DEF CON 2025: Solving Modern Cybersecurity Problems with AI

Michael Glass

About Michael

- Bluescreenofwin
- Hacker from early 2000s
- Started working for local government 2009 during recession, found a niche, kept working it
- Went to college for cybersecurity
- Lead the security team for one of the world's largest technology streaming services company
- Currently running enterprise security for a financial startup
- Also volunteer and run infrastructure for WRCCDC, mentor cybersecurity students, also certified professional brewer (yes I make beer)

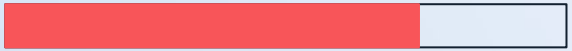
Class Introductions

Tell us about yourself!

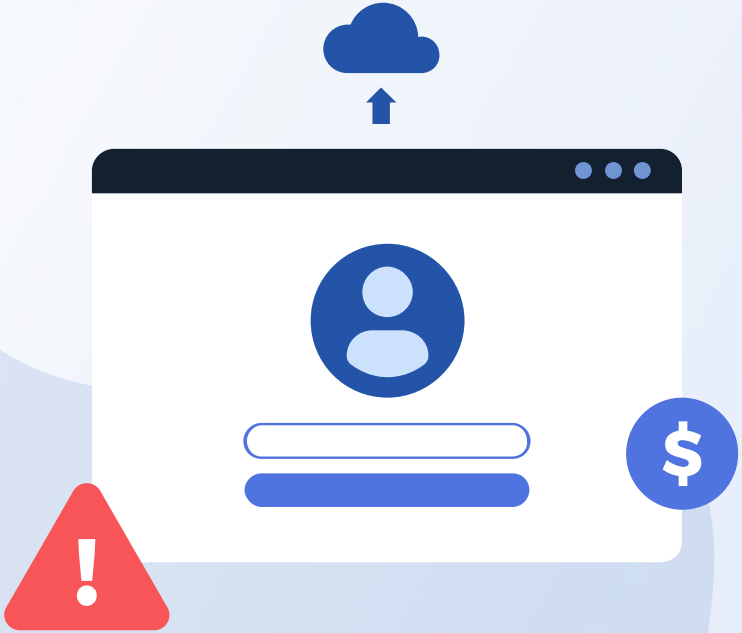
- Name or handle

- Current role (student or none is also acceptable!)

- Years of experience with Cybersecurity



- Level of comfort with generative AI/LLMs

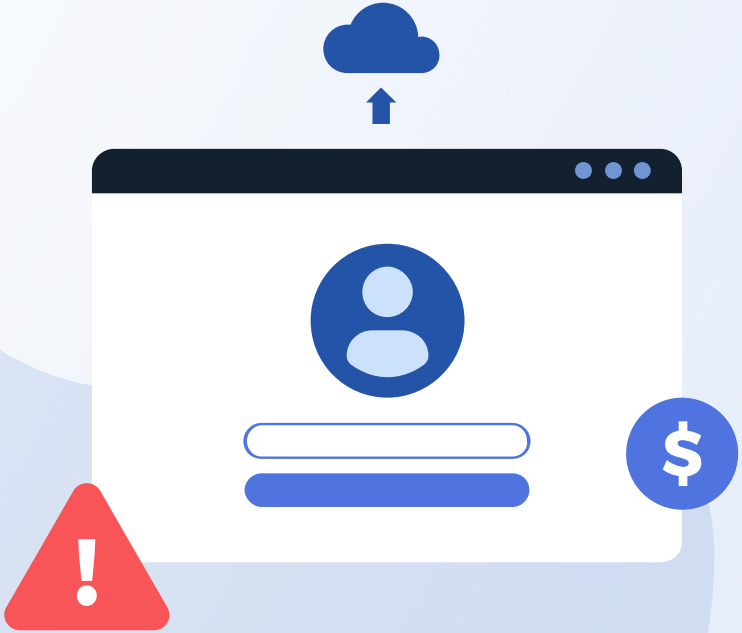


Access to WiFi

WiFi Information provided by DefCon

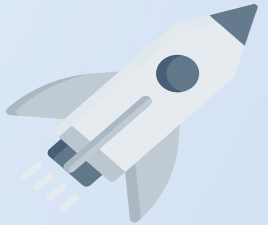
SSID: **DCTLV2024SAHARA**

PW: **R*u26TcnQ(9vnP<G!**



Access to Git

- https://github.com/DCT2025-SCSPWAI/Training-Materials
- We will need everyone’s github account so we can add you
- After being added please clone the repository
- After this we will begin!



