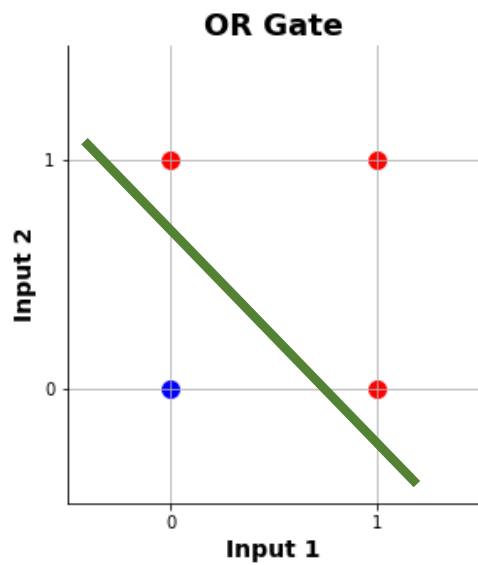
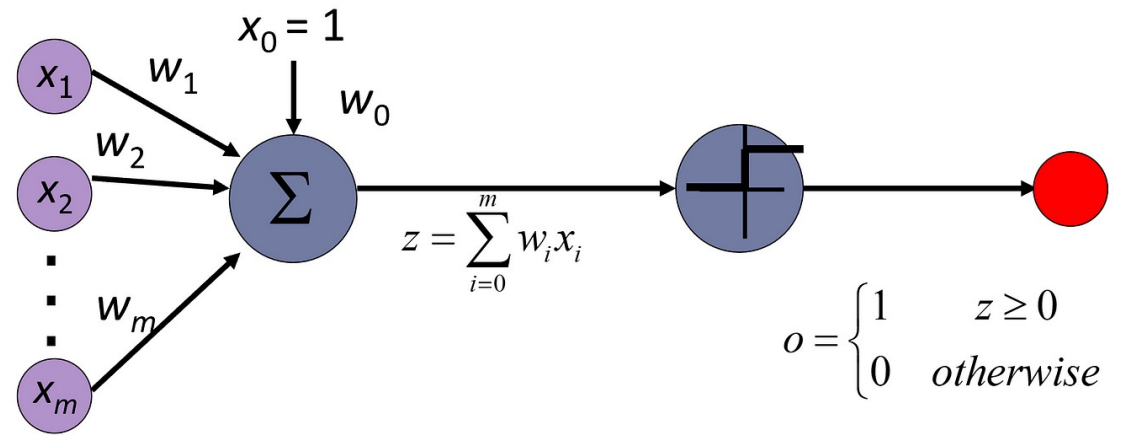


and a diameter-limited perceptron ψ whose support sets have diameters like those indicated by the circles below:



Minsky, M., & Papert, S. (1969). An introduction to computational geometry. *Cambridge tracts in mathematics*, 479(480), 104.

Rosenblatt's Perceptron



First AI Winter: Mid-1970s to mid-1980s

The first AI Winter lasted from the mid-1970s to mid-1980s when interest in AI research diminished significantly, along with a corresponding reduction in funding and perceived optimism about the field's potential. This downturn was primarily due to unmet expectations and the realization that the ambitious goals set in the early days of AI were far more complex and difficult to achieve than initially anticipated.

1. **Limited Computational Power:** The hardware available at the time was not powerful enough to support the complex computations needed for AI, limiting the development and testing of AI systems.
2. **Research Limitations:** Early AI research was heavily reliant on symbolic AI and heuristic approaches, which struggled with real-world complexities and lacked the ability to learn from data as effectively as modern machine learning techniques.
3. **Evaluation Challenges:** AI systems at the time were not robust and struggled outside controlled environments, leading to disillusionment with their capabilities.
4. **Funding Cuts and Criticism:** Influential reports, such as the *Lighthill Report*, criticized the progress and promise of AI research, leading to significant reductions in funding from governments and private institutions. *“In no part of the field have the discoveries made so far produced the major impact that was then promised.”*

The first AI Winter taught researchers the importance of **setting realistic expectations, understanding the technical limitations**, and ensuring sustained funding and support for **gradual but steady progress** in the field. **Or did it?**

Lighthill, J. (1973, April). Artificial intelligence: A general survey. In *Artificial Intelligence: a paper symposium*. London: Science Research Council.

Backpropagation, 1986 (but also 1970 and 1974)

The backpropagation algorithm was described several times before they thought to apply it to multilayer perceptrons.

- Seppo Linnainmaa (1970), *Algoritmin kumulatiivinen pyöristysvirhe yksittäisten pyöristysvirheiden Taylor-kehitemänä* [The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors], Doctoral dissertation, Master's Thesis, University of Helsinki
- Paul Werbos (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088), 533-536.

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams*

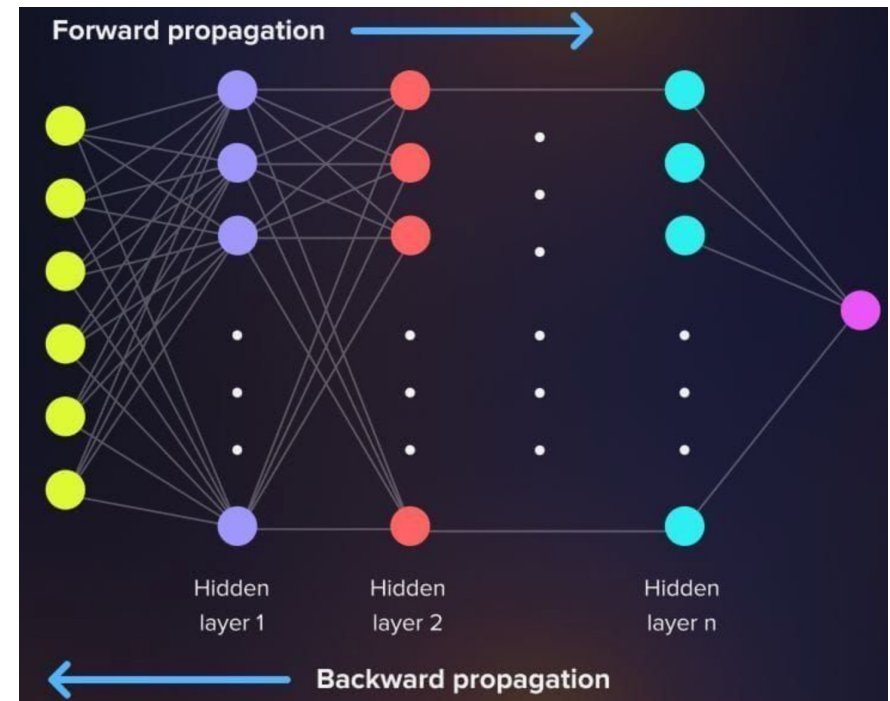
* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA

† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure¹.

Backpropagation, 1986 (but also 1970 and 1974)

- **Chain Rule in Action:** Backpropagation uses the chain rule from calculus to pinpoint how much each weight in the network contributes to the overall error.
- **Gradient Descent:** Calculates the gradient of the error with respect to each weight, providing the direction to adjust the weights to reduce error. This gradient guides the optimization algorithm.
- **Backward Propagation:** The process works by propagating the error backwards through the network, layer by layer, starting from the output.
- **Weight Update:** Using the calculated gradients, the weights are adjusted in the direction that minimizes the error. The learning rate controls the size of the adjustment.
- **Iterative Refinement:** Backpropagation is iterative, repeating the forward pass, backward pass, and weight updates until the network reaches the desired performance.



Backpropagation, 1986 (but also 1970 and 1974)

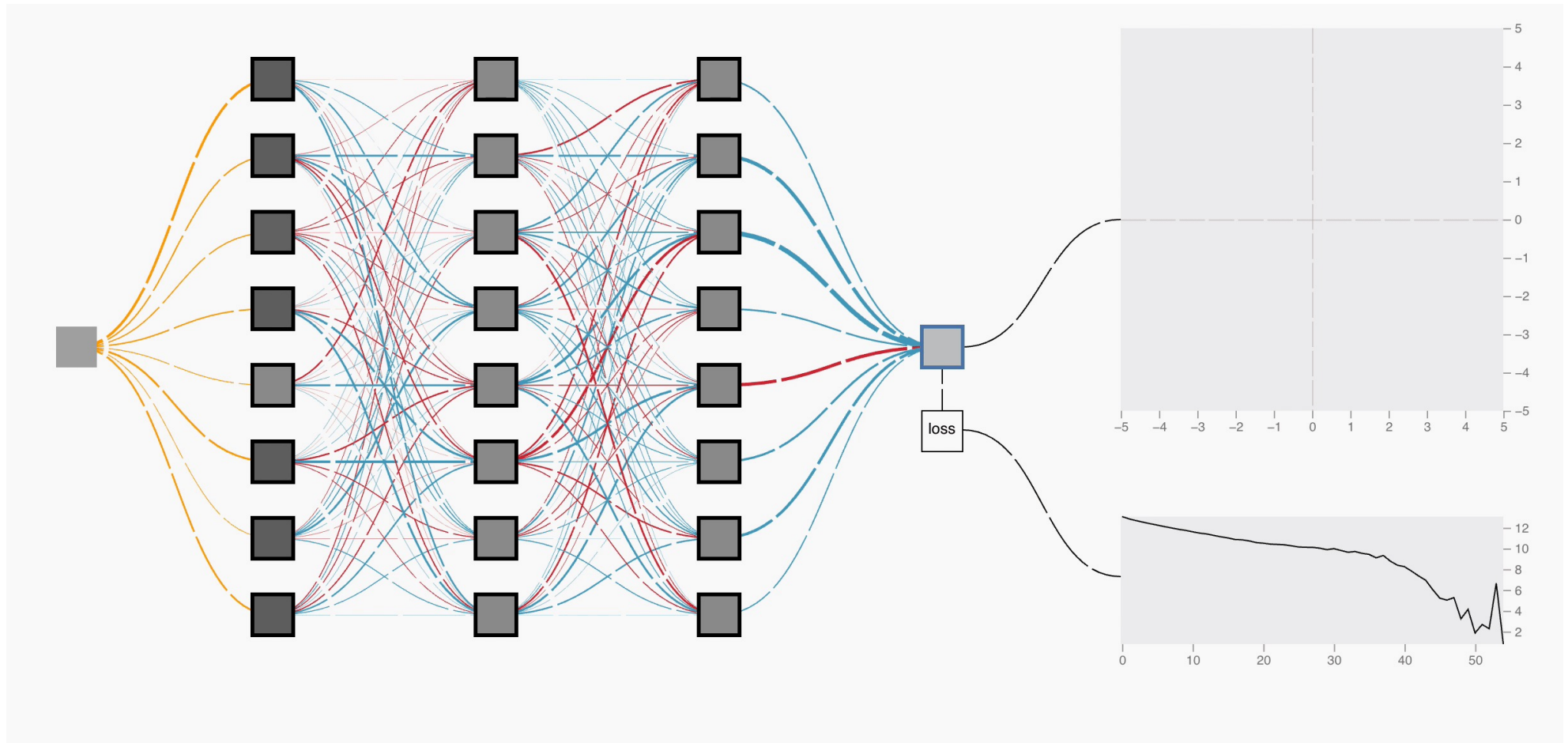
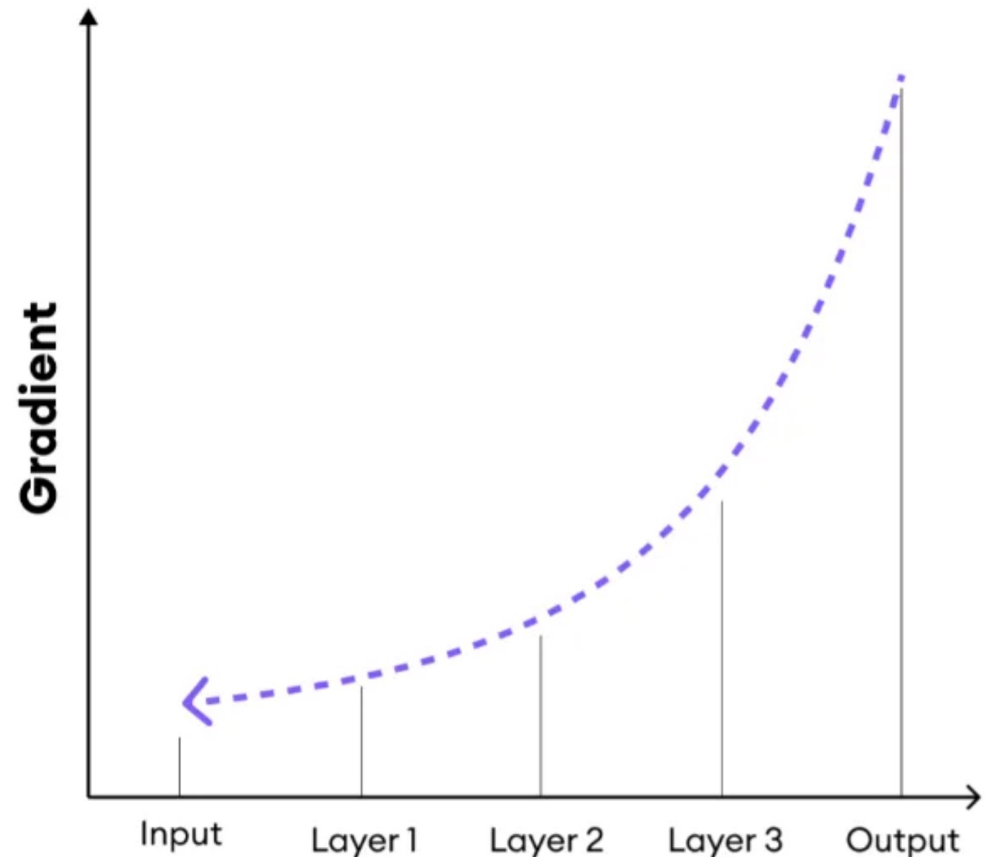


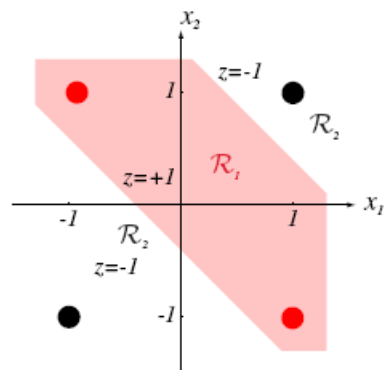
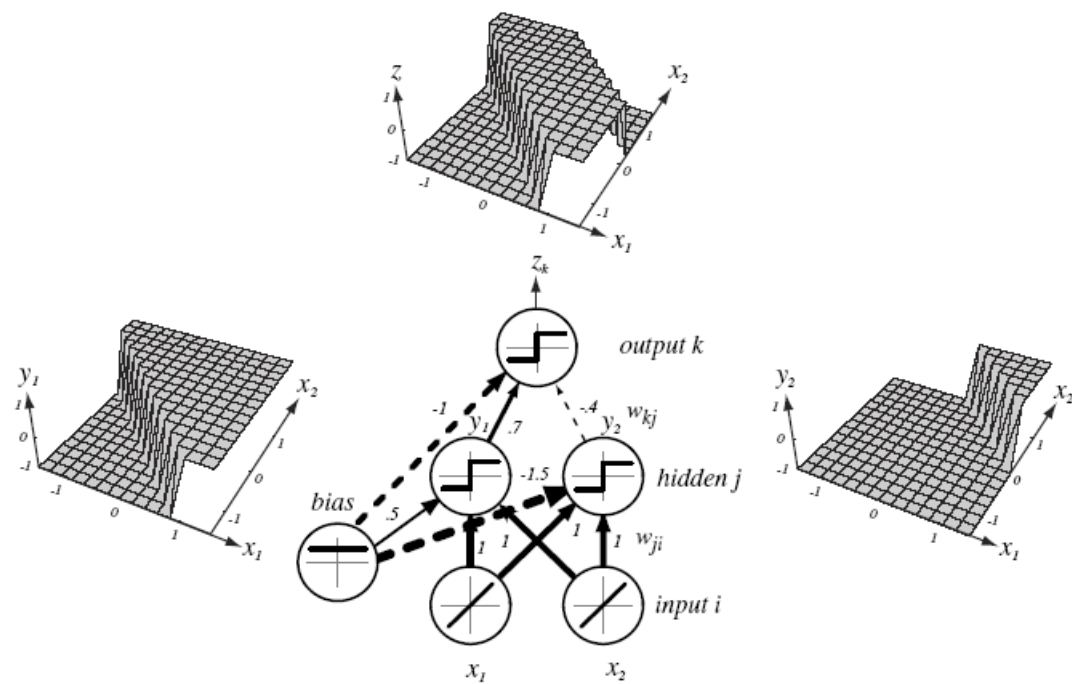
Image source: <https://xnought.github.io/backprop-explainer/>

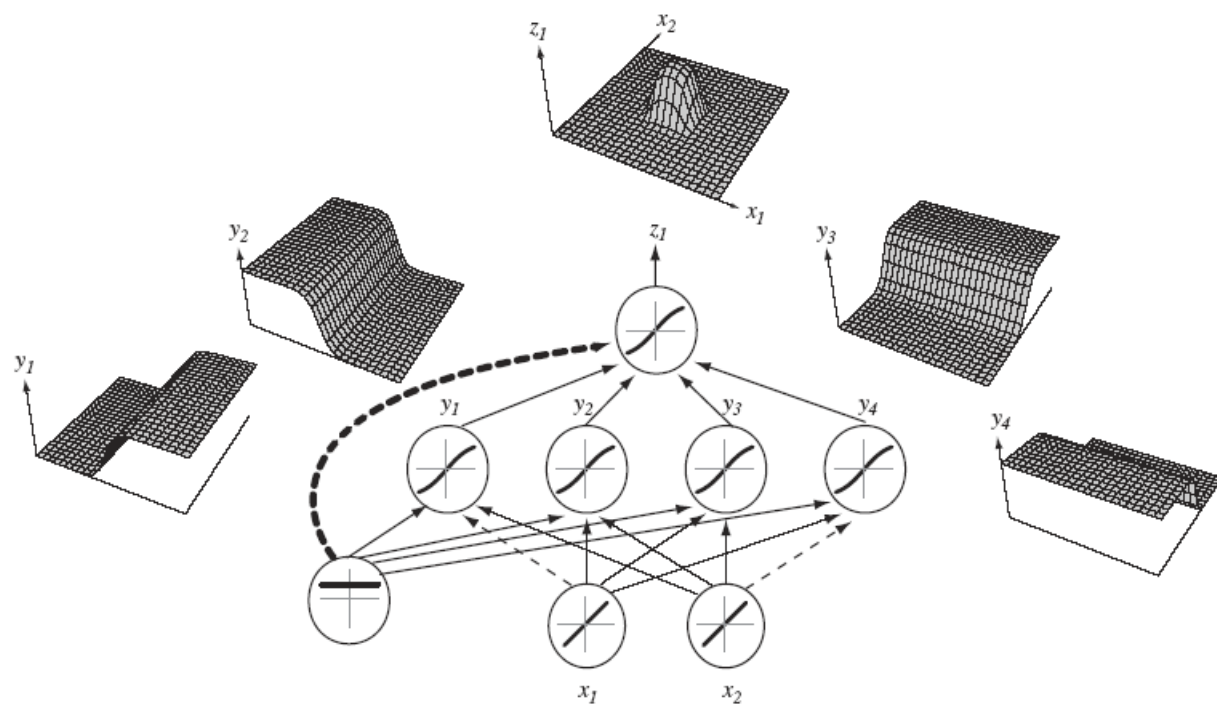
Backprop Issues

Vanilla backpropagation, while foundational, have limitations that can hinder its effectiveness in practice.

