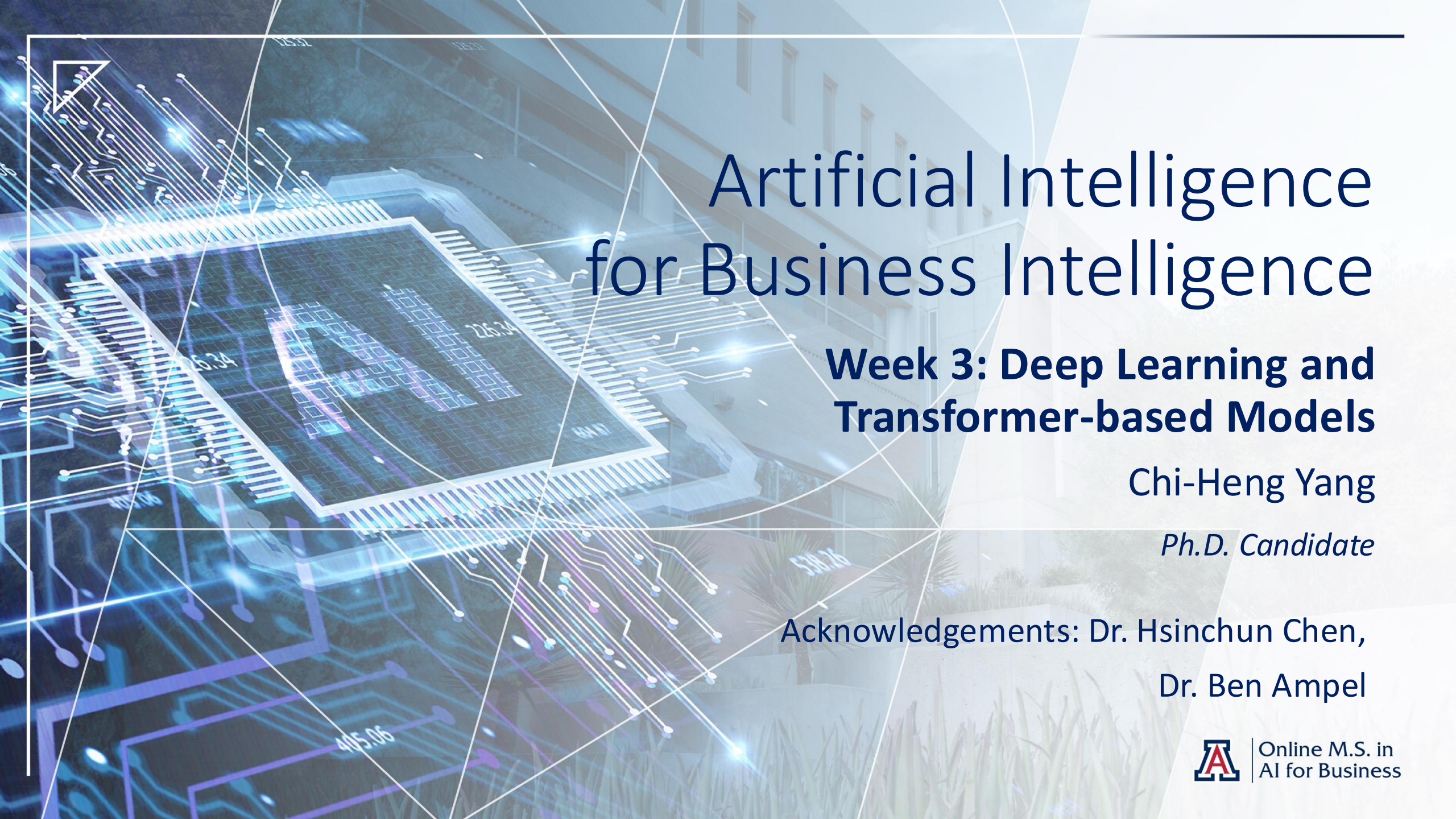


Artificial Intelligence for Business Intelligence

Week 3: Deep Learning and Transformer-based Models

Chi-Heng Yang

Ph.D. Candidate

Acknowledgements: Dr. Hsinchun Chen, Dr. Ben Ampel

Online M.S. in AI for Business

Today's Learning Objectives

- **Module 1: Deep Learning**
  - Understand the fundamental structure of deep learning models.
  - Explain how artificial neural networks learn through the feedforward and backpropagation algorithms.
  - Explain how MLP, CNN, and RNN handle different types of data.

- **Module 2: Transformer-based Models**
  - Understand the attention mechanism in Transformers.
  - Explain how SFT, LoRA, prompt engineering, prompt learning, and RAG help task adaptation.

Module 1: Deep Learning

Deep Learning

Nature, 27 January 2016:

- "…AlphaGo program applied **deep learning** in neural networks (convolutional NN) — brain-inspired programs in which connections between layers of simulated neurons are strengthened through examples and experience."

.S. in AI

Deep Learning – Applications

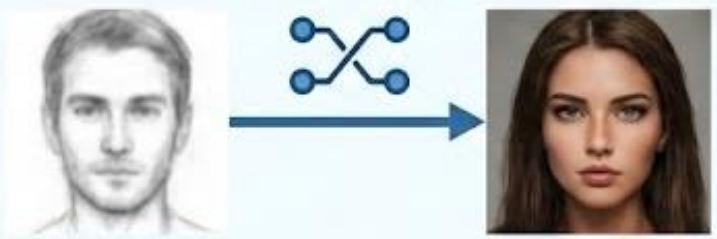
1. Image Classification & Detection (CNNs)

Automated feature learning for vision tasks

2. Natural Language Processing (Transformers/RNNs)

Understanding context & nuance in text/speech (e.g., machine translation).

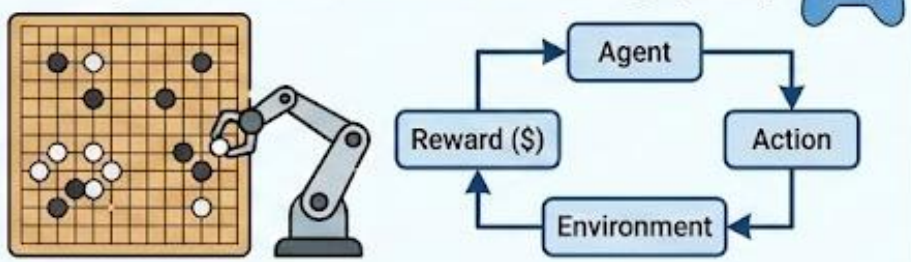
3. Generative Modeling (GANs/Diffusion)



Noise/Sketch — Generated Image

Creating new data from learned patterns (e.g., synthetic art, deepfakes).

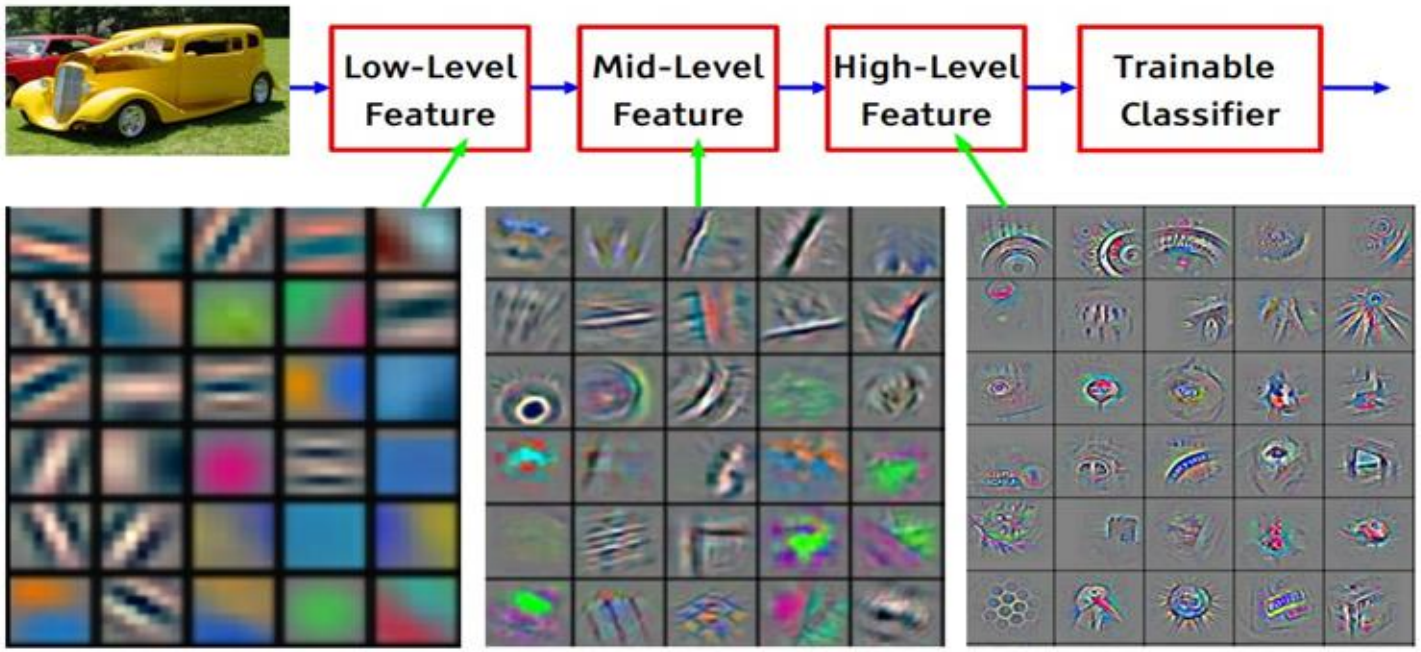
4. Deep Reinforcement Learning (DRL)

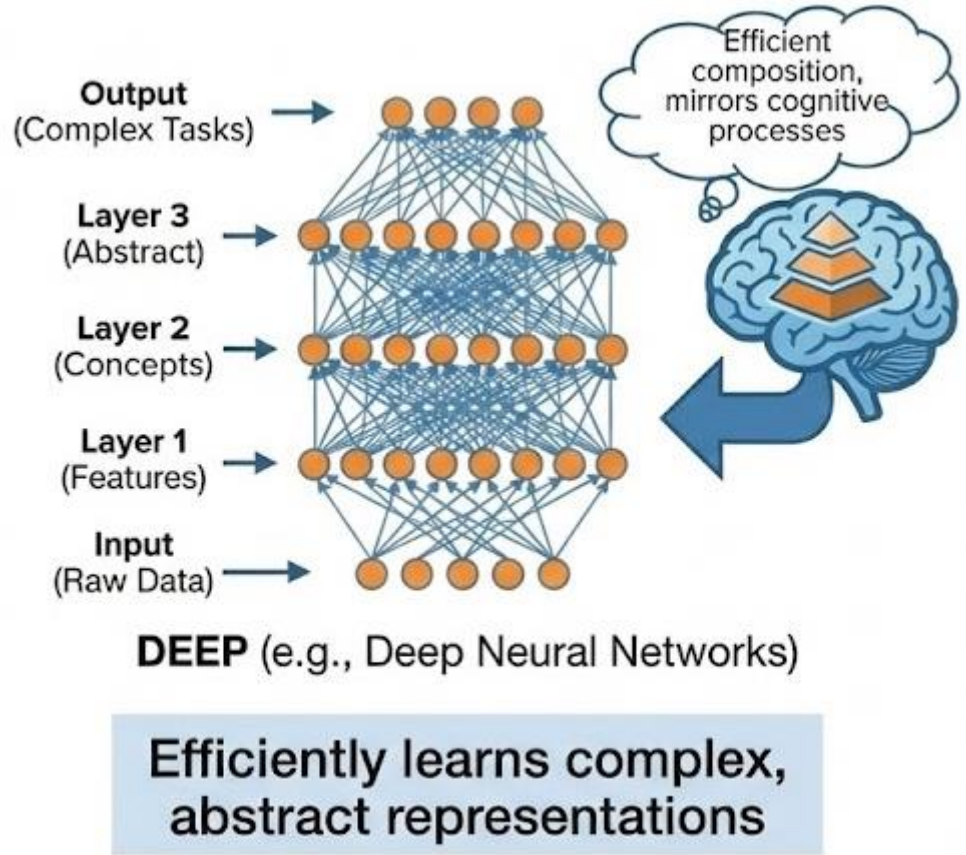


Learning through trial & error to achieve goals (e.g., game playing, robotics).

Deep Learning – Why deep architecture?

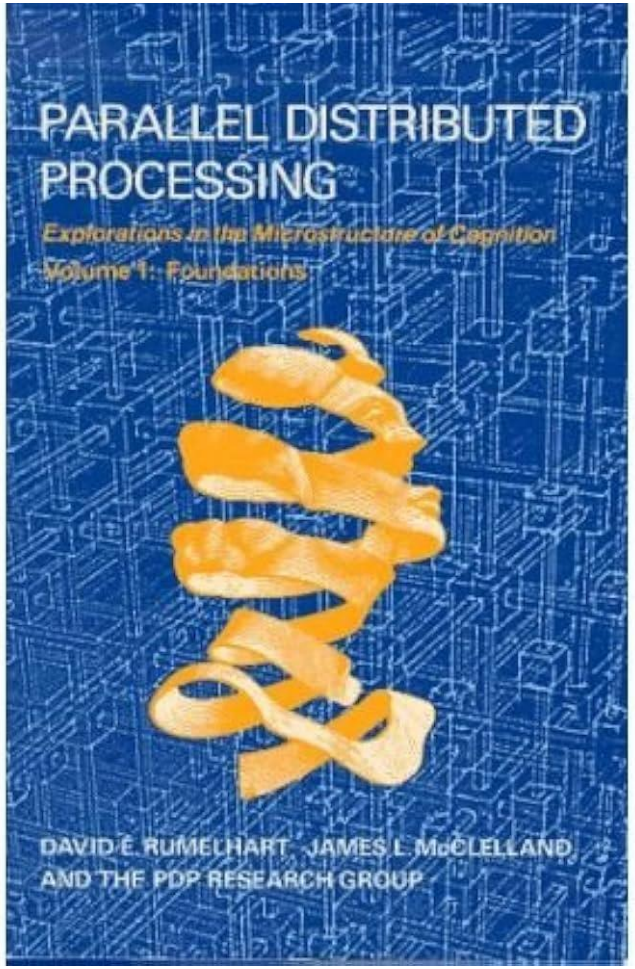
- Deep Learning = Learning Hierarchical Representations





Artificial Neural Networks – History

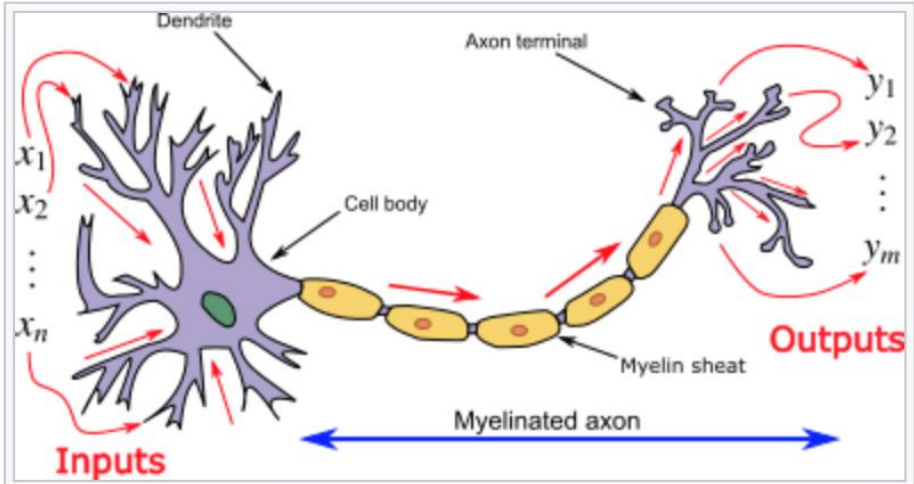
- Inspired by biological neural networks (animal brains), **Artificial Neural Networks (ANN)** was originated in 1950 to simulate intelligence through simple units and basic patterns.
- **John Hopfield** introduced **associative memory** in **1982**, using physics-based loops to store and retrieve information (become the foundation of recurrent neural networks).
- **PDP Research Group** at UCSD formalize the ANN concept in **1986** by modeling intelligence as distributed patterns across network connections (Rumelhart et al. 1986).
  - **Geoffrey Hinton** pioneered the **backpropagation** algorithm to train these patterns.
- **Hopfield and Hinton** were jointly awarded the **2024 Nobel Prize in Physics** for these foundational contributions to neural computation.



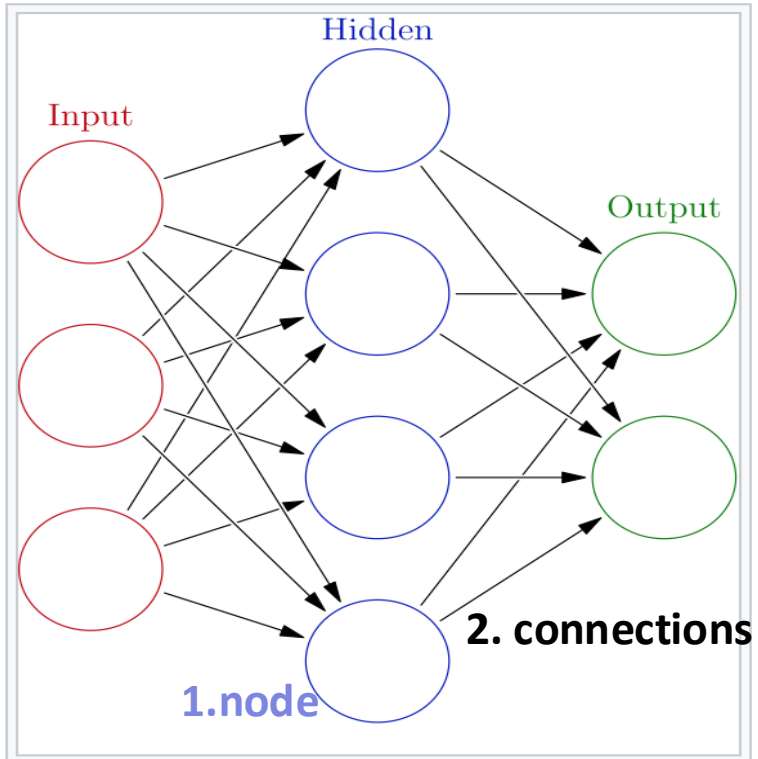
Reference: Rumelhart, D. E., McClelland, J. L., & PDP Research Group. (1986). Parallel distributed processing: Explorations in the microstructure of cognition: Foundations. The MIT press.

Artificial Neural Networks - Architecture

- **Key Components in ANN**:
  1. **Nodes:** Artificial *neurons* (processing units)
  2. **Connections:** Synapses (signal transmission)



Neuron and myelinated axon, with signal flow from inputs at dendrites to outputs at axon terminals



An artificial neural network is an interconnected group of nodes, inspired by a simplification of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

Artificial Neural Networks - Neurons

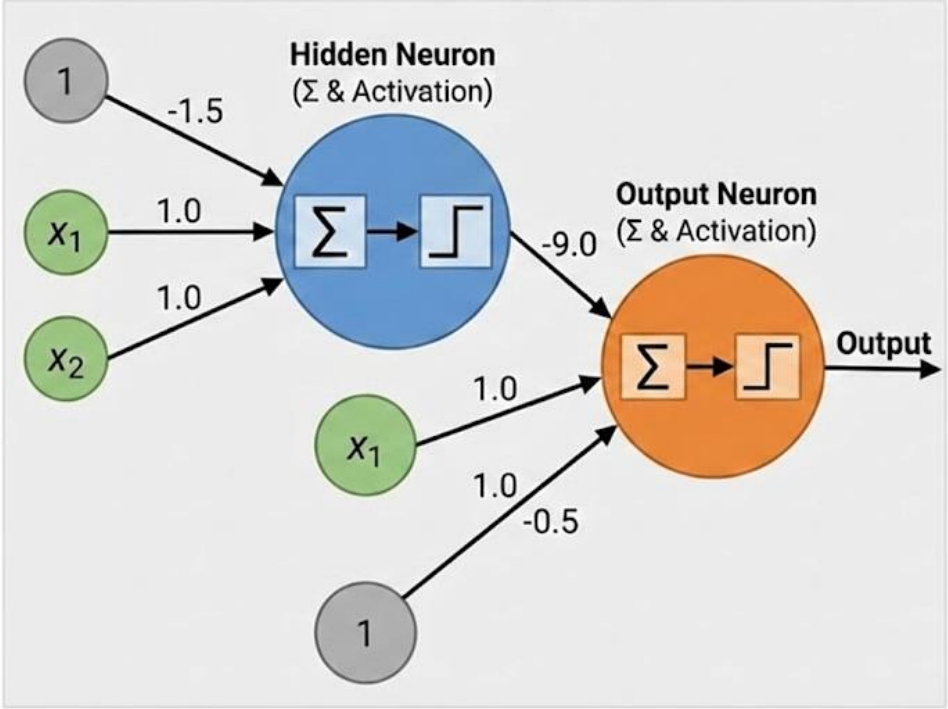
- **A neuron** is a computational unit in the neural network that exchanges messages with other.

