Unveiling the Mystery of Deep Learning: Past, Present, and Future

Dr. Elham Barezi,
AI Research scientist

Co-Sponsored by Rosen Center for Advanced Computing (RCAC), and IPAI

Spring and Summer 2025

PURDUE UNIVERSITY

Building a Supportive AI Community!

- We're committed to build a strong, inclusive, and resourceful AI community for Purdue!
- We want to help you use AI confidently, ethically, and creatively!
- We're ready to do even more.

Tell us what would make this community more valuable for you:

- **Consulting & Mentorship**: Expert guidance, office hours, project feedback?
- **Technical Support**: Help with models, tools, GPUs, or data?
- **Learning Resources**: Tutorials, workshops, reading groups?
- **Community Events:** Meetups, speaker sessions, collaborations?
- **Anything Else?**

Share Your Needs & Ideas With Us!

Course Outline

1. History and Basics of DNN
    a. From traditional ML to DNN
2. Fundamental deep learning: from discriminative to generative
    a. CNN, RNN, Autoencoders, attention,
    b. Deep learning for Representation Learning and feature extraction
    c. Discriminative vs generative deep learning: VAE, GAN, Diffusion Models
3. Transformers Era
    a. self-attention, encoders, decoders, masking,
    b. self attention vs old attention
    c. Selection Criteria
4. LLMs in Practice
    a. Prompt Engineering Methods, RAG, Agents,
    b. Fine-tuning Methods: instruct tuning, RLHF, Adapters
    c. Deep learning for different domains
5. AI safety and Governance

PURDUE UNIVERSITY

Course Outline-first session-March 5th 2025

1. History and Basics of DNN
    a. AI hypes and winters
    b. Deep learning from 1950s
    c. From single neurons to deep networks
    d. Deep learning challenges solved from 1950-present
        i. Model overfitting
        ii. Activation function saturation
        iii. Vanishing/exploding gradient
    e. Deep learning weaknesses

Course Outline-Second Session-April 18th 2025

2. Fundamental deep learning models, from discriminative to generative:

   a. CNN,
   b. RNN,
   c. Earlier version of **attention**,
   d. Deep learning for **Representation Learning and feature extraction**
   e. Earlier **Pre-Training** models
   f. Discriminative vs Generative deep learning

PURDUE UNIVERSITY

Course Outline-3rd session-July 14th 2025

1. From RNNs to Transformers

- Brief recap of:
  - Sequence modeling with RNNs
  - Traditional attention mechanisms
- Motivation for Transformers

2. Transformer Architecture Overview and Core components:

- Tokenization
- Self-attention
- Positional encoding: Sinusoidal vs learned positional encodings
- Layer normalization & residual connections

3. LLM Variants and Evolution

- Encoder-Only models,Architecture, Training and Prediction.
- Encoder-Decoder models,Architecture, Training and Prediction.
- Decoder-only models, Architecture, Training and Prediction.

4. LLM Tips and Tricks

- Context Window Size
- Inference and Next Token Prediction
- Selection Criteria for LLMs

Course Outline-4th session-Aug 6th 2025

1. LLM Engineering
    a. Prompt Engineering
    b. RAG
    c. Agents
    d. Fine-Tuning
        i. Full fine-tuning
        ii. Instruct tuning
        iii. Reinforcement Learning with Human Feedback (RLHF)
    e. Parameter-Efficient Fine-Tuning (PEFT)
        i. Prefix/Prompt tuning
        ii. Adapters
        iii. LORA
2. MultiModal Transformers: Image, Audio, Video, 3D Transformers

Course Outline

1. LLM Engineering
    a. Prompt Engineering
    b. RAG
    c. Agents
    d. Fine-Tuning
        i. Full fine-tuning
        ii. Instruct tuning
        iii. Reinforcement Learning with Human Feedback (RLHF)
    e. Parameter-Efficient Fine-Tuning (PEFT)
        i. Prefix/Prompt tuning
        ii. Adapters
        iii. LORA
2. MultiModal Transformers: Image, Audio, Video, 3D Transformers

Pre-Training vs Fine-Tuning

Pretraining is the initial training phase where a model learns general-purpose patterns from **large, unlabeled datasets** (e.g., predicting missing words, next tokens, etc.). **It builds foundational knowledge useful across many tasks.**

Fine-tuning adapts a pretrained model to a **specific task using smaller, labeled datasets** (e.g., sentiment analysis, question answering). **It specializes the model by continuing training on task-relevant data**.

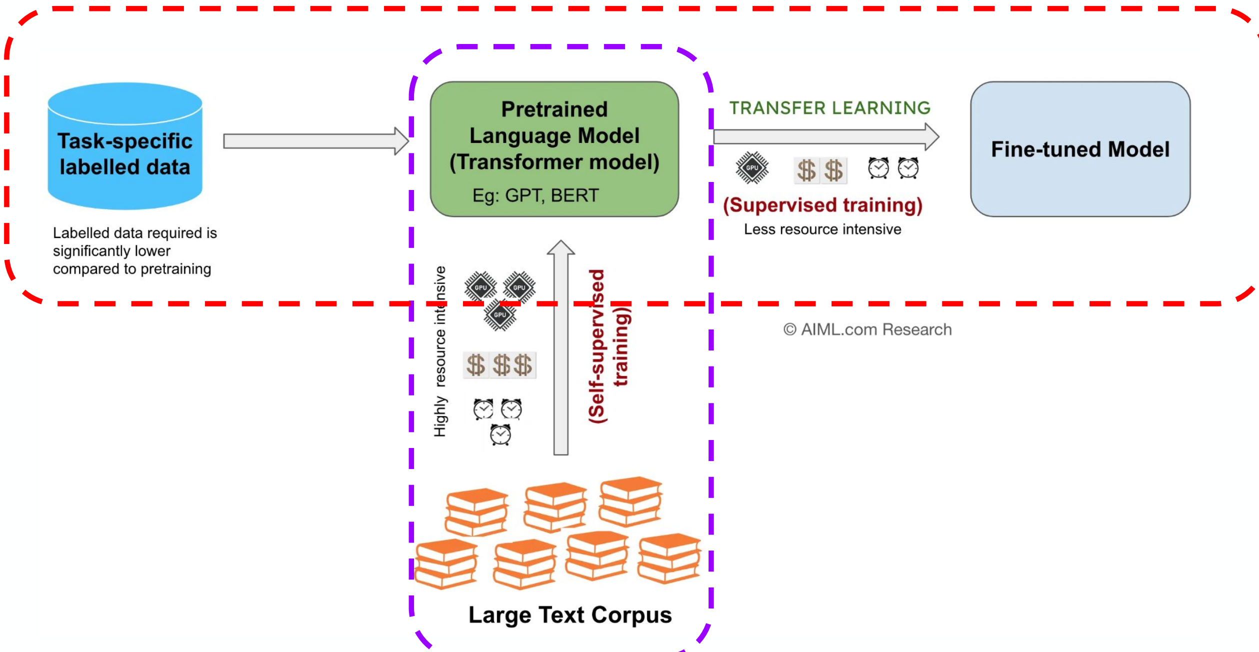


Photo from AIML.com with some modifications

Multi-task Learning

A Step Forward, but Not Enough

- In 2019, multitask learning gained popularity, with models like T5 being trained on multiple tasks simultaneously.
- T5 proposed that every NLP task is cast as a text-to-text problem and **Standardizing tasks** as natural language instructions.
- Showing that **changing the prompt** (not model weights) could change behavior.
- This approach improved performance on the training tasks, but failed to address cross-task generalization: **models were unable to generalize to new, unseen tasks**.

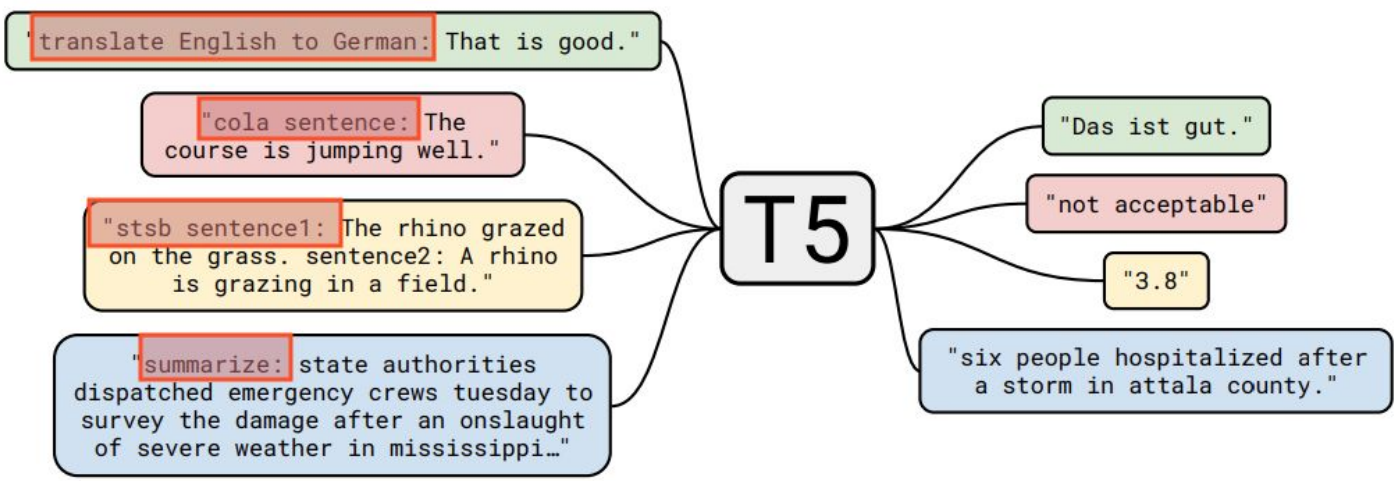


Photo from T5 paper with modifications.

Prompt Engineering

The launch of GPT-3 in 2020 showcased the remarkable capabilities of LLMs with 175 billion parameters, allowing them to effectively perform tasks through **few-shot prompting.**

- If Prompt Engineering is not accurate enough, you can pay cost for fine-tuning.

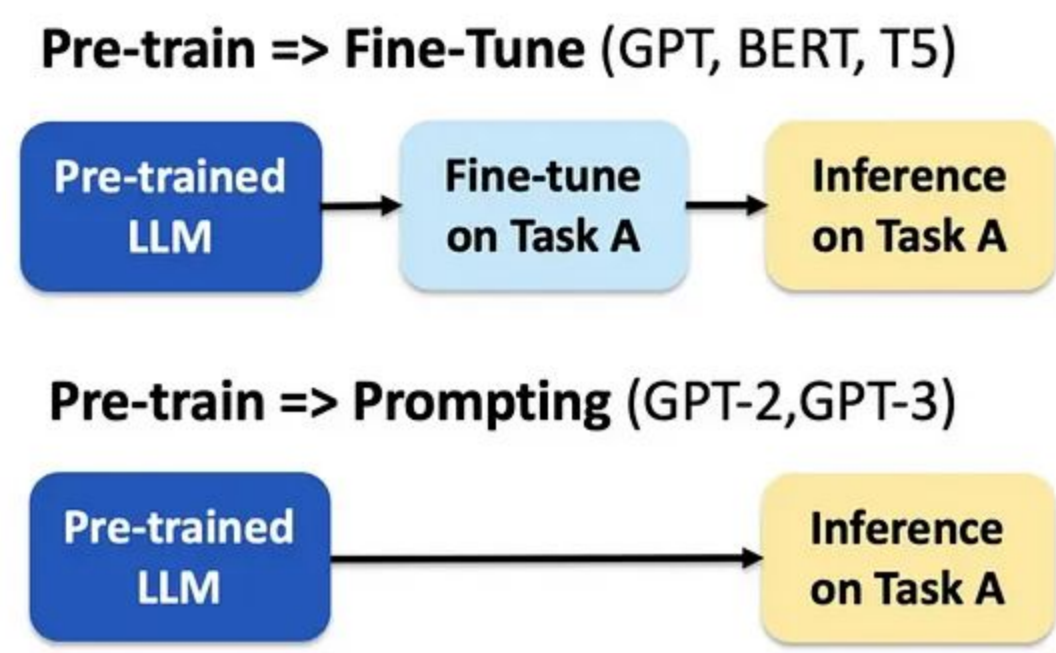


Photo from
https://medium.com/@lmpo/an-overview-instruction-tuning-for-llms-440228e7edab

PURDUE UNIVERSITY

Prompt in LLM

- **Prompting** prepares a frozen pretrained model for a specific downstream task by including a text that **describes the task** or **even demonstrates an example of the task**.
- **Prompt engineering** refers to the process of **refining a model's input to produce the desired output**, **without updating the actual weights of the model** as you would with fine-tuning.
- **Prompt engineering** is the **art of asking the right question to get the best output from an LLM.** It enables direct interaction with the LLM using only plain language prompts.

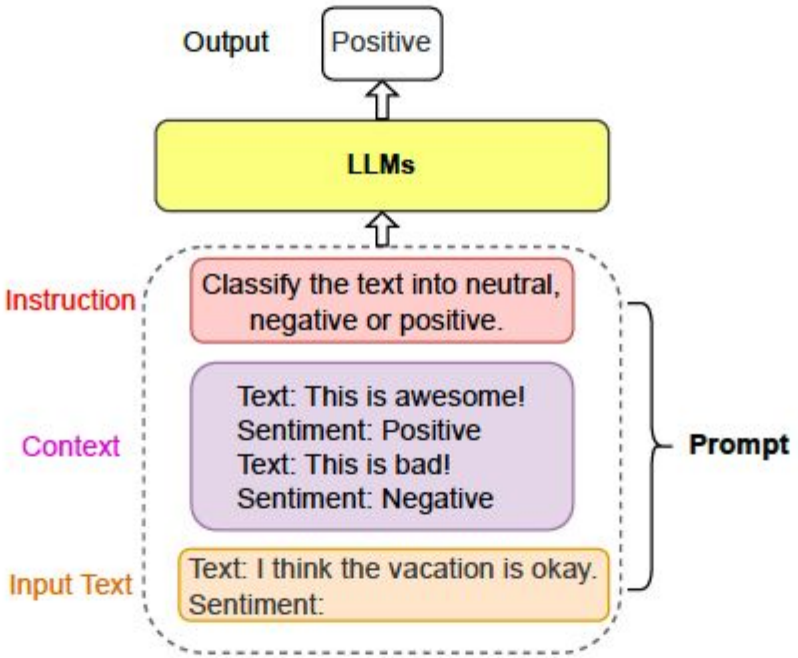


Photo from Unifying large language models and knowledge graphs: A roadmap.

PURDUE UNIVERSITY

Few-Shot and Zero-Shot Prompting

- In **Few-shot prompting,** you show an LLM a few examples of how a task should be done **in the prompt itself**, to guide LLM's behavior, without changing the model's weights.
  - For unfamiliar tasks and structured outputs
- In **Zero-shot prompting**, you ask an LLM to perform a task **without giving any examples**, relying only on the instruction.
  - For easy questions and general knowledge.

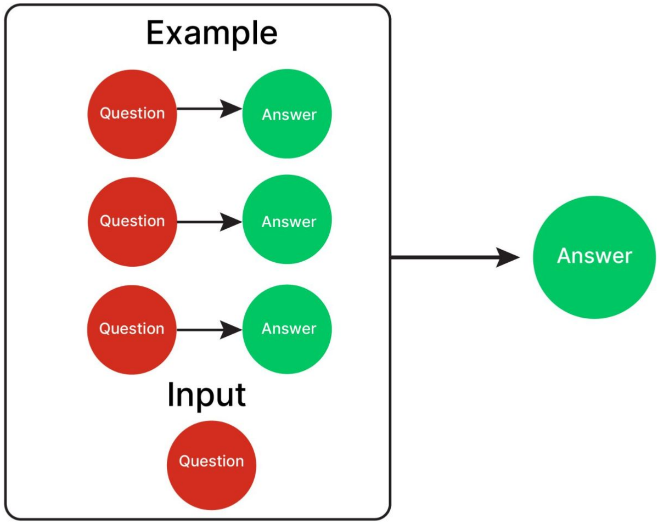


Photo from: https://www.analyticsvidhya.com/

Prompt Engineering

- Prompting is underestimated because the right prompting techniques, when used correctly, can get us very far.
- It is overestimated because even prompt-based applications require significant engineering around the prompt to work well.
- We need more advanced prompt engineering methods:
    - Chain of Thoughts
    - Tree of Thoughts
    - Self Consistency

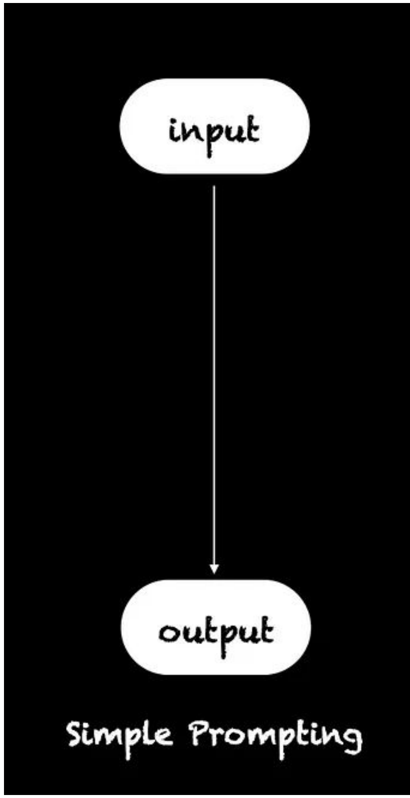


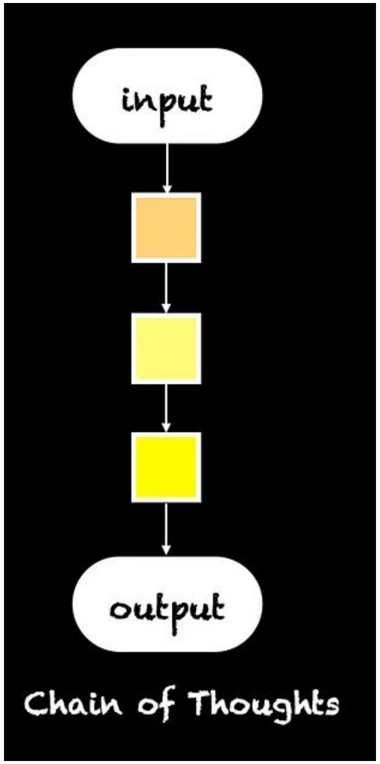
Photo from https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6

Chain Of Thoughts (COT)

We need to take advantage of the **reasoning** abilities of the LLM, if the direct approach of simple prompting does not give us what we expect.