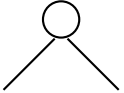
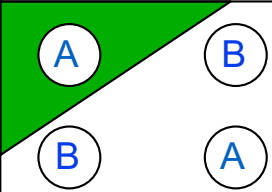
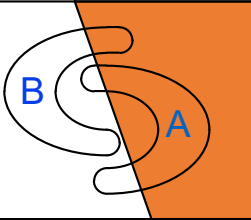
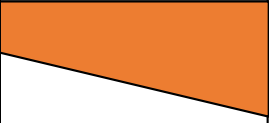
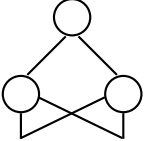
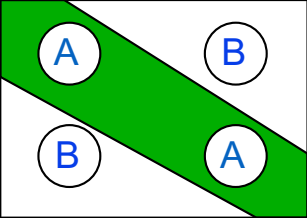
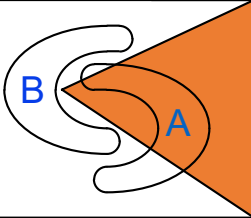
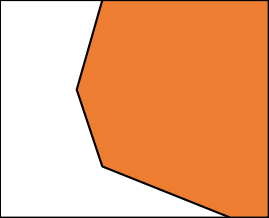
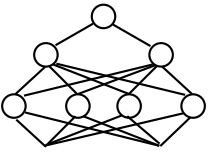
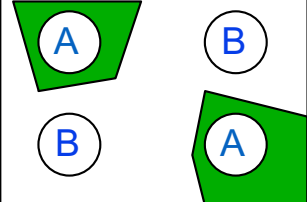
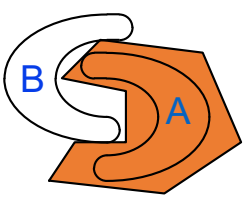
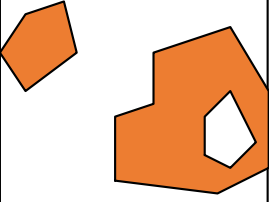
- **Vanishing Gradients:** In deep networks, gradients can become very small as they are propagated backward through layers with activation functions like the sigmoid or tanh. This results in very slow learning for the layers early in the network, as the updates to weights become negligible.
- **Exploding Gradients:** This is the opposite of the vanishing gradient problem, where gradients can become excessively large. This often leads to numerical instability and causes the model's parameters to overflow, making learning unstable.
- **Gradient Descent Inefficiencies:** Vanilla gradient descent may converge very slowly, especially in non-convex optimization landscapes common in deep learning, due to inefficient or inappropriate learning rate choices.
- **Lack of Regularization:** Vanilla backpropagation does not inherently include mechanisms for regularization, making it prone to overfitting if additional techniques such as dropout, weight decay, or batch normalization are not used.
- **Computational Efficiency:** Without optimizations or modifications, vanilla gradient descent can be computationally expensive due to its full pass over all data points in each iteration, especially with large datasets.
- **Parameter Sensitivity:** The algorithm's sensitivity to hyperparameters, such as the learning rate, can significantly affect the model's ability to learn relevant patterns efficiently.







Different Non-Linearly Separable Problems

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	<i>Half Plane Bounded By Hyperplane</i>			
Two-Layer 	<i>Convex Open Or Closed Regions</i>			
Three-Layer 	<i>Arbitrary (Complexity Limited by No. of Nodes)</i>			

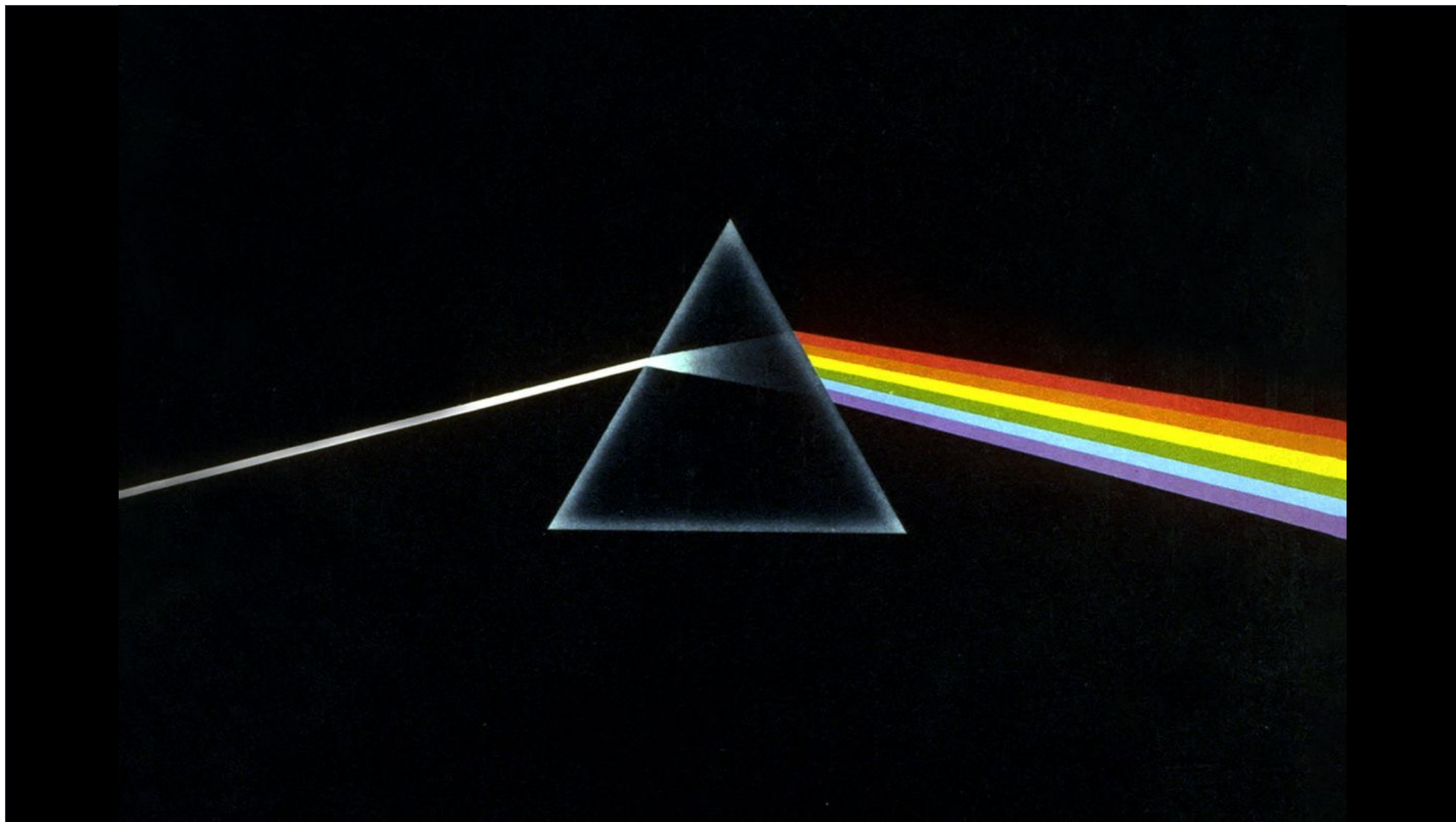
Second AI Winter: ~1987 to ~1993

- **Overhyped Expert Systems:** These AI systems, while initially promising, couldn't live up to expectations, proving brittle, hard to maintain, and limited in scope.
 - **Brittle:** Unable to adapt to changing conditions or handle situations outside their narrow domain.
 - **Difficult to Maintain:** As expert systems grew more complex, updating and expanding their rule-based systems became increasingly cumbersome.
 - **Limited in Scope:** They struggled with tasks requiring common sense reasoning or complex problem-solving.
- **Rise of Personal Computers:** Affordable PCs offered a more versatile and cost-effective alternative to specialized AI hardware, undercutting the market.
- **Funding Cuts:** Disillusionment with AI's progress led to government and private investors reducing funding for research.
- **Lack of Real-World Use:** AI struggled to demonstrate practical applications and tangible benefits, further discouraging investment.

The second AI Winter taught researchers the importance of setting realistic expectations, understanding the technical limitations, and ensuring sustained funding and support for gradual but steady progress in the field. **Or did it?**

End of **Second AI Winter** (early 90s)

- **Increased Computing Power:** Moore's Law continued to deliver exponential growth in processing power, enabling more complex AI algorithms and larger datasets.
- **Focus on Specific Problems:** Researchers shifted from grand AI goals to tackling smaller, more manageable problems with clear applications, leading to tangible successes.
- **New Ties with Other Fields:** AI began integrating with disciplines like statistics, economics, and mathematics, leading to new approaches and hybrid solutions.
- **Emphasis on Mathematical Rigor:** A renewed focus on sound mathematical principles and scientific standards improved the credibility and reliability of AI research.
- **Emergence of Machine Learning:** Advances in machine learning, particularly with neural networks, started to yield promising results in areas like speech and image recognition.
 - **Shift in Focus:** Instead of trying to create general-purpose AI, researchers started focusing on specific, well-defined problems that machine learning could address. This led to more manageable goals and demonstrable progress.
 - **New Algorithms and Techniques:** Key advancements in machine learning algorithms, like backpropagation for training neural networks, unlocked new possibilities for tackling complex tasks like pattern recognition and prediction.
 - **Data-Driven Approach:** Machine learning emphasized learning from data rather than relying solely on hand-coded rules. This allowed AI systems to adapt and improve their performance as they were exposed to more data.
 - **Early Successes:** Machine learning started achieving notable success in areas like handwriting recognition, spam filtering, and fraud detection



Why is time interesting in an AI context?

Time series data are common and characterizing them with neural networks offers powerful capabilities.

Prediction & Forecasting:

- Financial Markets: Predicting stock prices, currency exchange rates, and market volatility.
- Weather Forecasting: Forecasting temperature, rainfall, and extreme weather events.
- Demand Forecasting: Predicting sales, customer traffic, and inventory needs for businesses.

Anomaly Detection:

- Fraud Detection: Identifying unusual patterns in financial transactions or user behavior.
- Manufacturing: Identifying defects in production lines.
- Cybersecurity: Detecting intrusions and malicious activity in network traffic.

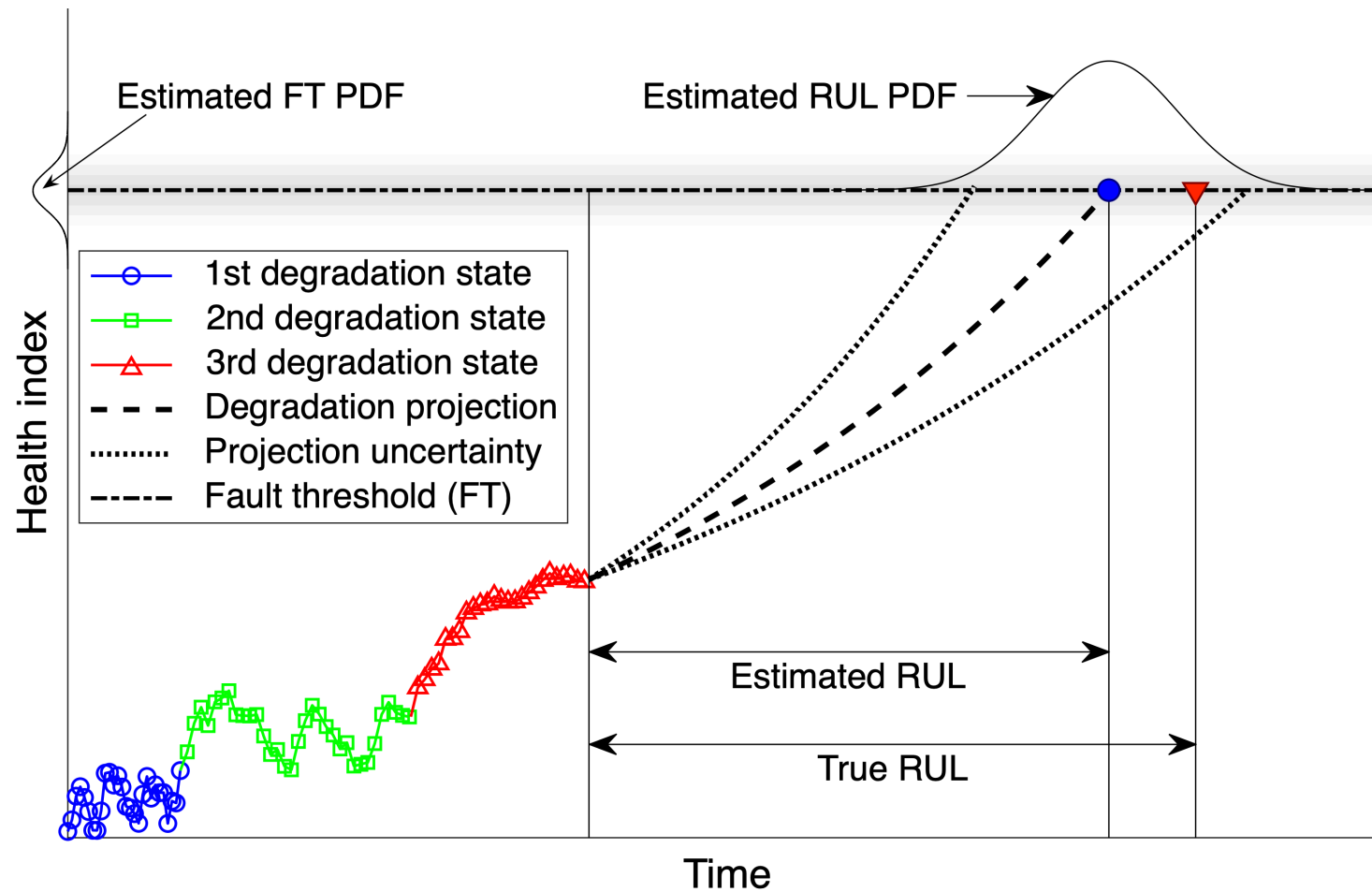
Classification & Pattern Recognition:

- Speech Recognition: Converting spoken language to text.
- [Natural Language Processing: Understanding the meaning and sentiment of text.](#)
- Gesture Recognition: Interpreting human gestures in video or sensor data.

Control & Optimization:

- Robotics: Controlling the movements and actions of robots in real-time.
- Process Control: Optimizing industrial processes to improve efficiency and quality.
- Autonomous Vehicles: Navigating and controlling self-driving cars.

PHM is all about time...



Camargos, M. O., Bessa, I., Junior, L. A., Coutinho, P. H. S., Leite, D. F., & Palhares, R. M. (2021). Evolving fuzzy system applied to battery charge capacity prediction for fault prognostics. *arXiv preprint arXiv:2102.09521*.

Recurrent Neural Networks (RNNs)

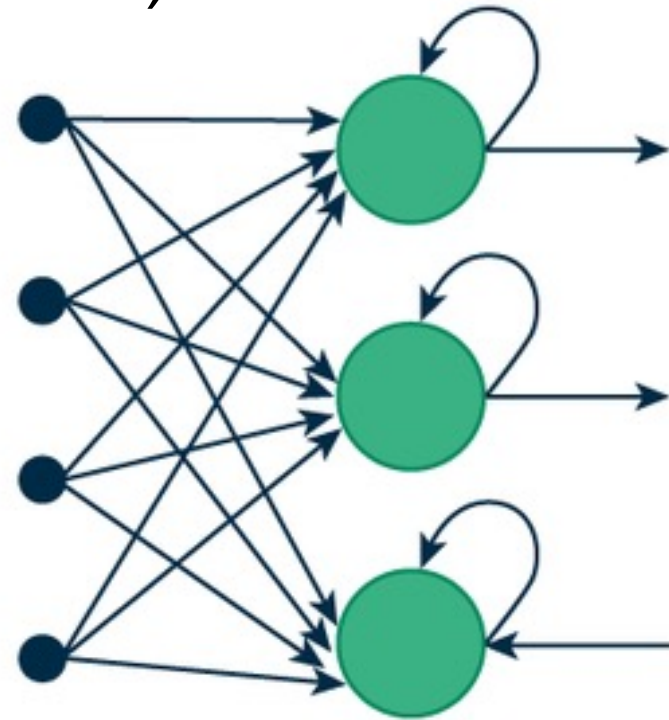
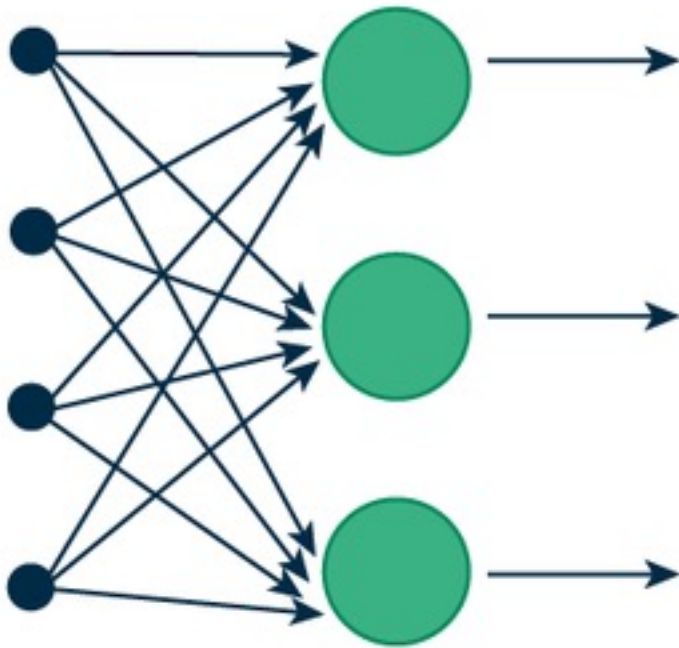
RNNs are a type of artificial neural network designed for sequential data.

- They have connections that form directed cycles, creating an internal state that acts as memory.
- Unlike traditional neural networks, RNNs can utilize past information to influence future predictions.
- Commonly used for tasks like time-series prediction, natural language processing, and speech recognition.
- Backpropagation Through Time (BPTT): A training algorithm adapted to handle sequence data, crucial for adjusting weights in RNNs.
- **Vanishing Gradient Problem:** A challenge where gradients diminish, making learning long-range dependencies difficult.