- Two components exist in each neuron:
  1. **Weighted sum**: Multiply each input by its weight and add them up

     $$z = (1 \times -1.5) + (x_1 \times 1.0) + (x_2 \times 1.0)$$

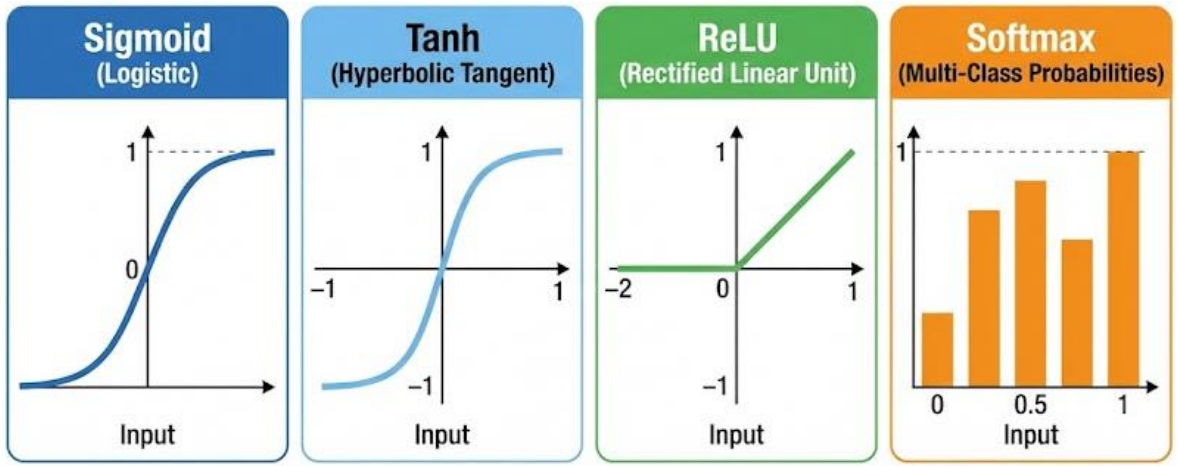
  2. **Activation function $f(z)$**: Add non-linearity to the weighted sum.

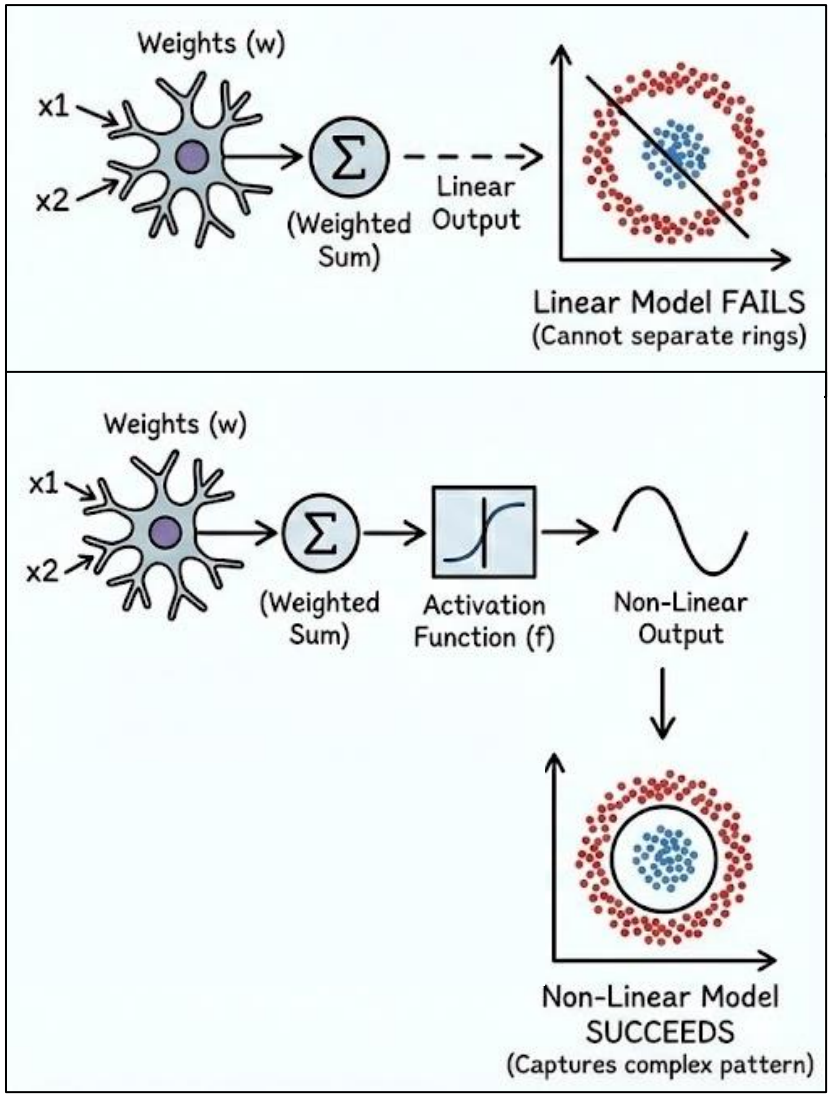
- The result after these two components is sent to the next neuron as the input through the connections.



Artificial Neural Networks - Activation Function

- Weighted sum alone means learning only linear rules.
- But not all real-world problems are linear.
  - As a customer gets older, they do not always become more likely to churn.
- Activation functions add "the bend," allowing the network to learn complex, non-linear shapes.





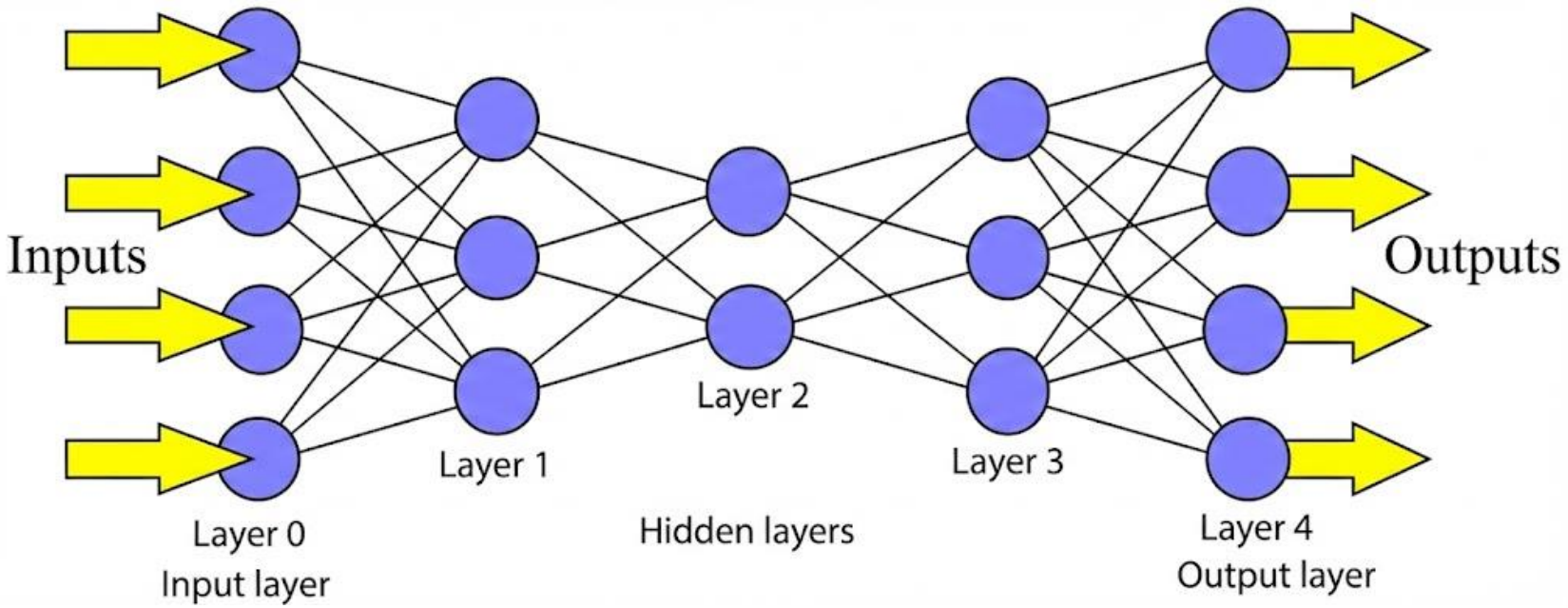
Artificial Neural Networks - Activation Function

- Activation functions are adopted in:
  - **Hidden layer** to help the model learn complex, non-linear patterns efficiently.
  - **Output layer** to match the format of the answer we need

Suitable Layer	Function	Formula	Output Value Range	Characteristics	Common Tasks
Output	Linear	x	$(-\infty, \infty)$	Passes the input directly to the output without changes.	Regression & Time Series
	Sigmoid	$\frac{1}{1+e^{-x}}$	$(0, 1)$	Perfect for predicting a single probability	Binary classification
	Softmax	$\frac{e^{x_i}}{\Sigma_j e^{x_j}}$	$(0, 1)$	Ensures all output probabilities sum up to exactly 1.0.	Multi-class classification
Hidden	ReLU	$\max(0, x)$	$[0, \infty)$	Extremely fast to compute and helps deep networks learn efficiently	N/A
	Tanh	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$(-1, 1)$	Zero-centered; usually performs better than Sigmoid in hidden layers.	N/A

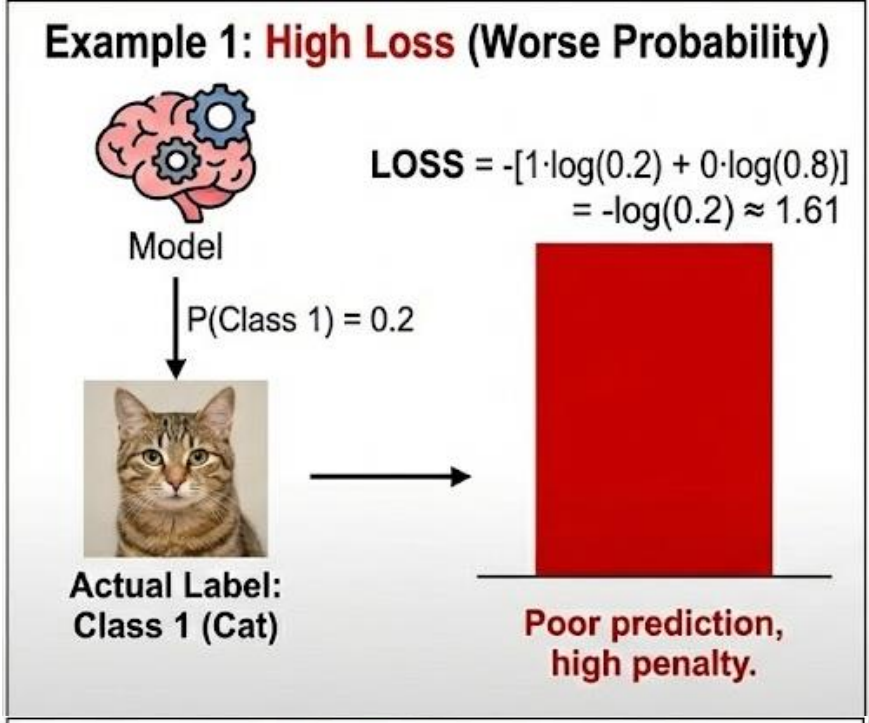
Artificial Neural Networks

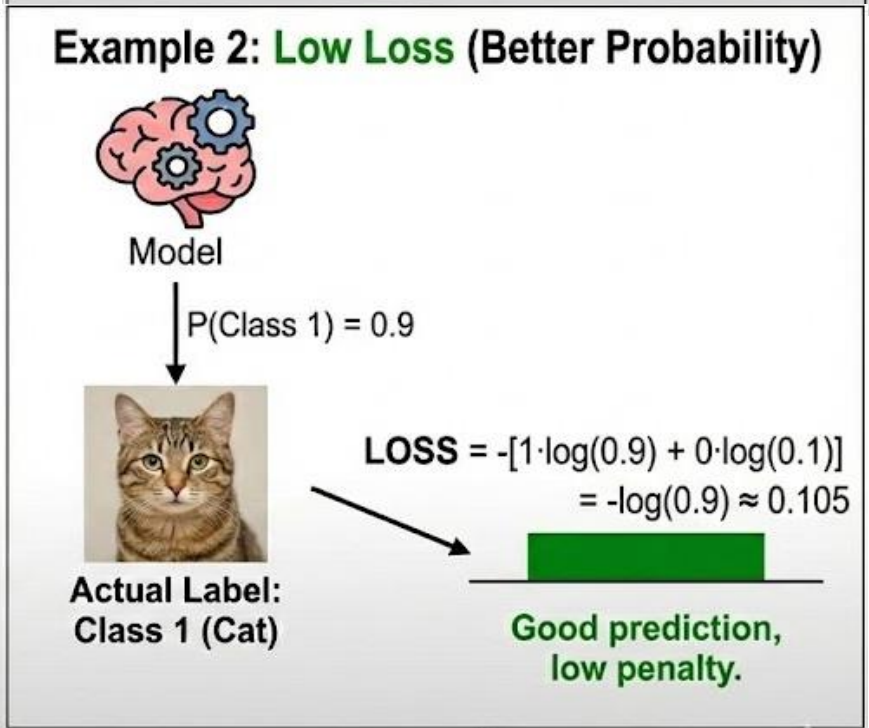
- **Feedforward algorithm**
  - Randomly initialize the connection weights.
  - Activate the neurons from the input to the hidden, then to the output units.



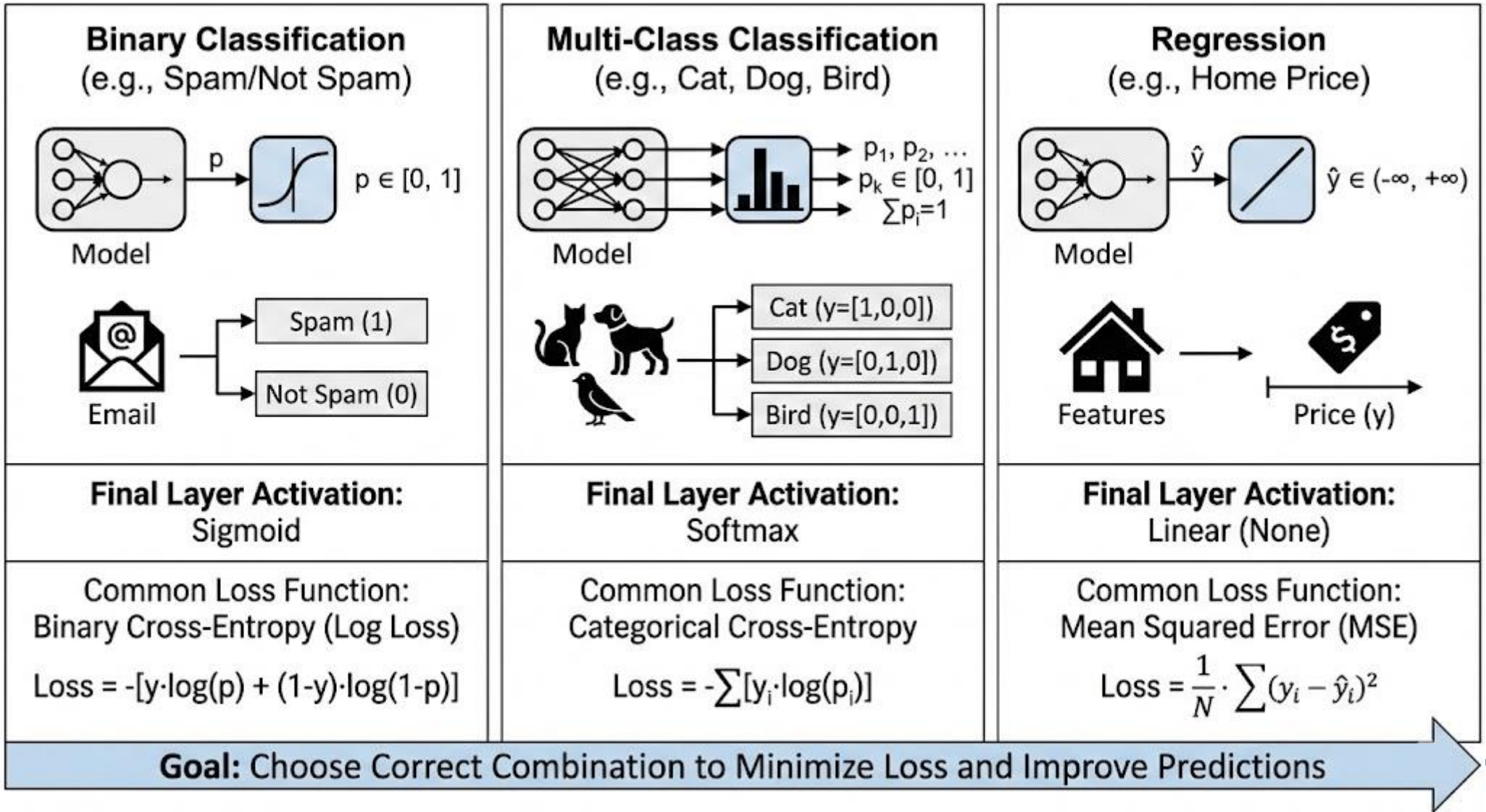
Artificial Neural Networks - Loss Function

- Once we have an output, we need to know how good it is.
- Loss function provides a “measurement” of how good the output is.
- For a classification task of cat (class 1) or not cat (class 0):
  - P(cat) = 0.2, P(not cat) = 0.8 ➔ not good
  - P(cat) = 0.9, P(not cat) = 0.1 ➔ good





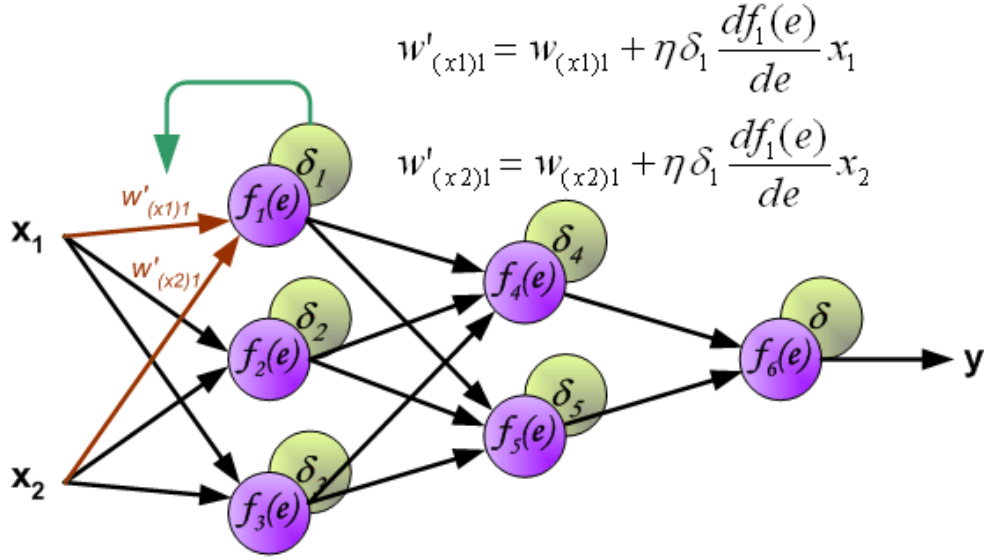
Artificial Neural Networks - Loss Function



.S. in AI

Artificial Neural Networks

- How to learn based on this measurement?
- **Backpropagation algorithm** providing the mathematical solution to train these interconnected layers (Rumelhart et al. 1986):
  - Calculate total error at the output with the loss function, f(e).
  - Then calculate contributions to error, δ_n, at each step by going backwards.
  - Correct errors with connection weights by going backwards with **gradient descent**.