Mini Git Tutorial

- Don’t know how to use git? We got you!
- Git is an open source version control system
- Allows one to track file changes and control source code
- Why do we use it for this class?
    - It’s awesome–everyone should learn how to use it
    - Allows us to easily push out changes and you can download the revisions to our material in realtime!

Mini Git Tutorial

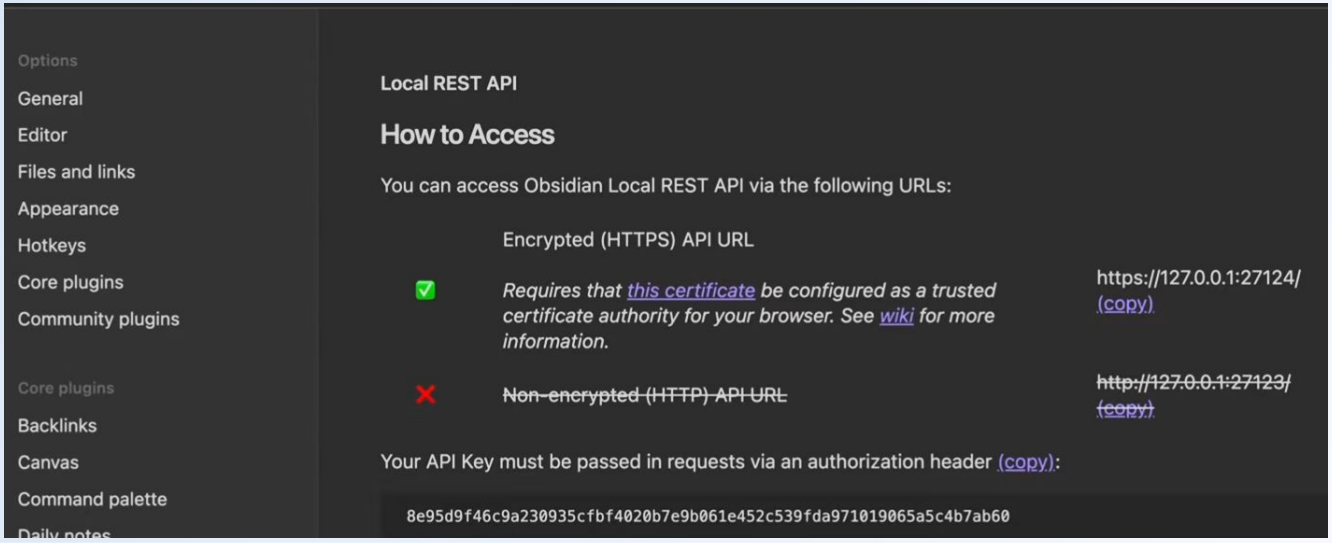
- Download git with your favorite package manager
- **Windows**: https://desktop.github.com/download/
- **Linux**:
  - https://git-scm.com/download/linux
  - Debian/Ubuntu: `apt-get install git`
- **Mac**:
  - https://git-scm.com/download/mac
  - Homebrew: `$ brew install git`

Mini Git Tutorial

- Make a directory for your git projects on your laptop
- Log into GitHub on a browser for session
- Type in the command
  - **git clone https://github.com/DCT2025-SCSPWAI/Training-Materials**
- If it was successful you will see Day 1 being pulled down
- If the CLI asks for authentication you may need to create an access token
  - Please ask for assistance if needed!