The most popular applications of CoT:

- **Question Answering**: Answering complex questions that require inferencing or **reasoning**.
- **Mathematical Problem Solving**: Solving math problems **step-by-step**, providing justification for each step.
- **Program Synthesis**: Generating source code based on natural language **instructions**.



PURDUE UNIVERSITY

Photo from https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6

COT Example

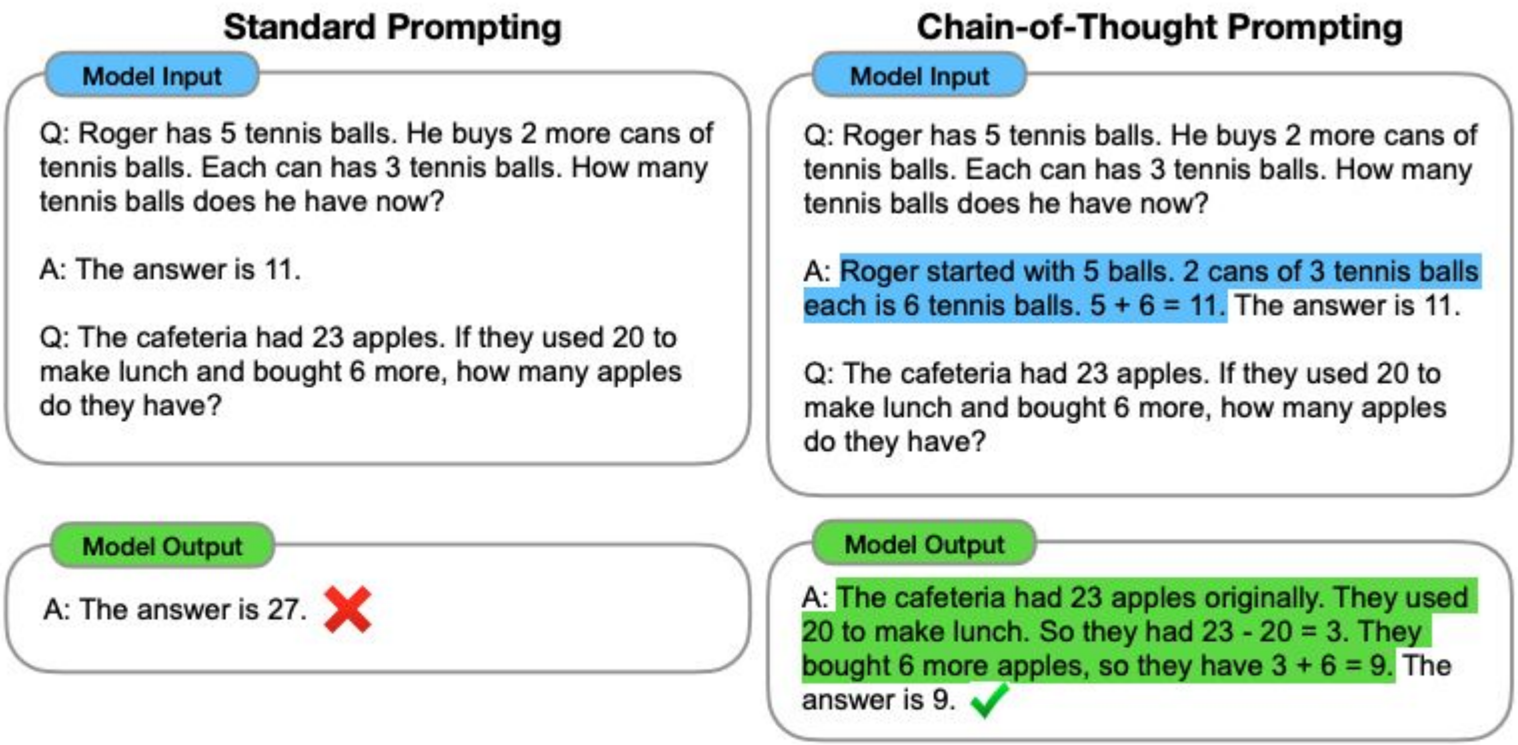


Photo from Chain-of-thought prompting elicits reasoning in large language models. *Neurips 2022.*

Tree of Thoughts (TOT)

Drawing inspiration from human cognitive processes, ToT facilitates considering a **spectrum of possible solutions** before deducing the most plausible one.

The LM's ability to generate and evaluate thoughts is **combined with search algorithms** (e.g., breadth-first search and depth-first search) to enable systematic exploration of thoughts with lookahead and backtracking.

Steps of TOT:

1. Generate multiple thoughts
2. Evaluate and select best
3. Expand and go deeper
4. Repeat if needed
5. Final solution

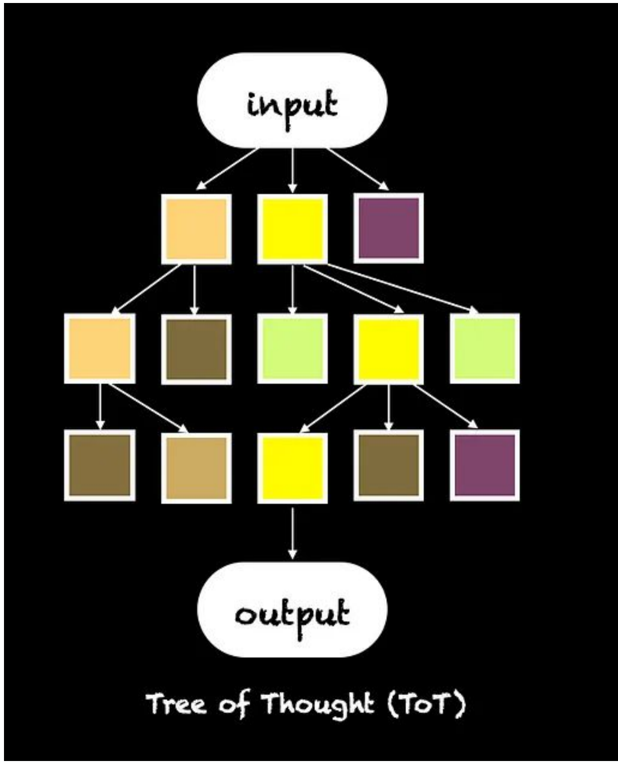
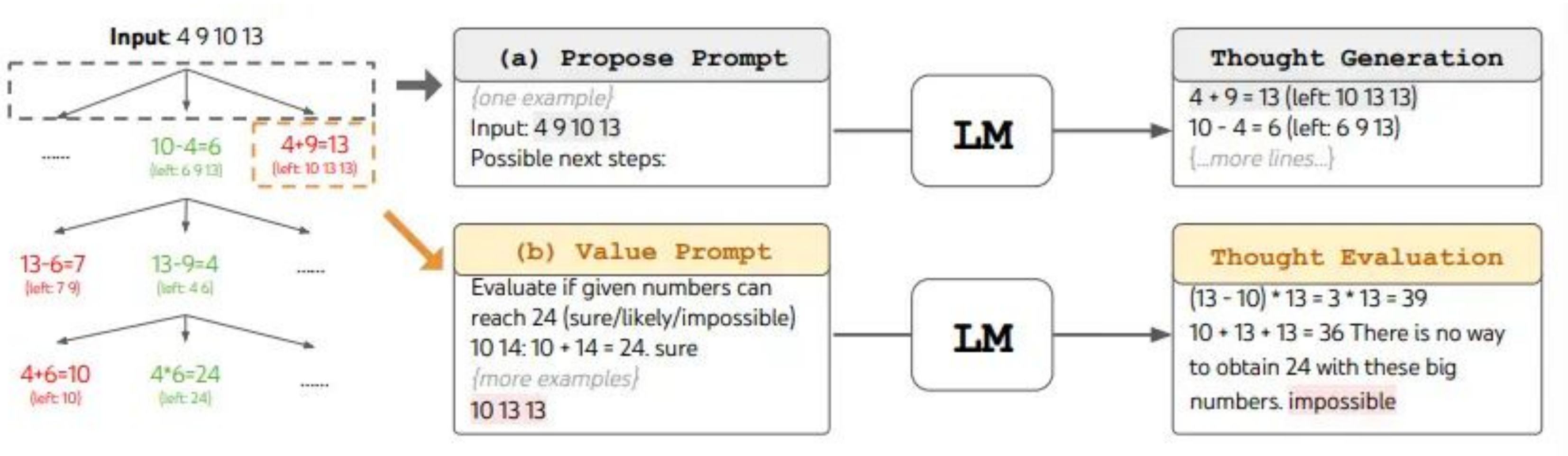


Photo from https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6

Tree of Thoughts (TOT)

ToT explores **multiple reasoning paths** in parallel, evaluates them, and **chooses the best one**.

There's **no clear single path to a solution**

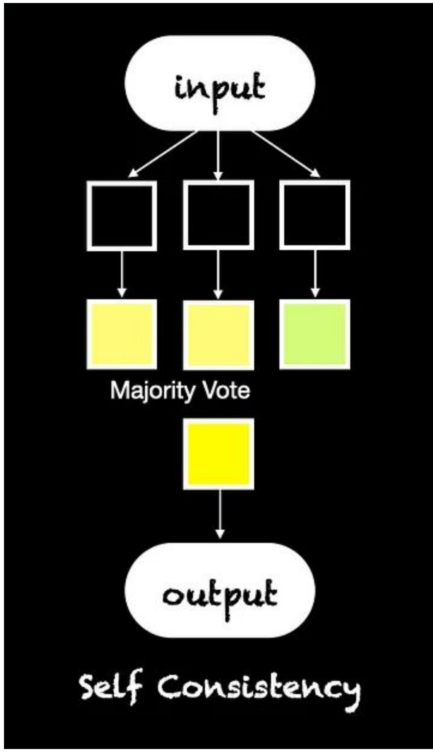


An example of Game of 24: The goal is to reach 24 using four given numbers and basic arithmetic operations.
Photo from [Yao et el. (2023)]

PURDUE UNIVERSITY

Self Consistency

- SC replace the **“greedy decode”** in CoT prompting by sampling from the language model’s decoder to generate a **diverse set of reasoning paths**; and marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.
- We can use temperature to generate several answers and reasons and find the best one.



$$P(\mathbf{r}_i, \mathbf{a}_i \mid \text{prompt}, \text{question}) = \exp^{\frac{1}{K}\sum_{k=1}^{K} \log P(t_k \mid \text{prompt}, \text{question}, t_1, \ldots, t_{k-1})},$$

r is reason, a is answer, and t are tokens.

Photo from https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6

Self-consistency Prompting Method

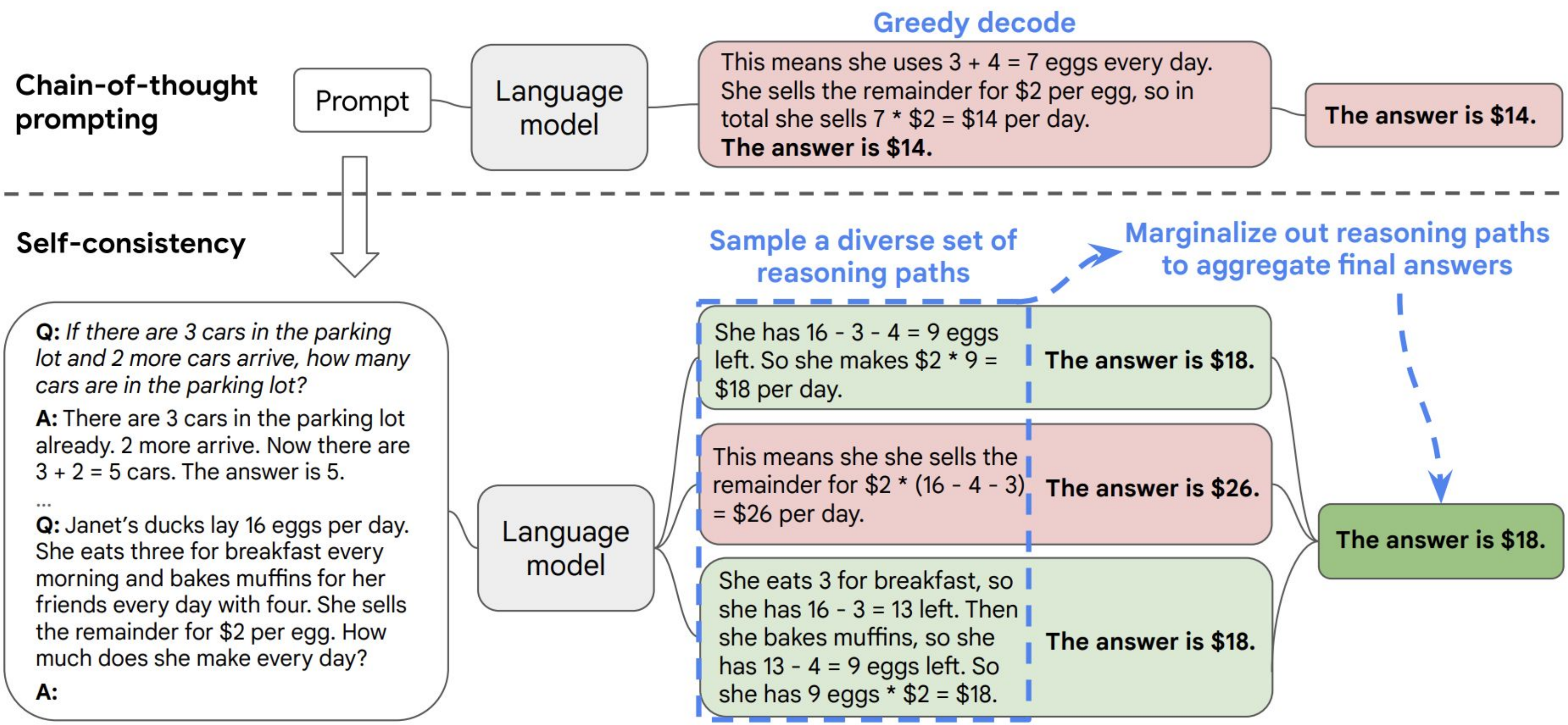


Photo from:
https://www.prompthub.us/blog/self-consistency-and-universal-self-consistency-prompting

Course Outline

1. LLM Engineering
   a. Prompt Engineering
   b. RAG
   c. Agents
   d. Fine-Tuning
      i. Full fine-tuning
      ii. Instruct tuning
      iii. Reinforcement Learning with Human Feedback (RLHF)
   e. Parameter-Efficient Fine-Tuning (PEFT)
      i. Prefix/Prompt tuning
      ii. Adapters
      iii. LORA
2. MultiModal Transformers: Image, Audio, Video, 3D Transformers

Retrieval-Augmented Generation(RAG)

- Pre-trained LLMs (foundation models) **do not learn over time**, often **hallucinate**, and may **leak private data from the training corpus**.
- **RAG** is a method for improving the response of an LLM by **injecting new data into the prompt** at the inference time, while **fine-tuning modifies the model**.
- RAG for LLMs aims to improve prediction quality by using an external datastore at inference time to **build a richer prompt that includes some combination of context, history, and recent/relevant knowledge (RAG LLMs).**
- RAG LLMs can outperform LLMs without retrieval by a large margin with much fewer parameters, and they can update their knowledge by **replacing their retrieval corpora**, and **provide citations for users to easily verify and evaluate the predictions**.