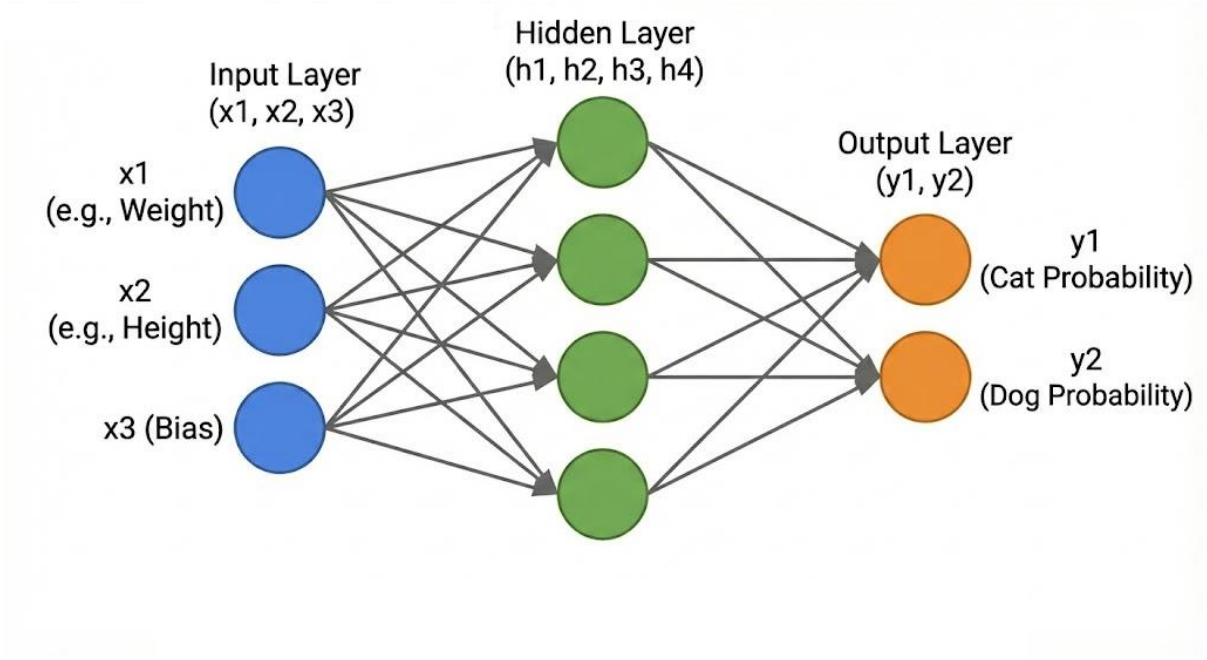


Reference: Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

Artificial Neural Networks – MLP

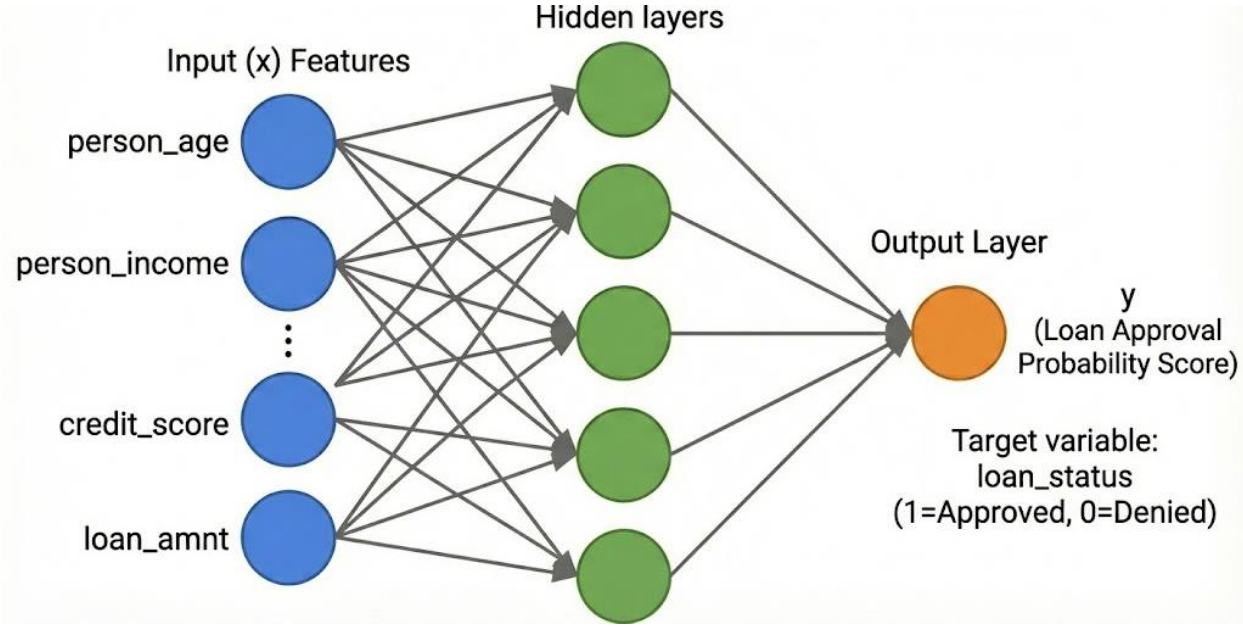
- **Multi-Layer Perceptron (MLP)** is the standard "Deep" realization of an ANN.

- It is organized into hierarchical layers:
  - **Input Layer**: Passes raw features ($x_1, x_2, \ldots$) into the network.
  - **Hidden Layers:** The "engine" where neurons compose simpler concepts into abstract features.
  - **Output Layer**: Aggregates final signals to produce a prediction (y).



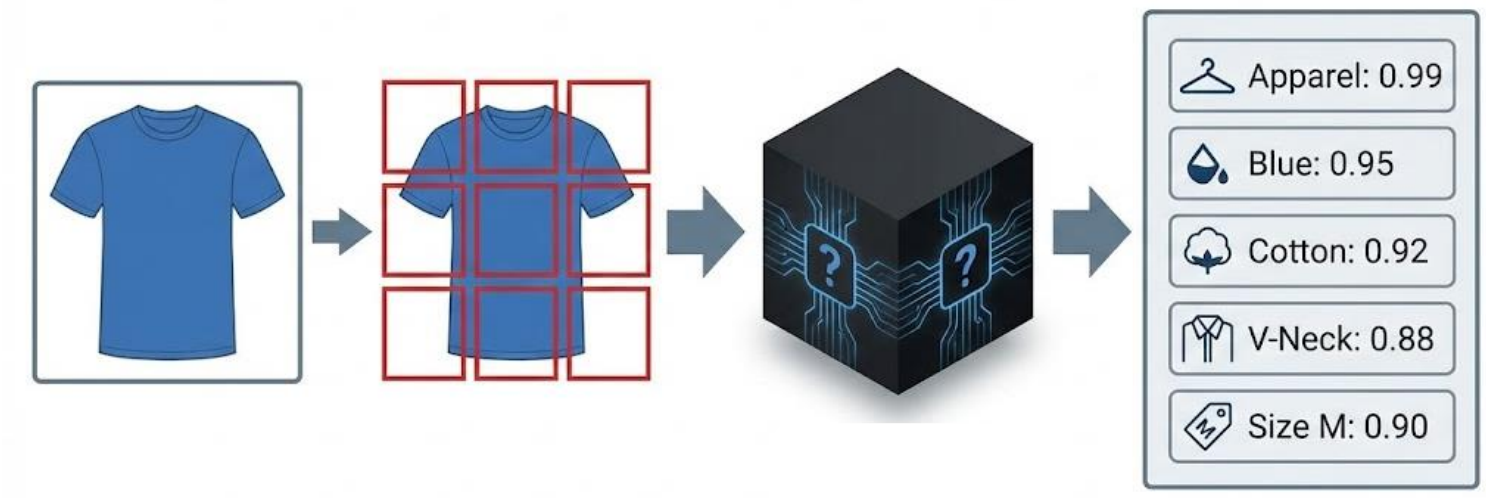
Artificial Neural Networks – Python Example

- **Python example**: loan application classification (revisited) using MLP
  - Refer to supplemental materials
- **Data**: past loan application records with customer information and decisions
- **Key Packages:** PyTorch



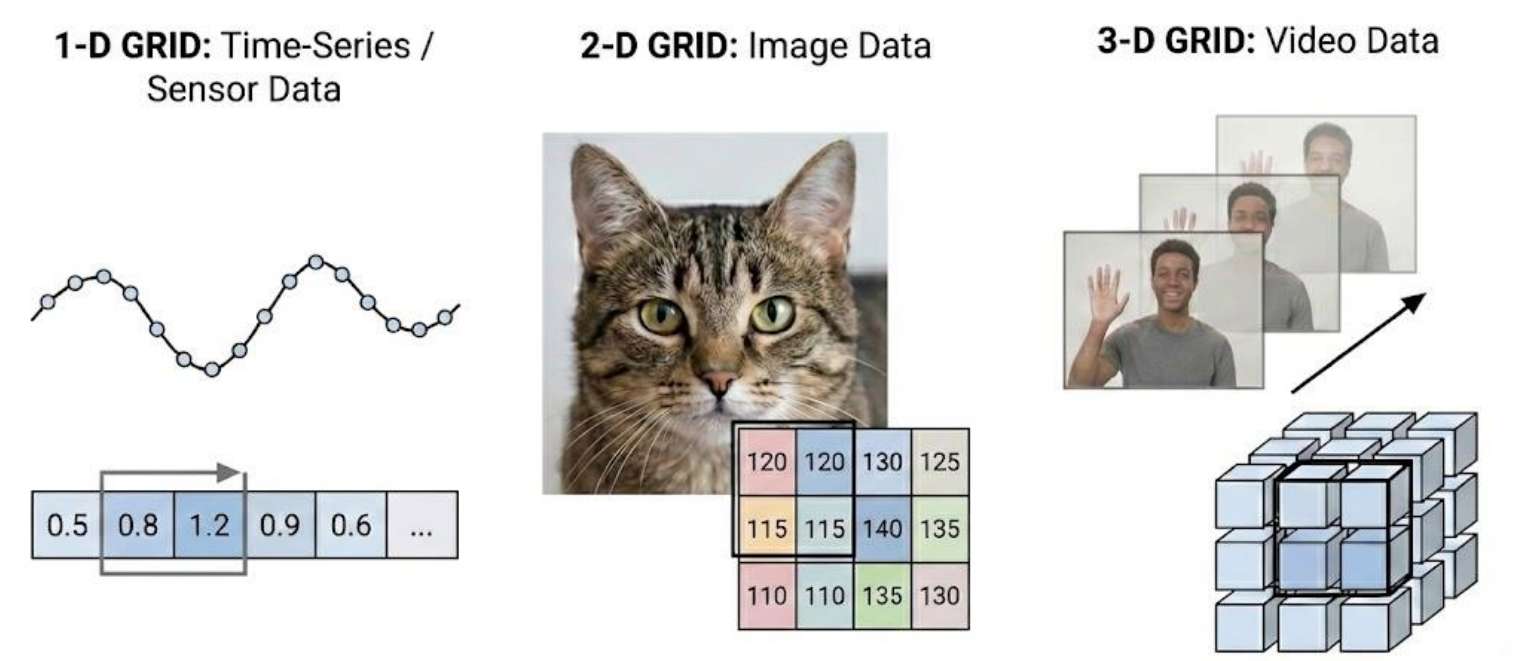
Convolutional Neural Networks (CNN)

- MLP works on classification with numerical or categorical features (e.g., income, age, gender).
- What if we want to classify images?
  - Product auto-tagging
  - Product defect detection



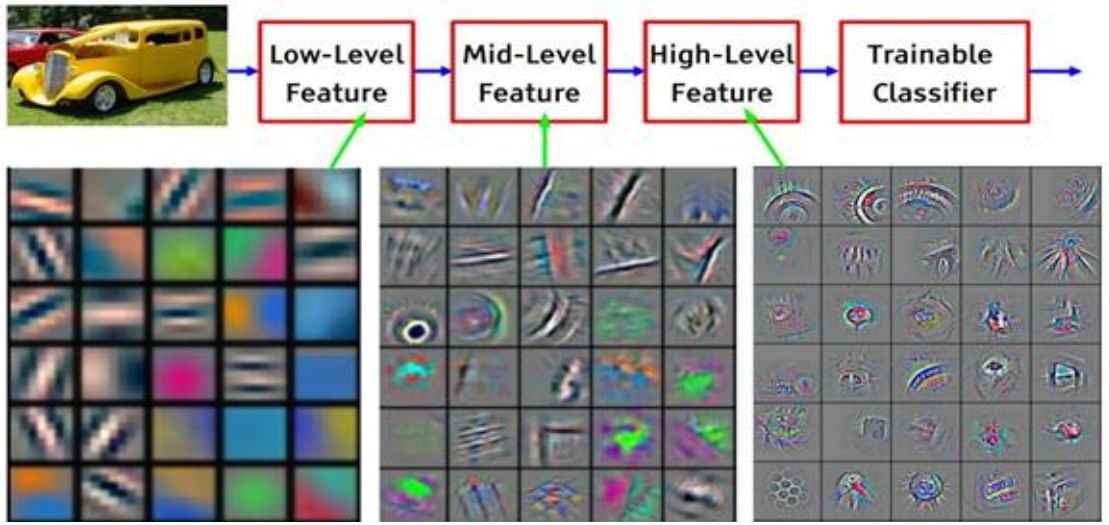
CNN

- **Convolutional Neural Networks** (or Convolutional Networks, or CNNs, or ConvNets) can process data with a **grid-like or array** topology
  - 1-D grid: time-series data, sensor signal data
  - 2-D grid: image data
  - 3-D grid: video data



CNN

- **Intuition**: Neural network with specialized connectivity structure
  - Stacking multiple layers of feature extractors, low-level layers extract local features, and high-level layers learn global patterns.
- The 2012 **AlexNet** model proved the superiority of deep CNNs by winning the **ImageNet** competition (Krizhevsky, Sutskever, & Hinton, 2012).
  - This breakthrough effectively ended the industry's reliance on manual feature engineering.



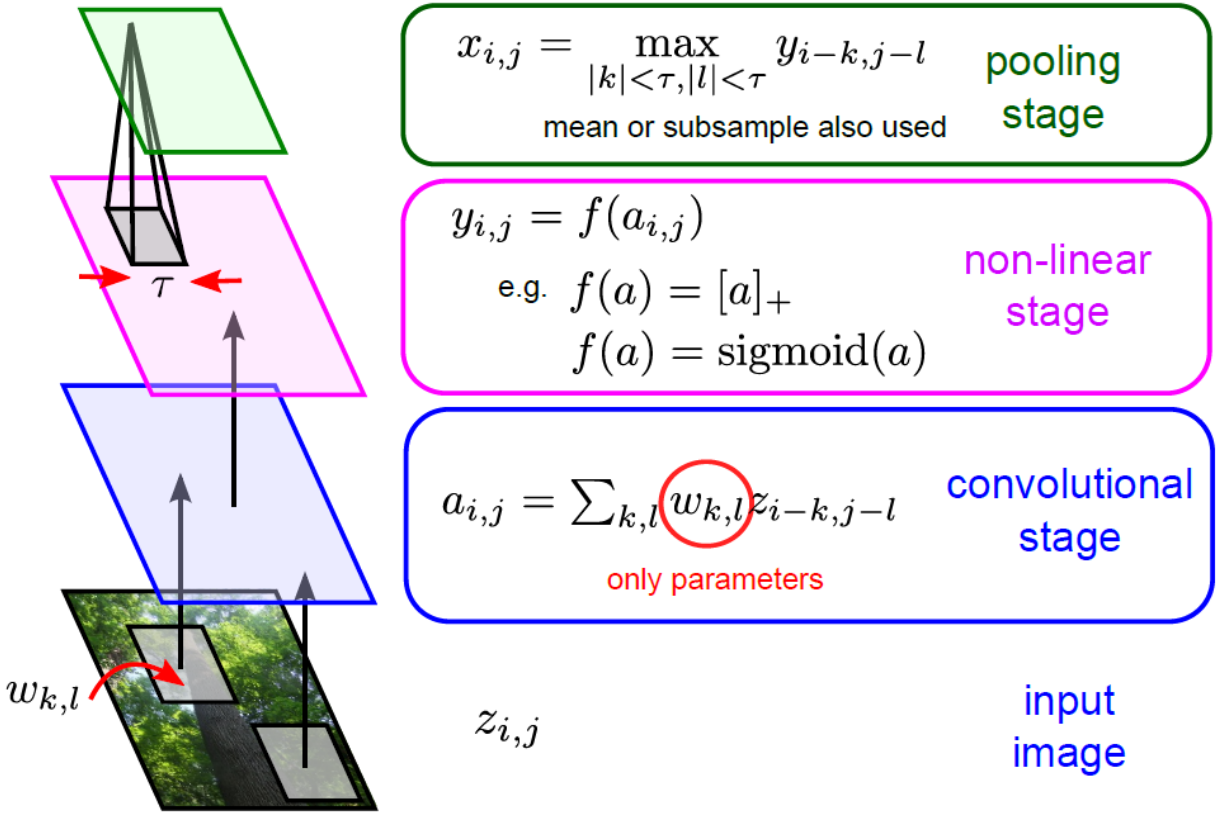
Feature Visualization of CNN trained on ImageNet (Zeiler & Fergus 2014)

References:

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.
- Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Cham: Springer International Publishing.

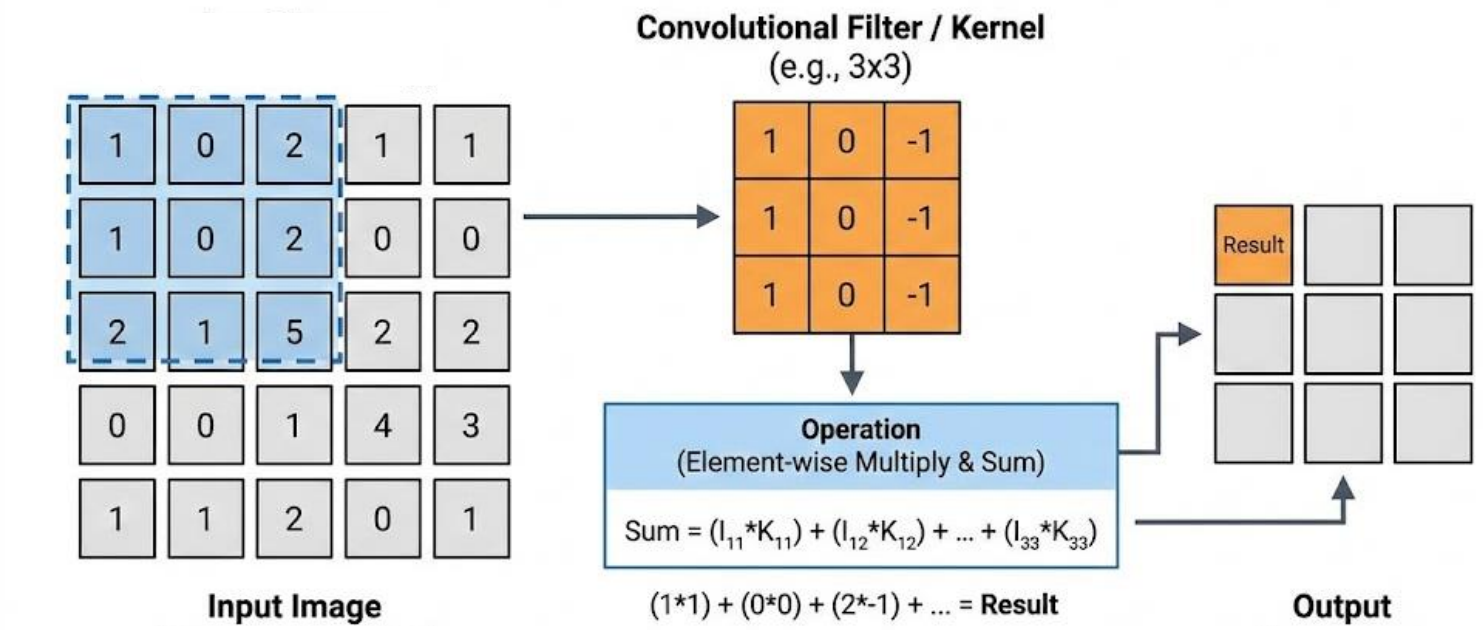
CNN – Architecture

- Three major stages:
  - **Convolutional**: detecting local features through filters (discrete convolution)
  - **Non-linear**: normalization via Rectified Linear Unit (ReLU)
  - **Pooling**: merging similar features



CNN – Architecture (Convolutional Stage)

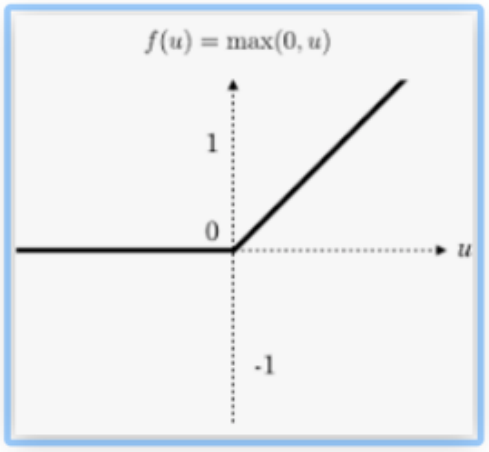
- **Goal**: Extract patterns from the input image
- **Mechanism**: Slide a **filter** across the input data to perform element-wise multiplication and summation.