Applications:

- Language modeling and text generation
- Sentiment analysis
- Machine translation
- Predictive analytics in financial markets
- Audio processing and music composition.

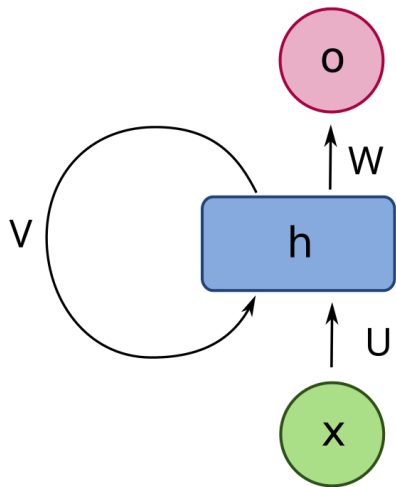
Recurrent Neural Networks (RNNs)



RNNs are a class of neural networks designed to process sequential data by maintaining a form of memory through recurrent connections, allowing them to effectively handle tasks where context and sequence order are crucial, such as language modeling and time-series prediction.

A one-unit RNN

From bottom to top: input state, hidden state, output state.
 U, V, W are the weights of the network.



RNNs & the Vanishing Gradient Problem

- Gradients shrink exponentially during BPTT, hindering learning of long-term dependencies.
- Occurs due to repeated multiplication of small values in the chain rule.
- RNNs struggle to retain information from earlier time steps.
- Affects tasks requiring long-term context (e.g., natural language understanding, speech recognition).



Long Short Term Memory (LSTMs, 1997)

LSTMs are explicitly designed capture long-range dependencies more effectively than traditional RNNs.

- **Cells:** LSTM networks consist of units called cells, which include three gates: input, forget, and output gates.
 - **Input Gate:** Determines the extent to which new information is added to the cell state.
 - **Forget Gate:** Decides what information is discarded from the cell state.
 - **Output Gate:** Controls the output and how much of the cell state affects the output.
- **Cell state** is a key component that flows through the entire network, allowing information to be retained over long periods.
- The **gating mechanisms** rely on a sigmoid activation function to decide which information to keep, update, or forget, achieving a balance between memory retention and contextual updating.

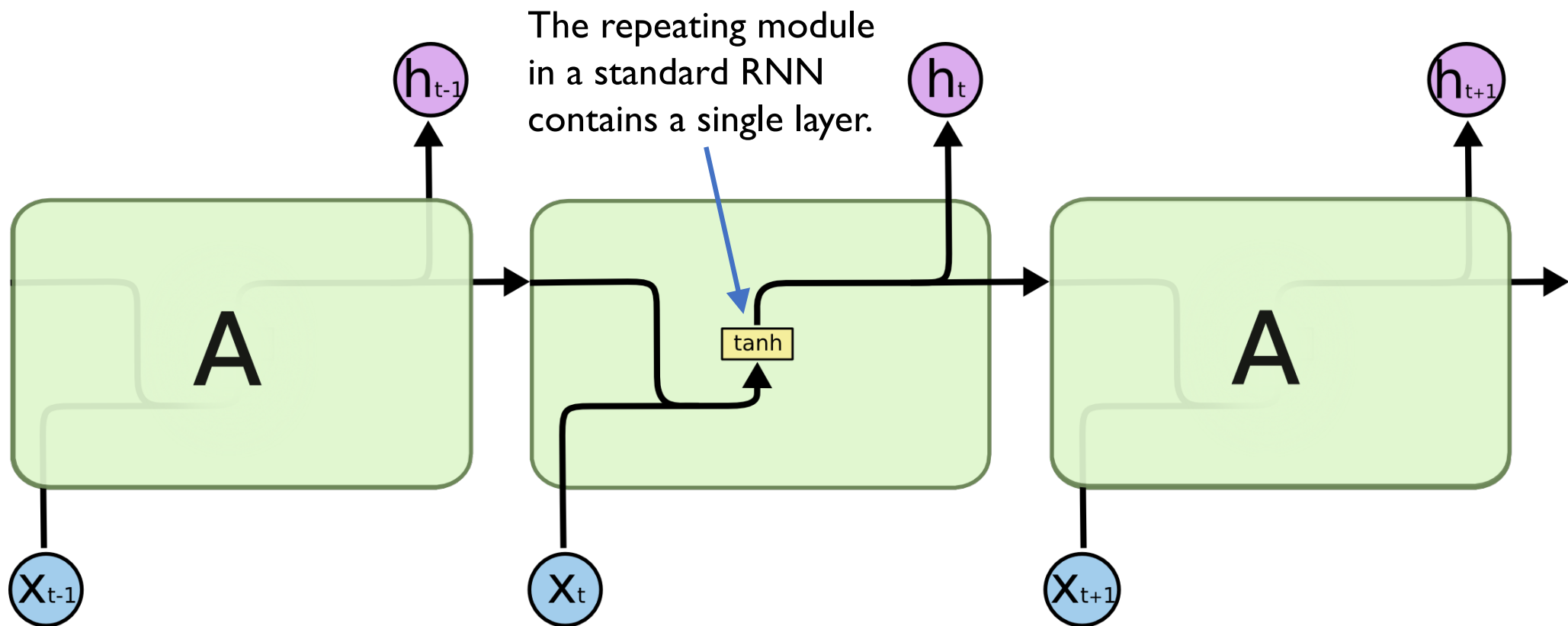
Advantages:

- **Capable of learning long-term dependencies** and retaining information across extensive time steps.
- Useful in tasks where context over time is important, like language modeling, machine translation, speech recognition, and time-series prediction.

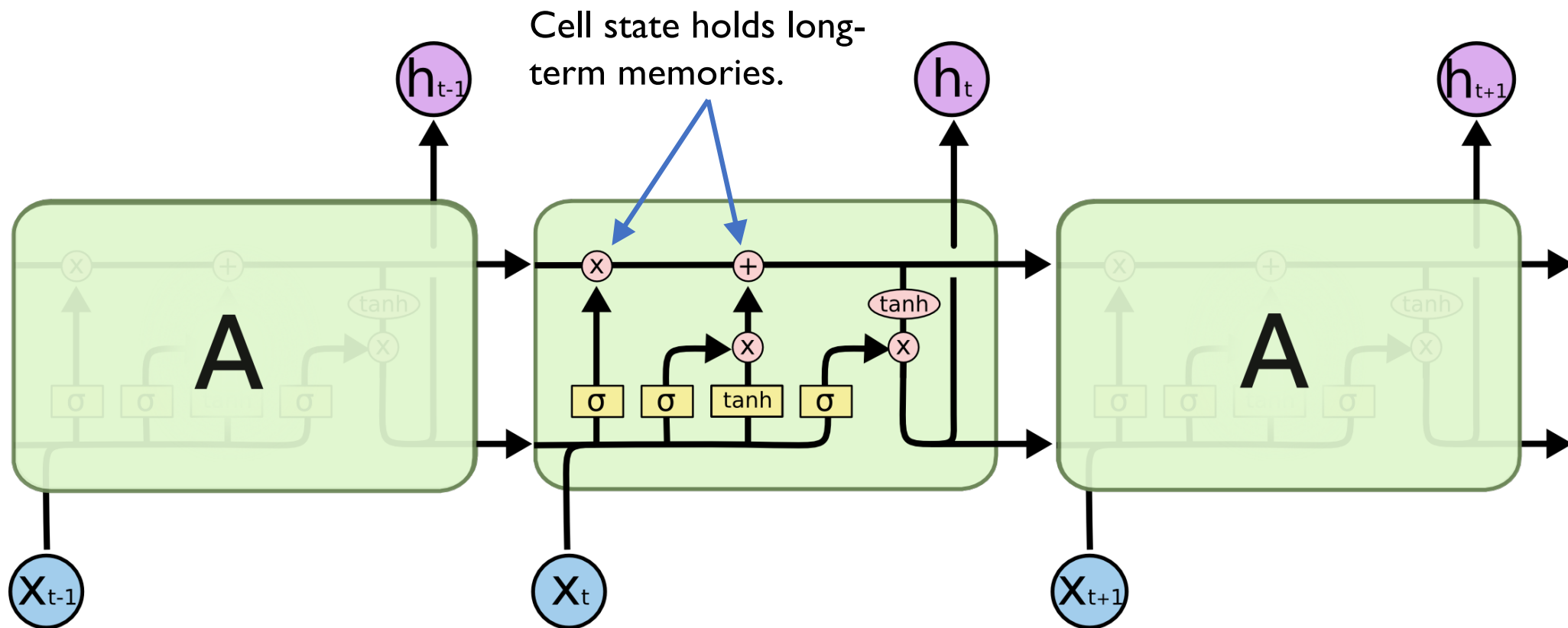
Challenges:

- Computationally intensive and require more resources than simple RNNs.
- **Tuning hyperparameters like learning rate, dropout rate, and number of hidden layers can be complex.**

Recall: the RNN

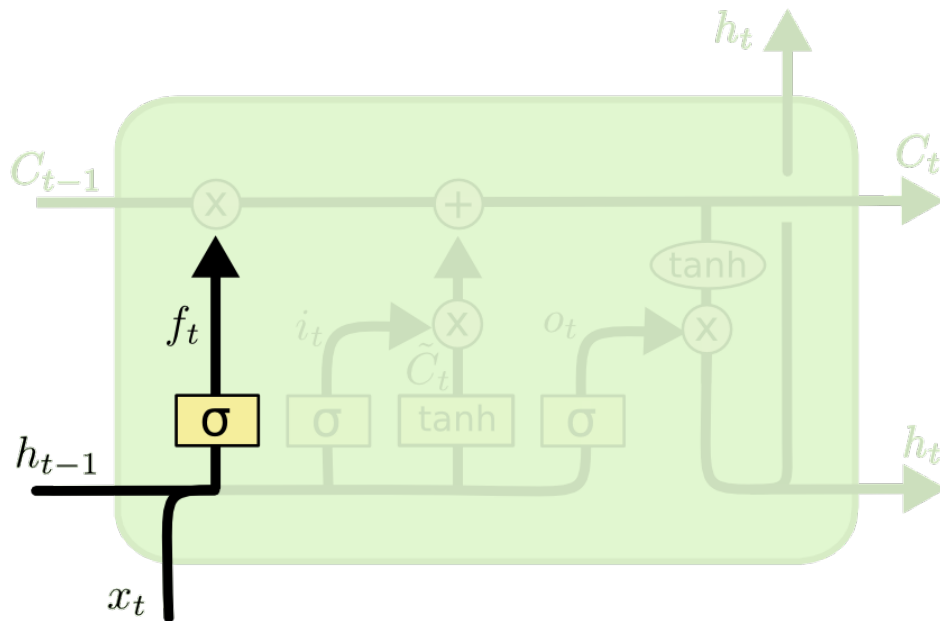


LSTM repeating module



The repeating module in an LSTM contains *four* interacting layers.

Forget Gate



The forget gate determines which information from the previous cell state should be removed or forgotten as the network processes new input.

Two inputs:

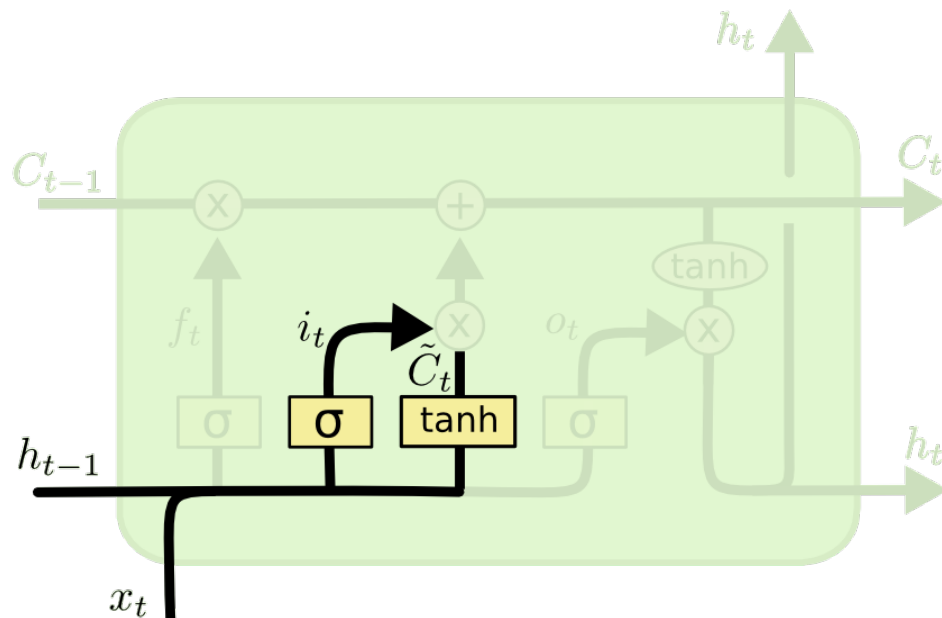
- The previous hidden state h_{t-1} from the previous time step.
- The current input x_t at the current time step.

The output of the sigmoid function is a vector where each element is a value between 0 and 1. These values represent the degree to which each part of the cell state will be forgotten:

- A value close to 0 means “forget this information entirely.”
- A value close to 1 means “keep this information intact.”

The forget gate allows LSTMs to selectively clear unneeded information while retaining important context, addressing the issue of "forgetting" in traditional RNNs over long sequences. This mechanism is key to LSTMs' ability to model long-range dependencies in sequential data.

Input Gate



The input gate controls the extent to which new information from the current input is allowed to alter the cell state.

Same two inputs as forget gate:

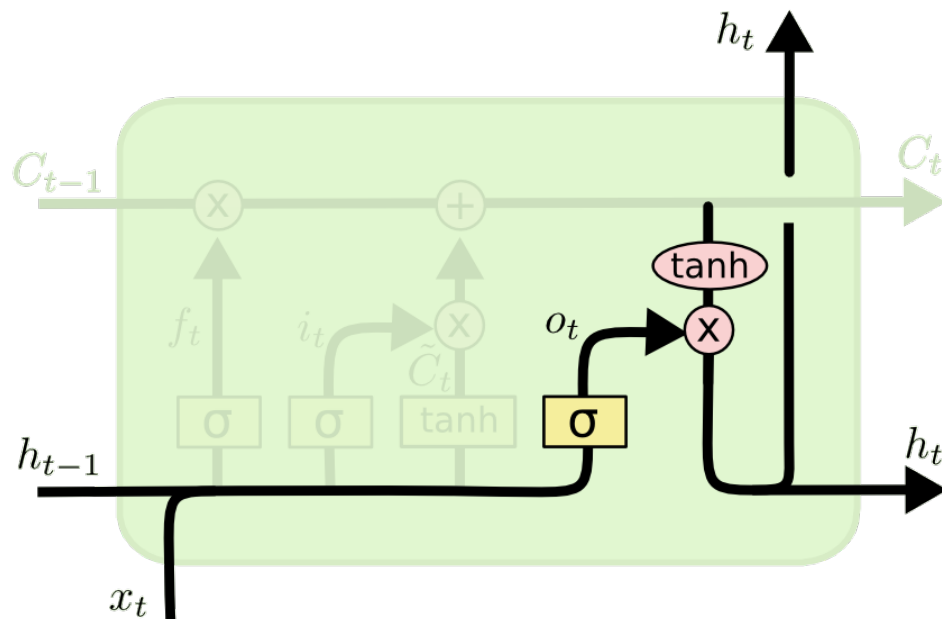
- The previous hidden state h_{t-1} from the previous time step.
- The current input x_t at the current time step.

The output of the sigmoid function is a vector where each element is a value between 0 and 1. It represents how much of each element from the candidate cell state should be accepted into the actual cell state:

- The element-wise multiplication of i_t with $\tilde{C}_t$ scales the new candidate values, determining their impact on the updated cell state.

The new cell state is formed by combining the scaled candidate values from the input gate with the existing modified cell state (controlled by the forget gate). The input gate allows the LSTM to incorporate relevant new information while protecting the cell state from irrelevant data or noise.

Output Gate



The output gate determines what information will be transferred out of the LSTM cell at each time step and into the wider network where further processing can occur—whether for prediction or as an input to subsequent layers or timesteps.

The same two inputs:

- The previous hidden state h_{t-1} from the previous time step.
- The current input x_t at the current time step.

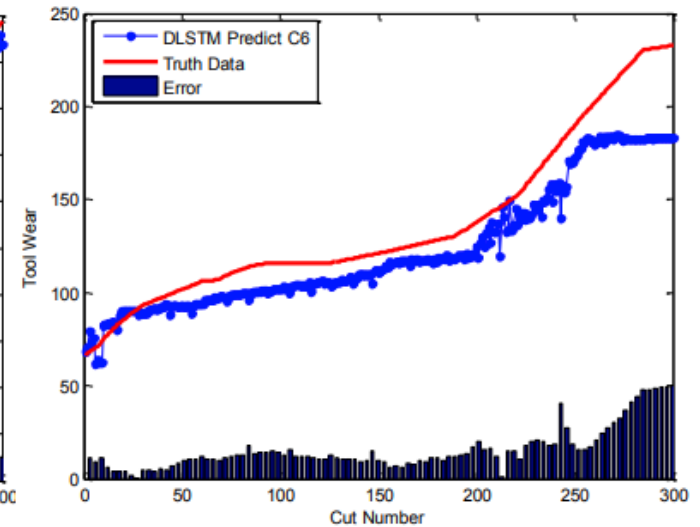
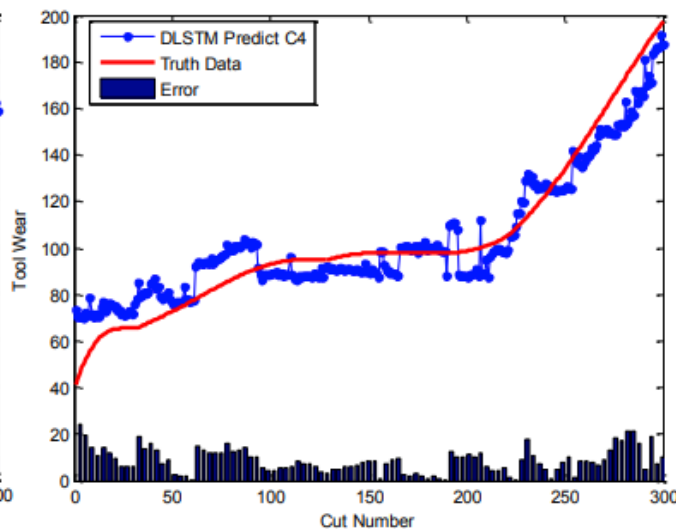
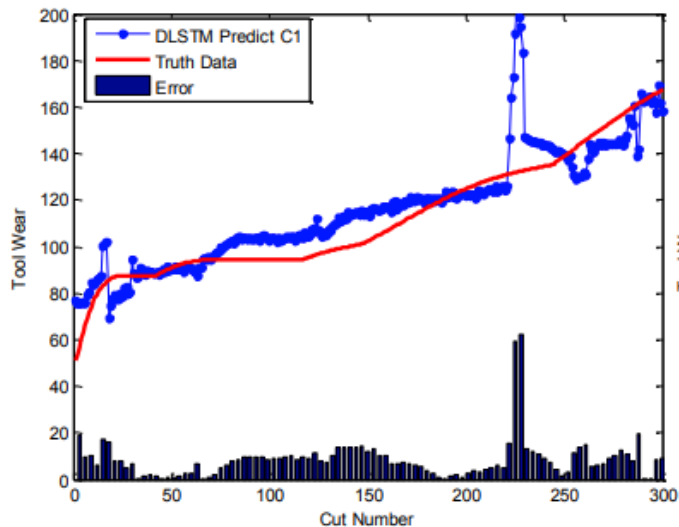
The element-wise multiplication of the output gate's activation o_t with the transformed cell state creates the current hidden state h_t , which also serves as the output from the LSTM cell.