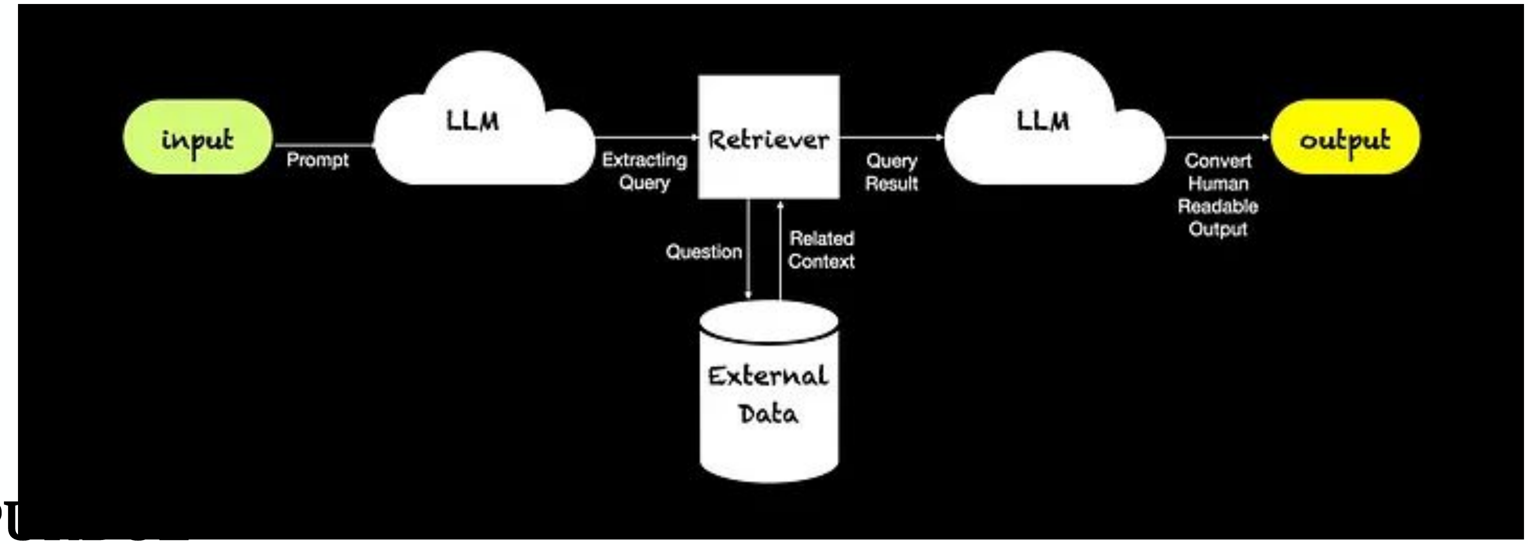


Photo from https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6

False Information by LLMs

- **Misinformation** involves the spread of false or inaccurate information **without malicious intent of the user**.
  - **Hallucination** refers to the generation of content that the model **invents or fabricates**.

- **Disinformation** is generating false information that is **intended to mislead**.

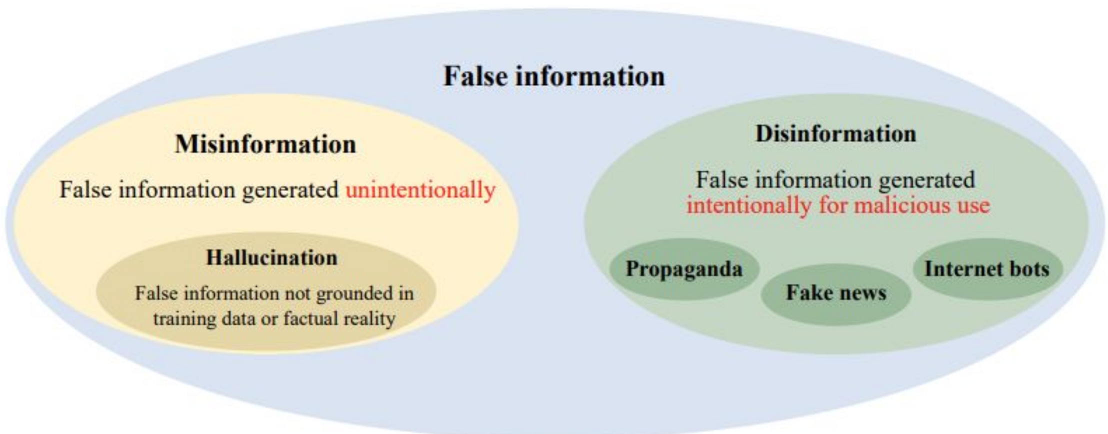


Photo from "Chua, J., Li, Y., Yang, S., Wang, C. and Yao, L., 2024. AI Safety in Generative AI Large Language Models: A Survey. *arXiv preprint arXiv:2407.18369*."

PURDUE UNIVERSITY

RAG use cases

Some examples of context information used by RAG include:

- **Real-time context** (the weather, your location, etc);
- **User-specific information** (orders the user has made at this website, actions the user has taken on the website, the user's status, etc);
- Relevant factual information (documents not included in the LLM's training data - either because they are **private** or they were updated after the LLM was trained).

RAG vs Fine-Tuning

- Compared to continuous pre-training or fine-tuning, RAG is easier and **cheaper** to keep retrieval indices up-to-date.
- If the retrieval indices have problematic documents that contain **toxic or biased content**, we can easily drop or modify the offending documents.
- RAG provides finer-grained **control** over how we retrieve documents.
    - For example, if we're hosting a RAG system for multiple organizations, by partitioning the retrieval indices, we can ensure that each organization can only retrieve documents from their own index. This ensures that we **don't inadvertently expose information from one organization to another**.

RAG vs Fine-Tuning

Selection Criteria

- **Budget:** fine-tuning involves retraining the model, which is more expensive.
- **Training vs Inference Cost:** since the weights are updated, fine-tuning requires more time commitment in the beginning but might be less time intensive in the long run.
    - RAG requires more compute during inference.

⇒ As a rule of thumb, RAG is an ideal strategy to start with. After that, if the task for the model becomes too narrow or specific, fine-tuning might be the next step.

RAG vs Fine-Tuning use cases

1. **Evolving domains with core tasks**:
    a. In medical imaging where there are standard diagnostic procedures (handled by fine-tuning) but also rapidly evolving research and new case studies (addressed by Visual RAG).
    b. **E-commerce and product recognition:** A fine-tuned model could recognize product categories, while RAG could retrieve up-to-date product information or similar items from a dynamic inventory.
    c. **Content moderation systems:** Fine-tuning can handle common violation types, while RAG can adapt to emerging trends or context-dependent violations.

LLM Training vs LLM Fine-tuning vs RAG

Some Comparisons

	Build from scratch	Finetune	RAG
Details	• data collection and preparation • design and train the model	• fine-tuning an existing model on domain-specific data	• enrich prompts using RAG by injecting domain-specific data to prompts
Pros	• accuracy	• accuracy • low-data volume	• accuracy • low-data volume • low computation cost
Cons	• data volume • high computation cost • high training time	• high computation cost	• reliance on data pipelines

Visual RAG

While traditional RAG processes text inputs and retrieves relevant textual information, Visual RAG works with images, sometimes accompanied by text, and retrieves visual data or image-text pairs.

The encoding process shifts from text encoders to vision encoders, and the knowledge base (i.e., a vector database) becomes a repository of visual information rather than text documents.

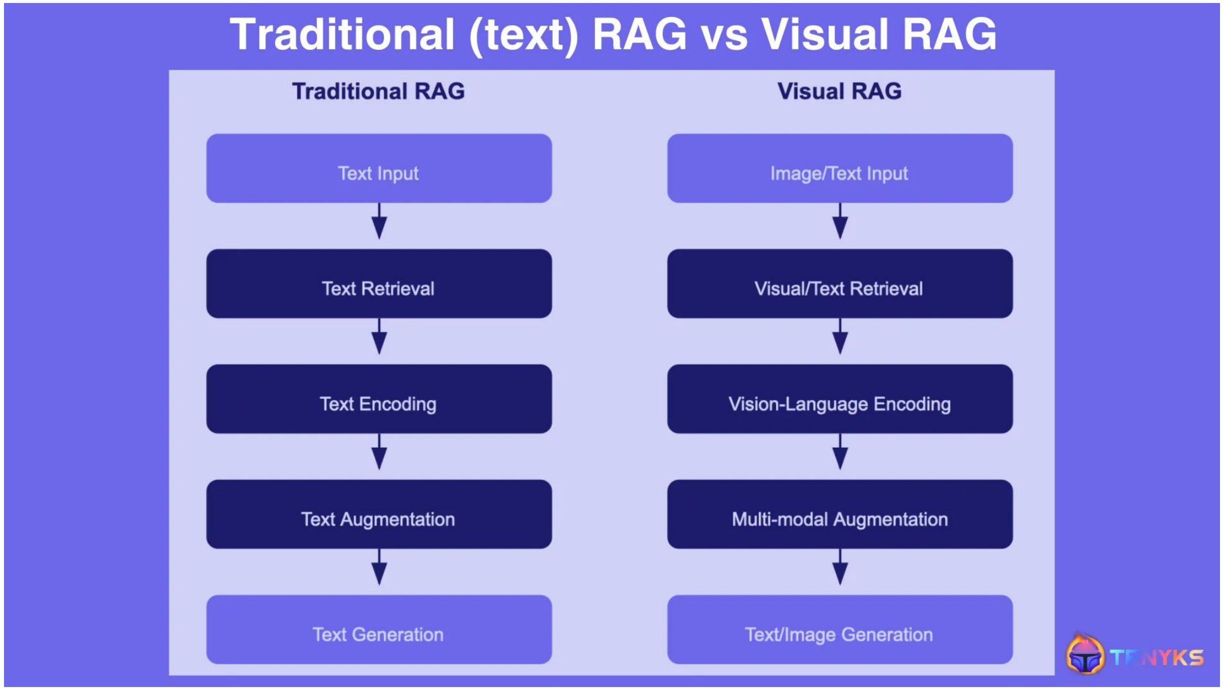


Photo from https://medium.com/@tenyks_blogger/rag-for-vision-building-genai-computer-vision-systems-6a812adfb20e

Visual RAG

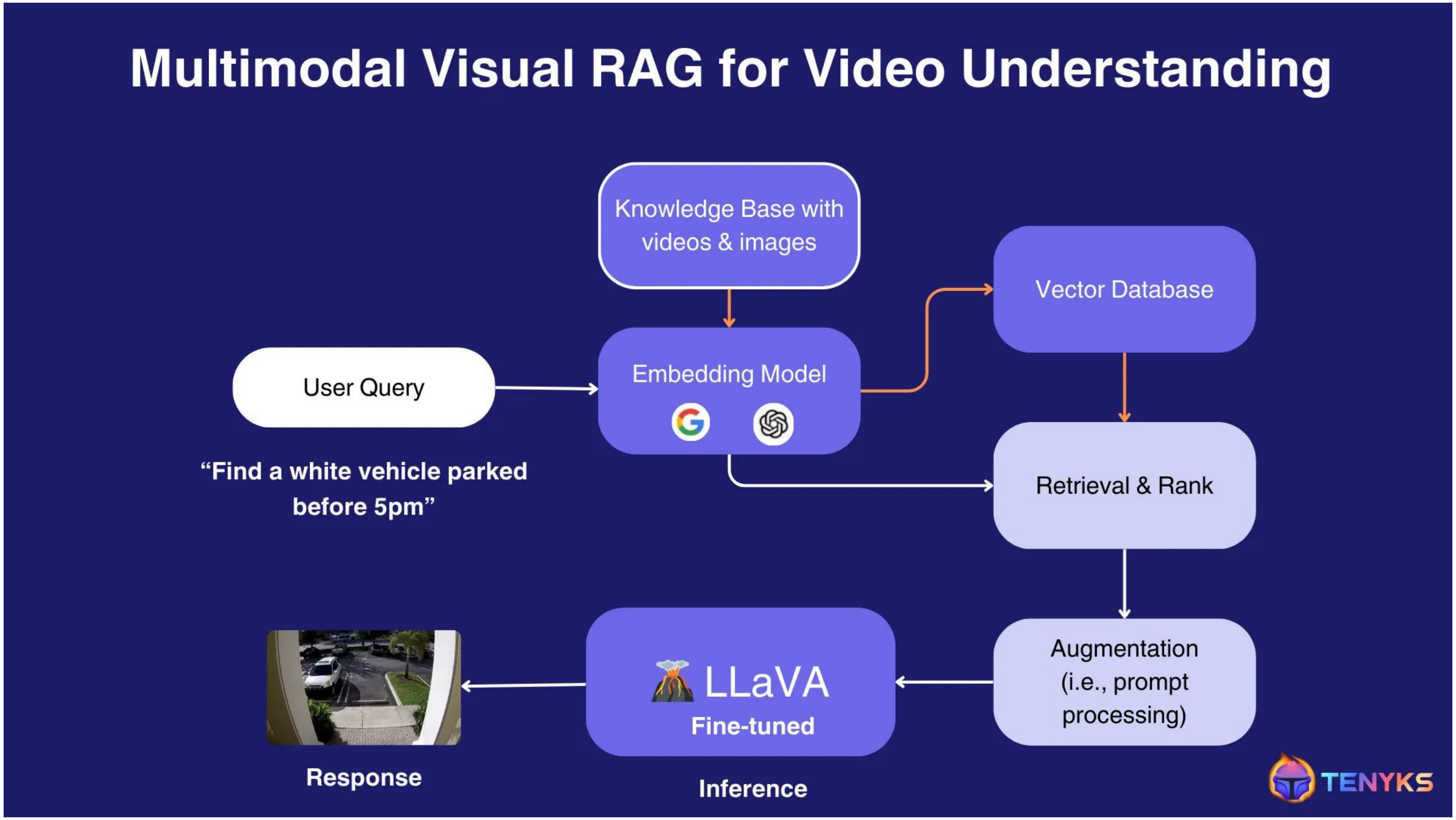


Photo from https://medium.com/@tenyks_blogger/rag-for-vision-building-genai-computer-vision-systems-6a812adfb20e

Course Outline

1. LLM Engineering
   a. Prompt Engineering
   b. Rag
   c. Agents
   d. Fine-Tuning
      i. Full fine-tuning
      ii. Instruct tuning
      iii. Reinforcement Learning with Human Feedback (RLHF)
   e. Parameter-Efficient Fine-Tuning (PEFT)
      i. Prefix/Prompt tuning
      ii. Adapters
      iii. LORA
2. MultiModal Transformers: Image, Audio, Video, 3D Transformers

Agents, An example

"Track stock's price and email me if it drops by more than 5% in a day."

A static LLM (like GPT-4 without tools or memory) **can't**:

- Access **real-time stock data**.
- Continuously **monitor changes** over time.
- **Trigger actions** (like sending an email).
- Persist memory or run code over time.

It can only explain *how* to do it, or give you a code snippet to run yourself.

An LLM **agent**, given the same goal, can autonomously:

1. Fetch current stock price (using a tool like Yahoo Finance API).
2. Store it and set a timer to check again later.
3. Compare current price to past price.
4. Decide: Has the price dropped >5%?
5. If yes → Compose an email and send it via email API.
6. Repeat this check on a schedule.

Agents can use Tools:

- Web API tool (for stock prices)
- Math tool (to calculate % drop)
- Memory (to track previous prices)
- Email tool (to send alerts)

Why Agents?