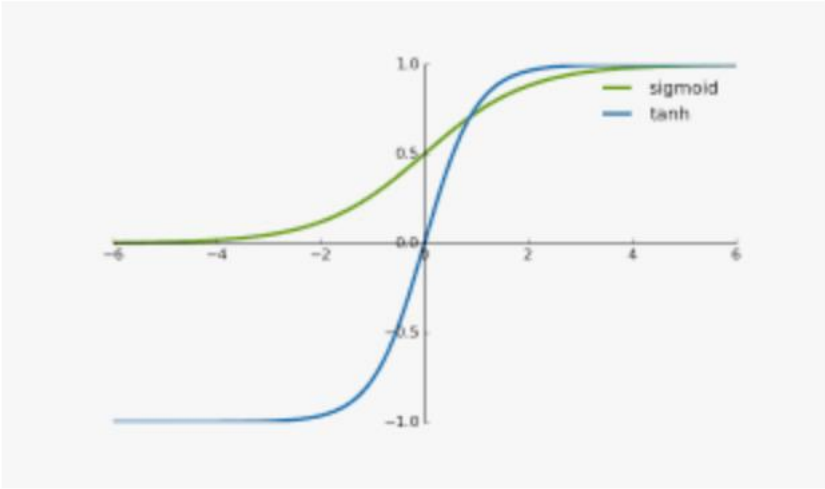


CNN – Architecture (Non-Linear Stage)

- **Goal**: Increase the nonlinearity of the entire architecture

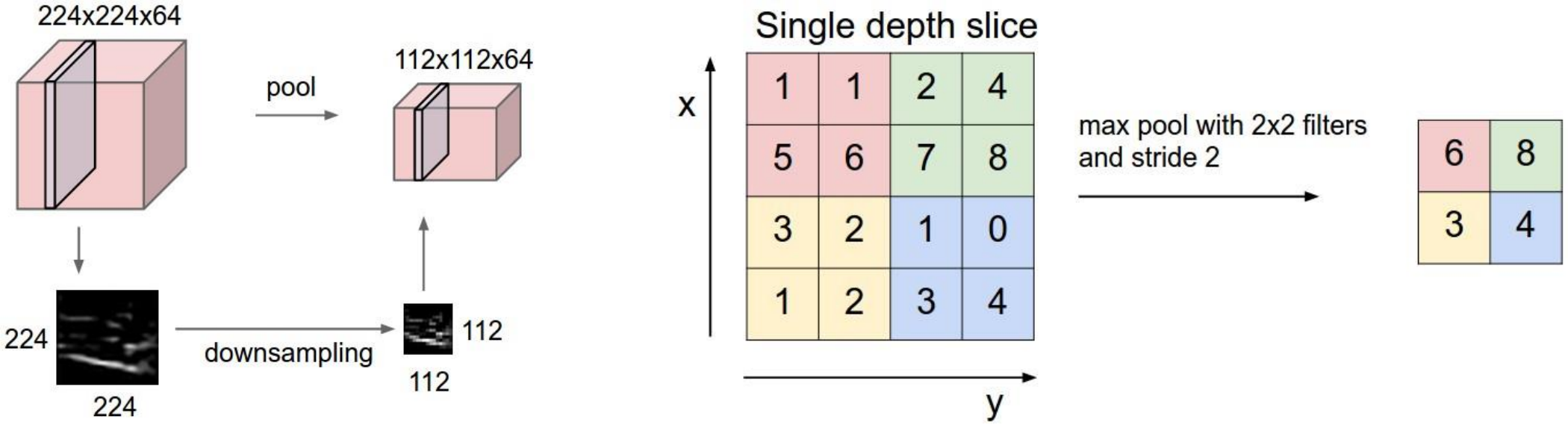
- **Mechanism**: A layer of neurons that applies non-linear activation functions





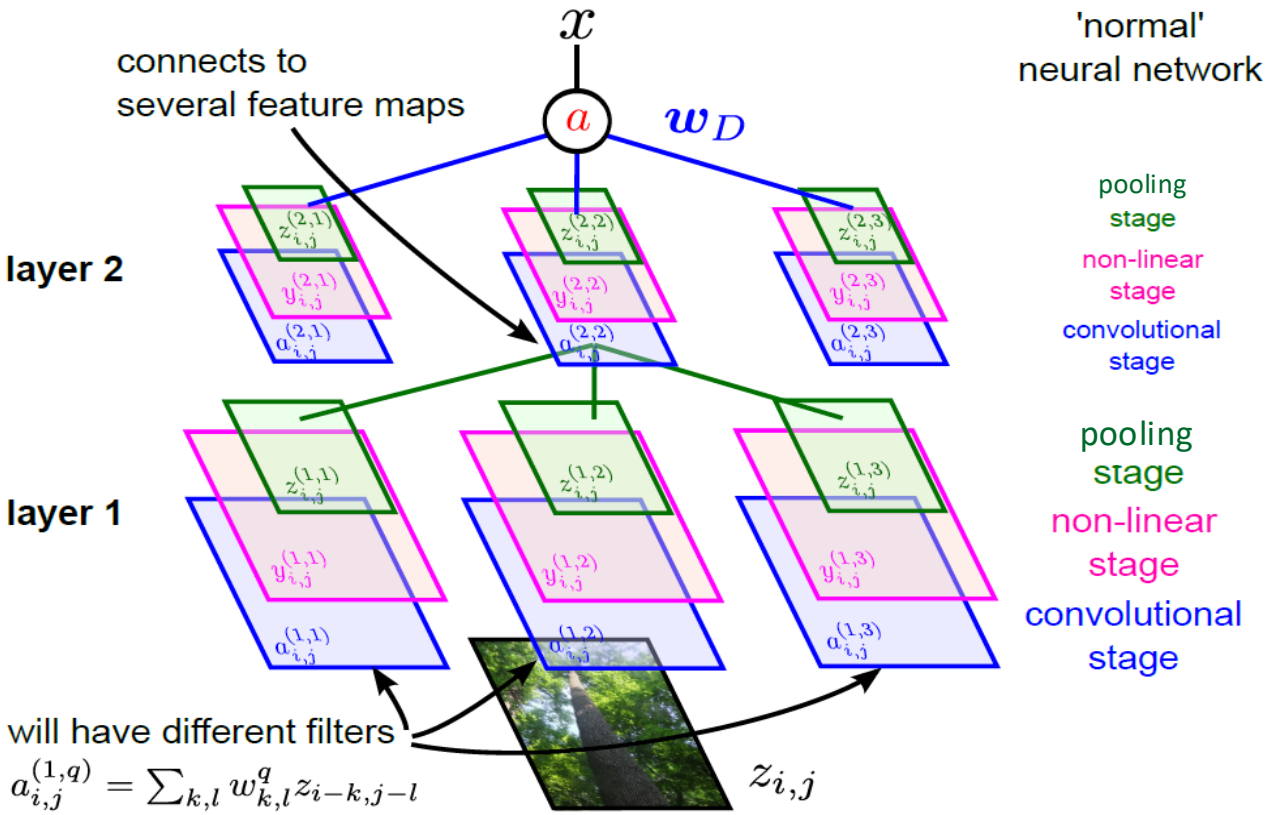
CNN – Architecture (Pooling Stage)

- **Goal**: Reduce the size of the representation (to reduce the amount of parameters and computation in the network).
- **Mechanism**:
  - Partition the input image into a set of non-overlapping rectangles for each such sub-region
  - Outputs the maximum value of the features in that region.



CNN – Full Architecture

- Multiple sets of these three stages can appear in a CNN design.
  - Input -> Conv. -> Non-linear -> Pooling -> Conv. -> Non-linear -> Pooling -> …-> Output

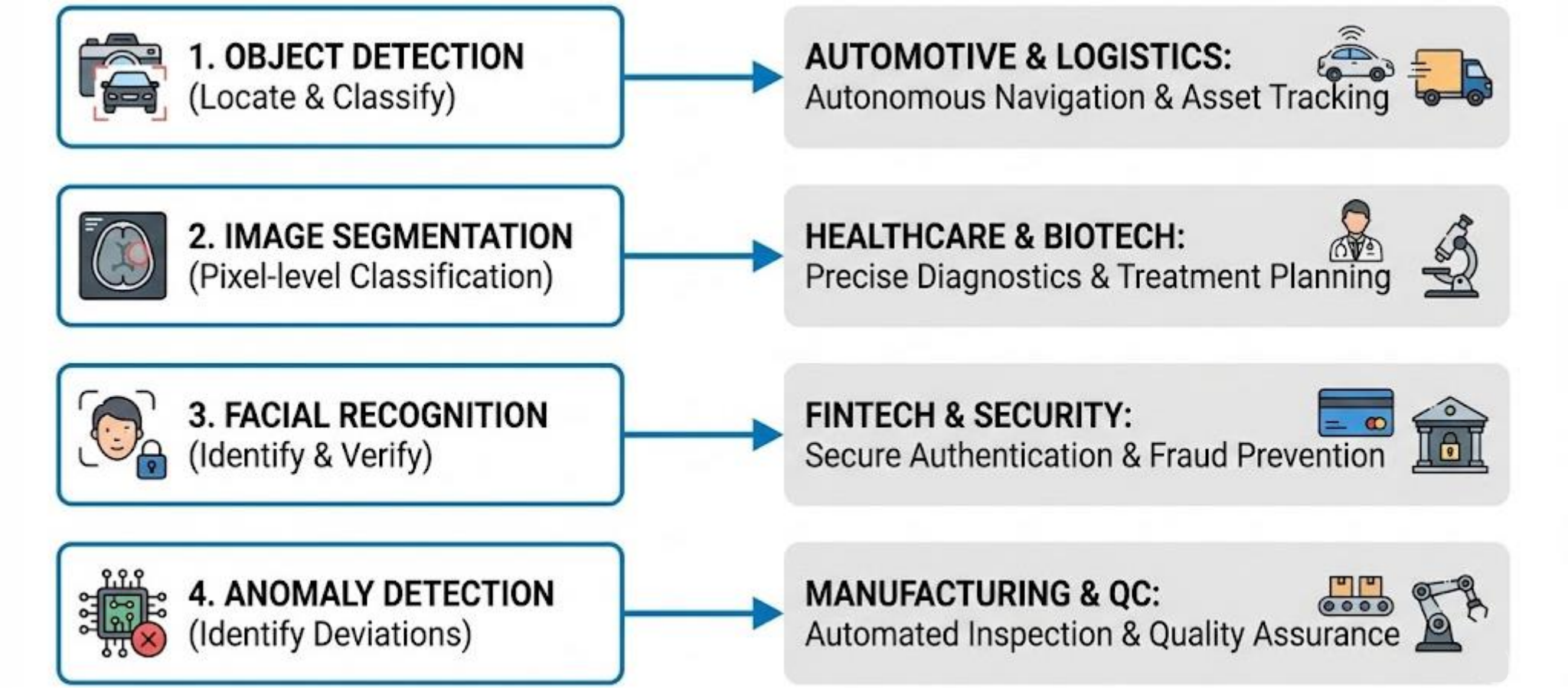


CNN – Full Architecture

- Recent CNN architectures have 10-20 such sets.

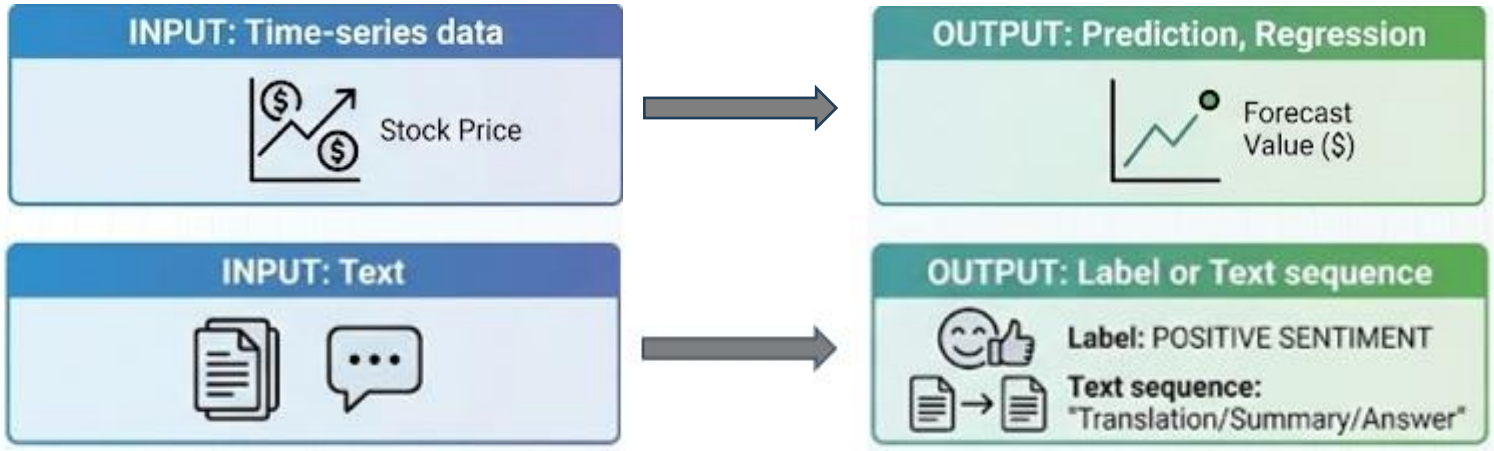
- After a few sets, the output is typically sent to one or two **fully connected** layers.
  - The typical activation function is the **sigmoid** function.
  - Output is typically class (classification)
    or real number (regression).

CNN – Applications



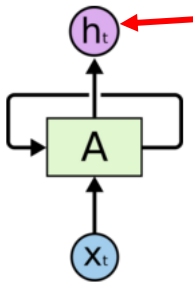
Recurrent Neural Networks (RNN)

- So far, the data we are dealing with does not have orders (e.g., customers, images).
- However, data from business operations can be sequential.
  - The data points have an order or timing (e.g., stock price, words in a sentence)
- We need a method that allows information to persist after processing each data point.



Recurrent Neural Networks

- **Recurrent Neural Networks (RNN)** are networks with loops, allowing information to persist.



Output is to predict a vector h_t, where $output\ y_t = \varphi(h_t)$ at some time steps (t)

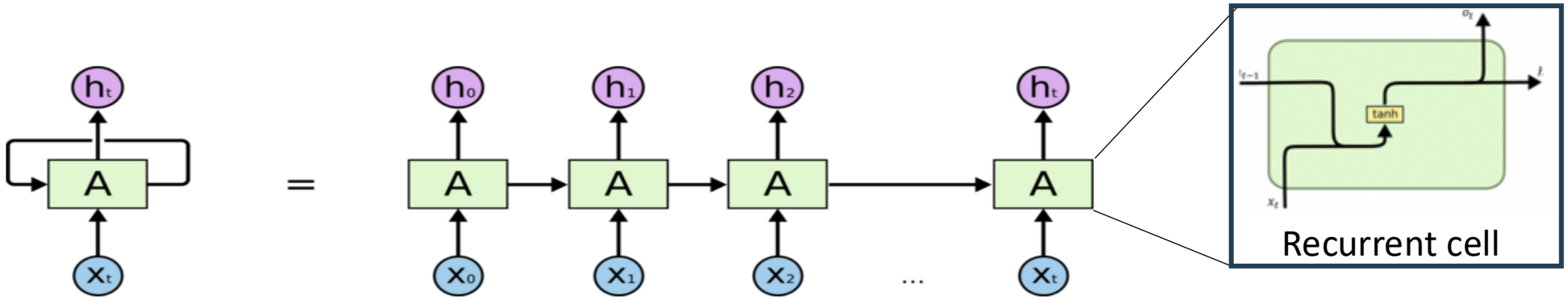
- The foundation for vanilla RNNs was established by introducing recurrent loops to process sequential information (Elman 1990, Hopfield 1982, Jordan 1997;).

References:

- Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554-2558.
- Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in psychology* (Vol. 121, pp. 471-495). North-Holland.

Recurrent Neural Networks

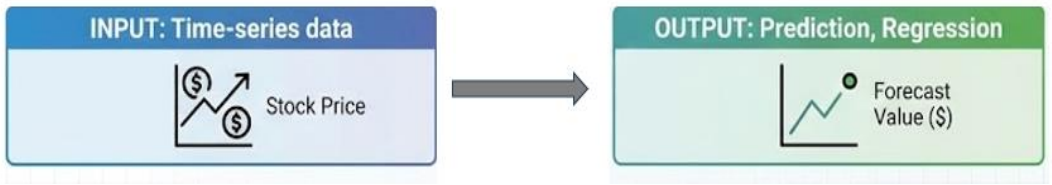
- RNN can be thought of as multiple copies of the same network, each passing a message to a successor.