This gate effectively controls the balance between maintaining memory and producing relevant short-term output.

LSTM in PHM: Rotating Machinery

Zhao, R., Wang, J., Yan, R., & Mao, K. (2016, November). Machine health monitoring with LSTM networks. In *2016 10th International Conference on Sensing Technology (ICST)* (pp. 1-6). IEEE.

Real CNC data



The background is a deep blue with a complex, abstract pattern. It features a grid of thin, light blue lines that intersect to form a mesh. Overlaid on this grid are several translucent, wavy, vertical bands that create a sense of depth and movement, resembling liquid or smoke. The overall effect is a modern, digital aesthetic.

Some Background to the Background

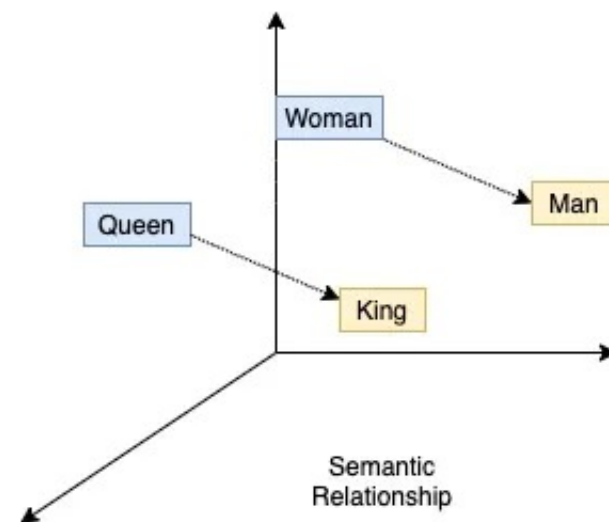
Word Embeddings, e.g., Word2Vec

A neural network-based technique for learning word embeddings, representing words as dense vectors.

- Converts words into continuous vector spaces based on their context
 - Skip-gram Model: Predicts surrounding words given a target word
 - CBOW (Continuous Bag of Words): Predicts a target word from surrounding context

Key Features

- Captures semantic relationships between words
- Word vectors can perform arithmetic operations (e.g., $\text{king} - \text{man} + \text{woman} \approx \text{queen}$)
- Highly efficient and scalable
- Revolutionized how text data is processed in machine learning
- Paved the way for further advancements in word embedding techniques, such as GloVe and BERT



Autoencoders

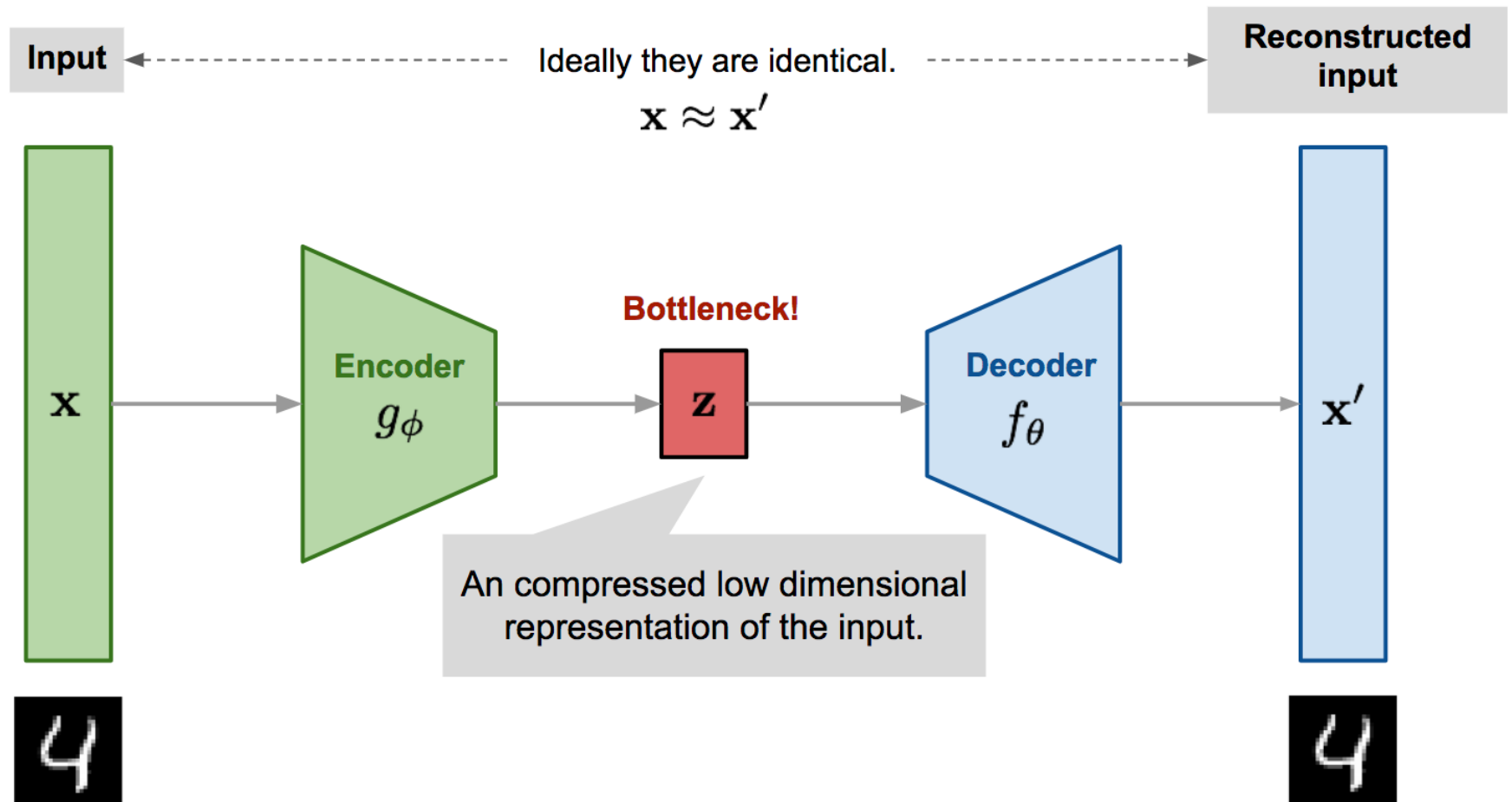
Autoencoders are neural networks trained to reconstruct their input data. Imagine them as a bottleneck for information, forcing the network to learn the most important features of the data.

- Encoder: Compresses the input data into a lower-dimensional representation called the "latent space" or "bottleneck."
- Decoder: Reconstructs the original input data from the latent space representation.

Applications:

- Dimensionality Reduction: Similar to PCA, but with non-linear capabilities.
- Feature Extraction: Learn relevant features from the data.
- Anomaly Detection: Deviations in reconstruction error can highlight unusual data points.
- Denoising Data: Trained to reconstruct clean input from noisy versions.
- Generative Modeling: Can generate new data similar to the training data by manipulating the latent space.

Autoencoder



Encoder-Decoders

Encoder-Decoder is a more general framework where the goal isn't necessarily to reconstruct the input, but to transform it into a different output. It still has two parts:

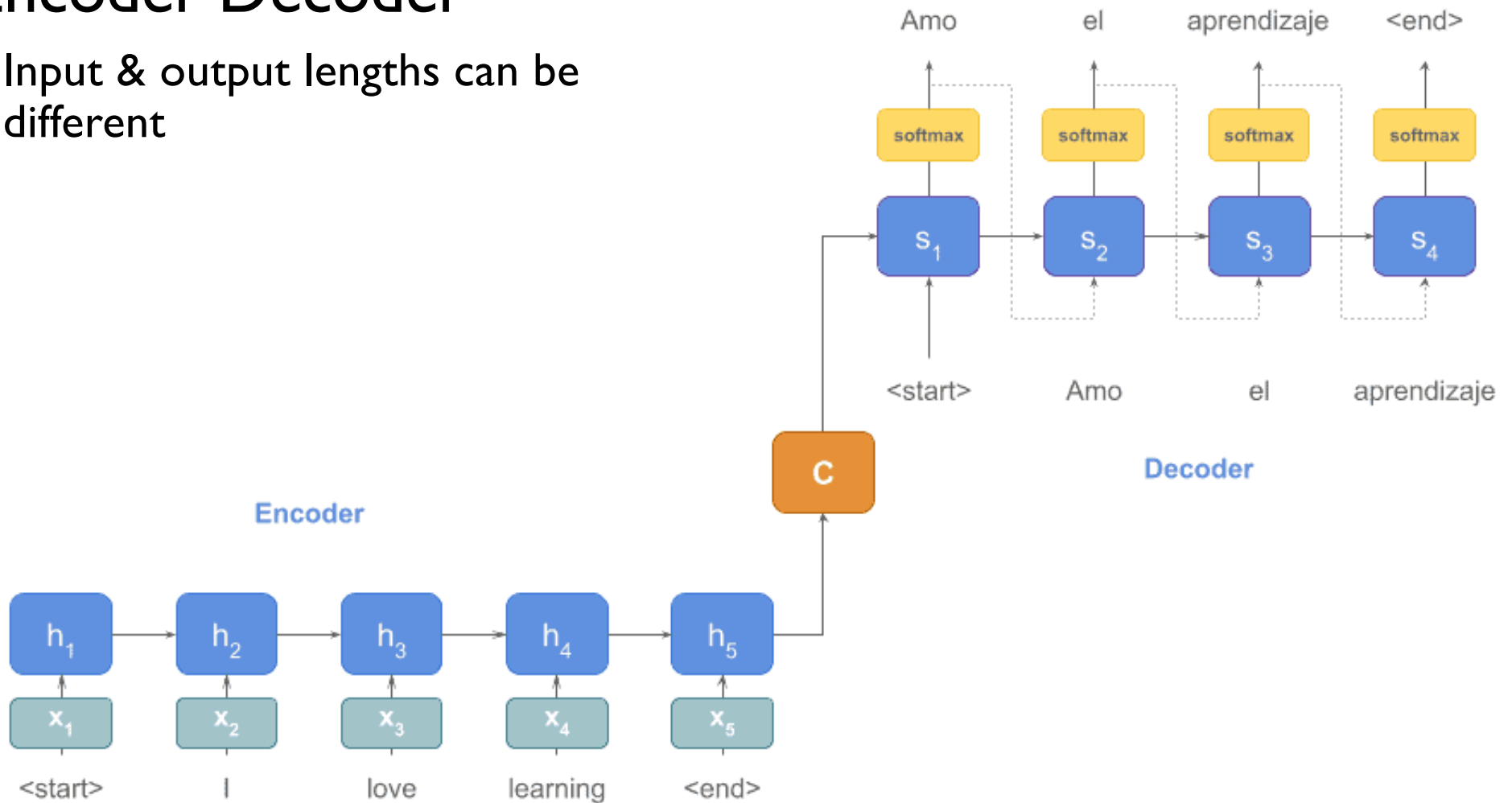
- **Encoder**: Similar to autoencoders, it processes the input and creates a compressed representation.
- **Decoder**: This part takes the encoded representation and generates a new output, which **could be in a different format or modality than the input**.

Encoder-Decoders learn a mapping from one domain to another.

- e.g., machine translation, where the encoder processes a sentence in one language and the decoder generates the translation in another language.

Encoder-Decoder

- Input & output lengths can be different



Bahdanau Attention: Helping Computers Focus (2016)

In machine translation, neural networks used an approach where the entire input sequence was compressed into a single, fixed-length vector. This was a bottleneck, especially for long sentences, as crucial information could get lost in the compression.

Bahdanau attention allows networks to dynamically focus on different parts of the input, leading to:

- Significantly improved accuracy of tasks like machine translation, especially for longer sentences.
- By examining the “attention weights,” we can see which parts of the input the network focused on, giving us insights into its decision-making process.

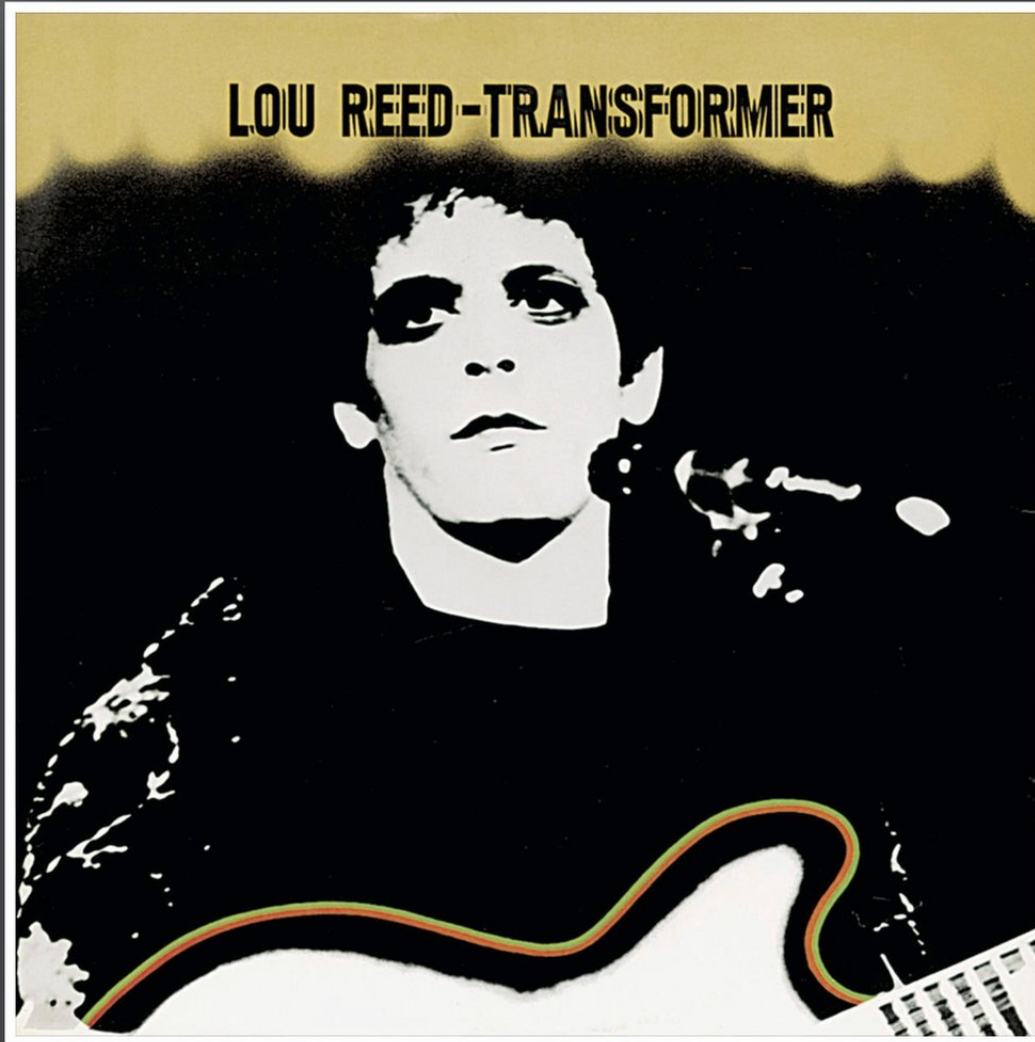
Imagine you're trying to translate a long sentence from English to French. Instead of trying to cram the entire English sentence into your brain at once, you might focus on a few words at a time, translate them, and then move on to the next few. That's essentially what Bahdanau attention does for computers!

Game changer:

- Overcame limitations of fixed-length vector encoding for long sequences.
- Boosted performance in machine translation and other tasks.
- Provided valuable insights into the model's decision-making process.

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural machine translation by jointly learning to align and translate*. arXiv. <https://doi.org/10.48550/arXiv.1409.0473>

LOU REED-TRANSFORMER



Transformers

A transformer is a type of neural network architecture that has revolutionized how computers process sequential data, like the words in a sentence or the notes in a musical piece.

- Transforms input sequences into output sequences: It takes a sequence of data (like words in a sentence) and produces another sequence (like a translation of that sentence from one language to another).
- Unlike previous models that processed data step-by-step, transformers can analyze an entire sequence at once, allowing them to understand the relationships between different parts of the data. **This is done through a mechanism called “attention.”**

Key components:

- The attention mechanism is the heart of a transformer. It allows the model to focus on different parts of the input sequence when processing a specific element. Imagine it like reading a sentence and paying attention to particular words that are most relevant to the word you're currently reading.
- Encoders and decoders: are the building blocks of a transformer. Encoders process the input sequence, while decoders generate the output sequence.
- Parallel processing: Transformers can process the entire input sequence in parallel, making them much faster than previous models.

Transformers have achieved state-of-the-art results in various tasks, including:

- Natural language processing: Machine translation, text summarization, question answering.
- Computer vision: Image recognition, object detection.
- Speech recognition: Converting spoken language to text.

Attention Is All You Need (2017) (1 of 2)

Attention Is All You Need was a **seminal paper that revolutionized NLP** by introducing a more efficient and powerful architecture for processing sequential data, paving the way for advancements in language understanding and generation.

- Introduced the **Transformer**, a new neural network architecture **relying solely on attention mechanisms**, discarding recurrent and convolutional layers.
- **Removed the inherent sequential nature of RNNs**, allowing for greater parallelization and faster training speeds.
- Enabled **better handling of long-range dependencies** in a sequence through self-attention.

Attention Is All You Need (2017) (2 of 2)

- Achieved superior performance in machine translation tasks compared to existing models, including RNN and CNN-based architectures.
- Allowed the model to focus on different parts of the input sequence simultaneously by using multiple attention heads.
- Introduced positional encodings, a method to incorporate positional information of words in the sequence, which is crucial for understanding word order.
- The [architectural basis for large language models](#) (LLMs) that have transformed natural language processing.
- Broad Applicability: Extended beyond machine translation to other NLP tasks, including text summarization, question answering, and sentiment analysis.

Transformers in PHM

Already seeing transformers used in PHM

- Capable of handling sequential and multivariate sensor data essential for predictive maintenance.
- Model complex relationships and dependencies in machinery data for accurate failure predictions.
- Enhance the accuracy and reliability of maintenance intelligence systems.

e.g.,

Ding, Y., & Jia, M. (2022). Convolutional transformer: An enhanced attention mechanism architecture for remaining useful life estimation of bearings. *IEEE Transactions on Instrumentation and Measurement*, 71, 1-10.