Obsidian

- I recommend **Obsidian** for note taking
- Private; local; data does not leave your PC unless you want it to
- We will also use it for a later lab
    - Install the **Local Rest API** community plugin and **enable**



Class Goals

- Our goal is to educate you on Artificial Intelligence, it's contemporary applications, and how to configure and use your own AI models and generative AI tools
- AI is a massive topic that spreads across many domains and we want to try to distill that down into the context of Cybersecurity
- Bring what you learn from the class back to your own home/work environments and apply the same knowledge there
- Learn to install, use, and modify popular Open Source tools in conjunction with AI to streamline usage
- Main software we will cover: Llama.cpp, LMStudio, OpenSearch

Day 1 Overview

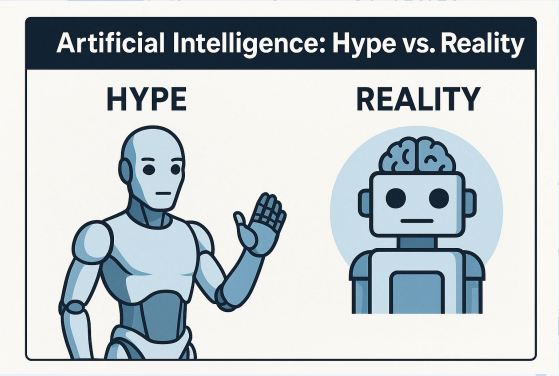
- AI, LLMs, and contemporary viewpoints on AI
- Primer on AI and LLMs
- Learn to navigate popular AI SaaS tools
- Low Rank Adaption and Fine-Tuning of Models
- Introduction and access to the AI environment
- Open source tooling for AI
- Introduction to OpenSearch

Day 2 Overview

- Threat modeling AI and common attacks with defense strategies
- Security frameworks for LLMs and Machine Learning Models
- Solve a series of problems using curated data to provide to our AI
  - XSS/SQLi
  - Application Data
  - Threat Intel Data
- Couple data together with RAG to solve more complex problems
- Integrate Threat Intelligence feeds to assist in answering unique problems
  - Data graciously provided by GreyNoise

Introduction into AI

What is “AI”



- Artificial Intelligence: Hype vs. Reality
  - **What is AI (in theory)?**
    A system capable of independent thinking and reasoning—true machine intelligence.
    Where are we now?
    Most modern AI systems are powered by **Large Language Models (LLMs)**—advanced pattern-recognition algorithms trained on vast text data.
    → They do **not** possess understanding, consciousness, or general reasoning.
  - **Not Quite "I, Robot"... Yet**
    We're far from achieving *Artificial General Intelligence (AGI)*—AI that can understand, learn, and apply knowledge across any task, like a human.
  - **The Marketing Reality**
    Many companies market their tools as “AI,” but they’re really deploying advanced automation with no real autonomy.
  - **The AGI Race**
    Organizations like OpenAI aim to develop AGI—a long-term goal often compared to the helper robots in *I, Robot*.

Levels of AI based on capabilities

1. **Artificial Narrow Intelligence**
    a. Can perform singular tasks
        i. Siri, Alexa, etc
2. **Artificial General AI**
    a. Commonly referred to as **AGI** or **Artificial General Intelligence**
    b. Can use previous learnings and skills to accomplish new tasks
    c. Only theoretical
3. **Artificial Super AI (Theoretical)**
    a. Reasoning, thinking, judging like a human being
        i. VIKI, ARIIA, Skynet
            1. I didn't intentionally only pick bad ones..