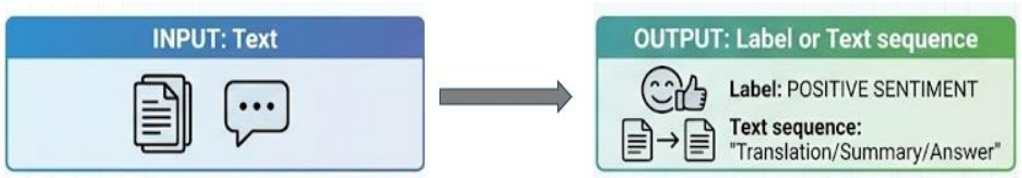


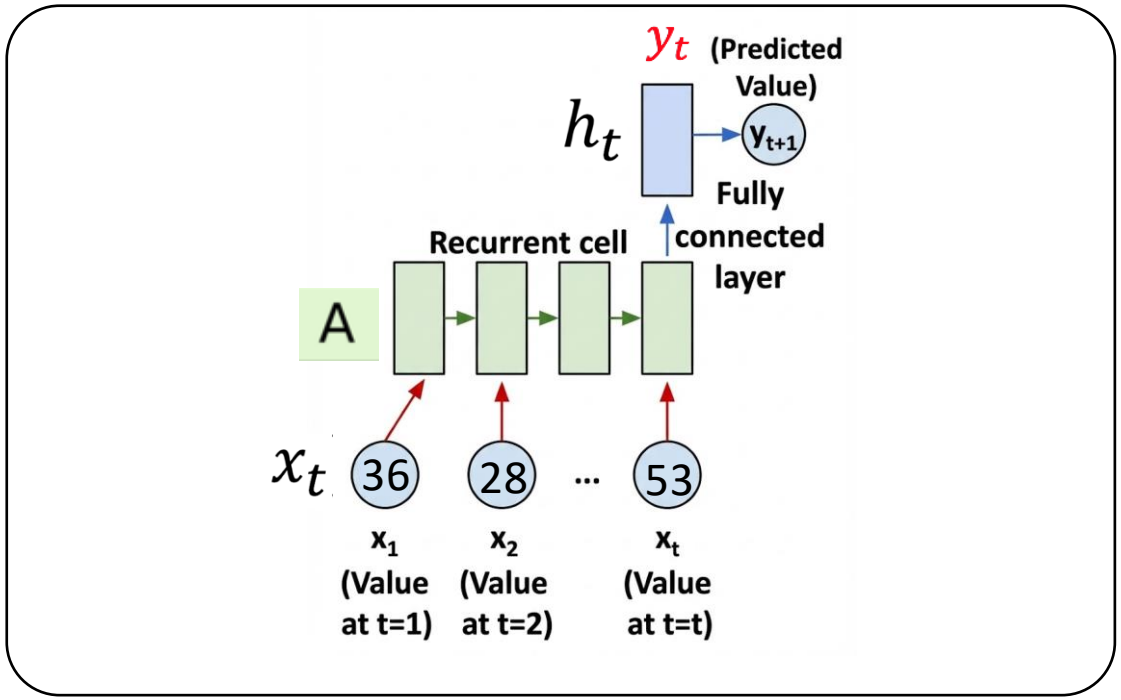
Recurrent Neural Networks

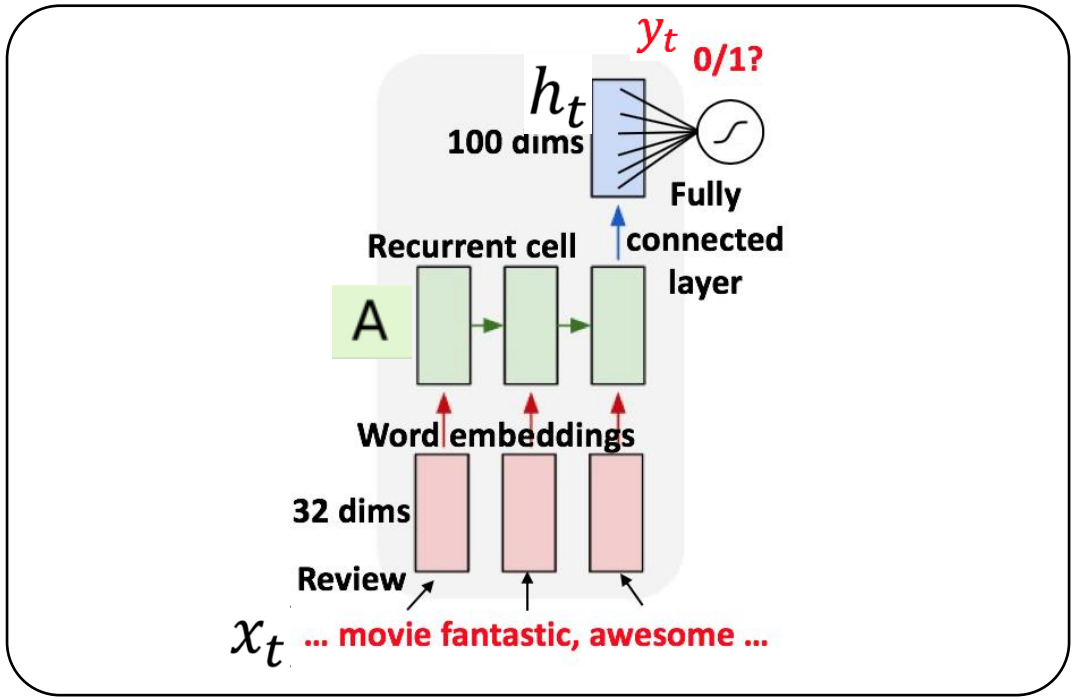
- The recurrent structure of RNNs exhibits the following characteristics:
  - Specialized for processing a sequence of values $x^{(1)}, \ldots, x^{(\tau)}$
    - Each value $x^{(i)}$ is processed with the **same network *A* that preserves past information**
  - Can scale to much **longer sequences.**
    - Reusing network ***A*** reduces the required amount of parameters in the network
  - Can process **variable-length sequences.**
    - The network complexity does not vary when the input length change.

Recurrent Neural Networks





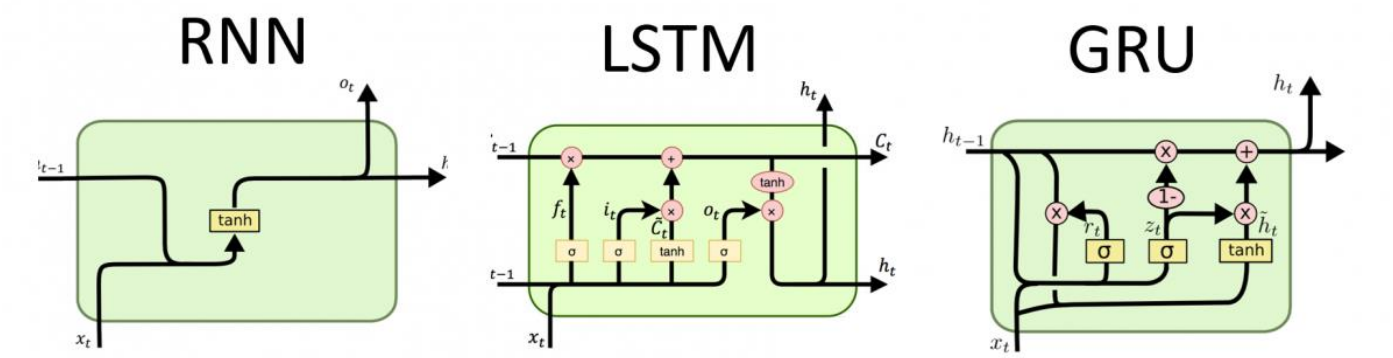




Networks with Memory

- Vanilla RNN operates in a “multiplicative” way.
  - **Vanishing gradient:** Signals fade away, preventing early layers from learning.
  - **Exploding gradient:** Signals grow chaotic, causing the model to crash.
  - Failure to capture long-term dependencies

- Two recurrent cell designs in an “additive” way were proposed and widely adopted:
  - **Long Short-Term Memory (LSTM)** (Hochreiter and Schmidhuber, 1997)
  - **Gated Recurrent Unit (GRU)** (Cho et al. 2014)



References:

- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (EMNLP) (pp. 1724-1734). Doha, Qatar: Association for Computational Linguistics..
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Module 2: Transformer-based Models

Transformer

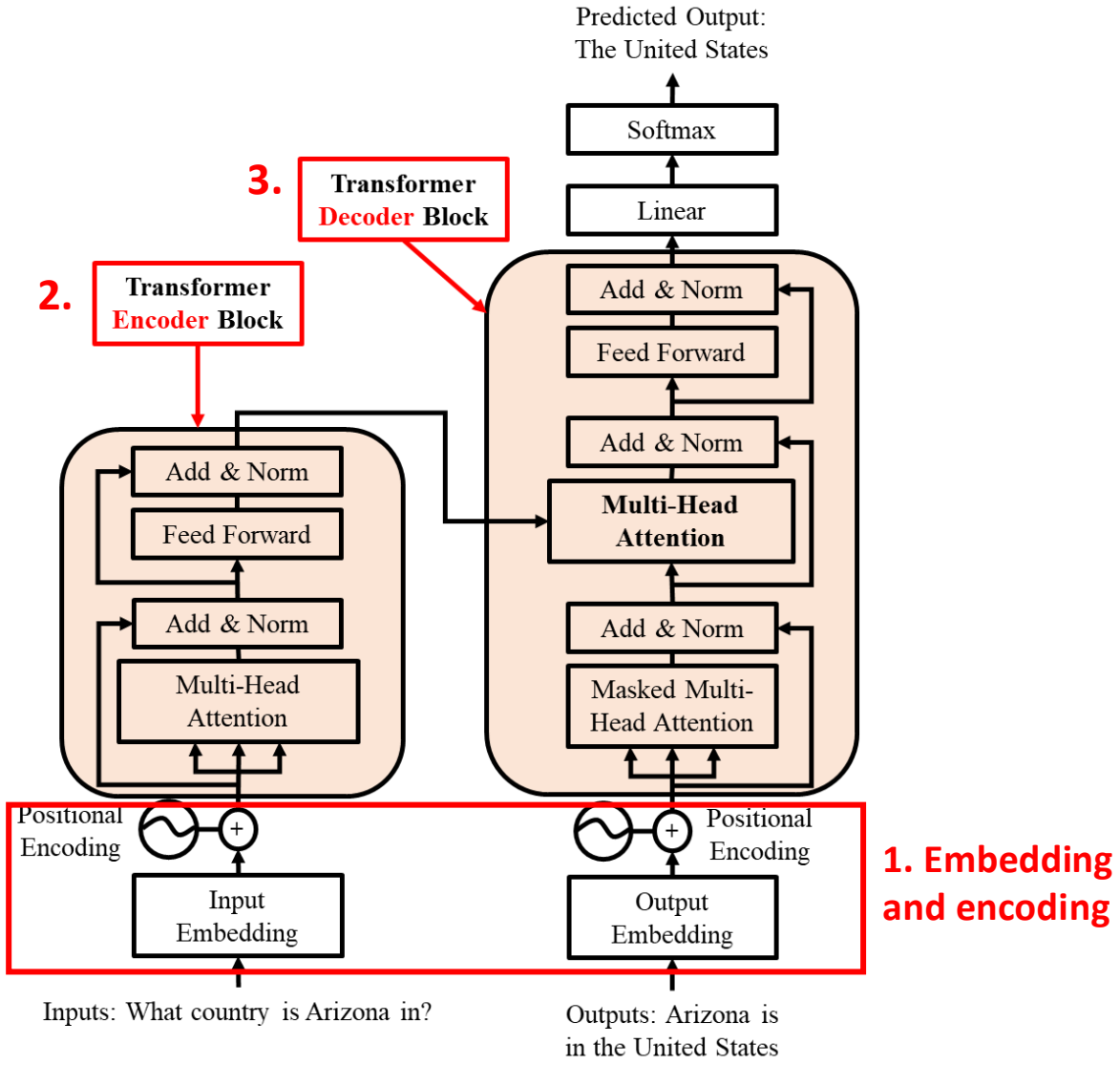
- Prior to 2017, **RNN** architectures (LSTM, GRU) had seen **state-of-the-art** performance for various **natural language processing** (NLP) tasks.
- However, **sequential computation** became **infeasible** for long sequences and massive datasets.
  - Reading inputs sequentially **prevents parallelization** in training and **forgets** past information.
- Instead of sequentially reading input, can we read the input all at once?
- In 2017, the Google Brain research team noted the success of **attention mechanisms** and built a new model known as the **Transformer** (Vaswani et al. 2017).
  - This model has no convolutions or recurrence.
  - The Transformer implements **attention mechanisms** to achieve state-of-the-art NLP performance.

Reference: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

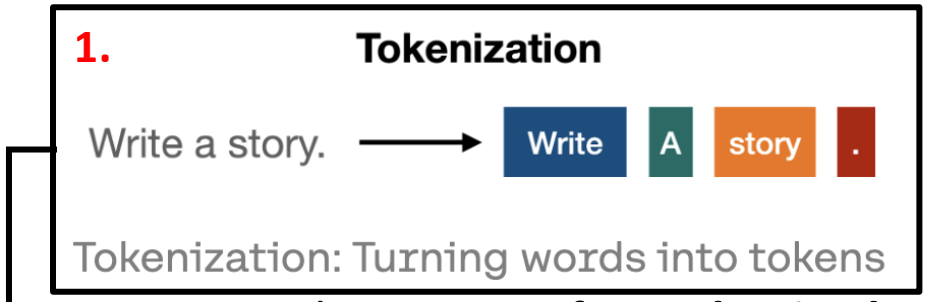
Transformer - Overview

- **Transformer** is designed to process every part of input sequence simultaneously.

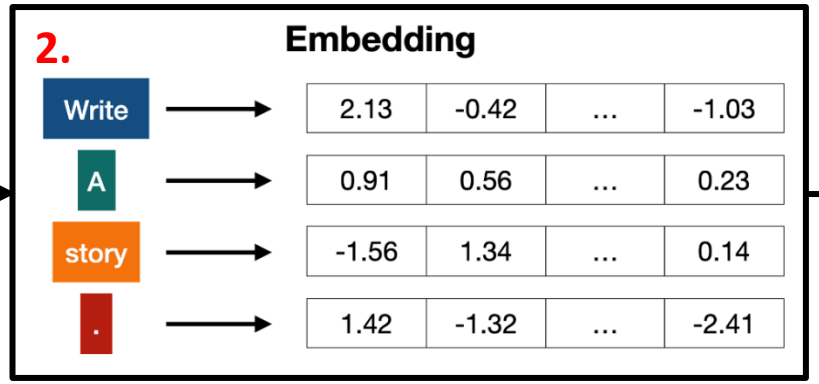
- Three major components:
  1. **Embedding and encoding**: transform textual input into numerical format
  2. **Encoder Block**: create latent representation of text
  3. **Decoder Block**: decode latent representation for task output



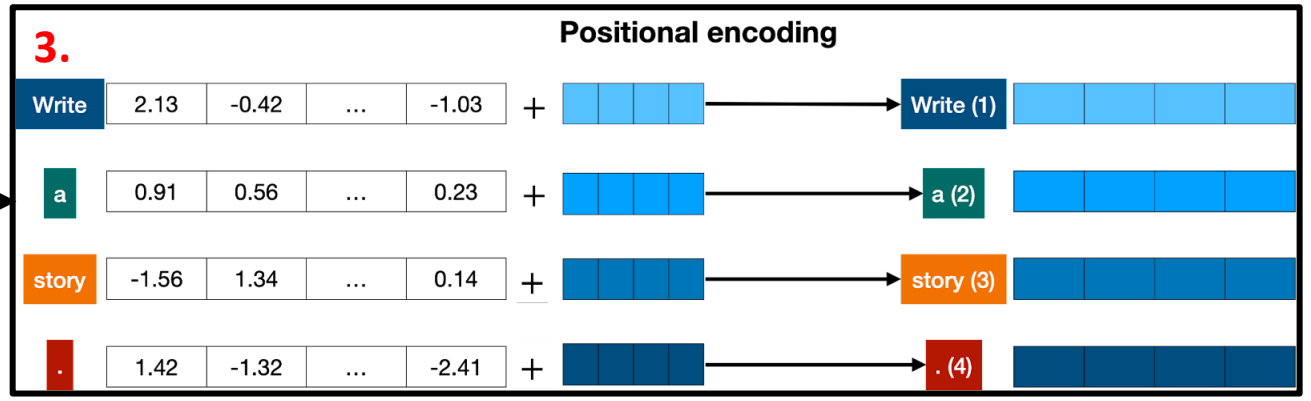
Transformer – Embedding and Encoding



1. Textual inputs are first **tokenized**, transforming sentences into individual tokens.



2. Tokens are **transformed** into **vectors** based on co-occurrence statistics (e.g., Word2Vec).

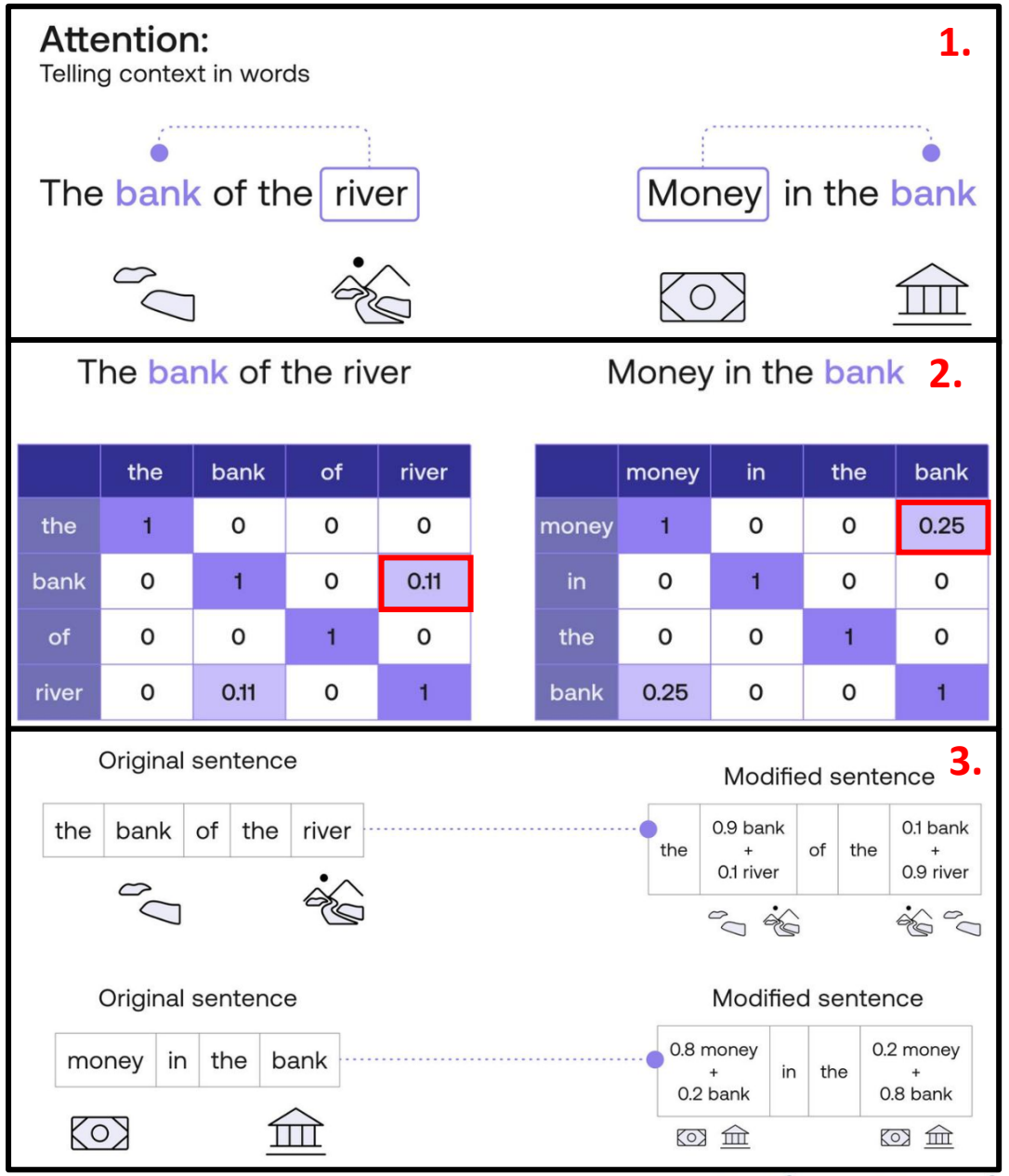


3. The **position** of each **token** in the sentence is added to the vector, such that sentences with the same words in a different order will have different vectors.

4. The vectors with added positional encoding are considered a **word embedding** and are input into the Transformer model.

Transformer – Encoder and Decoder Block

- A key component in the encoder and decoder is the **attention.**
- The word “bank” has numerous meanings depending on the context it is used.
  - We need context for the task.
- Attention helps identify context by determining the importance of nearby words.