Results on PRONOSTIA data set.

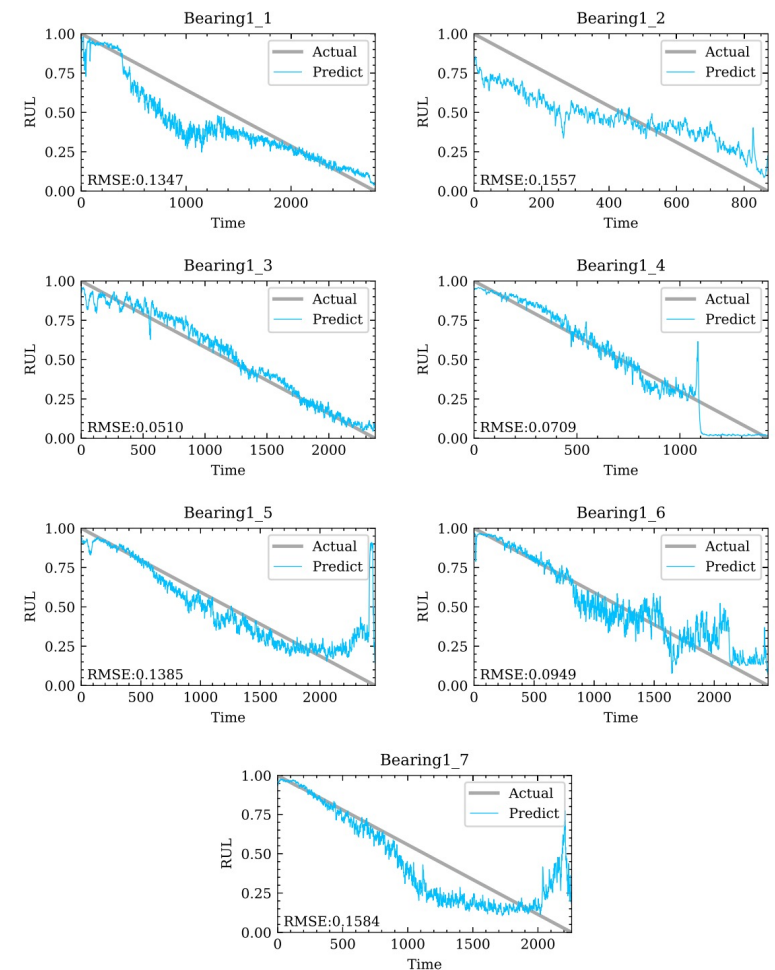
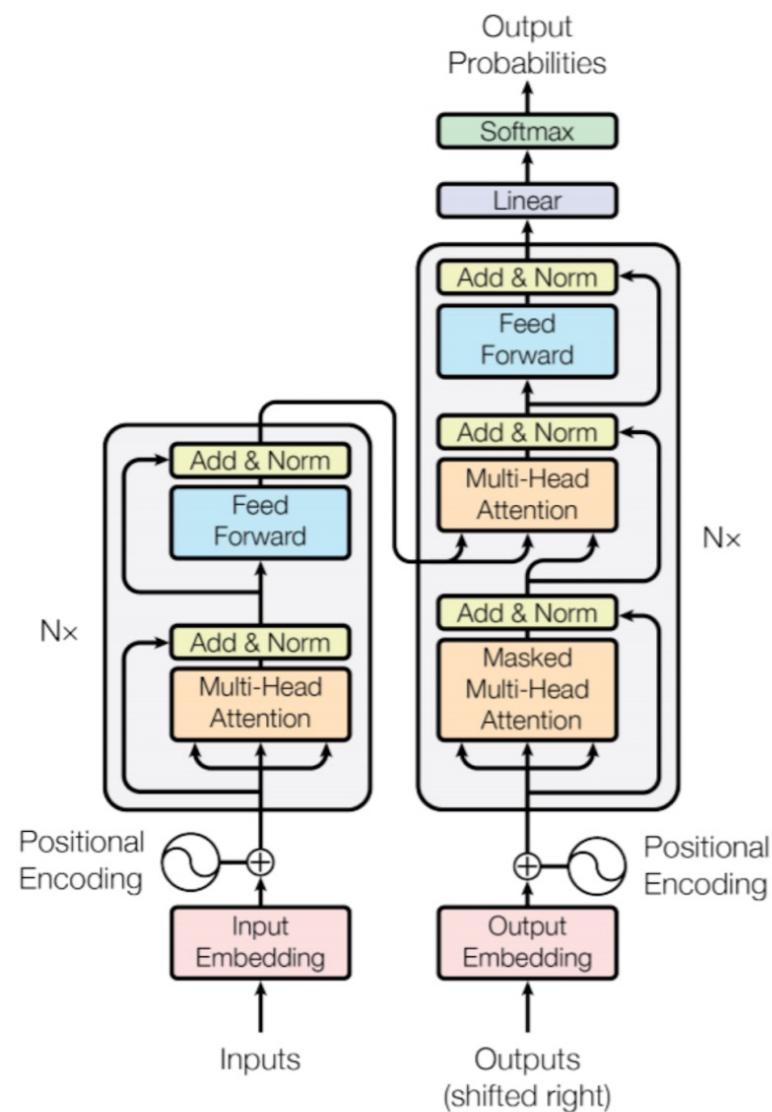


Fig. 10. RUL prediction with best performance on base CoT for Bearing1_1–Bearing1_7.

The Transformer Architecture

- Architecture diagram from Vaswani's 2017 paper
- Encoder on the left, decoder on the right
- A transformer network implementation will consist of multiple encoder and decoder blocks stacked on top of each other, with the output of one block being the input to the next block.



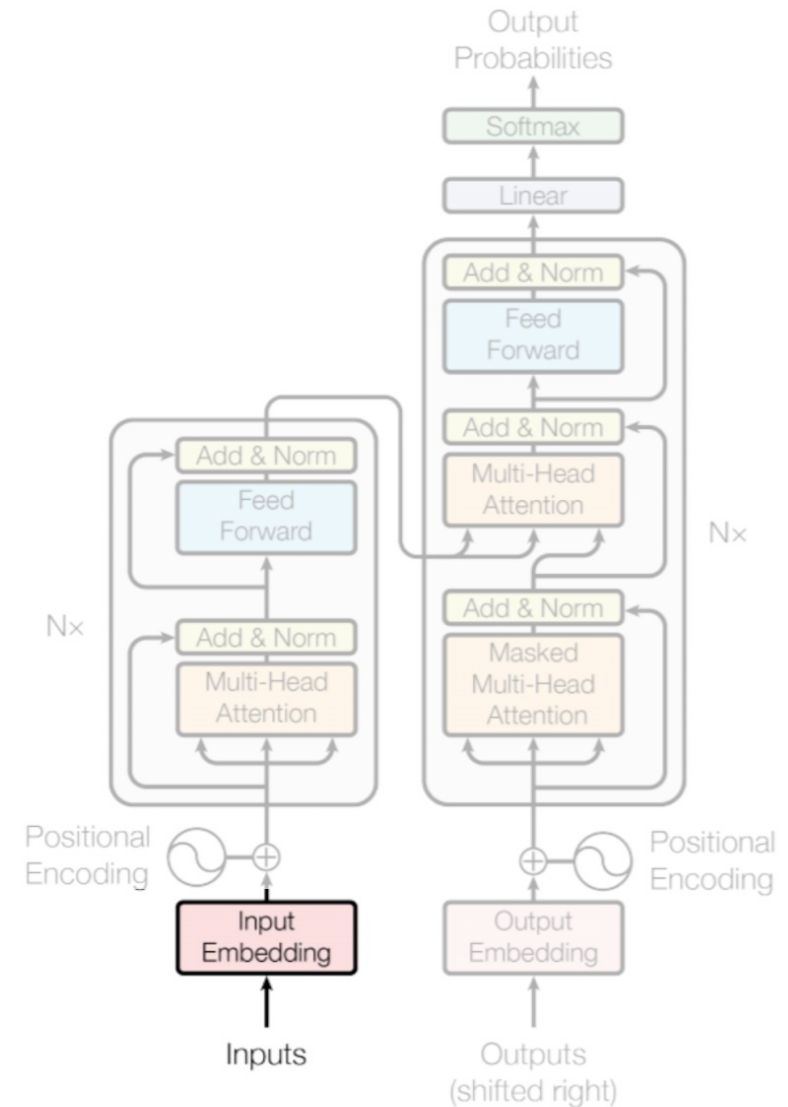
Inputs & Embedding

Inputs

- Whatever sequence you're trying to train a model on...
 - Time series sensor data
 - Audio
 - Video frames
- Will (mostly) focus on text processing in examples

Embedding

- Recall embedding models from above
 - representation of words as vectors, where semantically similar words are closer to each other in some high-dimensional space



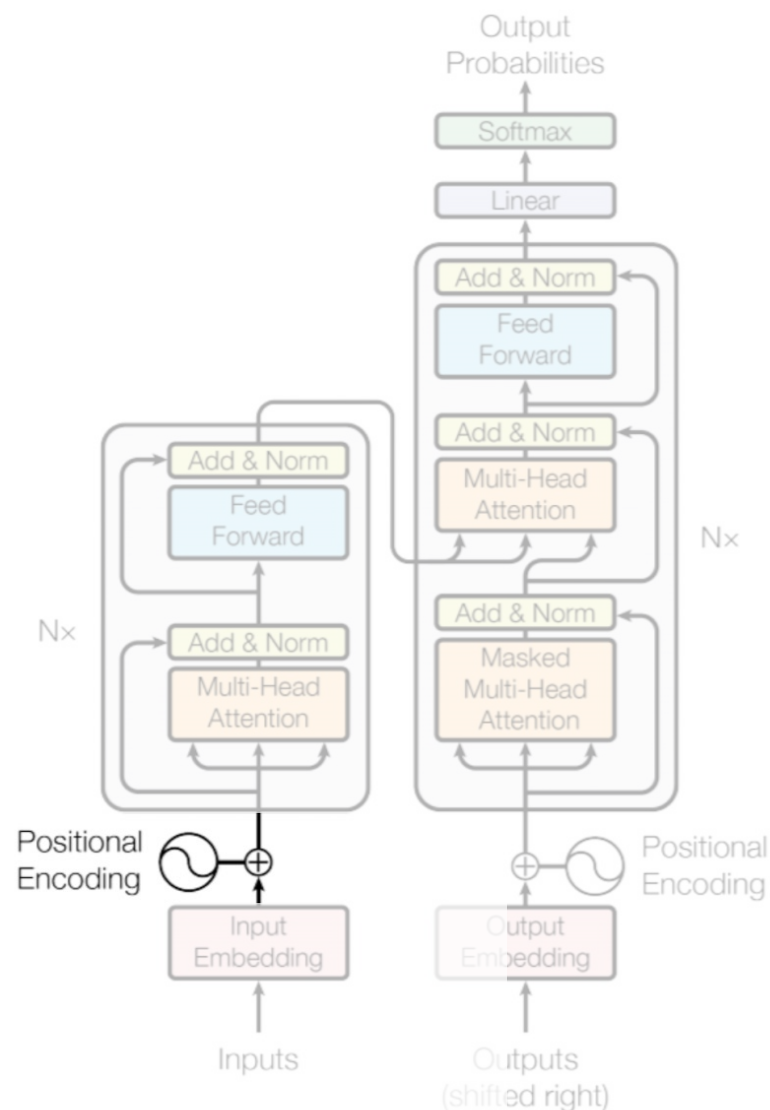
Positional Encoding

Positional encoding gives the transformer a sense of word order, enabling it to understand the meaning of a sentence based not just on the words themselves, but also on their *arrangement*.

- Crucial step that adds information about the position of words in a sequence to the model's input.
- Transformers **process all words in a sentence at the same time, which is highly efficient**. However, this means they lack the sequential information that RNNs inherently possess.
- The meaning of a sentence can change drastically depending on the order of words. Consider "The dog bit the man" versus "The man bit the dog." Positional encoding helps the transformer understand these differences.

How does positional encoding work?

- Each position in the input sequence is assigned a unique vector of values generated using sine and cosine functions with different frequencies.
 - **Allows the model to learn relative positions and distances between words.**
 - **Allows the model to generalize to sequences longer than those seen during training.**
- The positional encoding vectors are added to the word embeddings before being fed into the transformer's attention mechanism.



The Attention Mechanism

The attention mechanism in the encoder side of a transformer is called *self-attention*. It allows the model to weigh the importance of different words in the input sequence when encoding a specific word. Here's a breakdown of how self-attention works in the encoder:

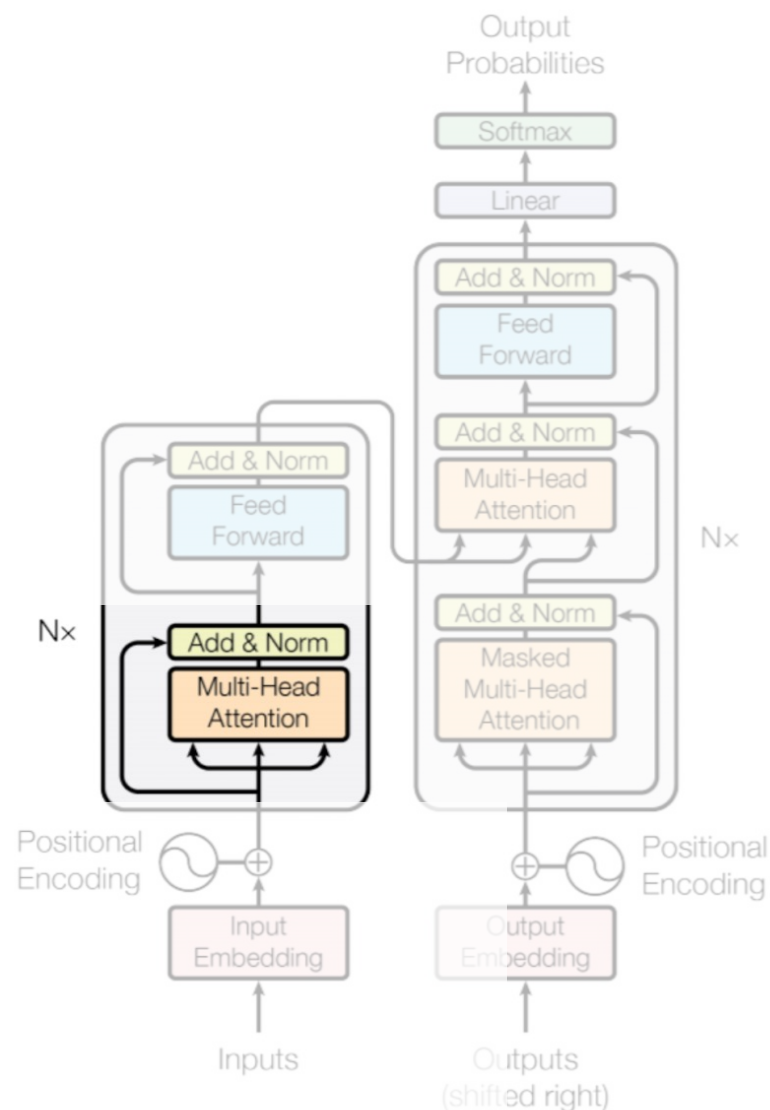
- Think of it as the model figuring out which other words in the sentence are most relevant to understanding the current word it's focusing on.

Creating Queries, Keys, and Values: For each word in the input sequence, three vectors are created:

- **Query (Q):** Represents the current word the model is focusing on.
- **Key (K):** Represents all other words in the sequence.
- **Value (V):** Holds the actual information (embedding) of each word.

Calculating Context:

- The query vector of each word is compared to the key vectors of all other words using a compatibility function (usually dot product). This produces attention scores, indicating how much each word relates to the current word.
- The attention scores are normalized using a softmax function, ensuring they sum up to 1 and represent probabilities.
- The normalized attention scores are used to weight the value vectors of all words. These weighted values are then summed up, producing a context vector for the current word.
- This context vector captures the relevant information from the entire input sequence, weighted by their importance to the current word. This enriched representation of the word is then passed on to the next layer of the encoder.



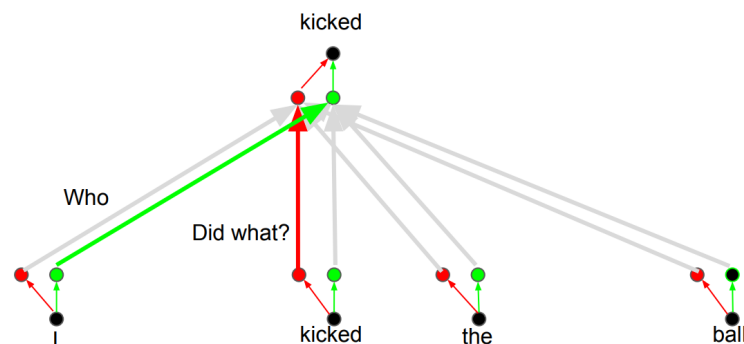
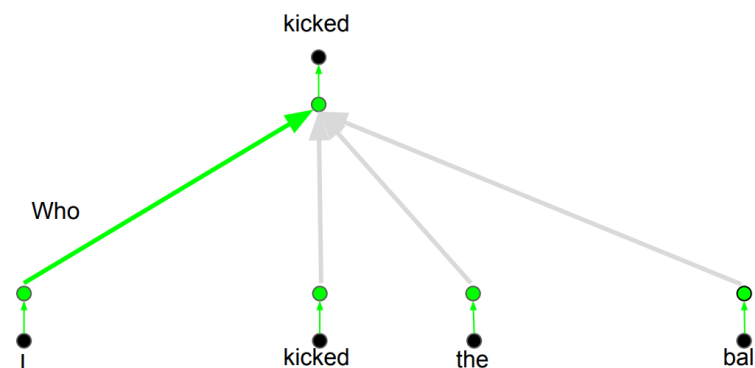
The Attention Mechanism (cont)

The attention layer picks out the relevant words and “highlights” them in the model, while diminishing the irrelevant words, to create a rich representation of text just as a human would.

- Imagine you're reading a sentence. As you focus on each word, your mind subconsciously considers other words in the sentence to understand its full meaning.
- Self-attention in transformers does something similar. It allows the model to weigh the importance of different parts of the input when encoding each word, capturing contextual relationships and dependencies between words.

Benefits of self-attention in the encoder:

- Captures long-range dependencies, in contrast to RNNs, which struggle with long sequences
- Attention scores for all words can be calculated simultaneously, making the encoding process highly efficient.
- Attention scores provide insights into which words the model is focusing on, offering some level of interpretability.