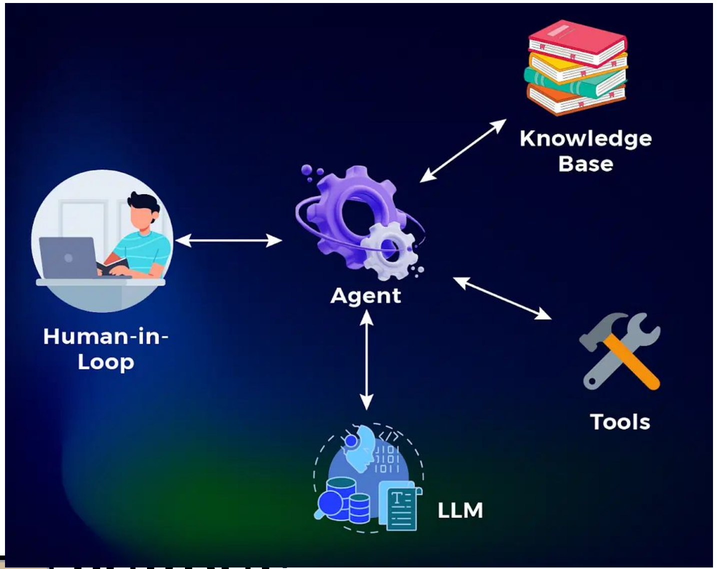
LLMs are powerful but limited by:

- No persistent memory
- No real-time access to tools or external data
- No autonomy—they only respond to prompts

Agents overcome this by:

- Using tools (like code interpreters, web search, file systems)
- Retaining memory/context across tasks
- Making decisions via planning or rules (e.g., ReAct or Tree of Thought frameworks)

⇒ **LLM Agents = LLM + Memory + Tools.**



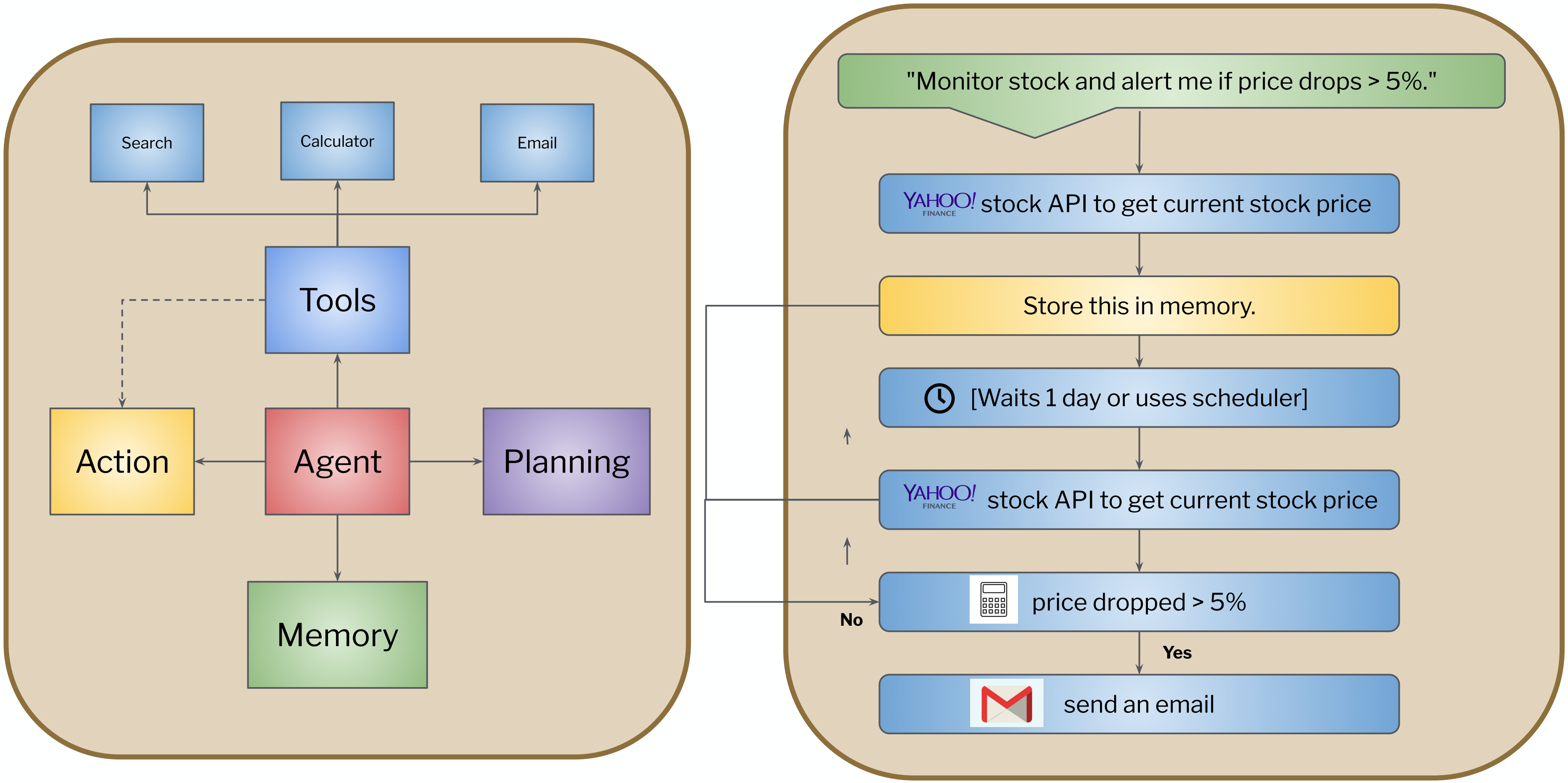
Real Examples

- **Customer Support Agent**: Uses knowledge base + email API to respond and resolve tickets.
- **Personal Assistants**
- **Coding Agent**: Writes, runs, debugs code autonomously.
  Data Analyst Agent: Pulls from a database, creates charts, interprets trends.

PURDUE UNIVERSITY

https://www.projectpro.io/article/llm-agents/1013

Agents, An example



LLM vs RAG vs Agents…

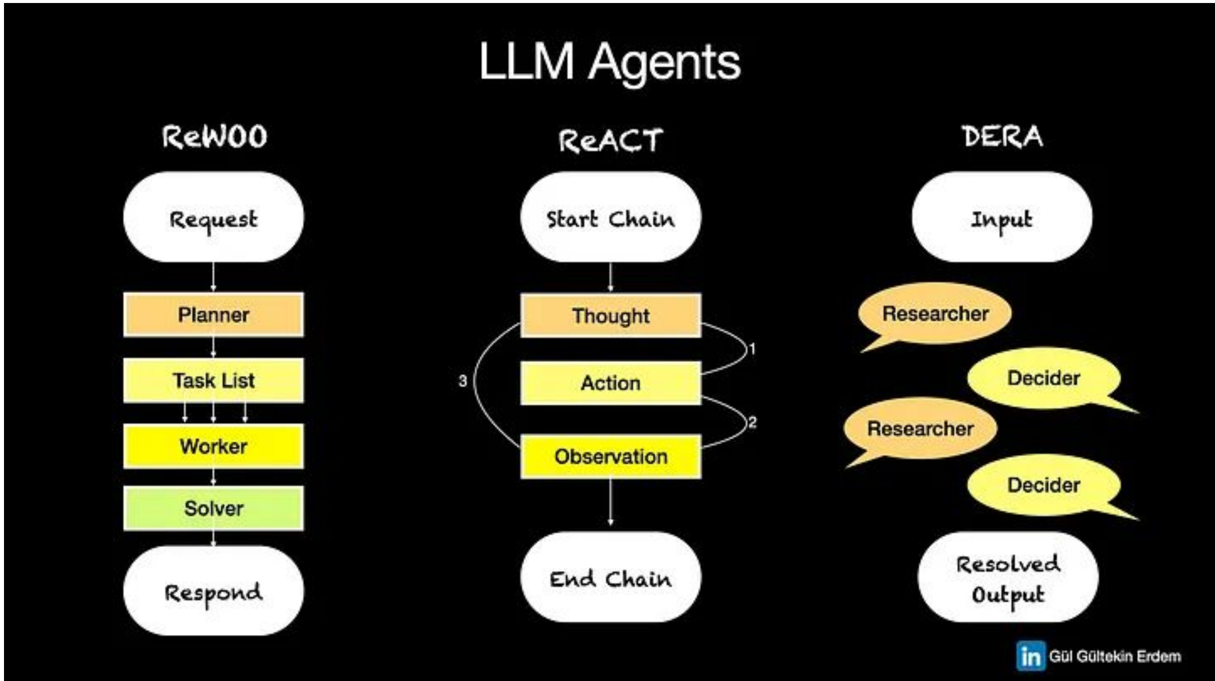
Capability / Task Requirement	Plain LLM	RAG	LLM Agent
Access real-time stock data	❌	❌ (no live API)	via API/tool use
Retain memory across time (track price over days)	❌	❌	via persistent memory
Calculate % drop and apply logic	Basic	Basic	can reason and decide
Trigger external actions (send email)	❌	❌	via tool/API
Run autonomously over time (scheduled checks)	❌	❌	via planning/execution
Retrieve relevant knowledge (e.g., from docs or PDFs)	❌	yes	If combined with RAG
Good for factual Q&A or document search	Basic	yes	yes
Acts independently without constant human input	❌	❌	yes

Agents

Planning, Tools use, combination

- **ReWoo (Reasoning with Workspaces):** Best for reasoning where **intermediate results are shared and reused** (e.g., planning, math).
- **ReAct (Reasoning + Acting)**: Best for **step-by-step agents that think, act, observe, repeat** (e.g., question answering with tools).
- **DeRA (Decomposed Reasoning Agent):** Tasks requiring explicit decomposition with **natural dialogue** + tool use

Photo from https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6



PURDUE UNIVERSITY

Selection Criteria: *RAG, Agents, Advanced Prompting*

- If you need to access any **external data source**, use **RAG**.
- If you need an **autonomous entity** that **make decisions** and **take action** on its own, then consider **LLM Agents**.
- If you don't need any of them, but want to use LLM to do a more **complex task than you can solve with a simple prompting**, employ **advanced prompting techniques.**

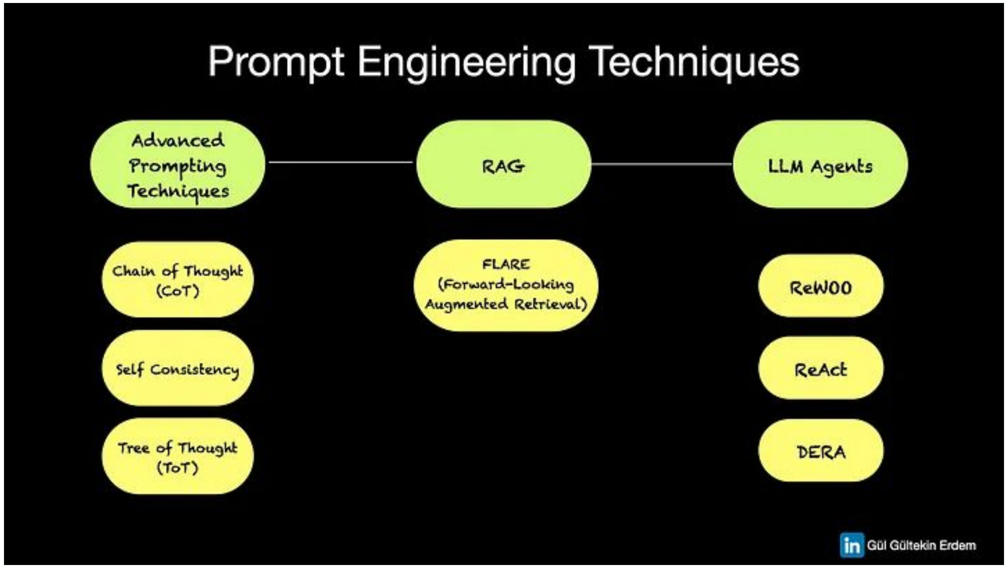


Photo from https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6

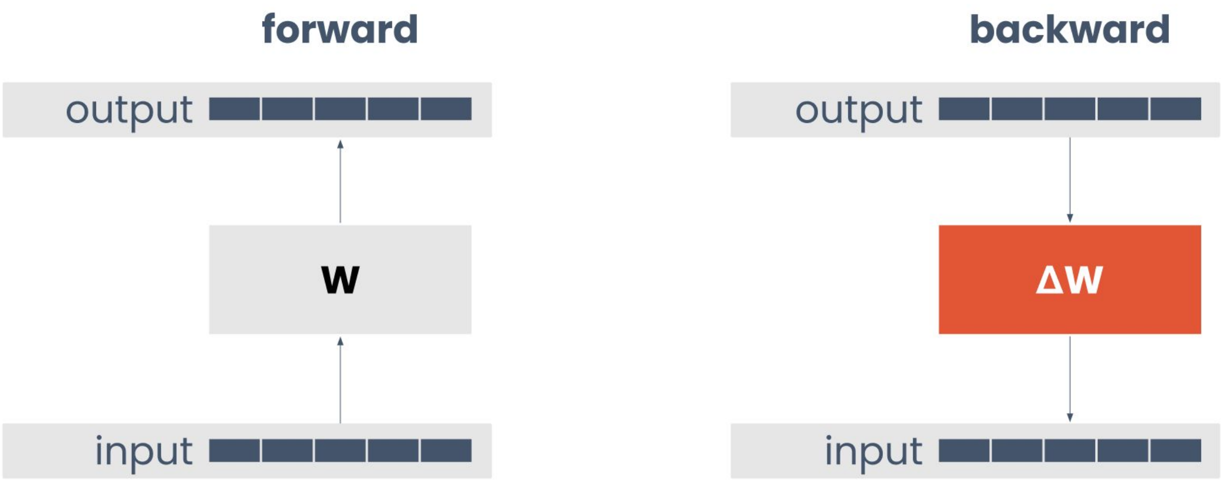
Course Outline

1. LLM Engineering
    a. Prompt Engineering
    b. Rag
    c. Agents
    d. Fine-Tuning
        i. Full fine-tuning
        ii. Instruct tuning
        iii. Reinforcement Learning with Human Feedback (RLHF)
    e. Parameter-Efficient Fine-Tuning (PEFT)
        i. Prefix/Prompt tuning
        ii. Adapters
        iii. LORA
2. MultiModal Transformers: Image, Audio, Video, 3D Transformers

Fine-Tuning

Showing your LLM new data and altering its weights

- Fine-tuning plays a vital role when dealing with extensive pre-trained open-source models.
- The process of fine-tuning involves updating the model's parameters by **feeding it new task-specific data and adjusting its weights based on the expected output**.
- Through **backpropagation**, a loss is calculated and the model's weights are adjusted to improve its performance on similar inputs in the future, aiming to enhance its task-specific performance without compromising its overall capabilities.



Full Fine-Tuning Cons

- Full fine-tuning entails adjusting all the model's weights.
  - This can be **slow** as it requires computing gradients across all weights to minimize the loss between the actual output and the model's generated output. This comprehensive modification also involves **billions of floating-point computations** and **constant data movement within the GPU memory hierarchy**.
  - Fine-tuning can also be **memory-intensive** as it requires **storing both gradients and optimizer states**, effectively doubling the memory requirements. Consequently, if your model fits within an NVIDIA A10G GPU, training may necessitate four such GPUs.
  - Given their enormous size, one needs extremely **big computing power** and **large scale datasets** to fine tune them on a specific task.
  - Fine-tuning LLMs on specific task may lead them to “forget” previously learnt information, a phenomena known as **catastrophic forgetting.**

https://www.labellerr.com/blog/fasten-up-your-data-annotation-process-with-pre-trained-models/

Partial Fine-Tuning

- Employ a model that has already been trained and fine-tune it partially.
    - **Training some layers while freezing others.**
- All we can do is **freeze the weights of the model's first layers** while only **retraining the upper levels**.
- How many layers should be frozen and trained can be experimented with?

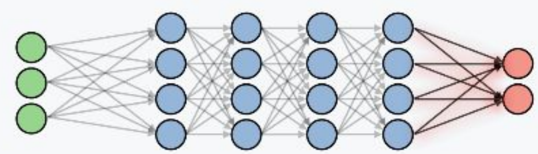
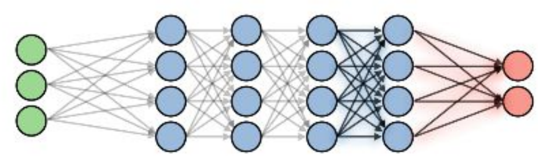
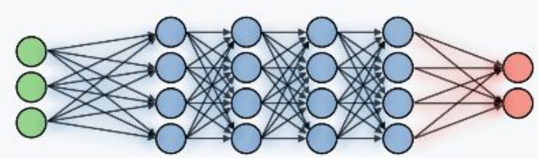
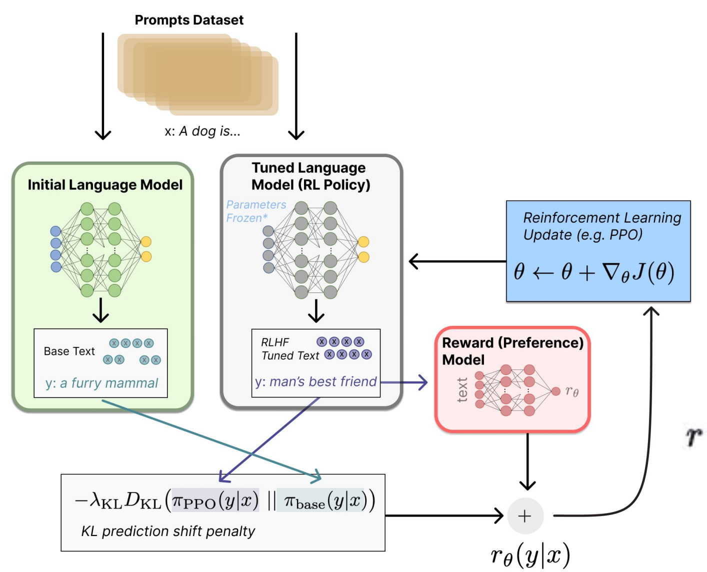
Illustration	Explanation
	Freezes all layers, trains weights on softmax
	Freezes most layers, trains weights on last layers and softmax
	Trains weights on layers and softmax by initializing weights on pre-trained ones

Photo from: https://www.labellerr.com/blog/fasten-up-your-data-annotation-process-with-pre-trained-models/

Reinforcement Learning with Human Feedback(RLHF)

- Most language models are still trained with a simple next token prediction loss (e.g. cross entropy).
- Wouldn't it be great if we use human feedback for generated text as a measure of performance or go even one step further and use that feedback as a loss to optimize the model?
- That's the idea of Reinforcement Learning from Human Feedback (RLHF); use methods from reinforcement learning to directly optimize a language model with human feedback.
- RLHF has enabled language models to begin to align a model trained on a general corpus of text data to that of complex human values.



$$r = r_\theta - \lambda r_{KL}$$

PURDUE UNIVERSITY

Photo from https://huggingface.co/blog/rlhf with some modifications

RLHF (Reinforcement Learning with Human Feedback)

- RLHF is a specific technique that is used in training AI systems to appear more human, alongside other techniques such as supervised and unsupervised learning.
- **A human assesses the quality of different responses from the machine, scoring which responses sound more human.**
- The score can be based on innately human qualities, such as **friendliness**, the right degree of contextualization, and **mood**.