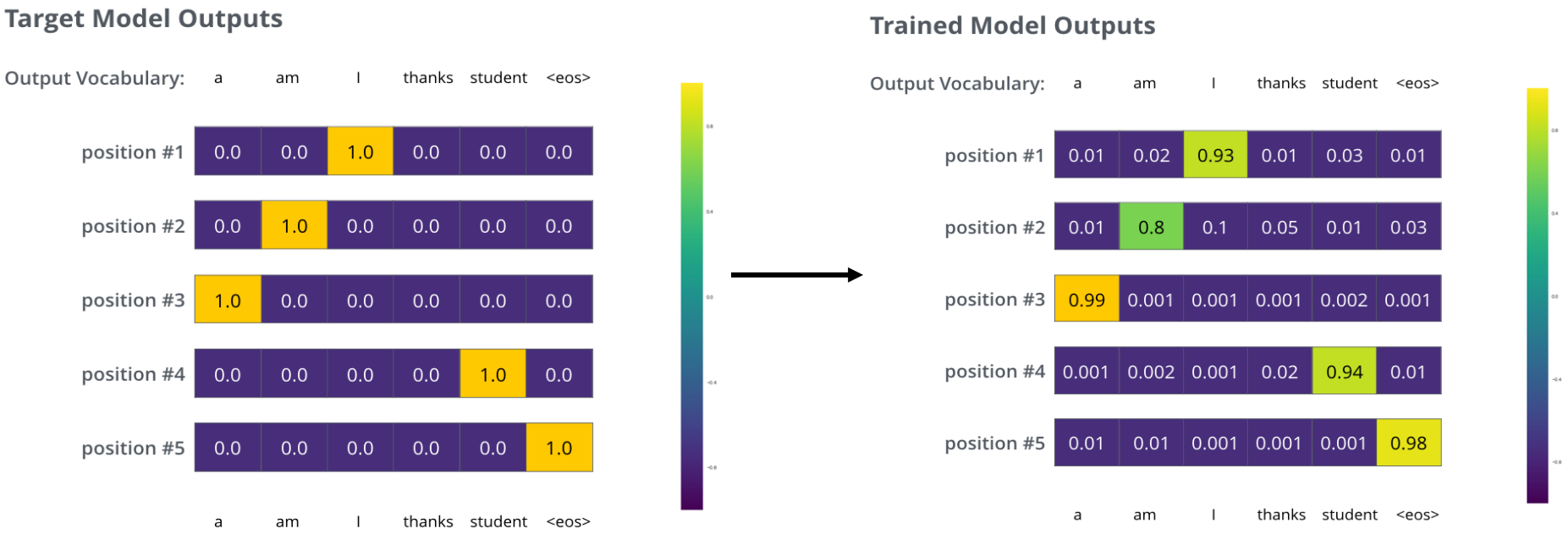


	the	bank	of	river
the	1	0	0	0
bank	0	1	0	0.11
of	0	0	1	0
river	0	0.11	0	1

	money	in	the	bank
money	1	0	0	0.25
in	0	1	0	0
the	0	0	1	0
bank	0.25	0	0	1

Transformer – Loss Function

- The transformer uses a simple **cross-entropy loss** to compare the target outputs and the predicted outputs.



(from https://jalammar.github.io/illustrated-transformer/)

Transformer

- Transformers are originally designed for sequence-to-sequence tasks (e.g., machine translation, summarization).
- However, real-world text is massive, dense, and uses highly specialized terminology.
  - E.g., 10K reports often have 100+ pages with financial terminologies
- We need trillions of words in the training data and millions of dollars of computational resources just to teach the model basic grammar.

Pretrained Language Models

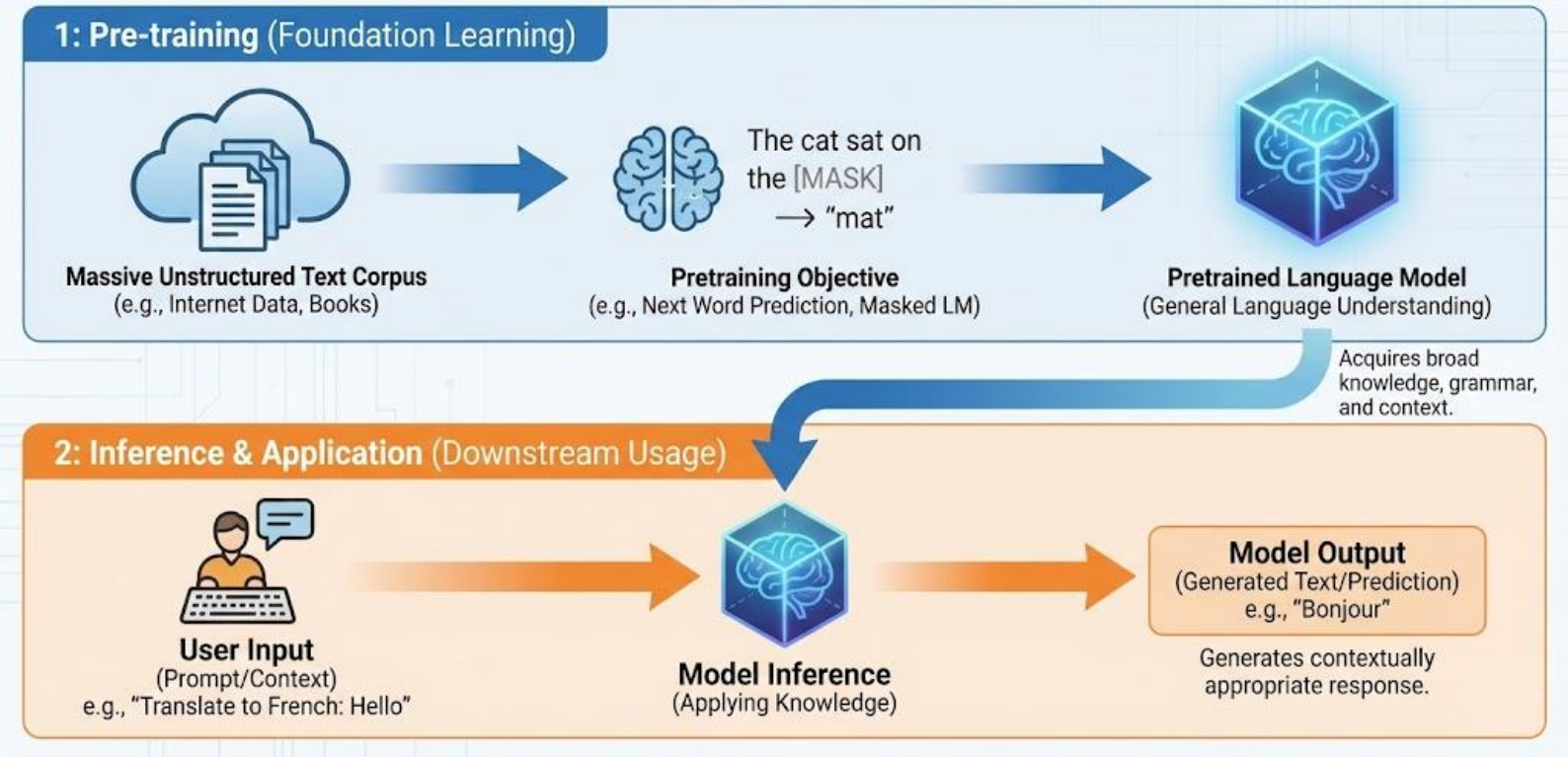
- Starting 2018, the field evolved from training task-specific models toward *pre-training* foundational architectures.
  - These models are trained on massive, unlabeled datasets to capture general linguistic patterns.
  - Efforts have been made in specializing Transformer's architecture (e.g., **Encoder** for deep bidirectional understanding, **Decoder** for generative scaling).

- In **2022**, the integration of human feedback unlock the **ChatGPT era,** producing models capable of following complex instructions and conversational reasoning (Ouyang et al. 2022).

Reference: Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, Maddie., Askell, A., Welinder, P., Christiano, P. F., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, *35*, 27730-27744.

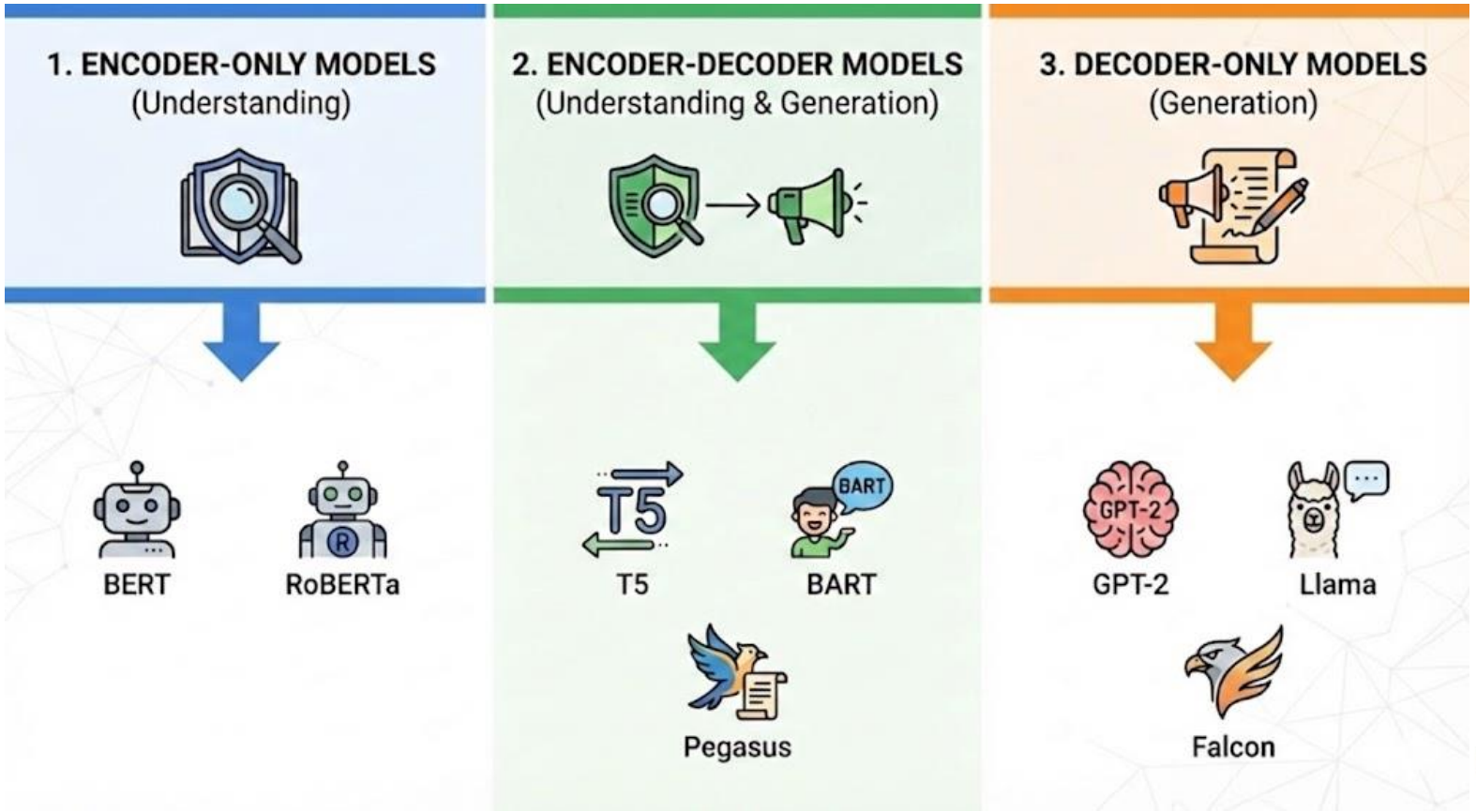
Pretrained Language Models

- Instead of starting from zero, we can **pretrain models** with natural languages (e.g., news, Wikipedia) to understand how human languages work.
  - E.g., when we see the term “social”, the next word is likely to be “media” or “interaction”



Pretrained Language Models – Categories

- **Pretrained language models (PLMs)** can be divided into three categories based on the components adopted from the transformer.



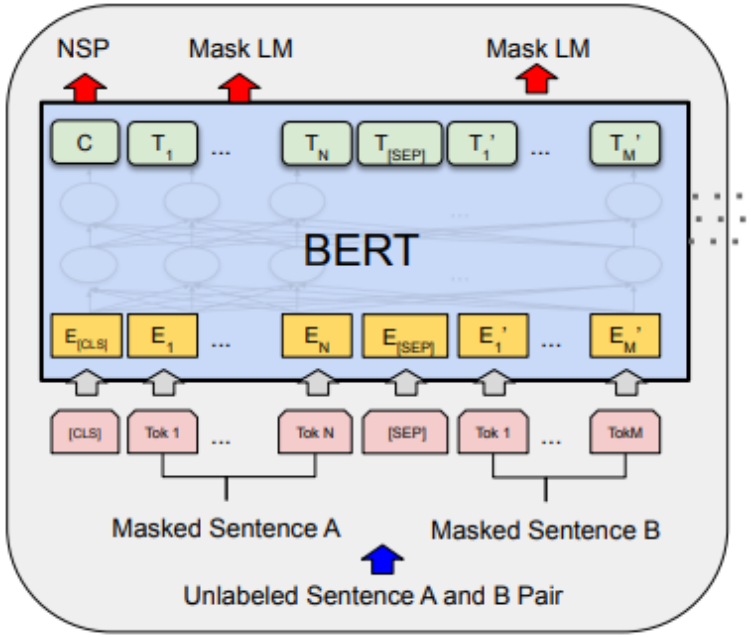
Pre-Trained Language Models – BERT

- BERT is a fundamental encoder-only model released in 2018 by **Google's AI Research team.**

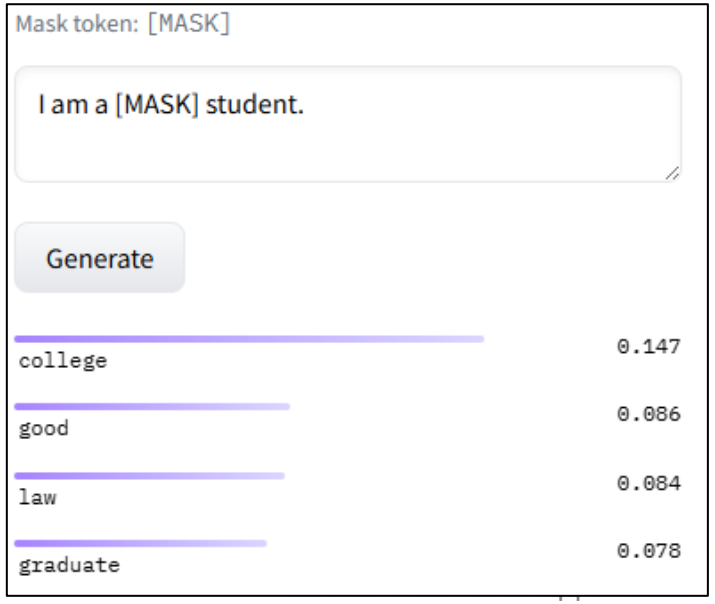
- BERT is pre-trained on the **BooksCorpus** (74 million sentences from 11,038 books) and **English Wikipedia** (2.5 billion words).

- Two-pretraining objectives:
  1. **Masked Language Modeling (Mask LM)**
     - A random word in a sentence is masked (i.e., hidden) from the model.
     - The model is trained to predict the masked word in the sequence.

  2. **Next Sentence Prediction (NSP)**
     - The training dataset is split into two-sentence pairs
     - The model is trained to predict if the second sentence follows the first.



BERT Pre-Training (from Devlin et al., 2018)

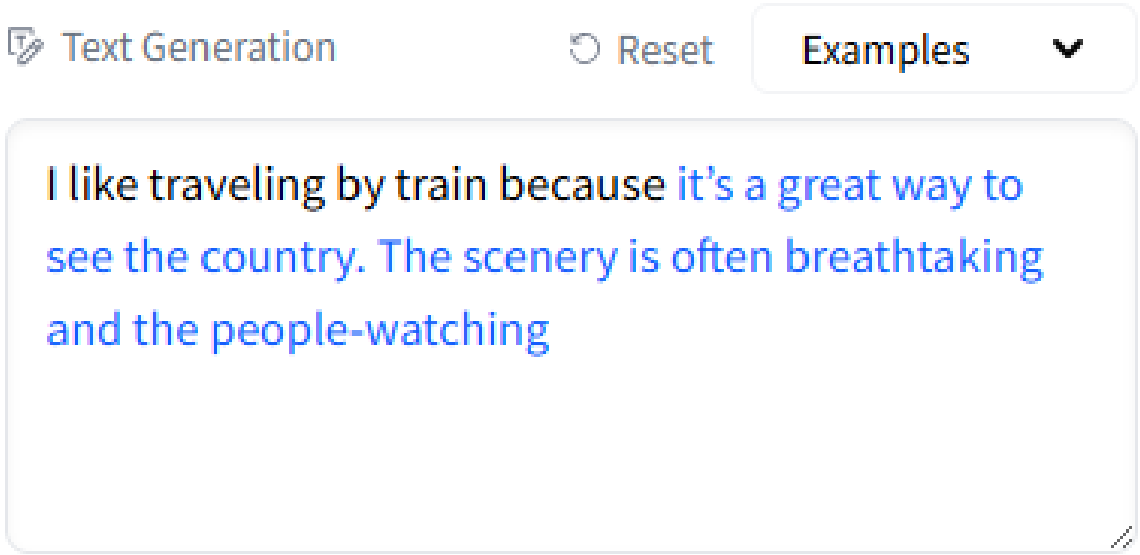


BERT Inference (from huggingface)

Reference: Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (pp. 4171-4186).

Pre-Trained Language Models – GPT

- **GPT** was introduced in **2018** by **OpenAI** as a paradigm-shifting architecture that uses an **autoregressive** training strategy (Radford et al., 2018).
  - GPT is pre-trained on the **BooksCorpus** (74 million sentences from 11,038 books).

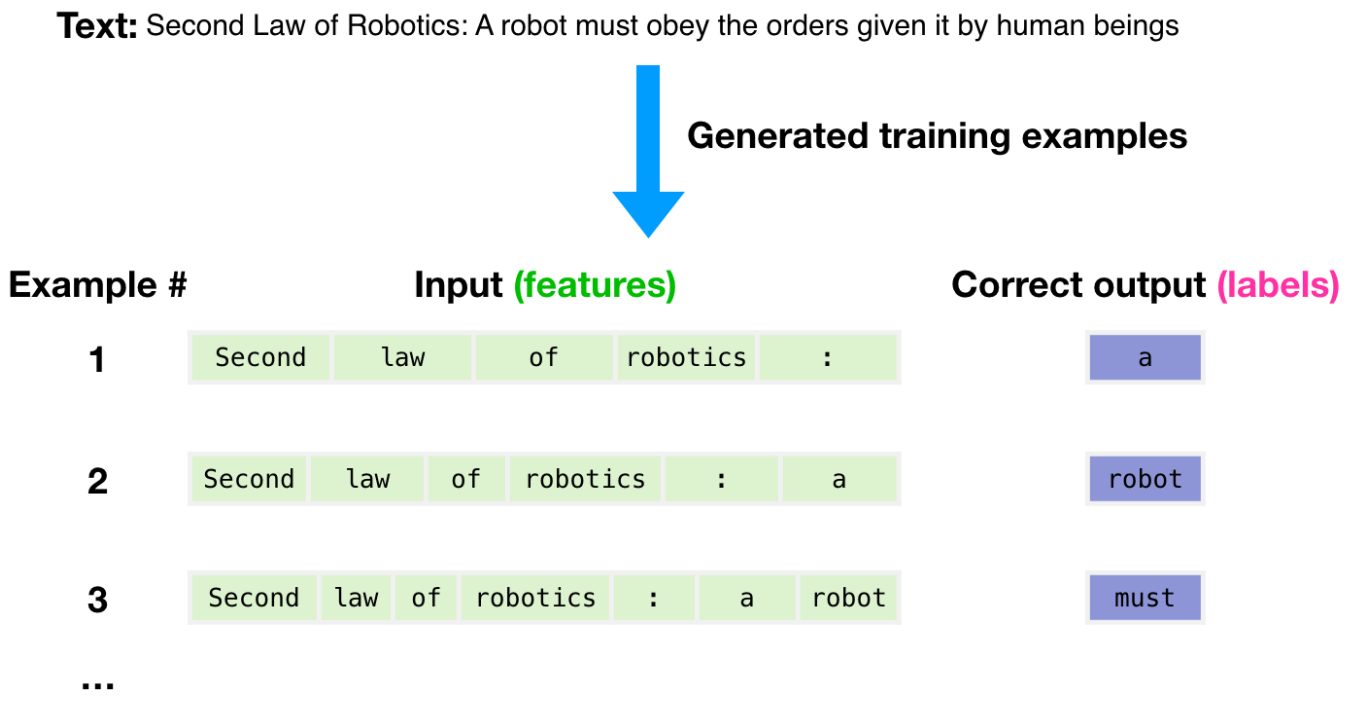


Text Generation Inference (from huggingface)

Reference: Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training*. OpenAI.