Feed Forward

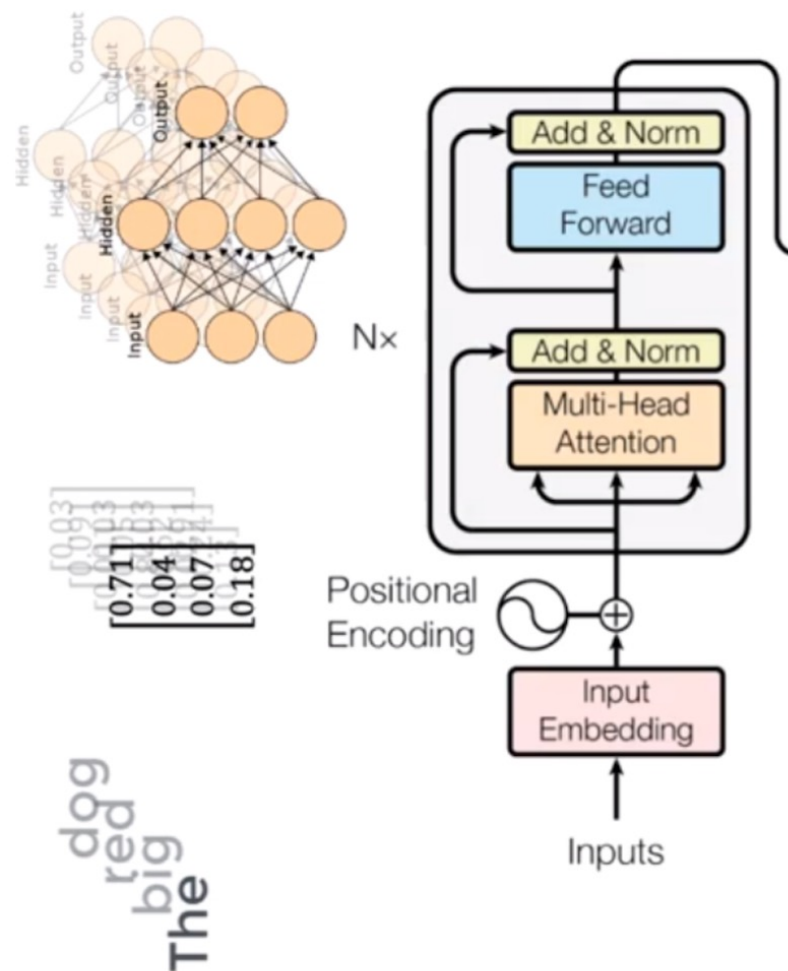
The feed-forward network (FFN) processes the output of the self-attention layer and introducing non-linearity and complexity to the model

- It takes the context vector produced by the self-attention layer as input. This vector represents a word enriched with contextual information from other words in the sequence.
- The context vector is passed through a series of fully connected layers with non-linear activation functions that allow the model to learn complex relationships and patterns in the data.
- The output of the FFN is a new representation of the word, further refined and transformed by the non-linear operations.

Why is the FFN important?

- The self-attention mechanism is primarily linear. The FFN introduces non-linearity, enabling the model to learn more complex functions and capture intricate relationships between words.
- The hidden layers in the FFN act as feature extractors, learning to identify salient features and patterns in the input data.
- By combining self-attention with the FFN, the transformer can create highly expressive representations of words, capturing both contextual information and complex non-linear relationships.

Think of the FFN as a “refining” step after the self-attention mechanism. It takes the contextually enriched representation of a word and further processes it, adding depth and complexity to the model's understanding. This allows the transformer to learn more nuanced and sophisticated relationships between words in a sentence



Outputs & Output Embedding

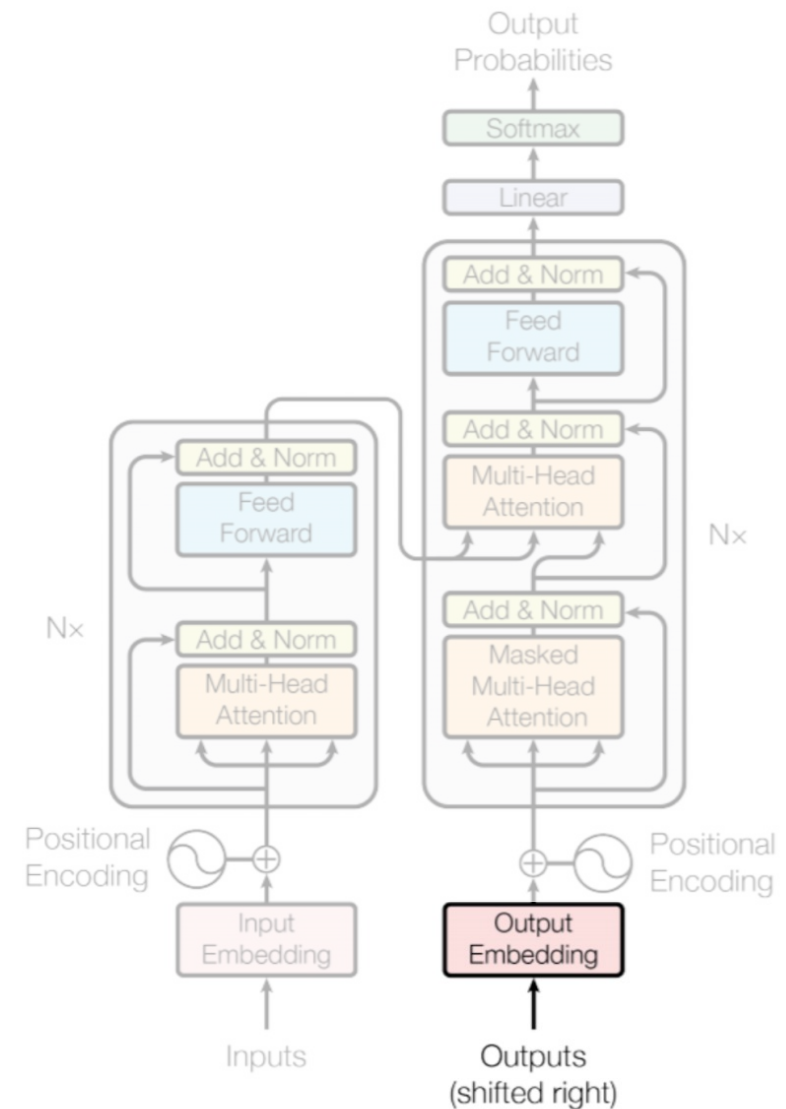
On the decoder side, the input is a time-shifted **output**.

- e.g., if we're training a translator for English to German, for training, we need to give an English sentence along with its translated German version for the model to learn. So, our English sentences pass through encoder block, and German sentences pass through the decoder block.

Decoder Output Embeddings:

- **Dynamic:** Unlike the encoder's fixed embeddings, the decoder creates embeddings on-the-fly for each word it predicts in the output sequence.
- The decoder generates the output sequence step-by-step. Each new word's embedding is influenced by the embeddings of the words generated before it. This is called "autoregressive" generation.
- **These embeddings aren't about representing the input. They are the decoder's guesses for the next word in the output.**

Encoder embeddings encode meaning; decoder embeddings **create** it.



Decoder Positional Encoding: It's All Relative

Decoders, like encoders, require positional information to correctly order the sequence they generate word-by-word. While **absolute positional encoding** (using fixed positions like 1st, 2nd, 3rd word) is commonly used – similar to encoders – **relative positional encoding** is another important approach, especially beneficial for generation tasks.

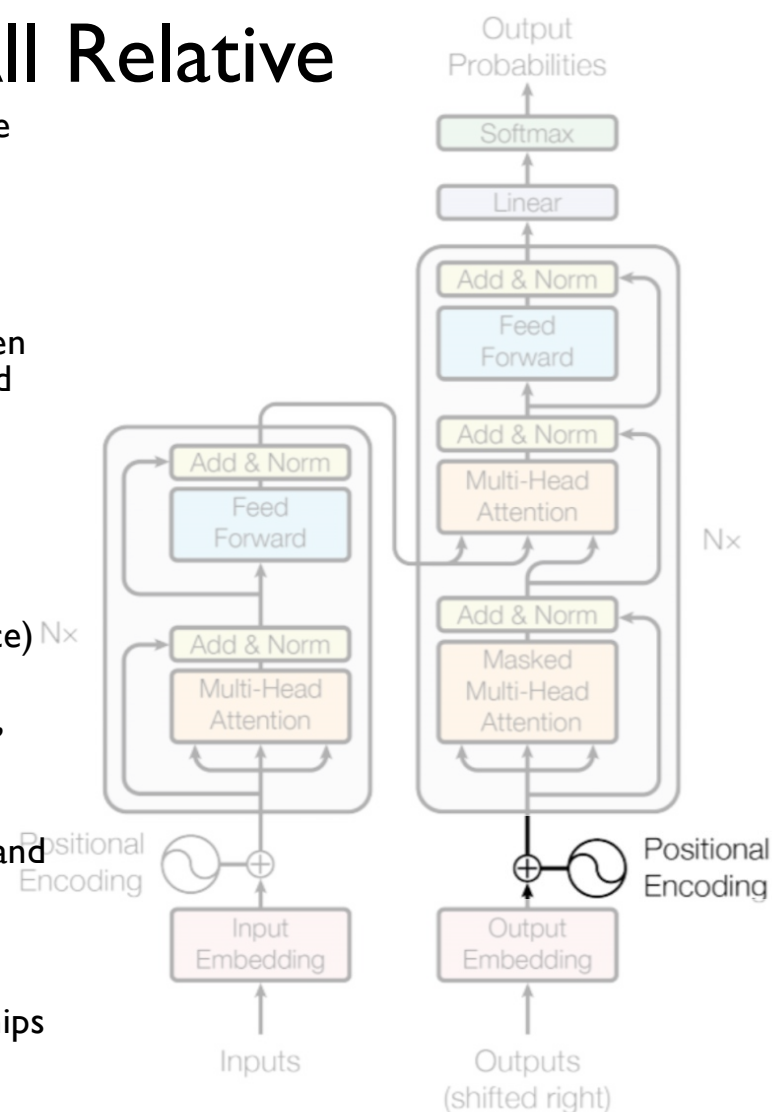
With relative encoding, the focus shifts to the distance or relationship between words in the output sequence as it's created step-by-step. Knowing “this word comes 3 positions after that one” becomes key information. This **focus on relative distance allows the decoder to be highly flexible with the length and structure of the output**, enabling it to generate sentences of varying lengths more dynamically than relying solely on fixed position assignments.

Think of it like building with LEGOs:

- Encoder: Looks at the complete instruction manual (the input sentence) $N \times$ – often using absolute positions.
- Decoder: Assembles the LEGO model brick-by-brick (word-by-word), focusing not just on the step number, but also on how each piece connects to those before it – like relative positions.

Some transformer models use absolute positional encoding in both encoder and decoder (as in the original architecture), while others have adopted relative positional encoding for improved flexibility and performance. The choice depends on the specific model design and application.

Relative positional encoding can help the decoder maintain correct relationships between words as it builds the output sequence, though both approaches are used in practice.



Decoder Multi-Head Attention: A Two-Pronged Approach

The decoder uses multi-head attention in two distinct ways, each serving a different purpose:

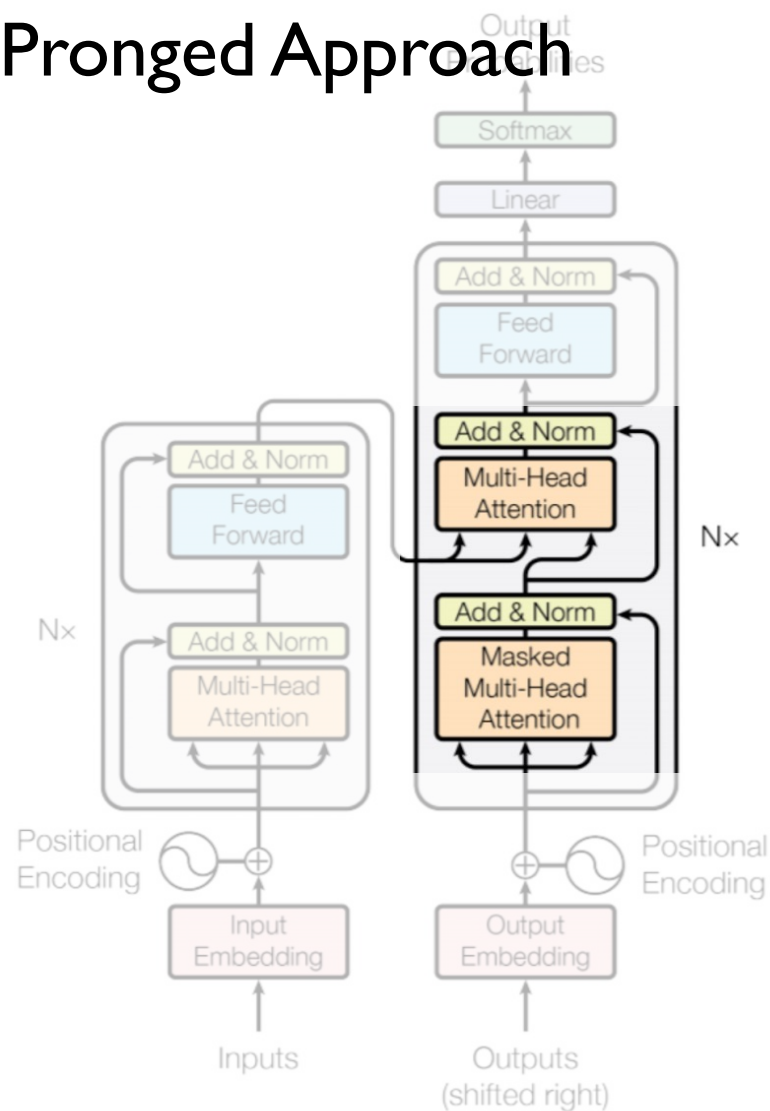
1. Masked Self-Attention: Focus on the Past

- Like the encoder, the decoder uses self-attention to understand relationships between words in the sequence. However, it wears a "mask" to prevent it from "seeing" future words.
- This is crucial because the decoder generates the output word-by-word. The mask ensures each word is predicted based only on the preceding context, not on words that haven't been generated yet.

2. Encoder-Decoder Attention: Bridging the Gap

- This attention layer allows the decoder to "look at" the encoder's output. It helps the decoder focus on the relevant parts of the input while generating each word of the output.
- Instead of treating the entire input equally, the decoder can pinpoint the most useful information for each step of the output generation process.

The decoder uses multi-head attention to both maintain the sequential order of output generation (masked self-attention) and connect to the encoded input (encoder-decoder attention). This dual functionality is key to the decoder's ability to generate coherent and meaningful sequences.



Decoder FFN: Refining Meaning, One Word at a Time

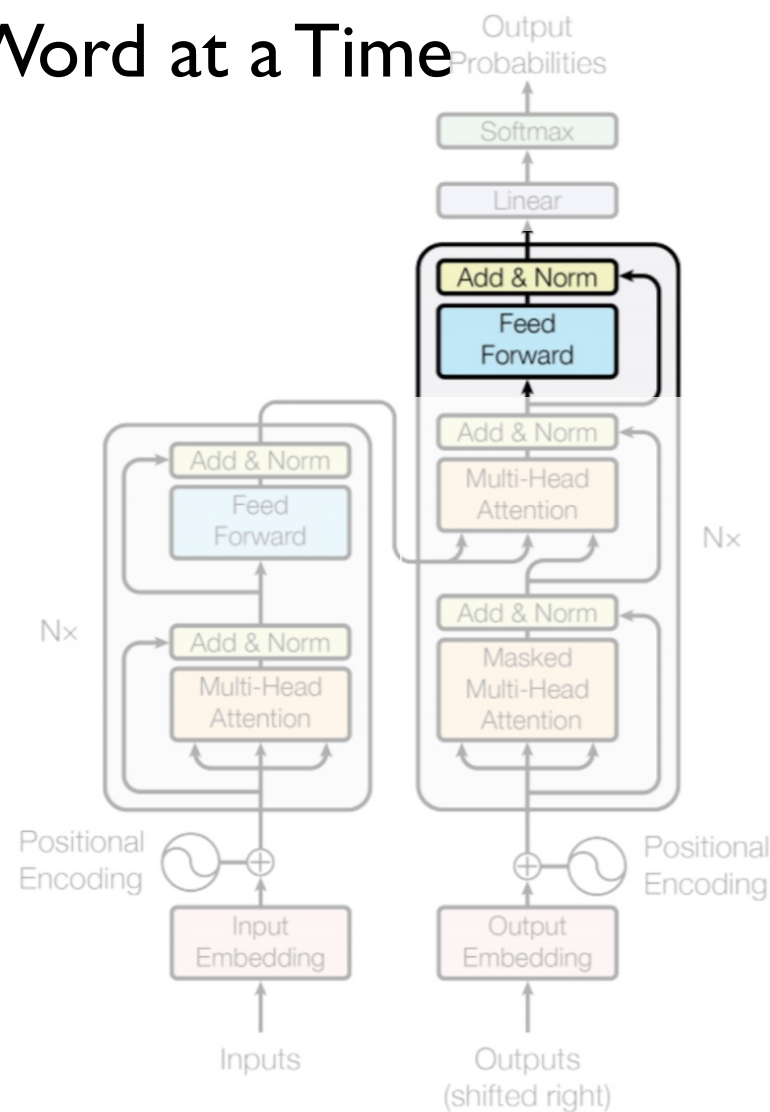
Just like in the encoder, the decoder's feed-forward network consists of fully connected layers with non-linear activation functions.

- It takes the output of the decoder's multi-head attention (which includes both masked self-attention and encoder-decoder attention) and further processes it.
- The FFN introduces non-linearity, allowing the decoder to capture more intricate relationships between words and generate more nuanced and expressive output.

Think of it as a sculptor refining a clay model:

- Multi-head attention roughly shapes the clay, capturing the basic form and relationships.
- Feed-forward network adds detail, smooths out rough edges, and refines the overall form to create a more polished and expressive final product.

The decoder's FFN plays a crucial role in refining the representations produced by the multi-head attention layers, ultimately contributing to the generation of more sophisticated and coherent output sequences.