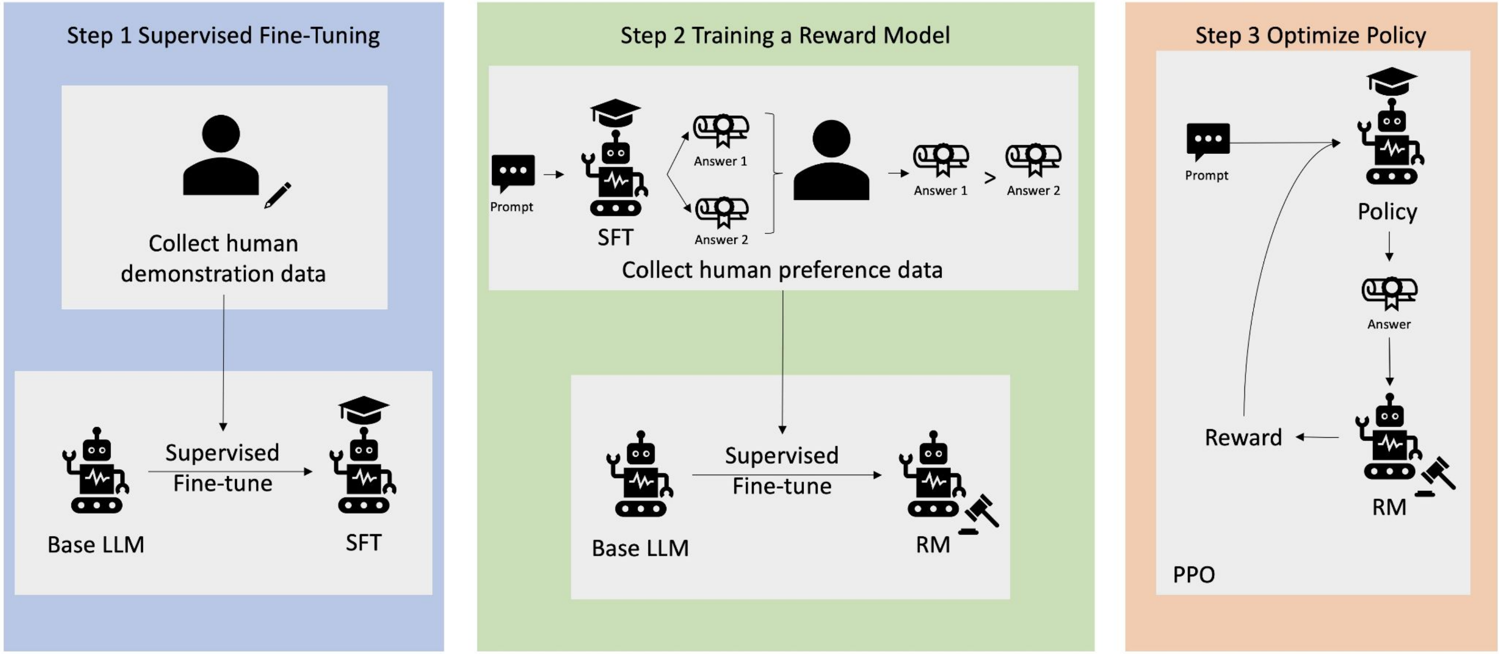


Photo from https://aws.amazon.com/what-is/reinforcement-learning-from-human-feedback/

Instruct Fine-Tuning

- *Instruction tuning* is a technique for fine-tuning LLMs on a labeled dataset of instructional prompts and corresponding outputs.
- **It improves model performance not only on specific tasks, but on following instructions in general.**
- The utility of instruction tuning lies in the fact that **pre-trained LLMs are not optimized for conversations or instruction following**. **In a literal sense, LLMs do not *answer* a prompt: they only *append text to it.*** Instruction tuning helps make that appended text more useful.

Example

Task: Rewrite the sentence "The cat sat on the mat." in a formal tone.

Output by next word prediction models: "The cat sat on the mat and then walked to the window."

Output by instruct tuning models: "The feline was positioned on the carpeted surface."

Instruct Fine-Tuning

- This method focuses on training models **to better understand and follow instructions**, as well as comprehend the relationships between task elements.
  - Instruction Tuning aims to finally resolve the long-standing **issue of cross-task generalization,** enabling models to adapt to new tasks more effectively

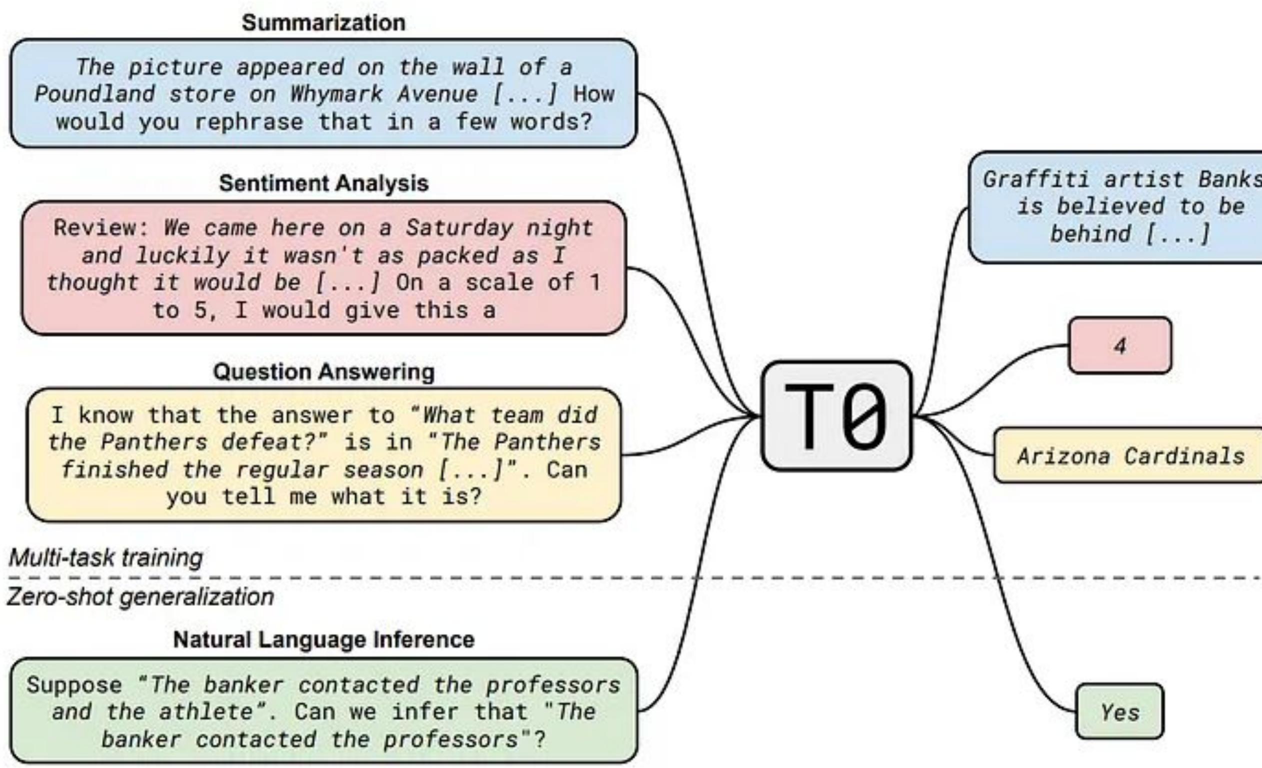


Photo from [39]

Course Outline

1. LLM Engineering
    a. Prompt Engineering
    b. Rag
    c. Agents
    d. Fine-Tuning
        i. Full fine-tuning
        ii. Instruct tuning
        iii. Reinforcement Learning with Human Feedback (RLHF)
    e. Parameter-Efficient Fine-Tuning (PEFT)
        i. Prefix/Prompt tuning
        ii. Adapters
        iii. LORA
2. MultiModal Transformers: Image, Audio, Video, 3D Transformers

Soft Prompting vs Hard Prompting

There are two categories of prompting methods:

- Hard prompts are manually handcrafted text prompts with discrete input tokens; the downside is that it requires a lot of effort to create a good prompt
- Soft prompts are **learnable tensors concatenated with the input embeddings** that can be optimized to a dataset; the downside is that they aren't human readable because you aren't matching these "virtual tokens" to the embeddings of a real word
    a. A disadvantage is the **lack of interpretability of soft prompts**.
    b. Unlike hard prompts, soft prompts **cannot be viewed and edited in text**. Prompts consist of an embedding, a string of numbers, that derives knowledge from the larger model.

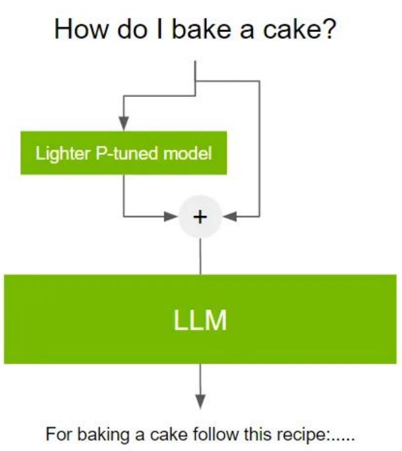


Photo from https://cobusgreyling.medium.com/prompt-tuning-hard-prompts-soft-prompts-49740de6c64c

Soft Prompting and Prompt Tuning

- Prompt tuning involves using a small trainable model before using the LLM. The small model is used to encode the text prompt and generate task-specific virtual tokens.
- **Prompt tuning created a smaller light weight model which sits in front of the *frozen* pre-trained model**. Hence soft prompts via prompt tuning is an additive method for only training and adding prompts to a pre-trained model.
- The trainable tensor (known as "soft prompt") would learn the task specific details. The soft prompt is optimized through gradient descent. In this approach rest of the model architecture remains unchanged.
- Soft prompts are created during the process of prompt tuning.

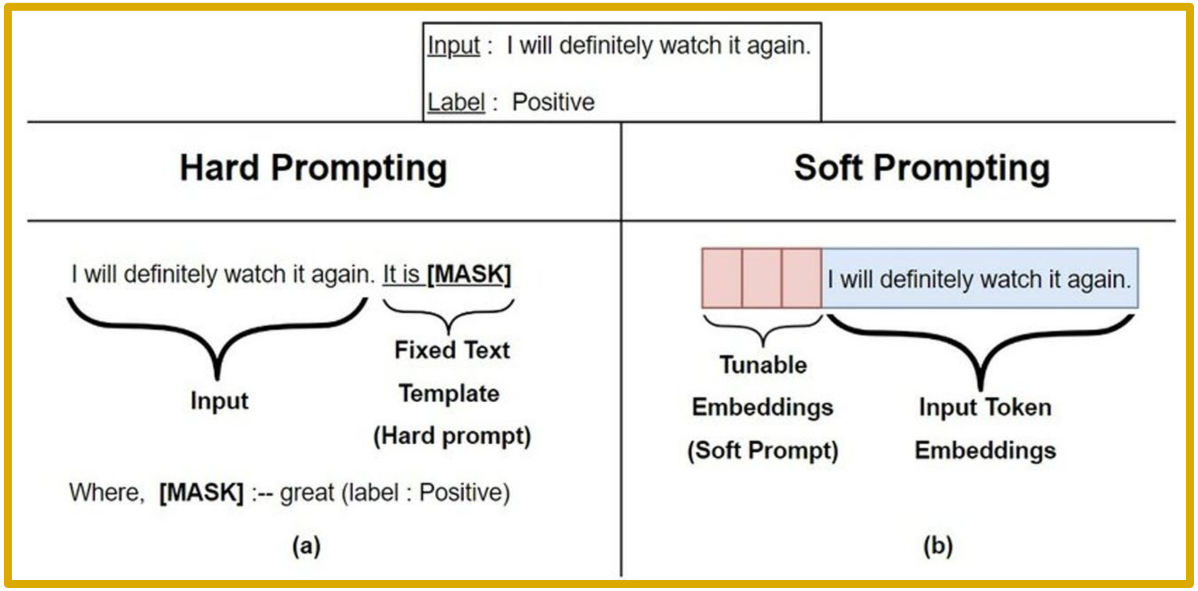


Photo from Senadeera, D., and Ive, J. Controlled Text Generation using T5 based EncoderDecoder Soft Prompt Tuning and Analysis of the Utility of Generated Text in AI. Dec. 2022.

Prefix Tuning

- Prefix Tuning is a similar approach to Prompt Tuning.
- Instead of adding the prompt tensor to only the input layer, prefix tuning adds trainable parameters are prepended to the hidden states of all layers.

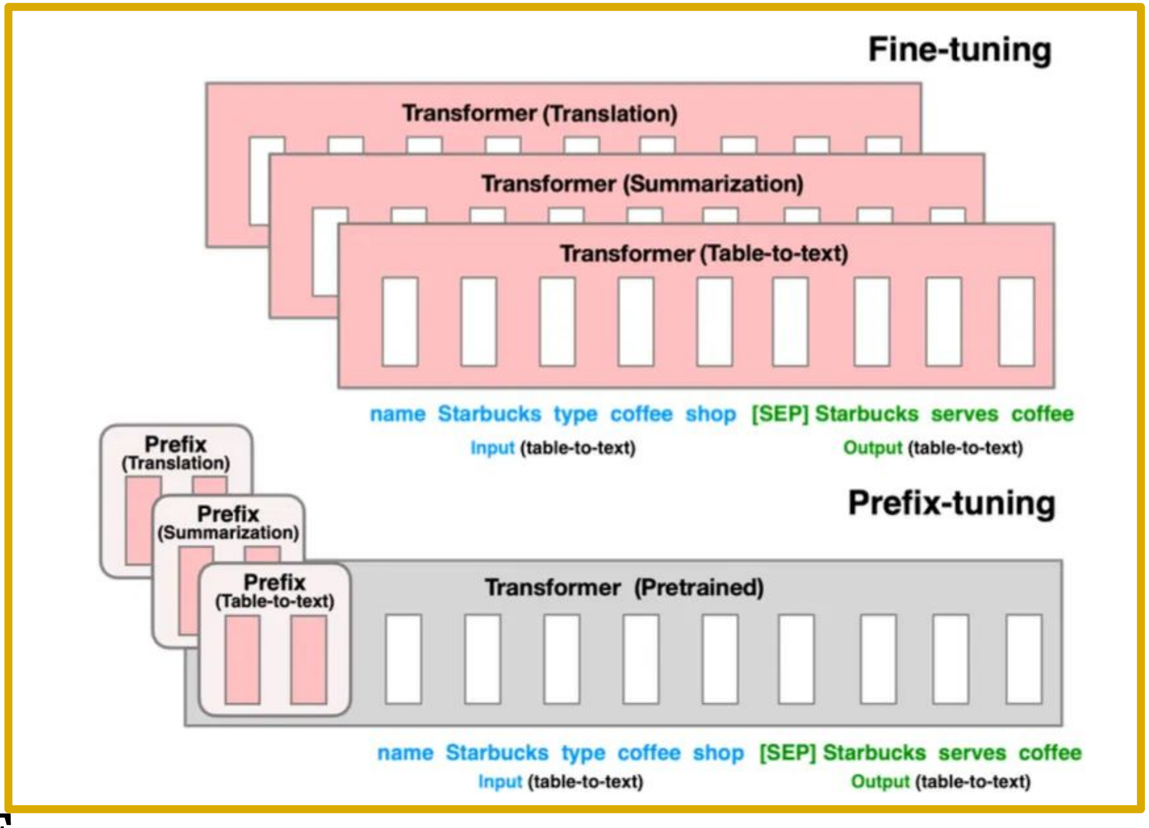


Photo from https://arxiv.org/abs/2101.00190

Adapters

Full Fine-Tuning

- High time, High memory, High computation.
- Fine-tuning requires storing both gradients and optimizer states, effectively doubling the memory requirements.
    - For instance, when using the Adam optimizer, a common rule of thumb suggests allocating **three times the GPU RAM as the model size in memory during the backward pass**.

Adapters

- Adapters allow the LLM to adapt to new scenarios **without changing its original parameters**.
- This **preserves the LLM's general knowledge and avoids catastrophic forgetting** when learning new tasks.
- Adapters can also reduce the number of parameters that need to be updated, reducing the time and compute required.
- **Training overhead can be reduced up to 70%** compared to full fine-tuning.

Adapters

- Adapters are new modules added between layers of a pre-trained network.
- In Adapter based learning ***only the new parameters are trained while the original LLM is frozen***, hence we learn a very small proportion of parameters of the original LLM.
- This means that ***the model has perfect memory of previous tasks*** and used a small number of new parameters to learn the new task.