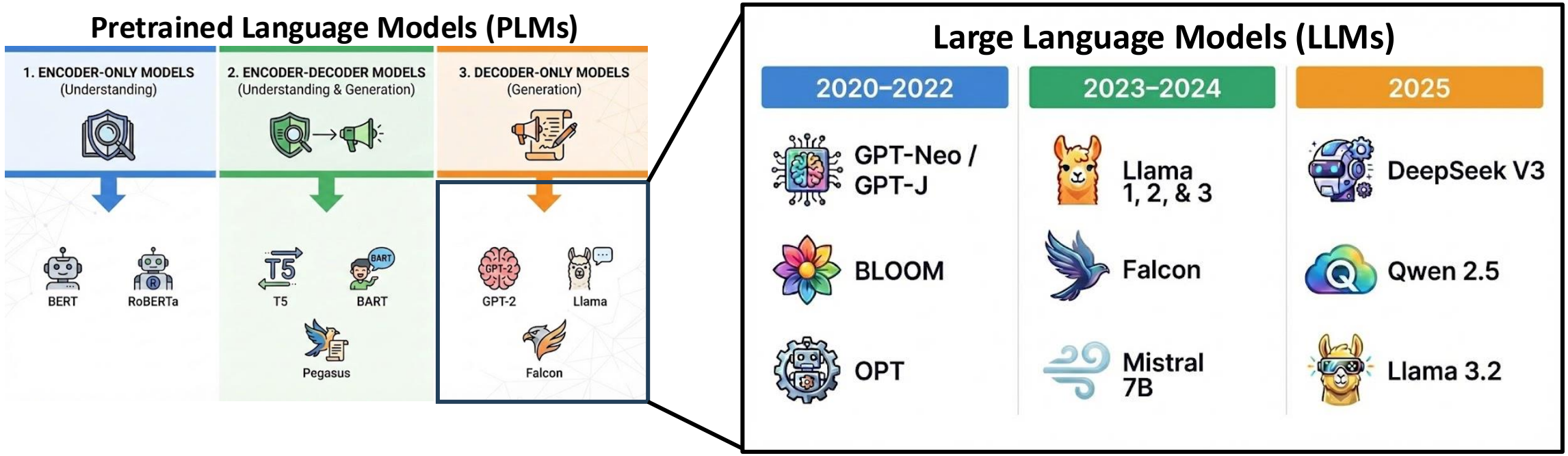
Pre-Trained Language Models – GPT

- GPT uses an autoregressive (predicts future values based on past values) training.
  - The goal of GPT is to **maximize** the **conditional probabilities** of each token given prior tokens as context.



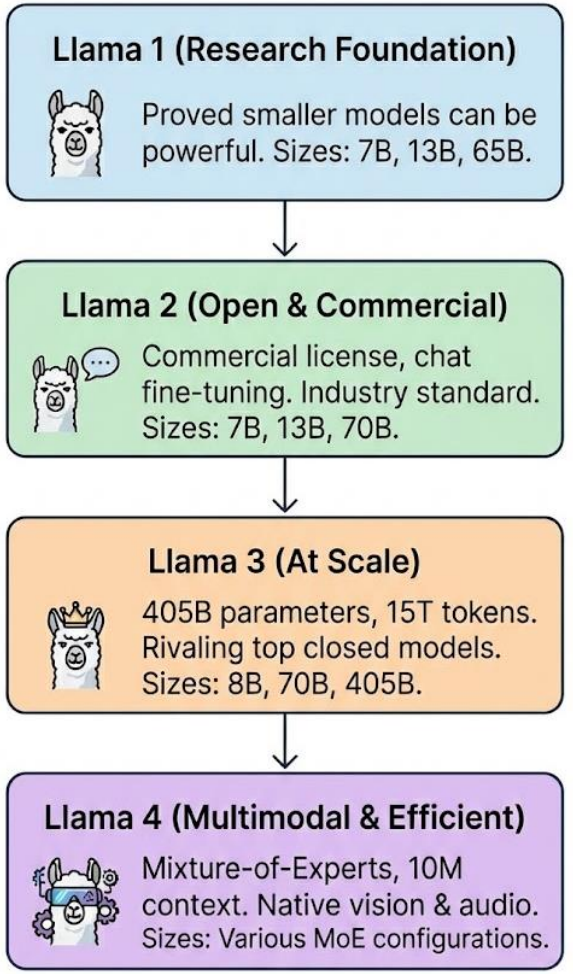
Large Language Models

- Recent Pretrained Language Models (PLMs) development have been largely focused on **decoder-only** architecture**.**
- As these models scale to billions of parameters, they exhibit emergent reasoning capabilities and are known as **Large Language Models (LLMs).**



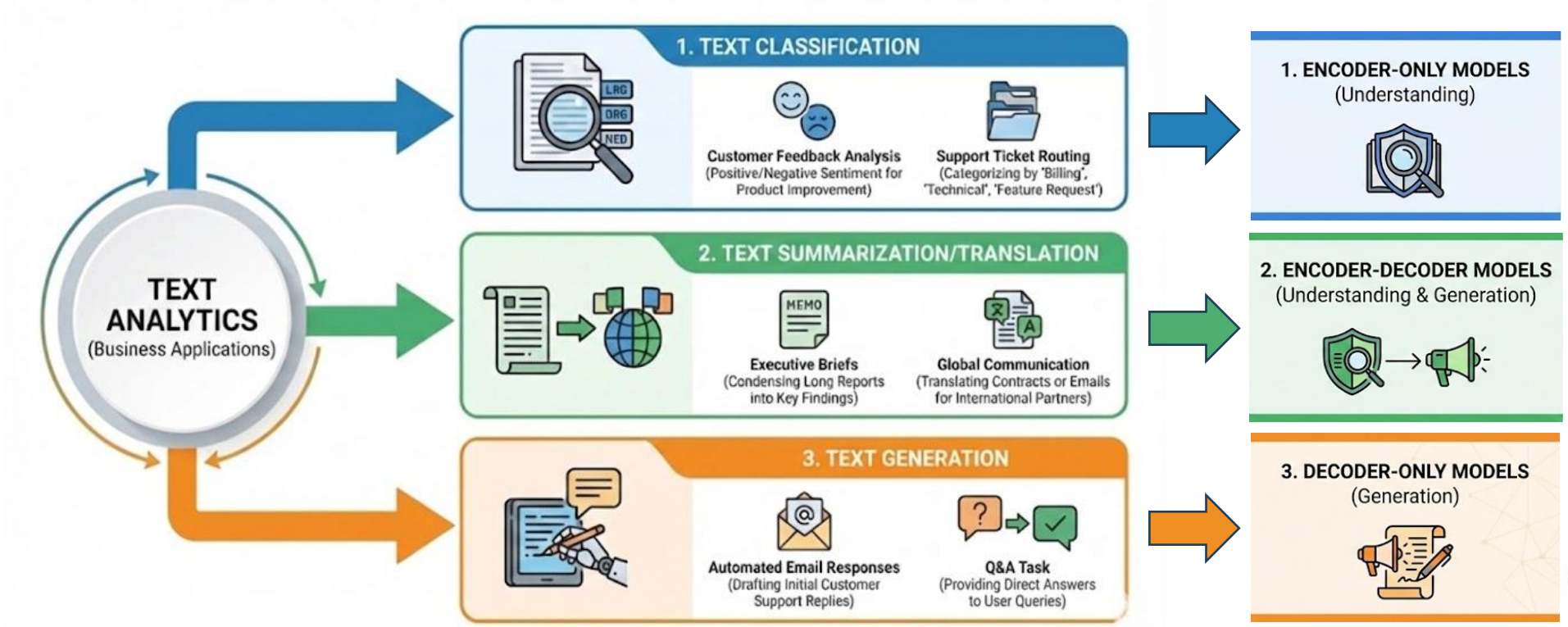
Large Language Models – Llama

- **Llama** is developed by Meta and is one of the most popular open-parameter LLM.
  - The model parameter is available on HuggingFace.
  - Developer can adapt it for specific tasks (e.g., advanced coding, financial statement analysis).
- It matches the intelligence of top-tier closed-source models (like GPT-4) while being significantly more efficient and cheaper to operate.
- Starting from Llama 3, it can now process images, understand charts, and handle massive amounts of data (like entire books).



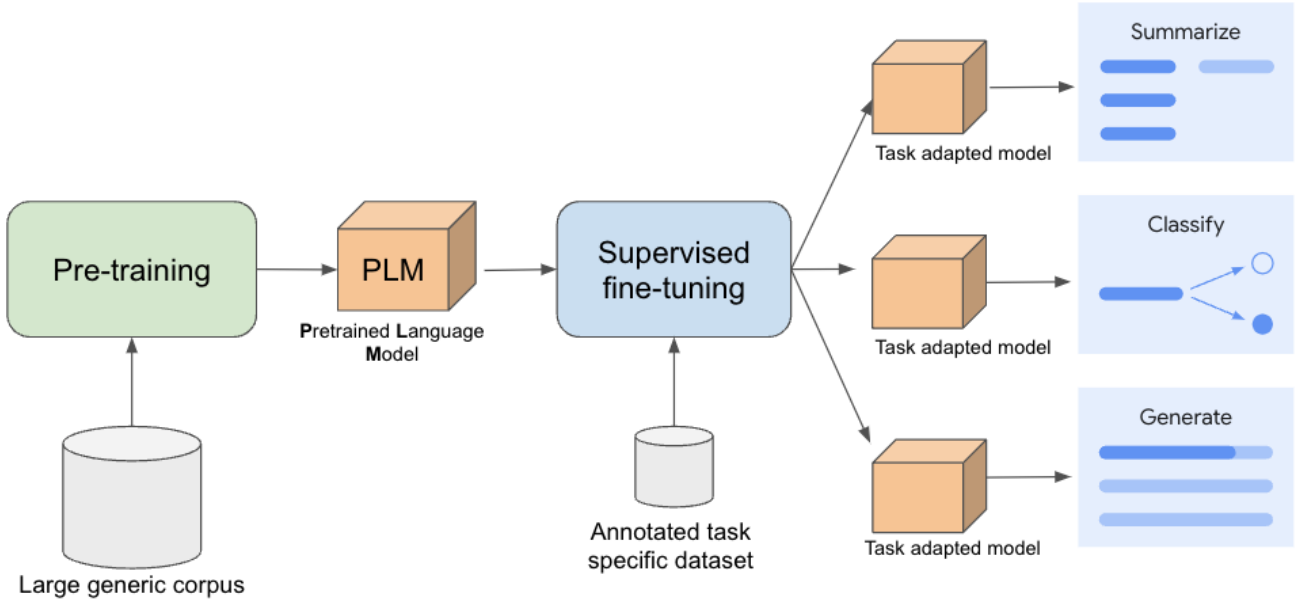
Task Adaption

- Pretraining alone (e.g., masked language modeling) is usually not sufficient for all kind of text analytic tasks.
  - Pretrained models also don't understand domain-specific terminologies.
- **Task adaption** aim to use additional training data to allow PLMs to perform effectively on specific domains and tasks.



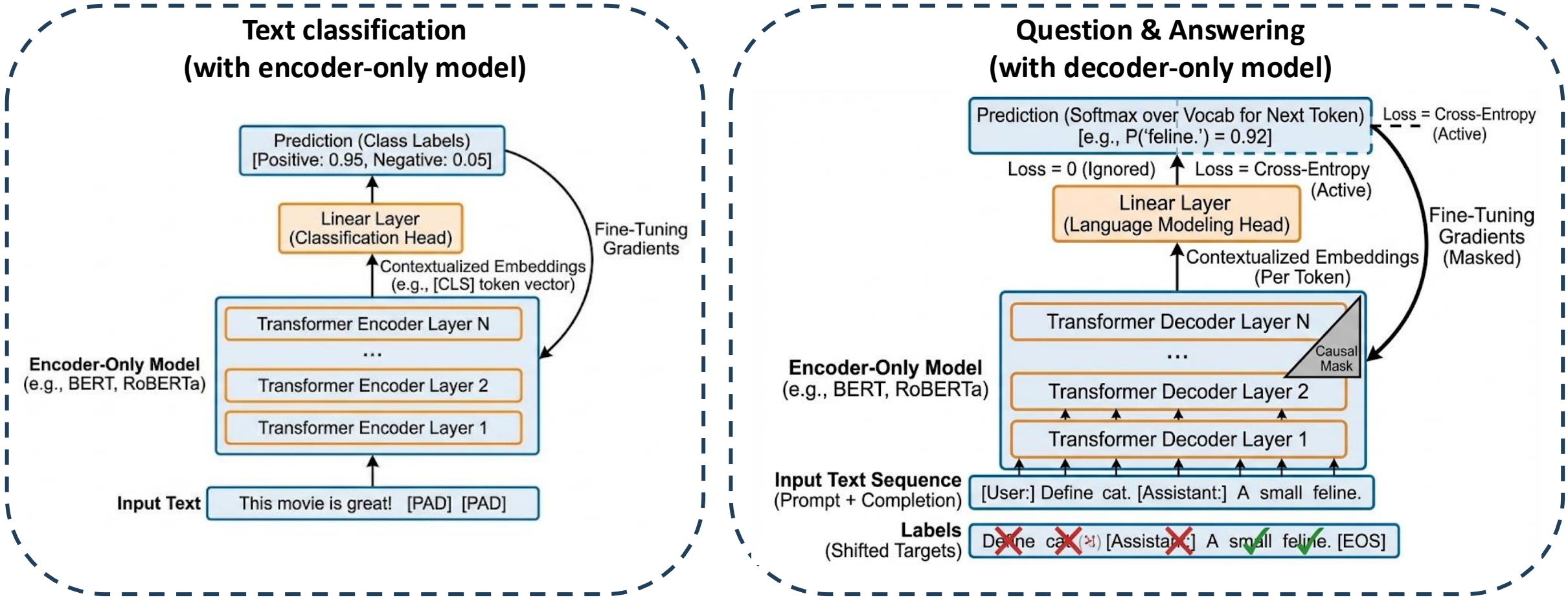
Task Adaption – Supervised Fine-Tuning

- Supervised fine-Tuning (SFT) is the most straightforward method for task adaptation.
- **Key idea**: use additional training data to slightly update the parameters of pre-trained model (W) to make it task specific (W').



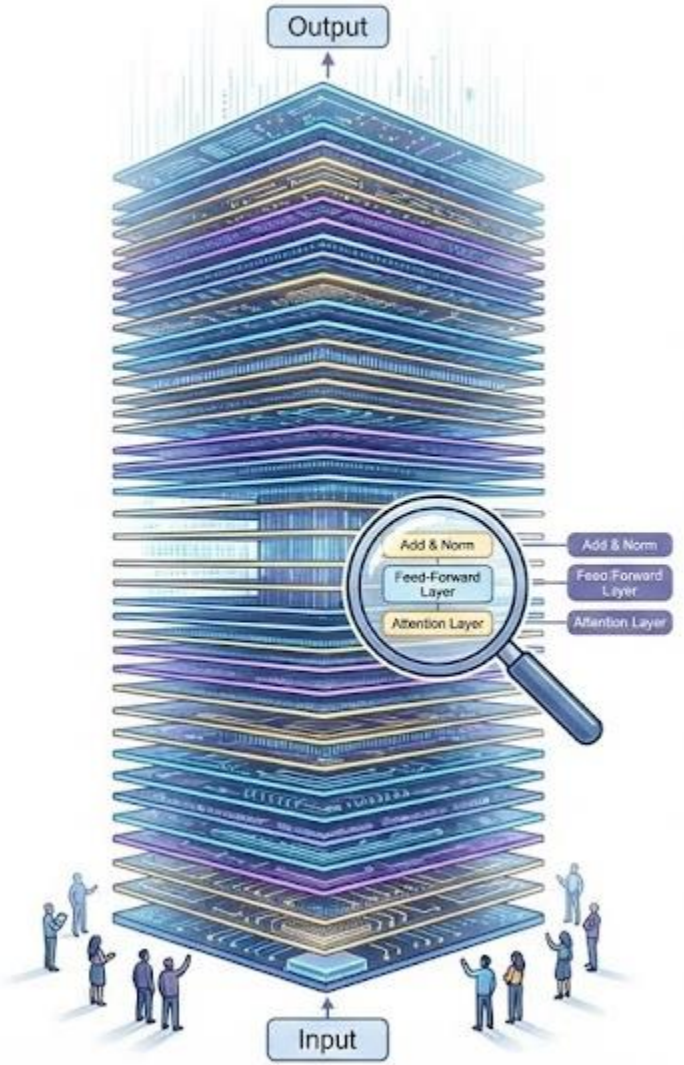
(From https://cloud.google.com/blog/products/ai-machine-learning/supervised-fine-tuning-for-gemini-llm)

Task Adaption – Supervised Fine-Tuning



Task Adaption – Supervised Fine-Tuning

- Modern LLMs (e.g., LLaMa) can have up to billions of parameters.
  - They often have hundreds or even thousands of layers of multi-head attention and feed-forward layers.
- It’s not realistic to fine-tune all of them.
- We need a **parameter-efficient fine-tuning (PEFT)** method.

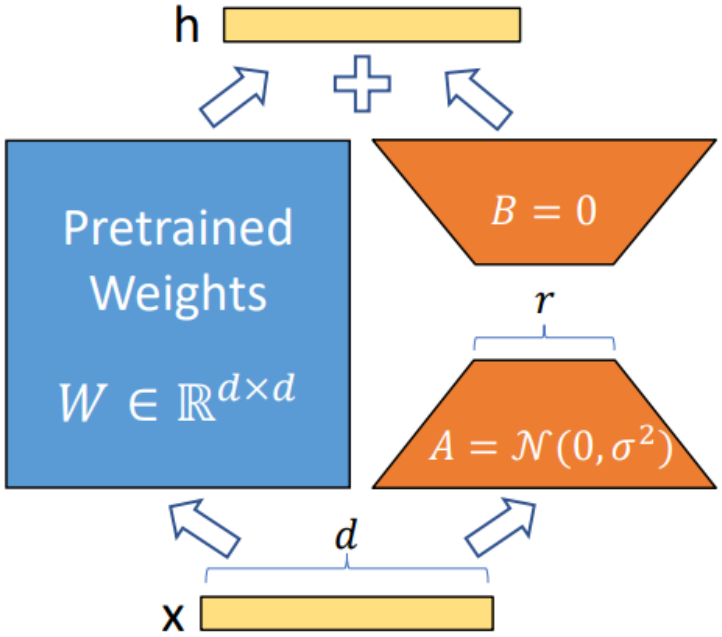


Task Adaption – Supervised Fine-Tuning

- Low-Rank Adaptation (LoRA) is a common PEFT method (Hu et al. 2021).
- Instead of updating W to W' directly, it update W by updating the changes ΔW only, where:

$$W' = W + \Delta W$$

- ΔW can be decompose into two matrices A and B, with an intrinsic *rank* r capturing the "essence" of parameter updates.
  - r is much smaller than the original dimension, this significantly reduce the number of parameters being updated.



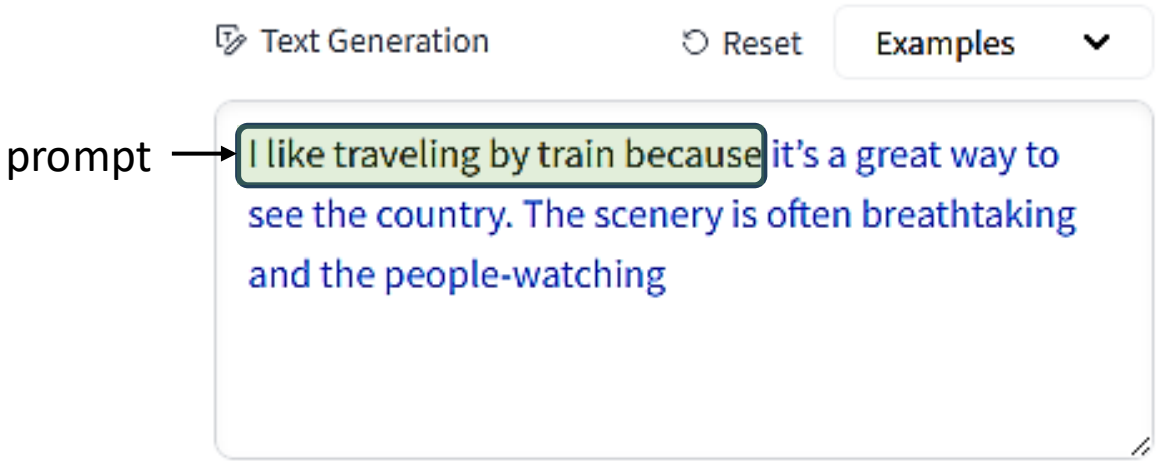
LoRA Reparametrization
(from Hu et al. 2022)

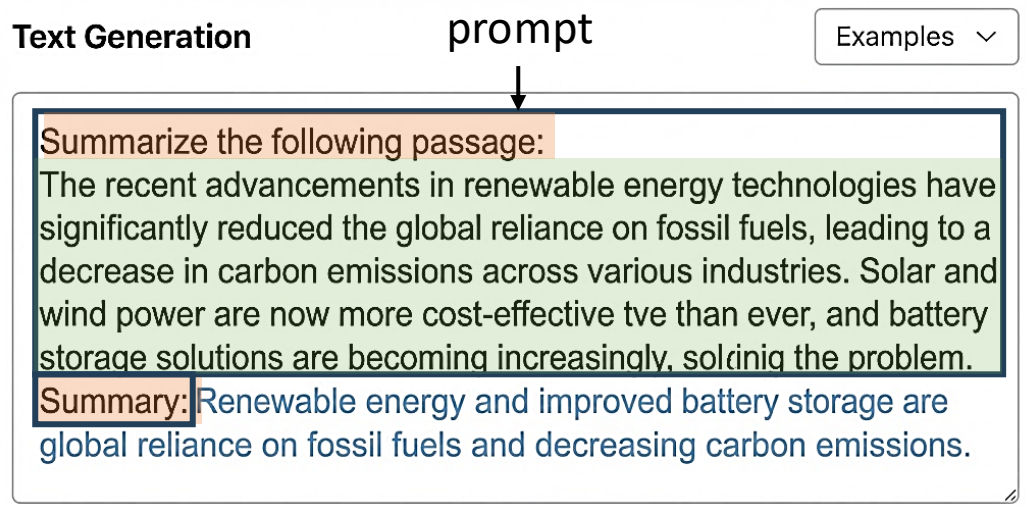
Reference: Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2022). Lora: Low-rank adaptation of large language models. ICLR, 1(2), 3.