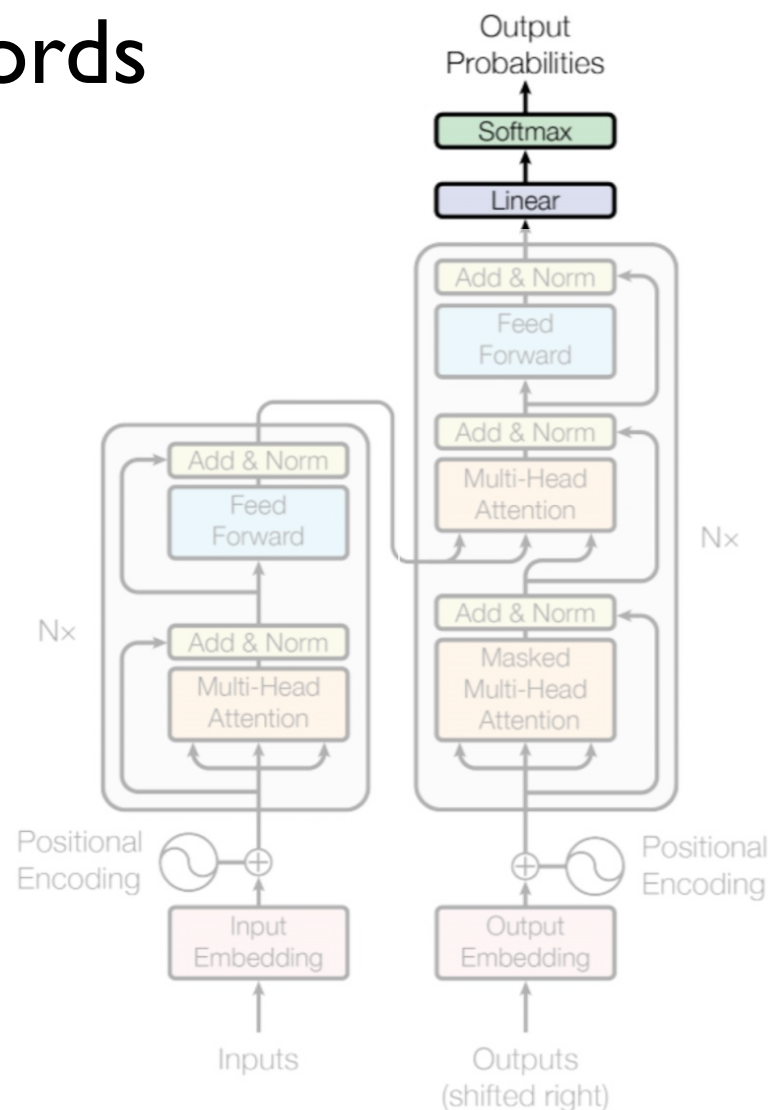


Decoder: From Numbers to Words

The decoder's final layers (linear, softmax, and output probabilities) are all about turning the decoder's internal representations back into actual words in the output sequence.

- Linear Layer acts like a translator, converting the decoder's final representation (a vector) into a format that can be understood in the context of your vocabulary. It maps the vector to a score for each word in your vocabulary.
- Softmax Layer takes those scores from the linear layer and turns them into probabilities. It ensures that:
 - Each probability is between 0 and 1.
 - All probabilities for all the words in your vocabulary add up to one
- Output Probabilities represent the decoder's "confidence" in each word being the correct next word in the sequence. The word with the highest probability is typically chosen as the decoder's prediction.

These final layers bridge the gap between the decoder's internal workings and the actual output text, allowing the model to generate human-readable sequences.

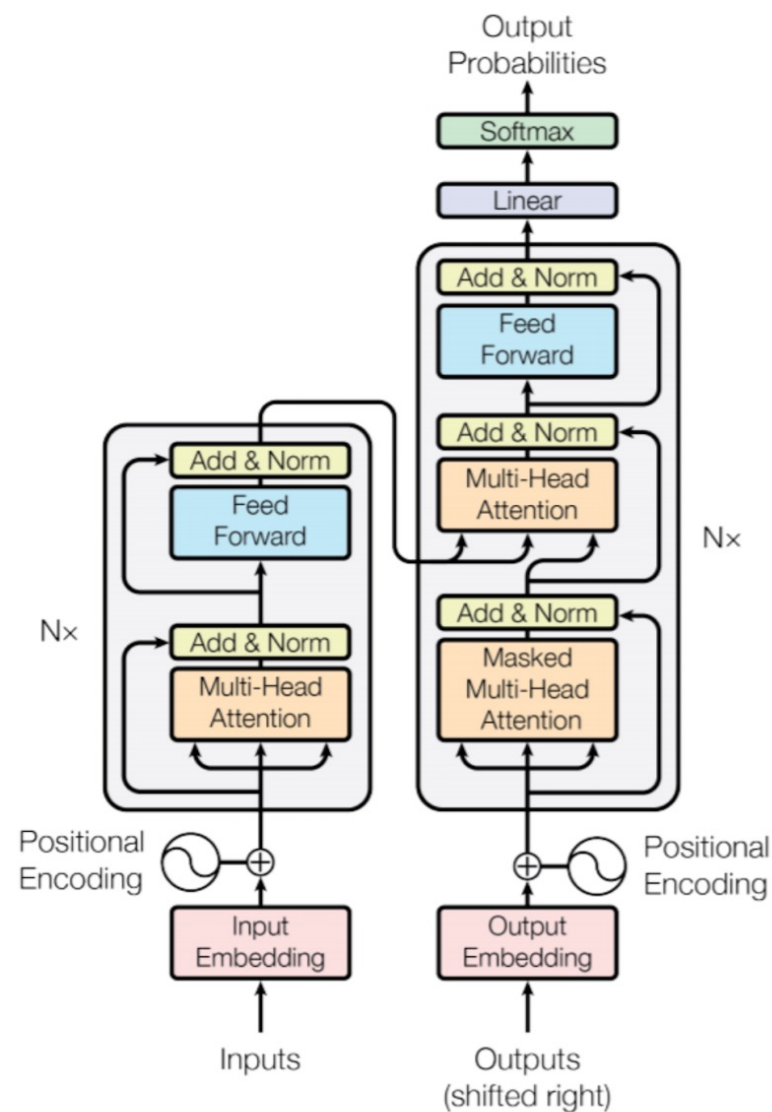


Applying Transformers for PHM

e.g., for vibration data,

- Preprocess as usual – cleaning, remove DC component, normalization, [(optionally) feature extraction]
- Segment raw vibration signals into fixed-length pieces called "tokens."
- Use a time-sliding window strategy to create sequences of tokens for the Transformer input
- Shape each data sample into blocks representing sequences for tokenized inputs.
- Apply labels (RUL, class) to each sequence, creating the dataset for training
- Embedding: Perform a linear projection for each token, adding a trainable [class] token at the start of sequences.

By combining these techniques, you can effectively represent your numerical time series data within the transformer architecture.



Advantages of Transformers for PHM

- **Global Contextual Awareness:** Captures long-range dependencies across entire sequences, which is critical for tracking complex degradation patterns in equipment.
- **Multi-Modal Compatibility:** Easily integrates with multiple data types (e.g., vibration, thermal, acoustic) in a single model, enhancing diagnostic capabilities.
- **Scalability:** Highly scalable for large datasets and can leverage pre-trained models, reducing the need for extensive labeled data.
- **Attention Mechanism:** Provides interpretability by highlighting which parts of the input sequence are most relevant for predictions, aiding explainability.
- **Adaptation:** Efficiently updates with new data, enabling dynamic Remaining Useful Life (RUL) predictions and real-time monitoring.
 - Incorporate new data and adjust its predictions dynamically.
 - Fine-tune with new incoming data without retraining the entire model, allowing the system to adjust to changes in the equipment's operational environment or degradation patterns.
 - With incremental updates, transformers support real-time fault detection and anomaly detection by quickly adapting to changes in the equipment's behavior.

Disadvantages of Transformers for PHM

- High Computational Cost
 - Transformers require substantial computational resources, particularly for large datasets and long sequences, which can limit real-time applications in resource-constrained environments.
- Data-Hungry
 - Training transformers effectively requires large amounts of labeled data, which may be difficult to obtain in PHM where failures are rare or unpredictable.
- Complexity in Model Tuning
 - Transformers have many hyperparameters and require careful tuning, which can be challenging and time-consuming for optimal performance.
- Lack of Temporal Bias
 - Unlike traditional time-series models, transformers do not inherently model sequential order, requiring additional mechanisms (like positional encoding) to handle time dependencies effectively.
- Limited Explainability
 - Despite attention mechanisms, transformers can be challenging to interpret fully, especially in applications where understanding failure mechanisms is essential.
- Overfitting Risk:
 - Transformers can overfit on small datasets, making it difficult to generalize in applications where labeled data is limited.

Application Architectures

Transformers can be combined in various ways to tackle complex tasks.

Stacked Transformers:

- Like building blocks, multiple transformers are stacked on top of each other.
- The output of one transformer becomes the input of the next.
- This allows for deeper processing and more complex reasoning.
- e.g., Some language models use this to achieve better understanding of long texts.

Ensemble Transformers:

- Multiple transformers are trained independently on the same (or different) data.
- Their predictions are combined (e.g., by averaging) to produce a final output.
- This can improve robustness and accuracy by leveraging the diverse strengths of each transformer.
- E.g., Used in machine translation to get more reliable translations.

Hierarchical Transformers:

- Transformers are arranged in a tree-like structure.
- Different transformers specialize in different levels of granularity (e.g., sentence-level, paragraph-level, document-level).
- This allows for efficient processing of hierarchical information.
- E.g., Used in document summarization to capture the essence of long texts.

Hybrid Architectures:

- Transformers are combined with other types of neural networks (e.g., convolutional neural networks, recurrent neural networks).
- This leverages the strengths of different architectures for specific tasks.
- e.g., Combining transformers with CNNs for image captioning.



Scaling Up: From Transformer to Large Language Model

An LLM is Essentially a Scaled-up Transformer

To get from the concept of a transformer to a large language model (LLM), the path involves:

- **Massive Data:** LLMs are trained on *truly* massive datasets of text and code. This could include books, articles, code repositories, and much of the internet. This vast data allows the model to learn patterns, grammar, and even some facts about the world.
- **Huge Model Size:** LLMs have a massive number of parameters (think billions or trillions). More parameters generally mean a more capable model.
 - OpenAI's GPT-3 uses 175 billion parameters.
 - OpenAI's o1-preview uses 2.8 trillion parameters (unconfirmed by OpenAI)
- LLMs are often trained using a technique called “generative pre-training.” This means they are trained to predict the next word in a sequence, which forces them to learn the underlying structure of language.
- Since LLMs can be computationally intensive, optimizations in serving the model (e.g., quantization, distillation) make them more practical for real-world applications by reducing the computational load without sacrificing performance.

While pre-training gives LLMs a broad understanding of language, they can be further fine-tuned for specific tasks like:

- Writing stories, articles, or summaries.
- Converting text between languages.
- Providing answers to questions based on given information.
- Writing code in various programming languages.

Multimodal LLMs

True to its name, a multimodal LLM can process and understand information from multiple modalities – like text, images, audio, and video.

- By combining information from different modalities, multimodal LLMs can build a richer and more nuanced understanding of the world, much like we do.
- They can analyze images, understand speech, generate text, and even create images or audio.
- They can bridge the gap between different forms of communication. Imagine an AI that can describe images to someone who is visually impaired, or translate spoken language into sign language.

This enables applications like:

- More intelligent and capable AI systems. By mimicking human perception, they can understand and interact with the world in more meaningful ways.
- Image captioning: Generating descriptive captions for images.
- Visual question answering: Answering questions about images.
- Text-to-image generation: Creating images from text descriptions.
- Interactive storytelling: Generating stories that incorporate images and sound.
- Searching not just with keywords, but with images or sounds. Control devices and applications with natural language and gestures.

Building a Multimodal LLM

- Images, text, and audio are fundamentally different kinds of data. An LLM trained on text can't inherently understand images or sounds.
- Integration: The key challenge is to find a way to represent these diverse data types in a way that a single model can understand and process them together.
- Need specialized encoders for each modality:
 - Text Encoder: A transformer-based LLM like those we discussed earlier.
 - Image Encoder: Often a Convolutional Neural Network (CNN) or a Vision Transformer (ViT) to extract features and create embeddings from images.
 - Audio Encoder: Models like Whisper that can convert audio into a sequence of representations.

Creating a Shared Representation Space

- The encoders need to translate their respective data types into a common “language” or representation space that the model can understand. This often involves mapping the outputs of each encoder into a shared embedding space.
- A multimodal LLM needs a mechanism to fuse or combine the information from different modalities. This might involve:
 - Simple Concatenation: Combining the embeddings from different modalities into a single input sequence.
 - Cross-Attention: Allowing the model to attend to information from different modalities, similar to how self-attention works within a transformer.
 - Multimodal Training: The entire model is then trained on data that includes multiple modalities. For example, it might be trained on image-caption pairs, audio recordings with transcripts, or video clips with descriptions.

Multimodal LLMs build upon the foundation of transformers and LLMs by incorporating specialized encoders for different modalities, creating a shared representation space, and training on multimodal data. This allows these models to process and generate information from multiple sources, leading to a deeper understanding of the world and exciting new applications.

Multimodal Transformer Systems (LLMs) for PHM

PHM relies on various types of data:

- Sensor Data: Vibration, temperature, pressure, acoustic emissions, etc.
- Inspection Data: Images, videos from visual inspections, X-rays, etc.
- Maintenance Records: Textual descriptions of repairs, replacements, etc.
- Operational Logs: Records of usage patterns, environmental conditions, etc.

Multimodal transformers can ingest and fuse these diverse data sources, capturing complex relationships that might be missed by traditional PHM methods.

Multimodal Transformer Systems (LLMs) for PHM (cont.)

Enhanced Feature Extraction

- Automatic Feature Engineering: Transformer can automatically identify relevant features from raw sensor data, reducing the need for manual feature engineering.
- Cross-Modal Feature Learning: They can learn correlations between different modalities, such as linking patterns in sensor data with visual anomalies in inspection images.

Improved Predictive Modeling

- By combining information from multiple sources, multimodal transformers can create more accurate and robust predictive models.
- They can detect subtle anomalies across different modalities that might signal impending failures, enabling proactive maintenance.

“Explainable” PHM

- Multimodal transformer can potentially provide insights into the underlying causes of failures by analyzing the relationships between different data sources.
- They can generate human-readable explanations of their predictions, increasing trust and transparency in PHM systems.

Challenges and Considerations

- Collecting and preparing diverse data for multimodal transformers can be challenging.
- Training large multimodal models requires significant computational resources.
- Integrating domain expertise into the development and deployment of multimodal transformers for PHM is essential.

Multimodal Transformers have the potential to significantly advance PHM by enabling more comprehensive data analysis, improved prediction accuracy, and enhanced explainability. This could lead to more efficient maintenance, reduced downtime, and improved safety across various industries.

From LLMs to Retrieval-Augmented Generation (RAG)

The Limitation of LLMs:

- LLMs are trained on a fixed dataset, so their knowledge is limited to what they learned during training.
- LLMs can sometimes generate incorrect or nonsensical information, especially when dealing with rare or unseen concepts.

The Need for External Knowledge:

- To overcome these limitations, we need a way to access and integrate external knowledge sources.
- This is where RAG comes in!

How RAG Works:

- **Retrieve:** Given a user query, RAG retrieves relevant information from external knowledge sources (e.g., databases, documents).
- **Augment:** This retrieved information is used to augment the LLM's input, providing it with the context and knowledge it needs to generate a more accurate and informative response.
- **Generate:** The LLM then generates a response based on both the user query and the retrieved knowledge.

Key Components of RAG:

- A collection of documents or a database containing relevant information.
- A retrieval model that can efficiently search the knowledge source and retrieve relevant documents.
- A large language model that generates the final response.

RAG Pros & Cons

Benefits of RAG:

- **Up-to-date Information:** Access to external knowledge sources allows RAG to provide information that is more current than the LLM's training data.
- **Reduced Hallucination:** Grounding the LLM's response in retrieved knowledge helps to reduce the risk of generating incorrect or nonsensical information.
- **Improved Accuracy:** RAG systems can achieve higher accuracy on tasks that require factual knowledge or specific domain expertise.

Downsides of RAG

- **Complexity:** RAG systems have more components than standalone LLMs, making them more complex to build and maintain. This includes managing the knowledge source, retrieval model, and the LLM itself.
- **Retrieving and processing information from external sources adds computational overhead,** which can increase latency and cost, especially with large knowledge bases.
- **The effectiveness of RAG depends *heavily* on the quality of the retrieval model.** Inaccurate or irrelevant retrievals can lead to poor or misleading responses.
- **Keeping the knowledge source up-to-date and ensuring the retrieval model remains effective requires ongoing maintenance and monitoring.** Although easier to maintain RAG than update an LLM

RAG in PHM

Framework:

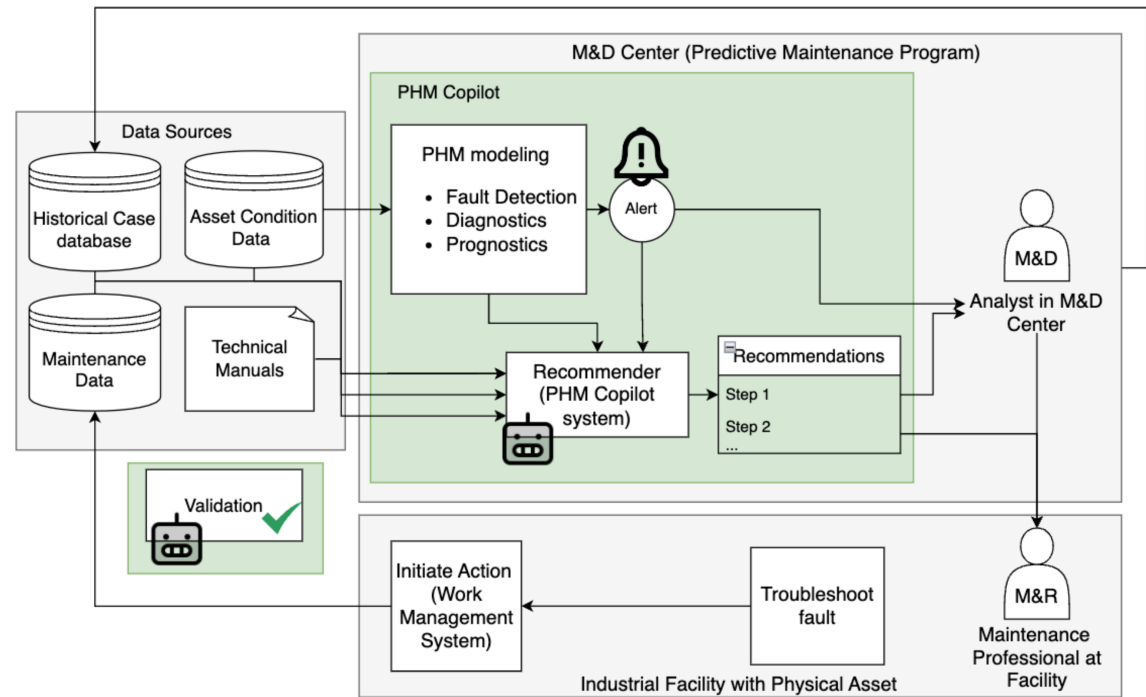
- Proposes integrating LLMs as in-the-loop agents in PHM workflows.
- The LLM-based PHM Copilot includes data processing, failure mode discovery, and structured maintenance recommendations triggered by PHM alerts.

Case Study:

- Utilized historical PHM data from GE Vernova for validation.
- Found that **integrating RAG significantly improved recommendation accuracy and relevance by drawing on similar past cases.**
 - RAG-enhanced recommendations were more likely to capture failure modes and did so in fewer steps than the LLM-only approach.
 - SMEs evaluated the LLM agents' outputs, finding the observation and failure mode extraction agents generally accurate, though complex cases required additional SME labeling.
 - The framework demonstrated potential for effective troubleshooting, though improvements in evaluator accuracy and retrieval diversity were noted as areas for future work.