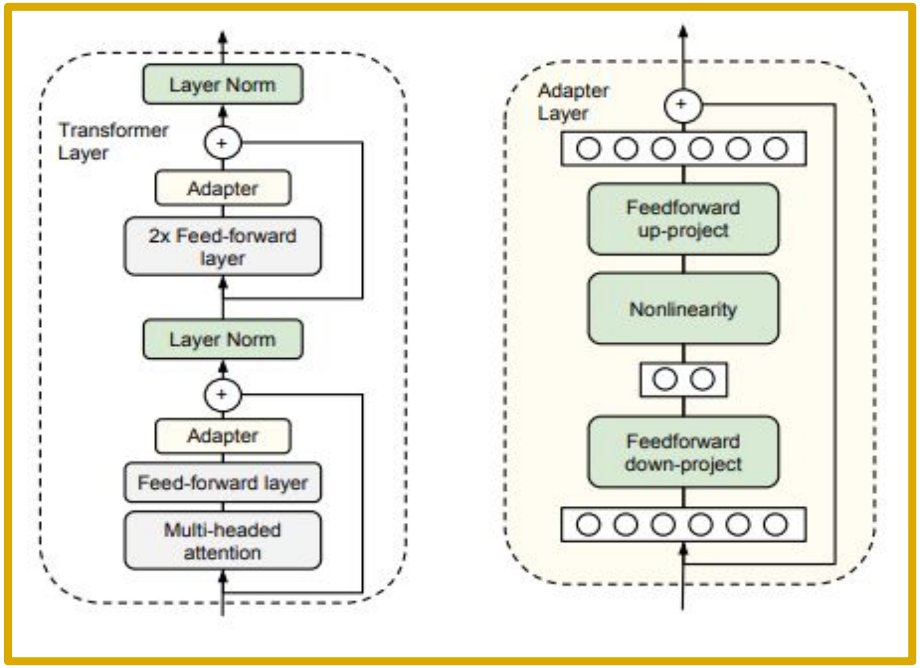


Photo from https://medium.com/dair-ai/adapters-a-compact-and-extensible-transfer-learning-method-for-nlp-6d18c2399f62

Adapters

- Authors suggest that Adapters on the lower layers have a smaller impact than the higher-layers.
  - Removing the adapters from the layers $0 - 4$ **barely affects performance**. Focusing on the upper layers is a popular strategy in fine-tuning.
  - One intuition is that the lower layers extract lower-level features that are shared among tasks, while the **higher layers build features that are unique to different tasks**.

LoRA: Low-Rank Adaptation of Large Language Models

- LoRA freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture,
  - greatly reducing the number of trainable parameters for downstream tasks. Compared to GPT-3 175B fine-tuned with Adam, **LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times**.
- LoRA performs **on-par or better than fine-tuning** in model quality, despite having fewer trainable parameters.
- **Unlike adapters, LoRA has no additional inference latency.**

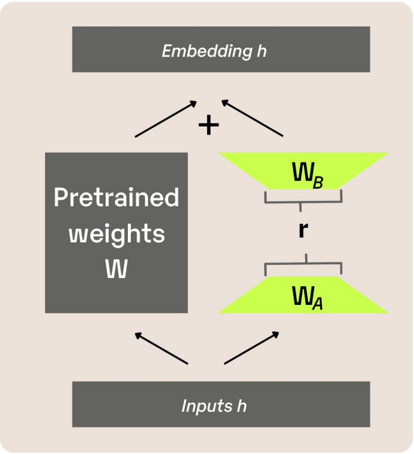


Photo from www.inferless.com

PURDUE UNIVERSITY

LoRA

1. No trade-off in performance, and even outperforming in some tasks

2. Extremely low memory footprint: 0.2%

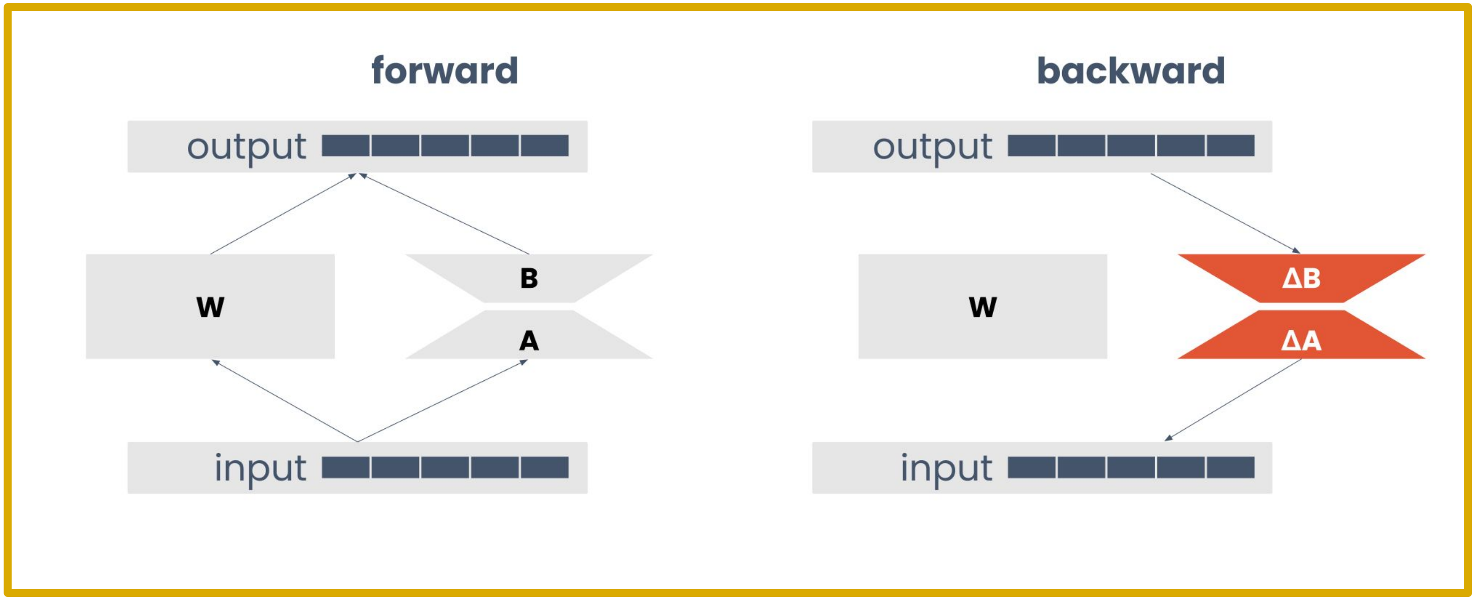
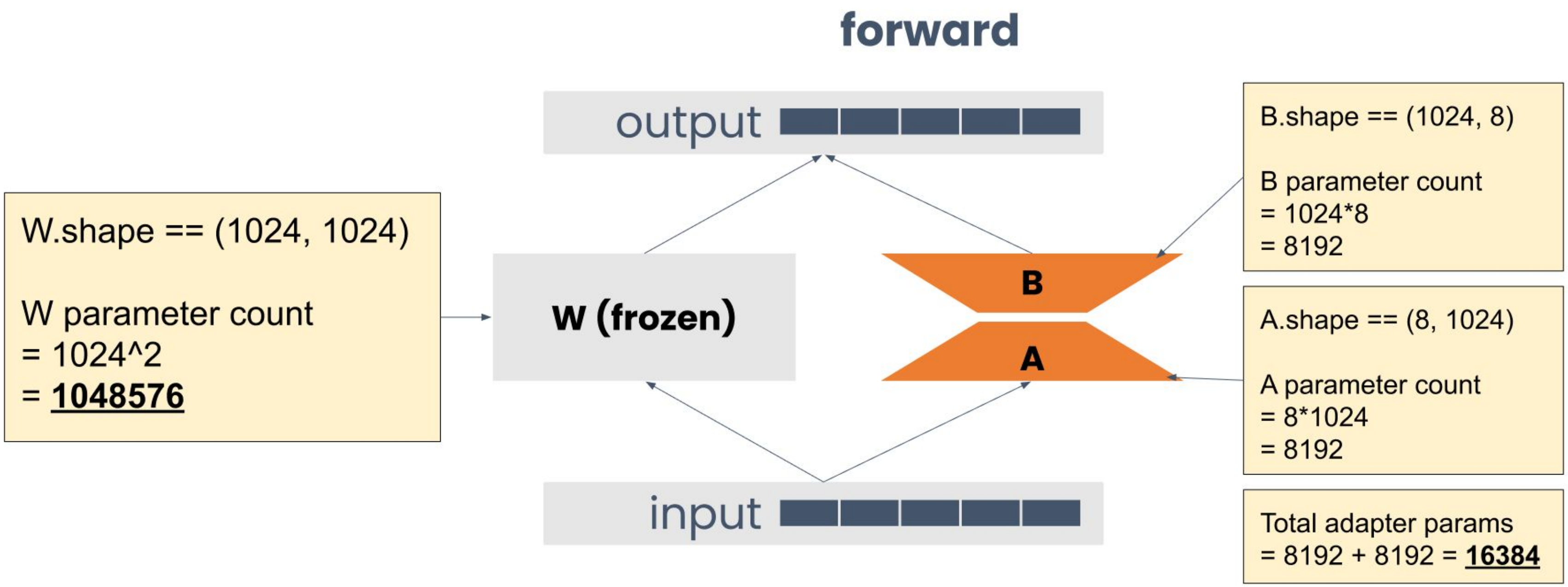


Photo from
https://medium.com/@infin94/understanding-the-seq2seq-model-what-you-should-know-before-understanding-transformers-e5891bcd57ec

PURDUE UNIVERSITY

LoRA Memory Footprint

- **With a reduced memory usage, the batch size can be increased, thereby accelerating model training.**
- This adjustment can make your training significantly faster by better utilizing your existing resources.



PURDUE UNIVERSITY

LoRA Intuition

LORA Authors (2021): We take inspiration from Li et al. (2018a); **Aghajanyan et al. (2020)** which show that the learned **over-parametrized models in fact reside on a low intrinsic dimension**. We hypothesize that the **change in weights during model adaptation also has a low "intrinsic rank"**, leading to our proposed Low-Rank Adaptation (LoRA) approach. LoRA allows us to train some dense layers in a neural network indirectly by optimizing rank decomposition matrices of the dense layers' change during adaptation instead, while keeping the pre-trained weights frozen.

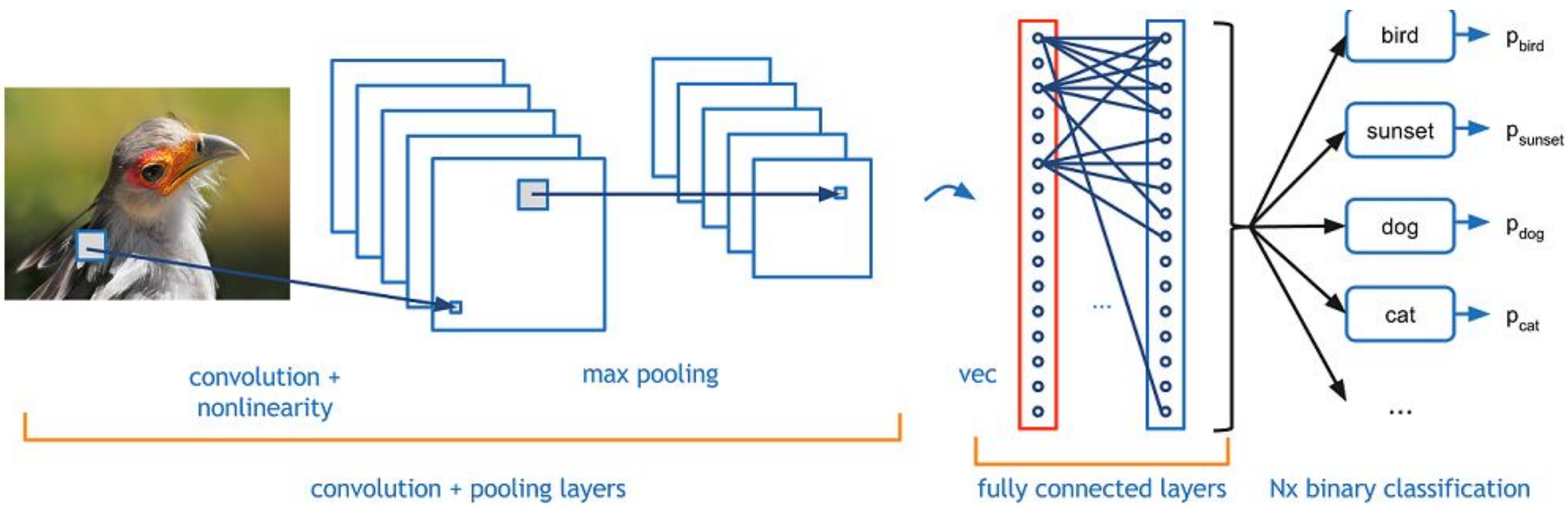
Aghajanyan et al, 2020: We empirically show that common **pre-trained models have a very low intrinsic dimension**; in other words, there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space. For example, by optimizing only 200 trainable parameters randomly projected back into the full space, we can tune a RoBERTa model to achieve 90% of the full parameter performance levels on MRPC. Furthermore, **we empirically show that pre-training implicitly minimizes intrinsic dimension and, perhaps surprisingly, larger models tend to have lower intrinsic dimension after a fixed number of pre-training updates**, at least in part explaining their extreme effectiveness.

Course Outline

1. LLM Engineering
   a. Prompt Engineering
   b. Rag
   c. Agents
   d. Fine-Tuning
      i. Full fine-tuning
      ii. Instruct tuning
      iii. Reinforcement Learning with Human Feedback (RLHF)
   e. Parameter-Efficient Fine-Tuning (PEFT)
      i. Prefix/Prompt tuning
      ii. Adapters
      iii. LORA
2. MultiModal Transformers: Image, Audio, Video, 3D Transformers

Computer Vision

Computer vision is a field of artificial intelligence (AI) that teach computers to derive meaningful information from digital images, videos and other visual input.



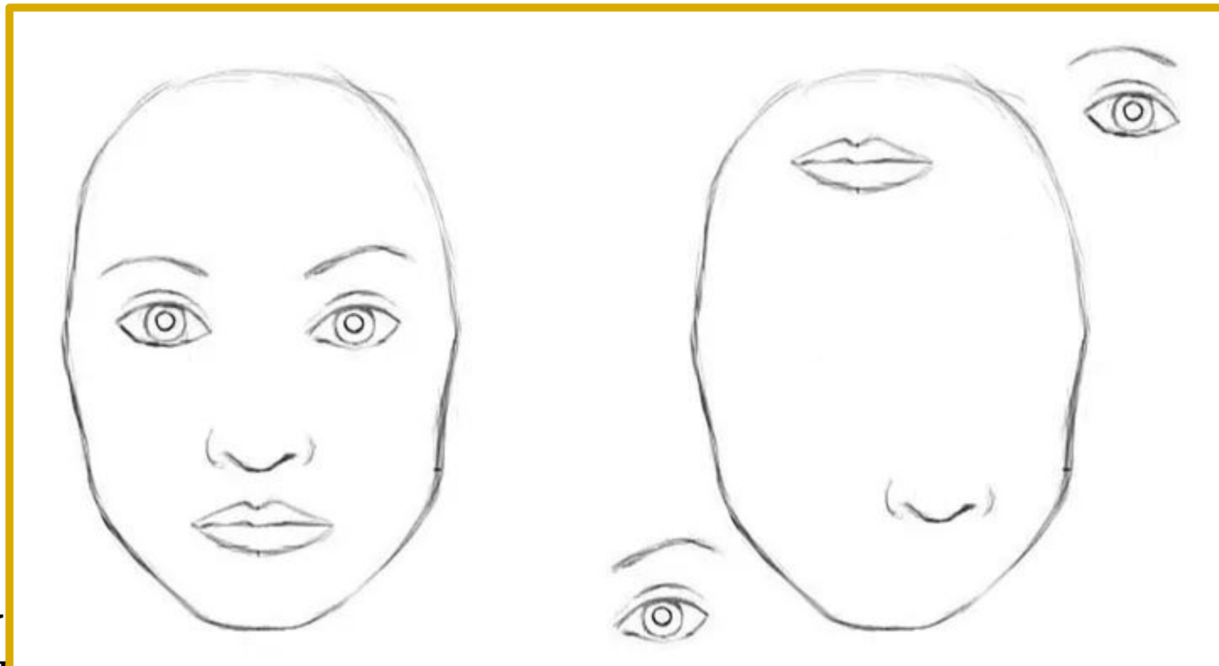
https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

PURDUE UNIVERSITY

From CNNs to Vision Transformers

For a CNN, both of these pictures are almost same.