Task Adaption – Prompt Engineering

- Recent pretrained models (especially LLMs) alone have shown great performance in text generation as their parameter size is large.

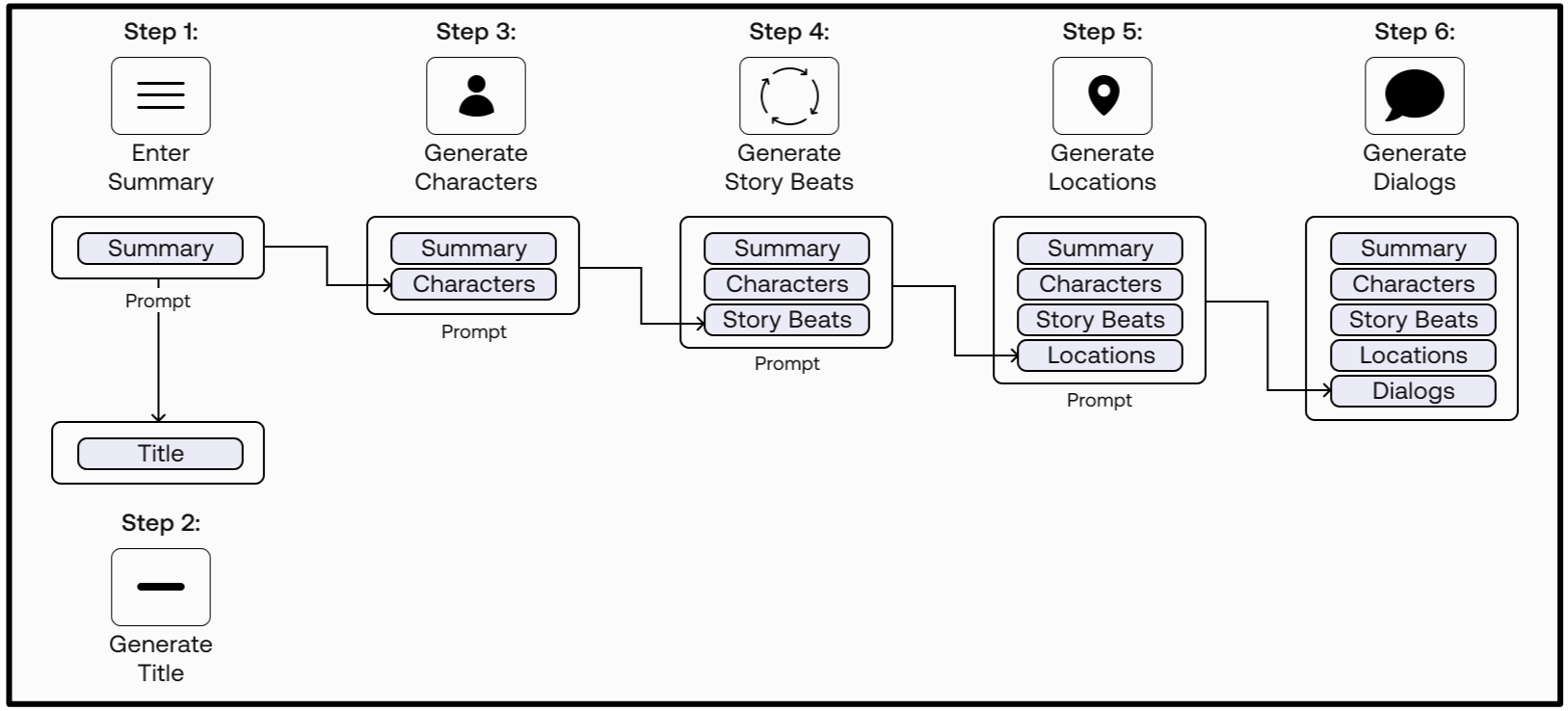
- We can get the desired output by curating the prompt given to LLM.
  - E.g., use a template (e.g., prefix) to specify the task (e.g., summarize the following passage) with the input (e.g., a passage).





Task Adaption – Prompt Engineering

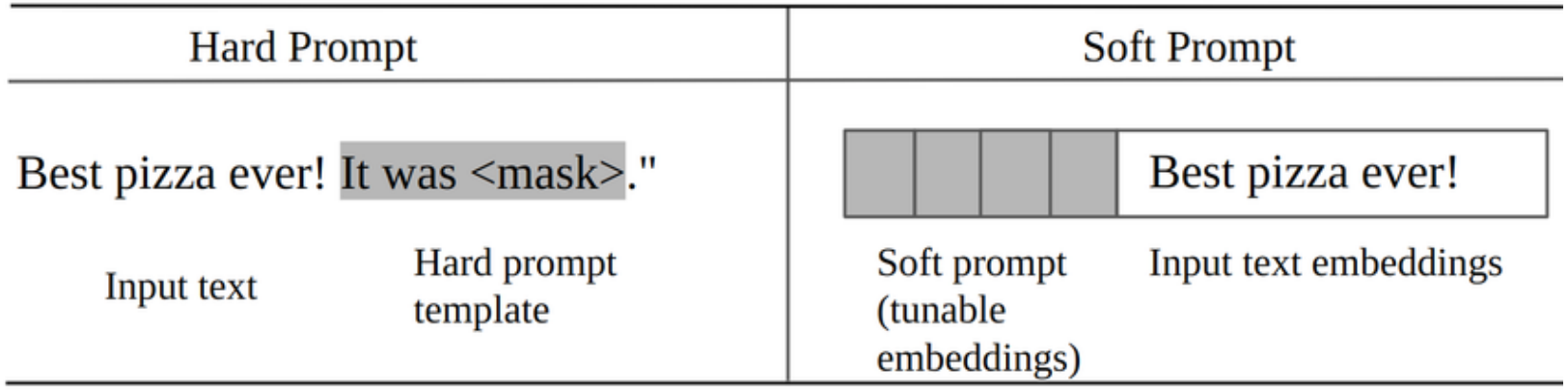
- Even with an engineered prompt, LLM's responses may not be ideal.
- **Chain-of-Thought (CoT)** prompting provides a series of reasoning examples to an LLM to help reasoning and achieve the desired output.



Task Adaption – Prompt Learning

- A “good” prompt can also be automatically identified through learning.

- Recall that each input token will be encoded into vector before the encoder and decoder blocks.
  - A prompt can be just vectors, not necessarily human readable.

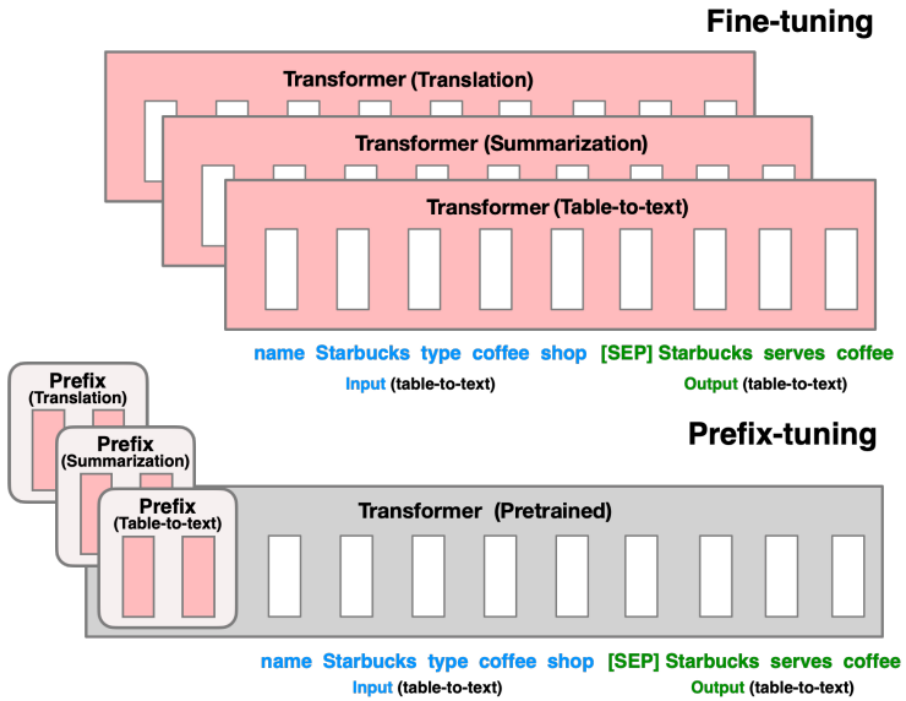


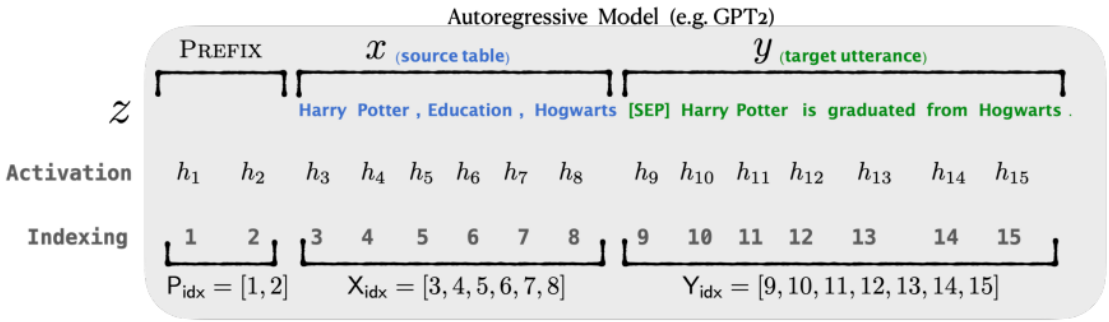
(From Xu and Sheng 2024)

Reference: Xu, Z., & Sheng, V. S. (2024). LecPrompt: A Prompt-based Approach for Logical Error Correction with CodeBERT. arXiv preprint arXiv:2410.08241.

Task Adaption – Prompt Learning

- **Prefix tuning** is a well-known prompt learning technique (Li and Liang 2021).
- A task-specific vector (**prefix**) is prepended to the input to guide the model's behavior.
- Only the prefix are updated during the learning process.





Summarization Example

```
Article: Scientists at University College London discovered people
tend to think that their hands are wider and their fingers are
shorter than they truly are.They say the confusion may lie in the
way the brain receives information from different parts of the
body.Distorted perception may dominate in some people, leading to
body image problems ... [ignoring 308 words] could be very
motivating for people with eating disorders to know that there was
a biological explanation for their experiences, rather than
feeling it was their fault."

Summary: The brain naturally distorts body image –
a finding which could explain eating disorders like
anorexia, say experts.
```

Reference: Li, X. L., & Liang, P. (2021, August). Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 4582-4597).

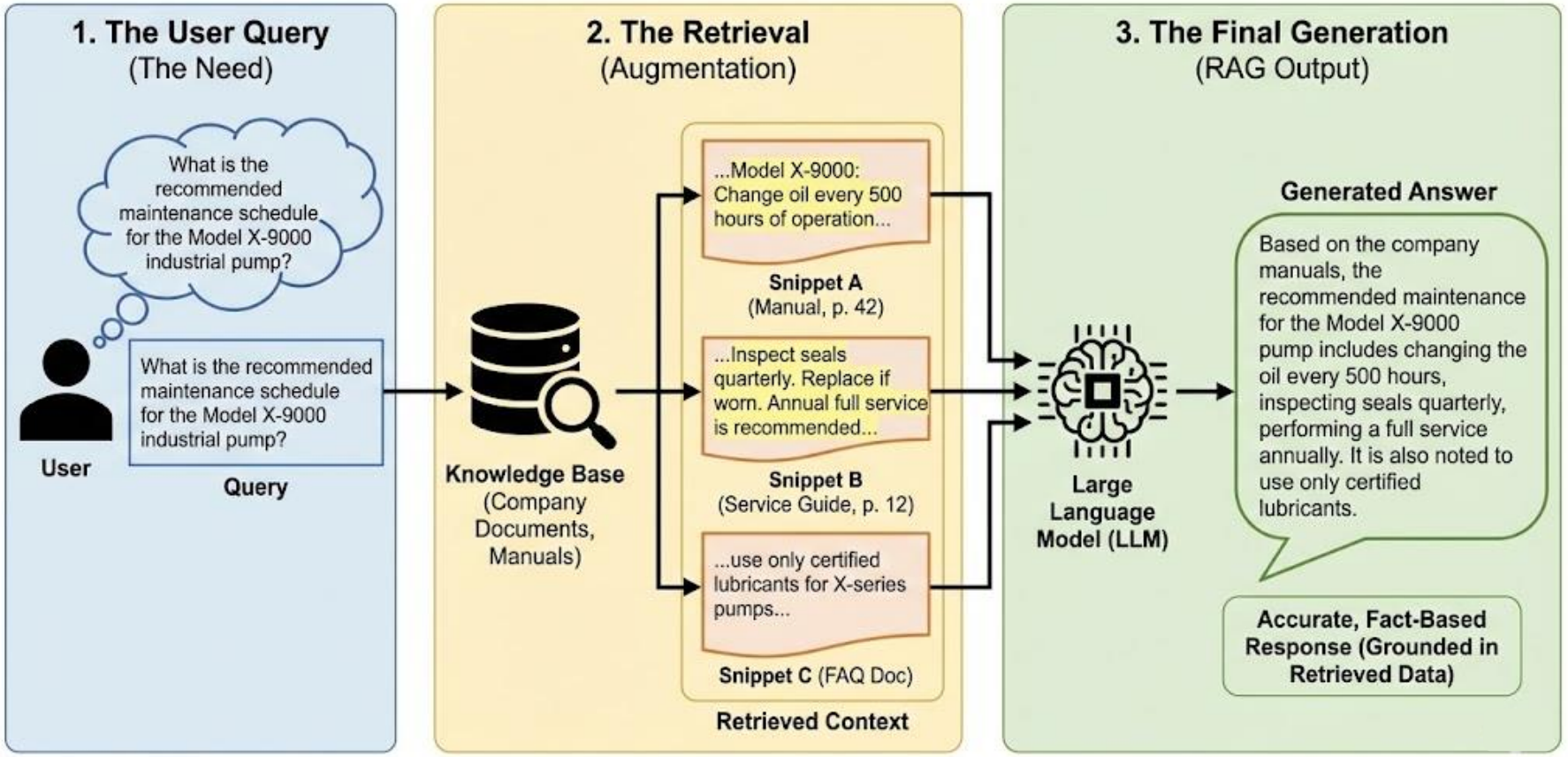
Task Adaption – RAG

- Models can be highly dependent from the data used in pre-training and fine-tuning.
  - If the training data is collected in 2023, it won't know what happened in 2026.
  - The model might make up some incorrect facts ➔ **Hallucination**

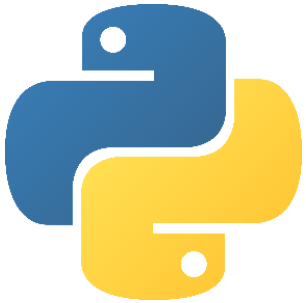
- We need real-time information from external sources to help generate the output ➔ **Retrieval augmented generation (RAG)**

Task Adaption – RAG

- RAG combines the prompt with retrieved, specific knowledge to generate an accurate, fact-based answer

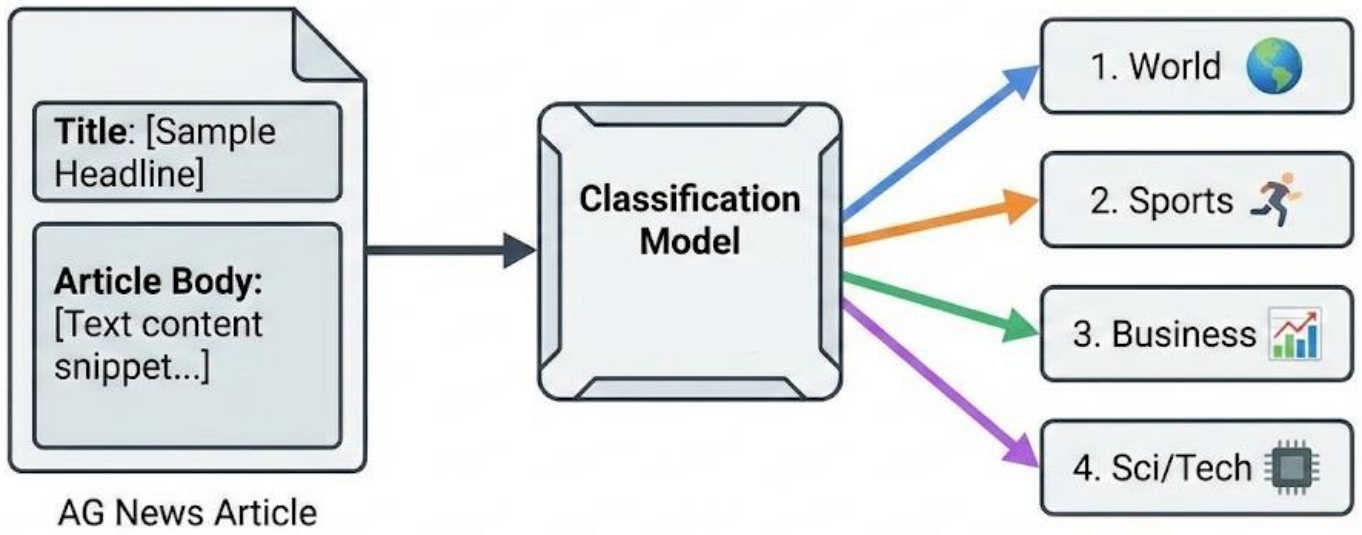


Python Example



Hugging Face

- **Python example**: news topic classification using BERT with zero-shot, SFT, and LoRA
- **Data**: AG news articles with title, article body, and class labels.
- **Key Packages**: PyTorch, transformer



Review Questions

- **Module 1: Artificial Neural Network**
  - In a binary classification task, what activation functions should be adopted in the hidden layer and output layer, respectively?
  - How does CNN handle pixels in image data?

- **Module 2: Transformer-based Models**
  - How does the attention mechanism capture “context”?
  - What’s the difference between SFT and prompt learning, in terms of the parameters updated?

Next Week

Problem Identification

Stakeholder

Scenario

Current Approach

Limitation

Data Collection and Extraction

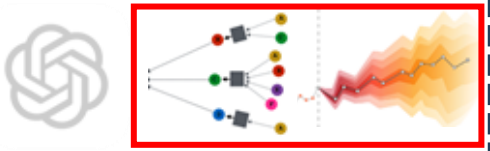
Data Collection

Data Extraction

AI-enabled Analytics

Machine Learning

Deep Learning



Advanced Techniques

(LLM, GNN, TSFM, RL)

Visualization

System and User Interface

Course of Action