LLMs have value in PHM for data quality improvement and supporting prescriptive decision-making.

Figure 1. Conceptual illustration of a PHM Copilot as part of a comprehensive PHM system.

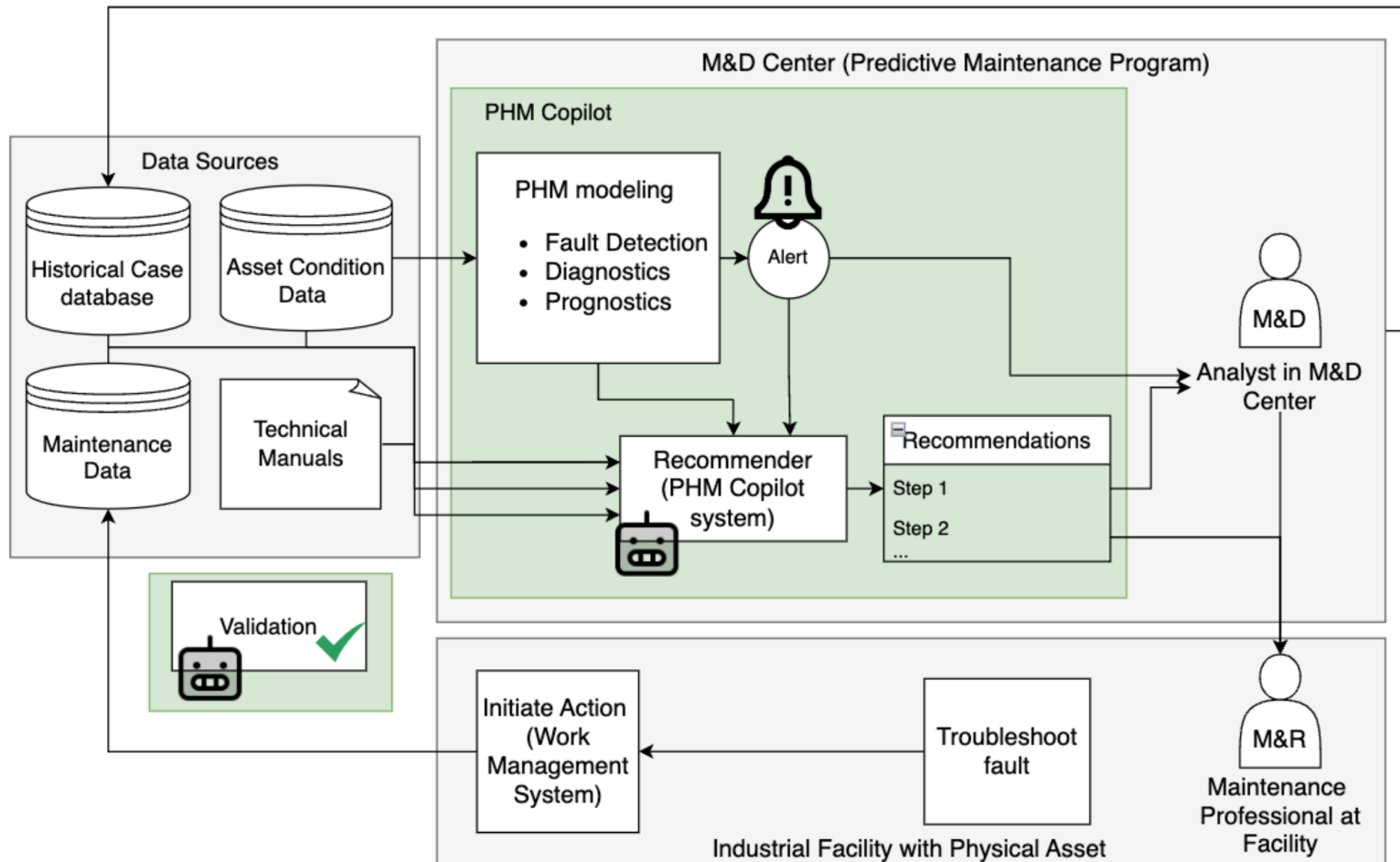


Lukens, S., McCabe, L. H., Gen, J., & Ali, A. (2024, November). Large Language Model Agents as Prognostics and Health Management Copilots. In *Annual Conference of the PHM Society* (Vol. 16, No. 1).

The background is a solid dark blue color. Overlaid on this are numerous thin, light blue, wavy lines that create a sense of depth and movement. These lines are arranged in a way that suggests a series of overlapping, translucent planes or a complex, flowing structure. The overall effect is a modern, abstract, and somewhat futuristic aesthetic.

Demos!

Multimodal LLMs “describe this image”



Multimodal LLMs “describe this image”

This diagram represents a Predictive Maintenance (PdM) framework integrated with a Prognostics and Health Management (PHM) system, known here as the PHM Copilot. It highlights the data flow and interaction between various components involved in monitoring, diagnosing, and recommending maintenance actions for physical assets in an industrial facility.

Key Components:

Data Sources:

Historical Case Database: Stores past cases for reference.

- **Asset Condition Data:** Collects real-time or periodic data on asset health.
- **Maintenance Data:** Documents maintenance history and procedures.
- **Technical Manuals:** Provides operational and troubleshooting guidance.

PHM Copilot:

Contains PHM modeling for **Fault Detection, Diagnostics, and Prognostics**.

- Generates an **Alert** when potential issues are detected based on the asset's condition.
- Passes information and data to a **Recommender** system for generating maintenance recommendations.

Recommender (PHM Copilot System):

Utilizes the data and insights from PHM modeling, historical cases, and manuals.

- Provides specific **Recommendations** with step-by-step actions (Step 1, Step 2, etc.).

Multimodal LLMs “describe this image”

M&D Center (Monitoring & Diagnostics Center):

Houses an **Analyst** responsible for receiving alerts and recommendations, assessing the situation, and coordinating actions with maintenance professionals.

Industrial Facility with Physical Asset:

Maintenance and Repair (M&R) professionals receive tasks and troubleshoot identified faults.

Actions are initiated through a **Work Management System** and relayed to M&R professionals for execution.

Validation:

The validation component, possibly an AI or system check, ensures that the recommendations meet quality or reliability standards before implementation.

Workflow Summary:

1. Data from historical cases, asset conditions, maintenance logs, and technical manuals feed into the PHM Copilot.
2. The PHM Copilot's models detect potential issues, generating an alert.
3. The Recommender processes data to provide actionable recommendations.
4. An analyst in the M&D Center evaluates the alert and recommendations, coordinating with M&R staff to address the fault.
5. Validated recommendations are passed to the Work Management System, triggering maintenance actions at the facility.

This system enhances proactive maintenance by leveraging historical data, real-time monitoring, and guided recommendations, aiming to reduce asset downtime and extend equipment life.

Chat With Data

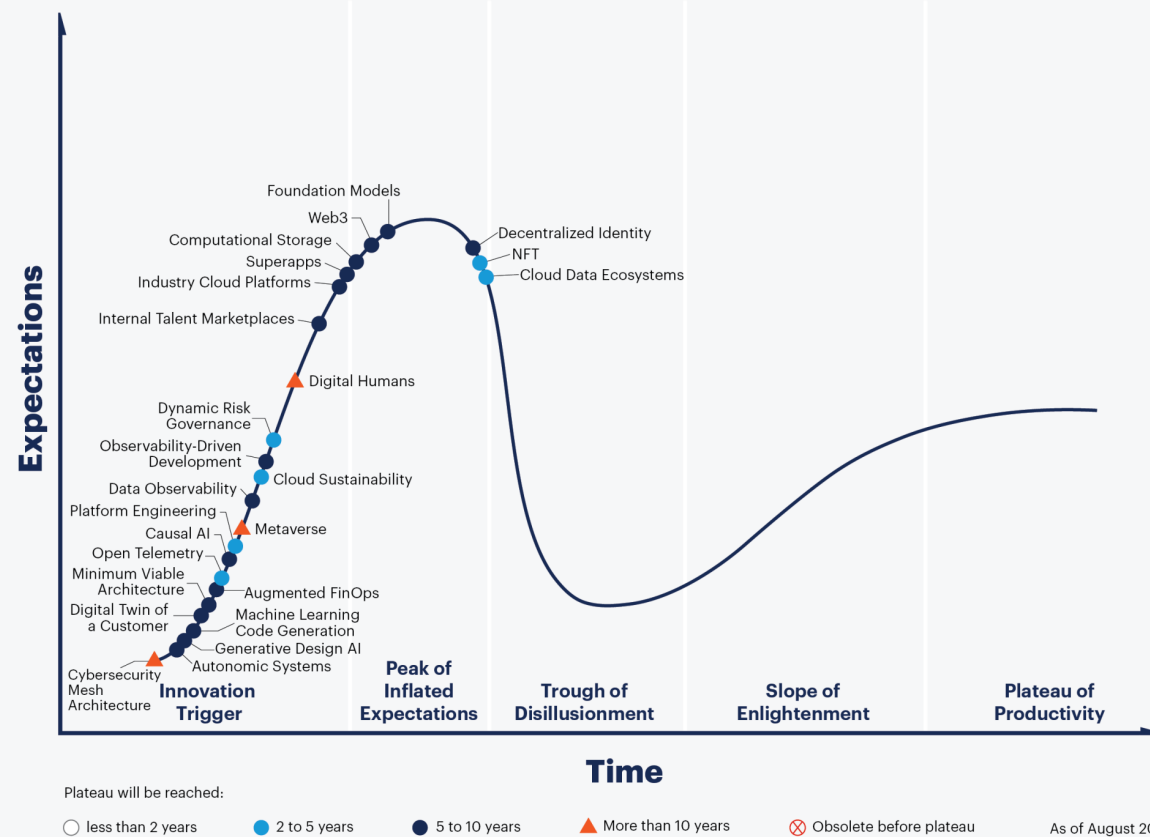
Chat With Docs

Unsupervised Clustering

The background is a solid dark blue color. Overlaid on this are numerous thin, translucent, light blue lines that form a series of overlapping, wavy, vertical shapes. These shapes resemble stylized, elongated 'S' curves or perhaps the edges of a spiral staircase, creating a sense of depth and movement. The lines are more concentrated in the center and fade out towards the edges.

Where is this going?

Hype Cycle for Emerging Tech, 2022

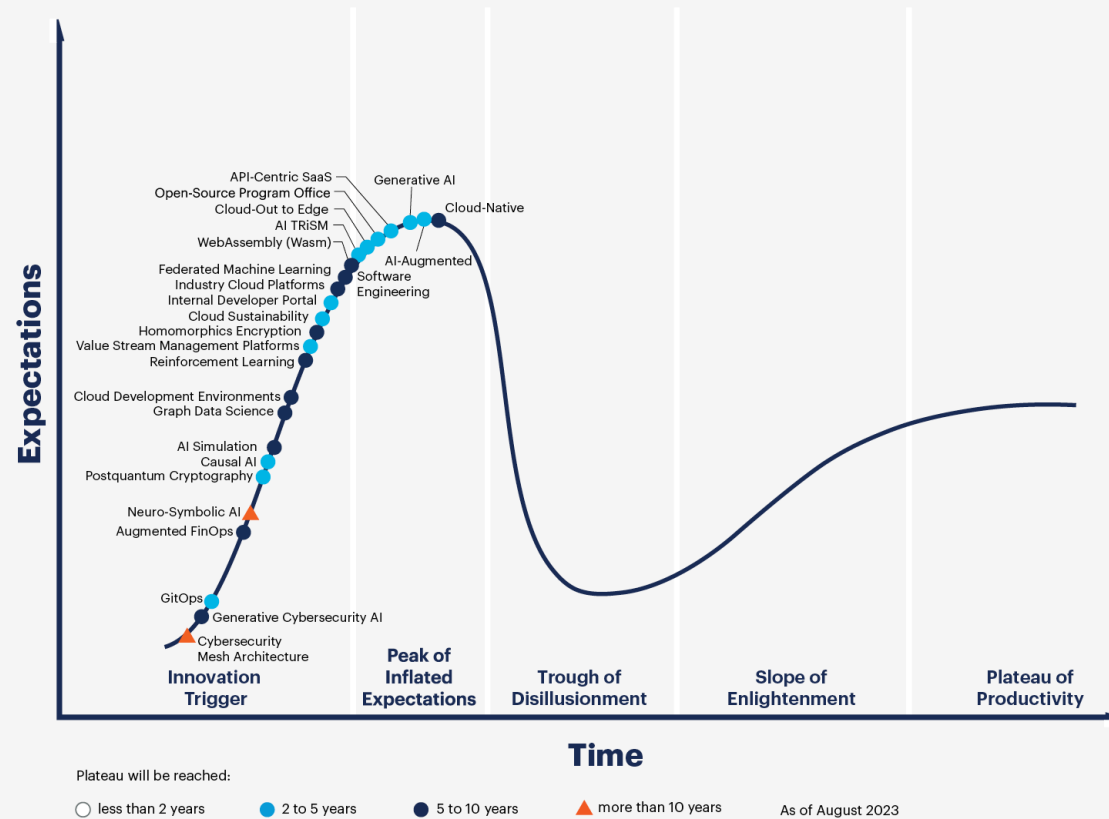


gartner.com

Source: Gartner
 © 2022 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner and Hype Cycle are registered trademarks of Gartner, Inc. and its affiliates in the U.S. 1893703

Gartner

Hype Cycle for Emerging Technologies, 2023

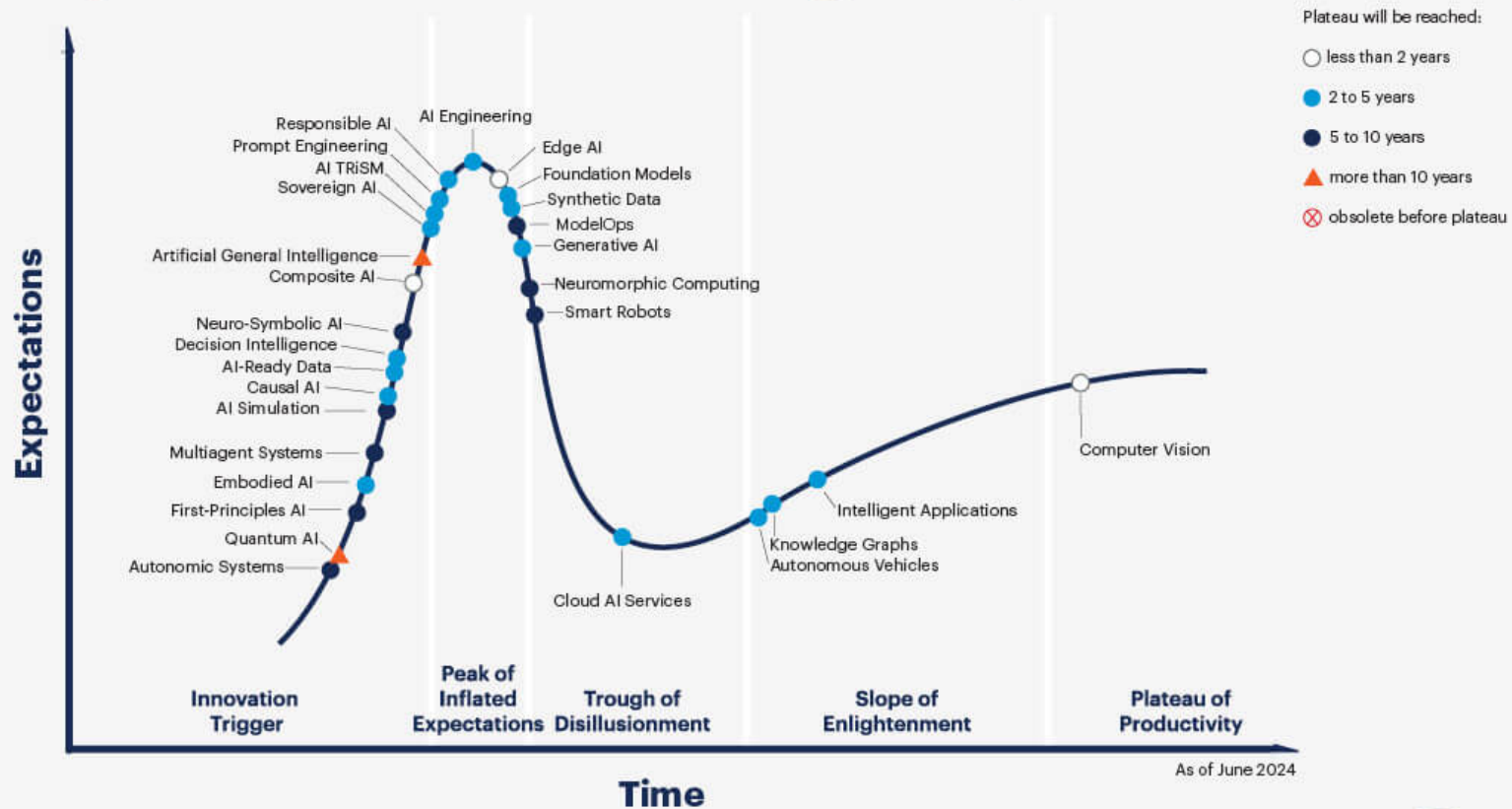


gartner.com

Source: Gartner
© 2023 Gartner, Inc. and/or its affiliates. All rights reserved. 2079700

Gartner

Hype Cycle for Artificial Intelligence, 2024



Source: Gartner
Commercial reuse requires approval from Gartner and must comply with the Gartner Content Compliance Policy on [gartner.com](https://www.gartner.com/content-compliance-policy).
© 2024 Gartner, Inc. and/or its affiliates. All rights reserved. GTS_3282450

Gartner®

Gartner's Take

GenAI has passed the Peak of Inflated Expectations

By the end of 2024, value will be largely derived from projects based on familiar AI techniques, either stand-alone or in combination with GenAI, that have standardized processes to aid implementation. Rather than focusing solely on GenAI, AI leaders should look to composite AI techniques that combine approaches from innovations at all stages of the Hype Cycle.

However, GenAI still has potential to be a transformational technology with profound business impacts on content discovery, creation, authenticity and regulation, automation of human work, and customer and employee experiences. Still, GenAI faces challenges, including ethical and societal concerns, limited security best practices and nefarious uses like deep fakes and disinformation.

<https://www.gartner.com/en/articles/hype-cycle-for-artificial-intelligence>

My Take

- LLMs are a tool, like any other
 - Very useful for certain applications
 - Good to have multiple, interacting agents
 - Don't get bullied into applying it inappropriately
- LLM hallucinations are a major issue. The more you know about a topic, the weirder the subtle errors seem.
- Transformers are *very* powerful
 - Training data & Compute power represent major challenges
 - GPUs are *very* expensive to rent or buy

The background is a solid dark blue color. Overlaid on this are several translucent, wavy vertical lines that create a sense of depth and movement. A faint, light blue grid pattern is also visible, intersecting with the wavy lines. The overall effect is a modern, abstract design.

Questions