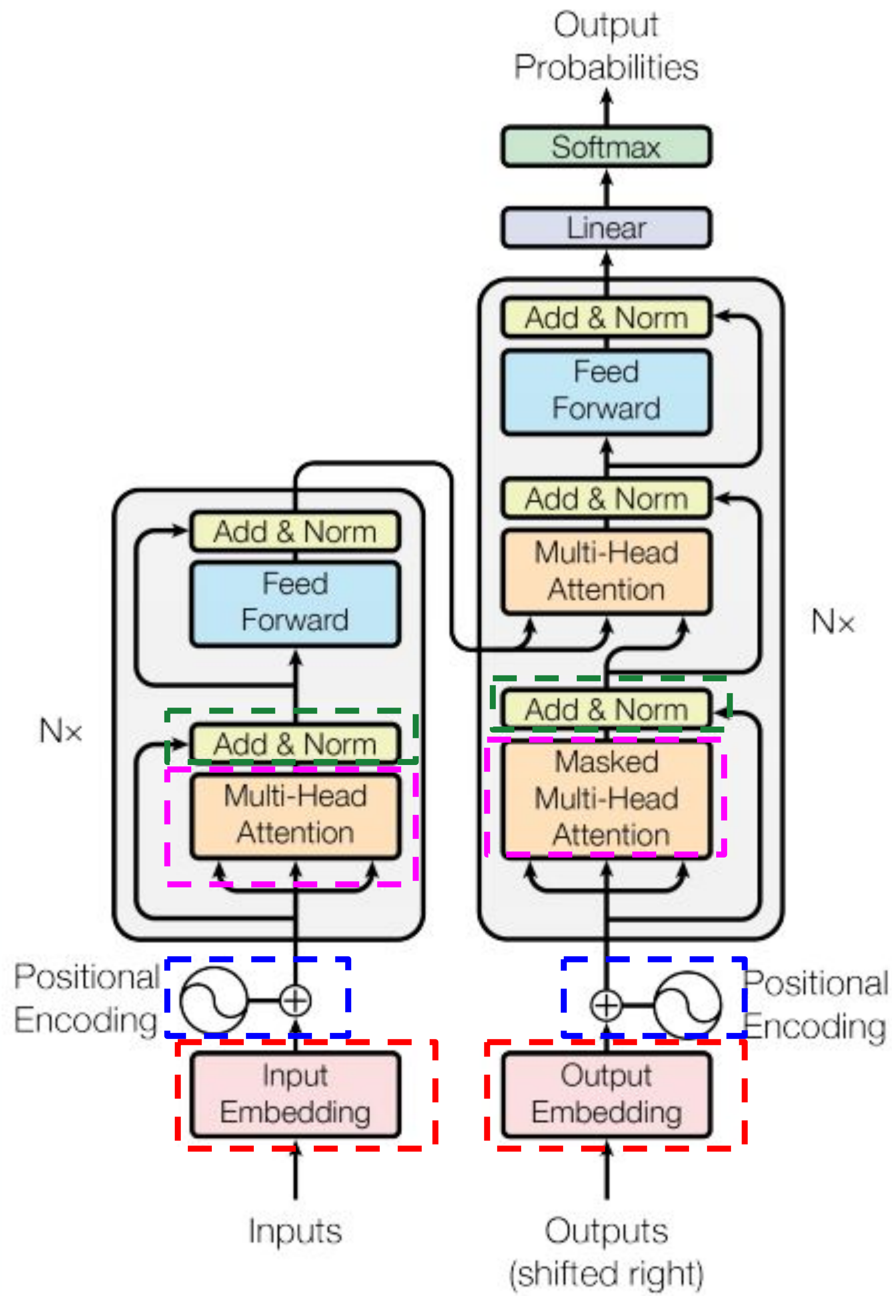
- **CNN does not encode the relative position of different features.**
- Large filters are required to encode the combination of these features.
- For examples:- to encode the information “eyes above nose and mouth” require large filters.



Transformer Details

1. Tokenization and Embedding **(Breaking your input to smaller units)**
2. Positional Encoding (**Extracting locality and neighborhoods in the input units**)
3. Residual Addition an Normalization
4. Attention
   a. self vs cross attention,
   b. single vs multi head attention,

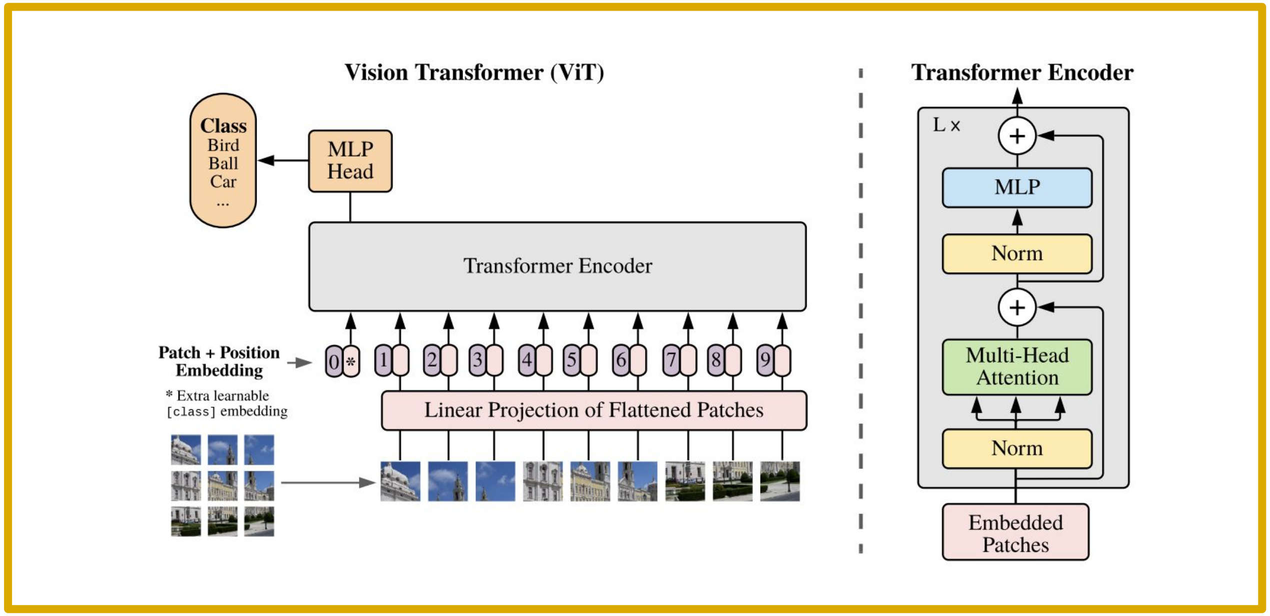
In text, we used **sub-words as units, and their indices for positional information**.



PURDUE UNIVERSITY

Transformers for Vision

- First split an image into **fixed-size patches,** linearly **embed** each of them, add **position embeddings**, and feed the resulting sequence of vectors to a **standard Transformer encoder.**
- **The image patches are being used same as sentence tokens.**
- In order to perform classification, use the standard approach of adding an extra learnable "classification token" to the sequence
- The model is pre-trained on both image-classification and patch embedding prediction.



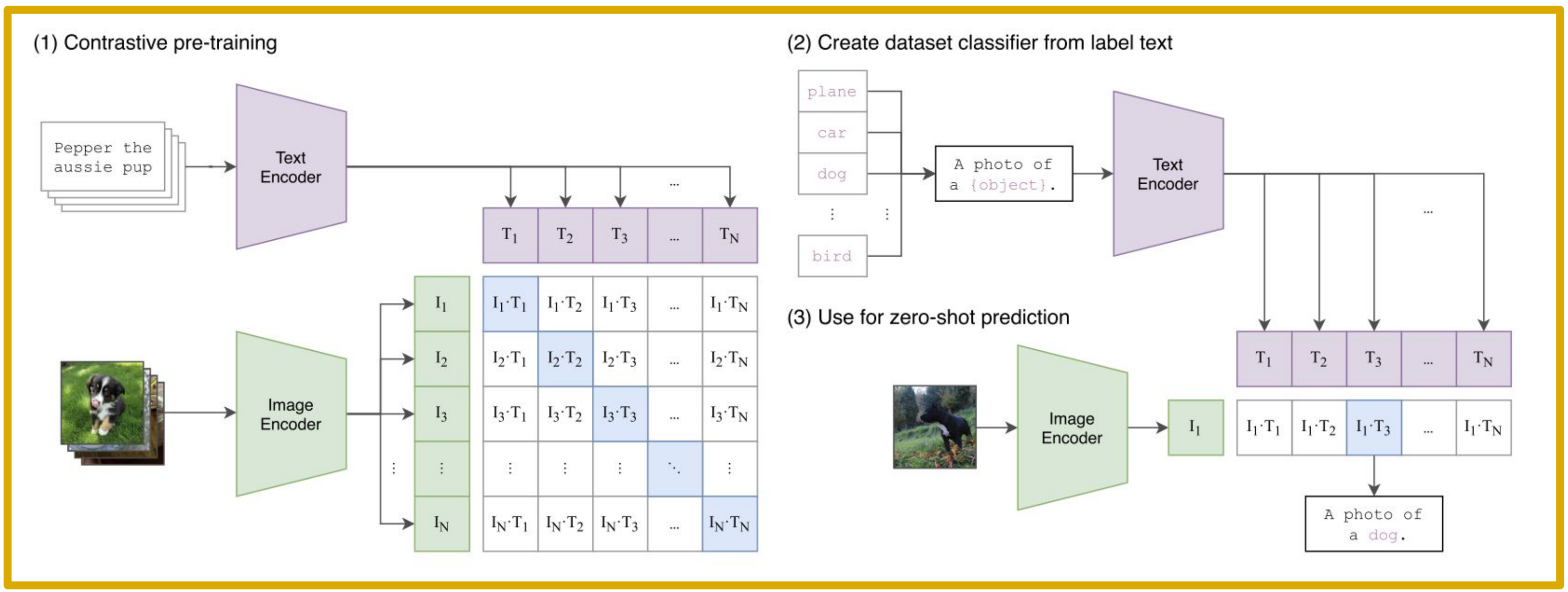
PURDUE UNIVERSITY

Photo from ViT, ICLR2021 (AN IMAGE IS WORTH 16X16 WORDS:TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE)

Transformers for Image-Text Integration

- CLIP learns a multi-modal embedding space by jointly training an image encoder and text encoder to **maximize the cosine similarity of the image and text embeddings of the N real pairs** in the batch while minimizing the cosine similarity of the embeddings of the (N^2 −N) incorrect pairings.

- At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.



PURDUE UNIVERSITY

Photo from CLIP paper, ICML2021, Learning Transferable Visual Models From Natural Language Supervision

Transformers for speech data

- We use CNN+quantizer **to generate a limited set of speech units** to help reconstructing and choosing output units.
- These quantized units (q) are equivalent to tokens embeddings in NLP transformers.
- The training paradigm is **masked token** learning.

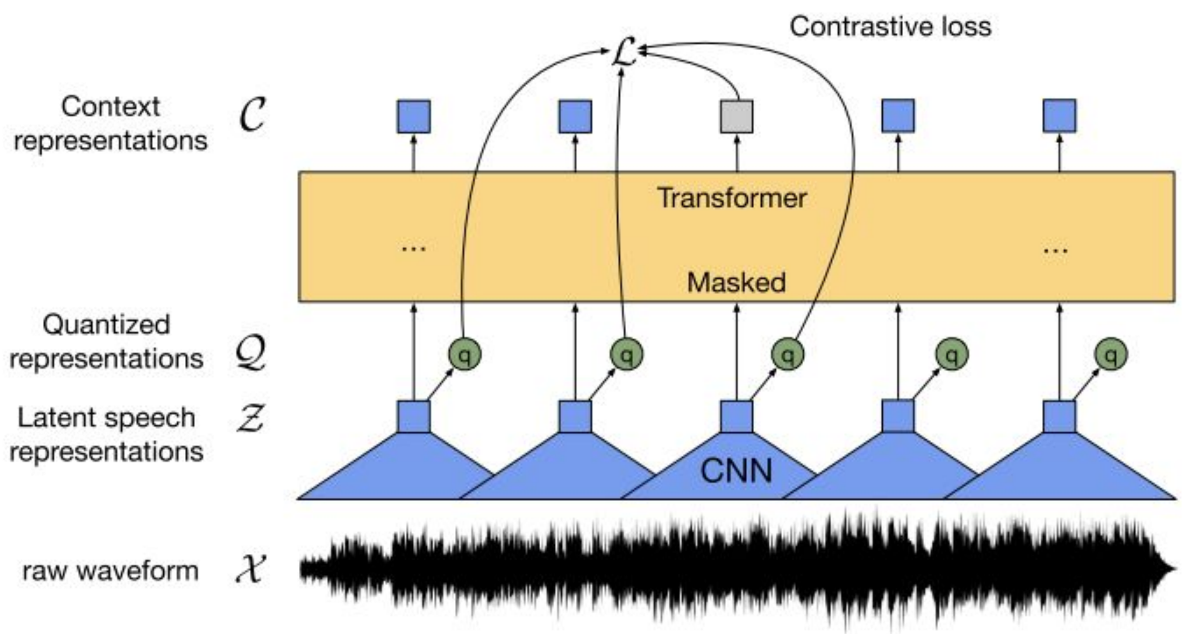


Photo from wave2vec 2.0 paper

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)}$$

PURDUE UNIVERSITY

Transformers for Video data

Each video has 3 dimensions, 2 from image frame, and one through time.

- How many options for self attention?

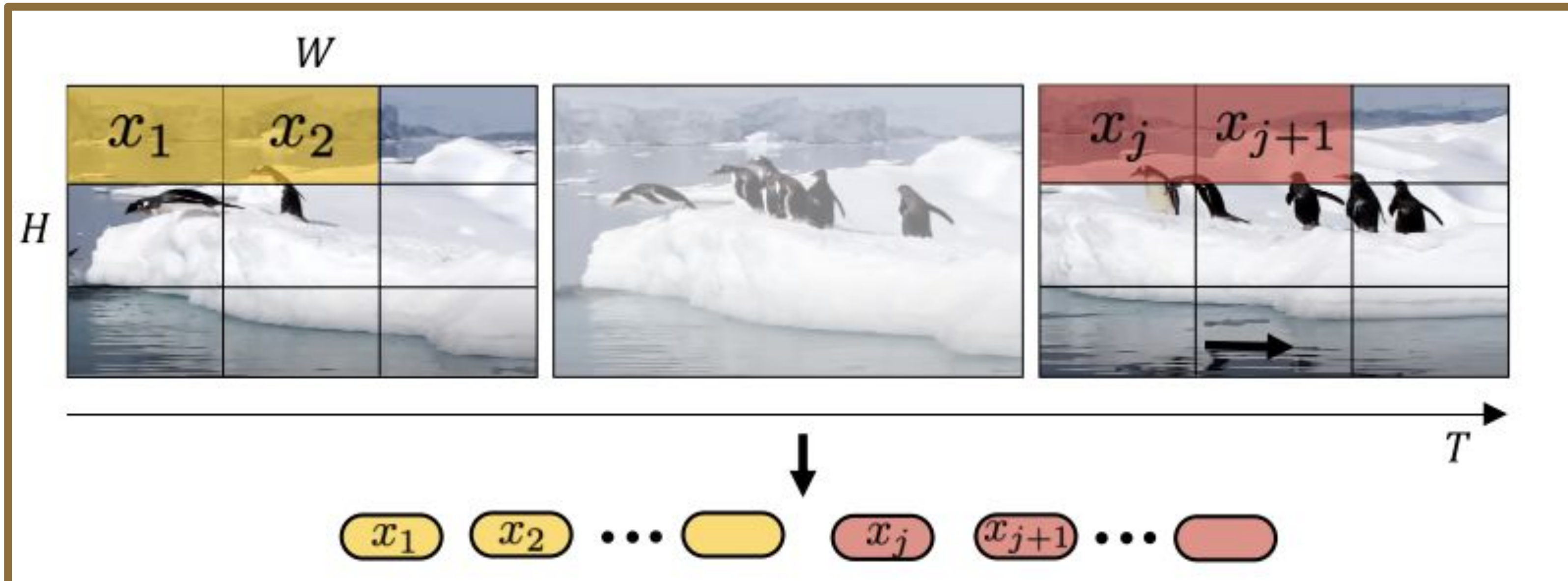
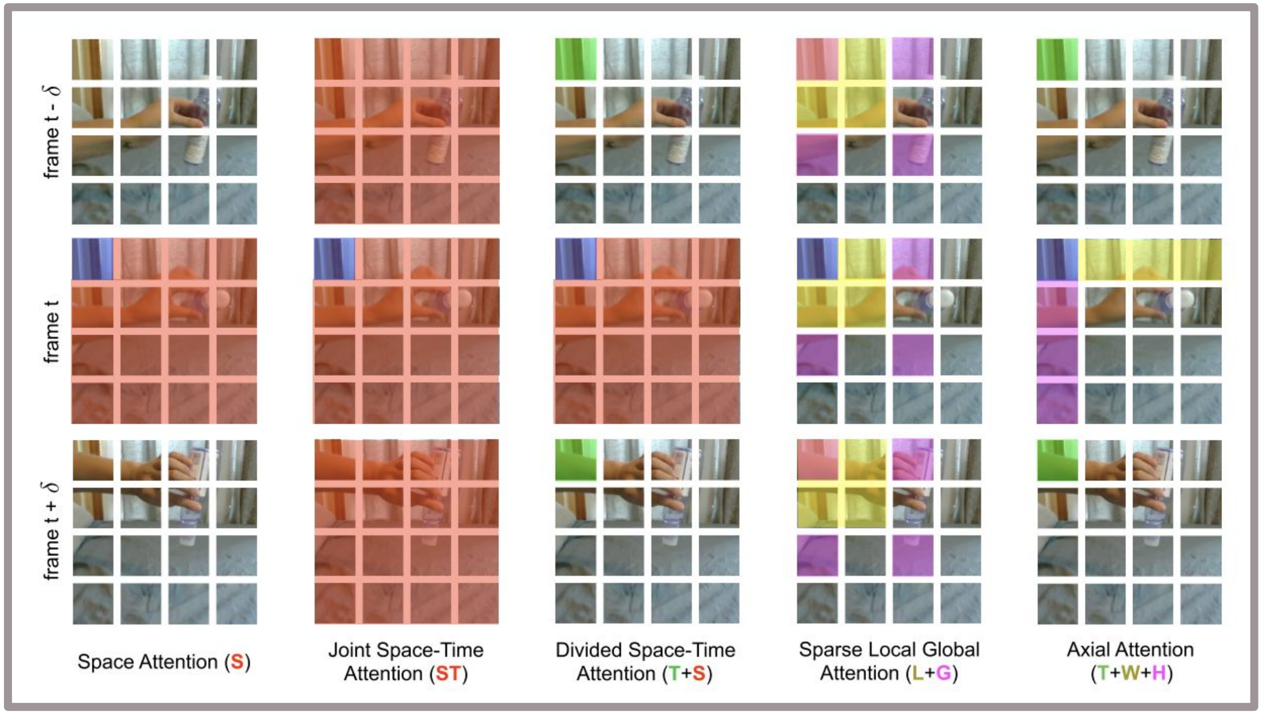


Photo from "Is Space-Time Attention All You Need for Video Understanding?
"

Transformers for Video data



- We denote in **blue** the query patch and show in **non-blue** colors its self-attention space-time neighborhood under each scheme.
- Patches without color are not used for the self-attention computation of the blue patch.
- Multiple colors within a scheme denote attentions separately applied along different dimensions.

PURDUE UNIVERSITY

Transformers for Video data

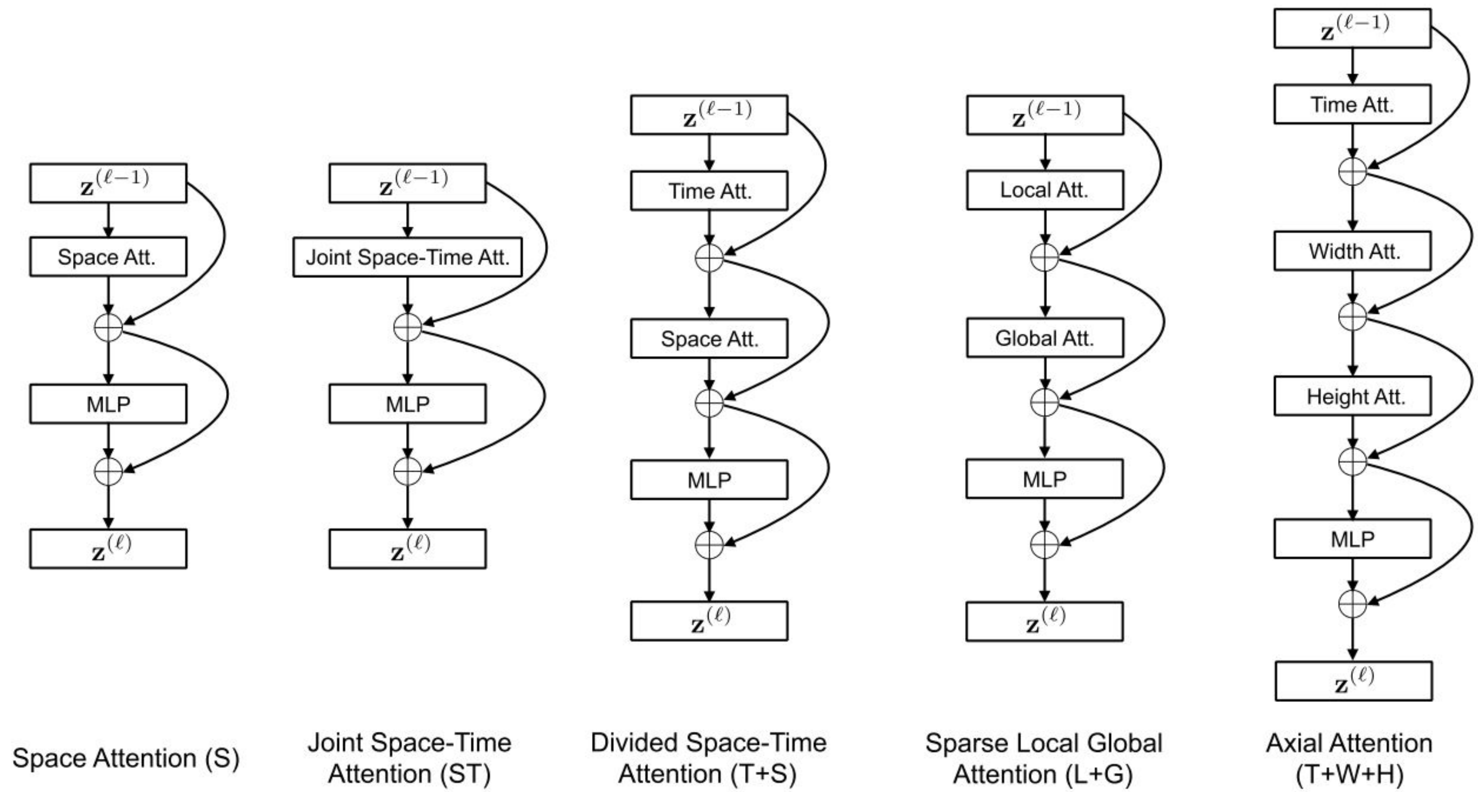


Photo from “Is Space-Time Attention All You Need for Video Understanding?
”

Transformers for 3D vision

- Due to the quadratic complexity of self-attention in Transformers, considering each point as one token is very computationally costly.
- They use a simple yet efficient implementation that groups each point cloud into several local patches and use it as input tokens for the transformers.

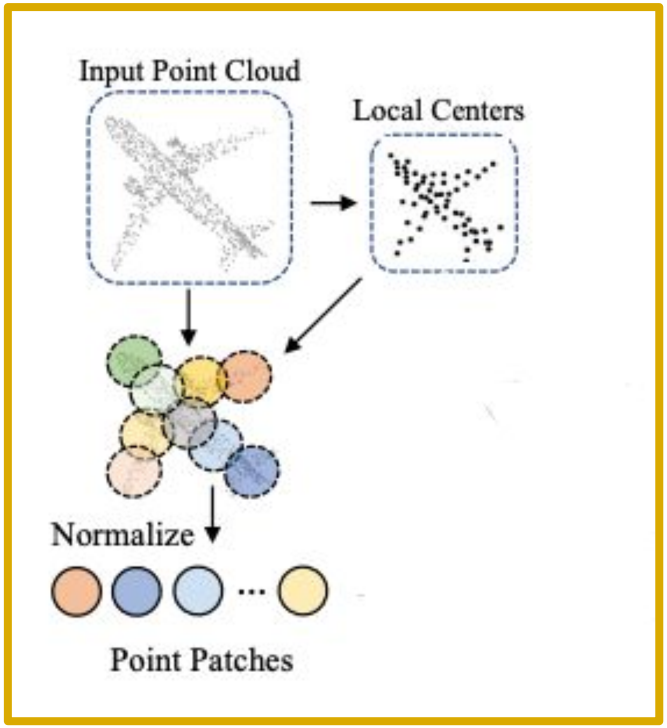


Photo from Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling, CVPR2022 with some modifications

Transformers for 3D vision

- Before pre-training, **a Tokenizer is learned** through dVAE-based point cloud reconstruction, where a point cloud can be converted into a sequence of discrete point tokens (the top part of the figure)
- During pre-training, the model is trained to recover **masked tokens** (the lower part of the figure).

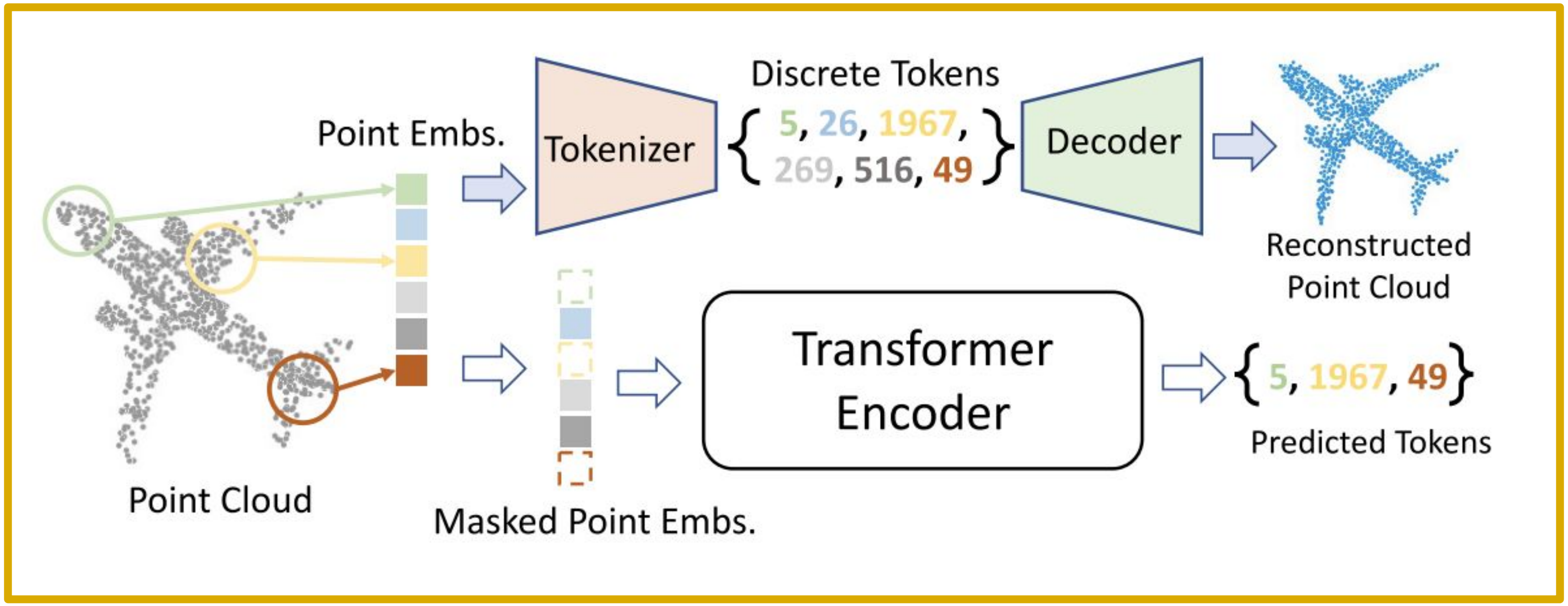


Photo from point-BERT paper

Transformers for different modalities

Aspect	NLP	Vision	Audio
Pre-training Task	Masked language modeling (BERT), next-word prediction (GPT), sentence ordering	Image classification (ImageNet), contrastive learning (CLIP),	Self-supervised waveform modeling (wav2vec, HuBERT)
Model Learns	Syntax, semantics, grammar, context	Shapes, edges, textures, object parts	Phonemes, pitch, tone, speaker identity
Architecture	Transformers (BERT, GPT, T5)	CNNs, Vision Transformers (ViT)	CNNs, RNNs, Transformers (wav2vec2, Whisper)
Fine-Tuning Use Cases	Sentiment analysis, NER, QA, translation	Medical imaging, satellite mapping, object detection	Speech-to-text, speaker ID, emotion recognition
Data Required (Pretraining)	Billions of tokens	Millions of images	Thousands of hours of audio

Cross-Modal Transformers

Mixing Several Modalities

Transformer-based cross-modal interactions:

(a) Early Summation,

(b) Early Concatenation,

(c) Hierarchical Attention (multi-stream to one-stream),

(d) Hierarchical Attention (one-stream to multi-stream),

(e) Cross-Attention, and

(f) Cross-Attention to Concatenation.

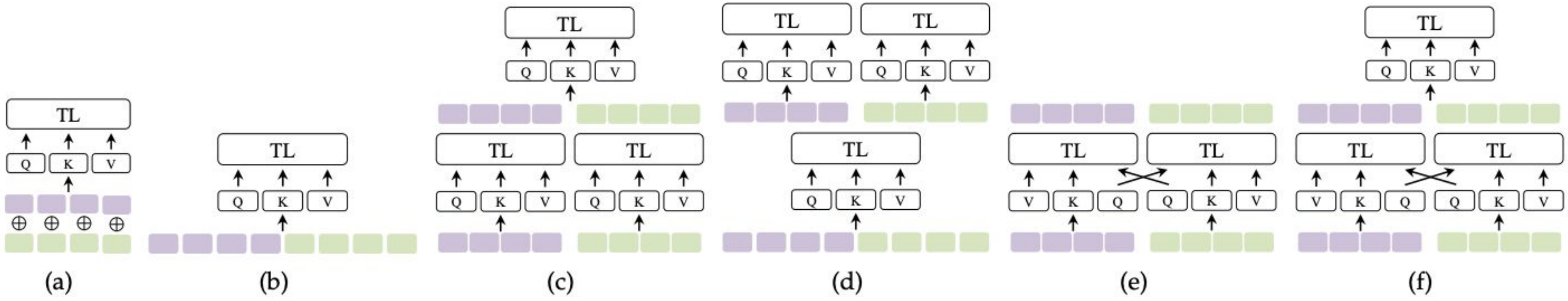


Photo from “Multimodal Learning with Transformers: A Survey”, 2023.

Next:

1. AI Ethics, Safety and Governance
2. Deep Learning tools (Python, HF, …)
3. Deep Learning for different Domains (Agriculture, Bio, …)
4. ?

PURDUE UNIVERSITY

References

1. Krishna, S.T. and Kalluri, H.K., 2019. Deep learning and transfer learning approaches for image classification. International Journal of Recent Technology and Engineering (IJRTE), 7(5S4), pp.427-432.
2. https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2
3. https://medium.com/@rushikesh.shende/autoencoders-variational-autoencoders-vae-and-%CE%B2-vae-ceba9998773d
4. Saha, M., Mitra, P. and Nanjundiah, R.S., 2017. Deep learning for predicting the monsoon over the homogeneous regions of India. Journal of earth system science, 126, pp.1-18.
5. https://yunfanj.com/blog/2021/01/11/ELBO.html
6. https://www.jeremyjordan.me/variational-autoencoders/
7. https://www.microsoft.com/en-us/research/blog/how-can-generative-adversarial-networks-learn-real-life-distributions-easily/
8. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
9. https://pub.towardsai.net/diffusion-models-vs-gans-vs-vaes-comparison-of-deep-generative-models-67ab93e0d9ae
10. https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder
11. https://cameronrwolfe.substack.com/p/language-model-training-and-inference
12. https://x.com/cwolferesearch/status/1689388468911132672
13. https://twitter.com/cwolferesearch/status/1671628210180698112?s=20
14. https://twitter.com/cwolferesearch/status/1659608476455256078?s=20
15. https://twitter.com/cwolferesearch/status/1692617211205022064?s=20
16. https://docs.cohere.com/docs/controlling-generation-with-top-k-top-p
17. https://x.com/cwolferesearch/status/1766180825173803516
18. Wang, W., Chen, W., Luo, Y., Long, Y., Lin, Z., Zhang, L., Lin, B., Cai, D. and He, X., 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*.
19. https://medium.com/@eugene-s/unleashing-the-potential-of-large-language-models-llms-with-chatgpt-8210f0cb063d
20. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), pp.1-67.
21. 21. Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M. and Hon, H.W., 2019. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, *32*.
22. 22. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F. and Liang, P., 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, *12*, pp.157-173.
23. 23. Dosovitskiy, A., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

References

24. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. and Krueger, G., 2021, July. Learning transferable visual models from natural language supervision. In *International conference on machine learning* (pp. 8748-8763). PMLR.
25. Baevski, A., Zhou, Y., Mohamed, A. and Auli, M., 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, *33*, pp.12449-12460.
26. Bertasius, G., Wang, H. and Torresani, L., 2021, July. Is space-time attention all you need for video understanding?. In *ICML* (Vol. 2, No. 3, p. 4).
27. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J. and Lu, J., 2022. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 19313-19322).
28. https://medium.com/@lmpo/an-overview-instruction-tuning-for-llms-440228e7edab
29. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), pp.1-67.
30. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J. and Wu, X., 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.
31. https://medium.com/@GULGULTEKIN/prompt-engineering-techniques-37a473ed25a6
32. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V. and Zhou, D., 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, pp.24824-24837.
33. Ovadia, O., Brief, M., Mishaeli, M. and Elisha, O., 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*.
34. Gupta, A., Shirgaonkar, A., Balaguer, A.D.L., Silva, B., Holstein, D., Li, D., Marsman, J., Nunes, L.O., Rouzbahman, M., Sharp, M. and Mecklenburg, N., 2024. RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture. *arXiv preprint arXiv:2401.08406*.
35. https://plan.seek.intel.com/generative-ai-webinars , Emerging frontiers in GenAI: Agentic AI workflows and Multimodal LLMs, Dillon laird, accessed Nov2024
36. https://medium.com/@infin94/understanding-the-seq2seq-model-what-you-should-know-before-understanding-transformers-e5891bcd57ec
36. https://predibase.com/blog/5-reasons-why-lora-adapters-are-the-future-of-fine-tuning
37. https://aws.amazon.com/what-is/reinforcement-learning-from-human-feedback/
38. https://huggingface.co/blog/rlhf
39. https://medium.com/@lmpo/an-overview-instruction-tuning-for-llms-440228e7edab
40. https://cobusgreyling.medium.com/prompt-tuning-hard-prompts-soft-prompts-49740de6c64c
41. https://medium.com/dair-ai/adapters-a-compact-and-extensible-transfer-learning-method-for-nlp-6d18c2399f62
42. https://x.com/cwolferesearch/status/1795888289896857836
43. https://medium.com/@tenyks_blogger/rag-for-vision-building-genai-computer-vision-systems-6a812adfb20e
44. https://magazine.sebastianraschka.com/p/understanding-and-coding-self-attention

Thank You

PURDUE
UNIVERSITY