Source: https://www.ibm.com/think/topics/artificial-intelligence-types

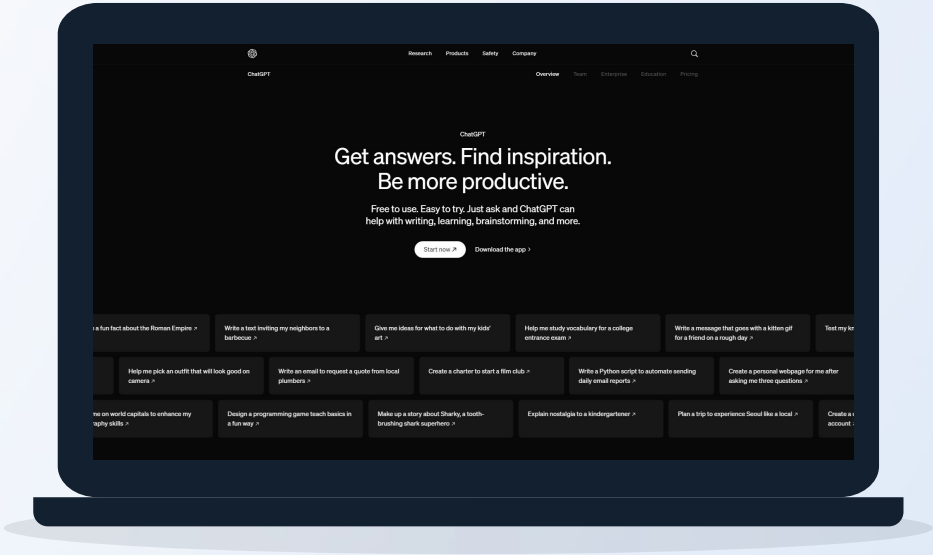
Levels of AI based on functionality

1. **Reactive Machine AI**
    a. Singular task, based only on available data, aka an algorithm
        i. Recommendation Engines for YouTube, Spotify, etc
2. **Limited Memory AI**
    a. Can recall past events/outcomes alongside current data
        i. ChatGPT, Bard, Alexa, Siri
3. **Theory of Mind AI (Theoretical)**
    a. Able to understand thoughts, emotions, motives
4. **Self-Aware AI (Theoretical)**
    a. See last slide with all the AIs who think humans need to either be killed or kept as pets

Source: https://www.ibm.com/think/topics/artificial-intelligence-types

Class Discussion
Pros and Cons of AI

Hypothetical: Giving up your privacy



Lab #1 Intro into Using ChatGPT

10:00am - Coffee/Bio Break

Return by 10:20am

Peeking Behind The Curtain

Pushing Past the Marketing

- LLMs are “simply” a neural network with access to a large dataset
- The Transformer model is the most popular model used for these neural networks
  - Chat *Generative Pre-trained Transformer*
- Initial design introduced by Google researchers in a paper in 2017
  - `Attention Is All You Need`
- That paper kicked off the race that we see today between a myriad of companies and organizations
- Several years later, I can now replicate with a Chatbot what I can’t in real life, a relationship

Bidirectional Encoder Representations from Transformers (BERT)

- Pre Trained Transformer
  - Based on the work done in the 2017 research paper
  - Trained on Wikipedia(~3TB)
- Officially launched in 2018
  - BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding”
- Encoder only, cannot be prompted or generate text
- Bidirectionality means it can take into account left and right context
- Can be easily fine-tuned to handle specific tasks
- Initially trained with a percentage of input tokens being masked. The model learns to predict the masked tokens based on context

What is a GPT?

- **Generative Pre-trained Transformer**
  - This is where “ChatGPT comes from”
- Type of Large Language Model (LLM) that uses lots of information (deep learning) to produce a natural language response
- If we asked ChatGPT to describe itself...

🤖 **Generative Pre-trained Transformer (GPT)**

- The model architecture behind **ChatGPT**
- A type of **Large Language Model (LLM)**
- Trained on massive datasets using **deep learning**
- Generates human-like responses in **natural language**

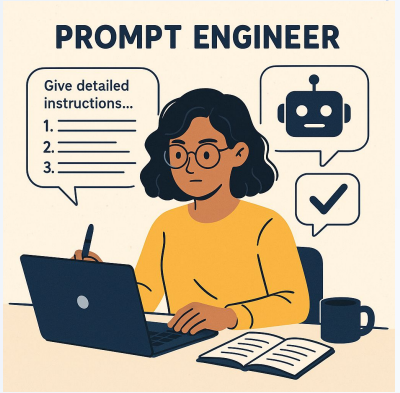
Think of GPT as a highly trained text prediction engine, fine-tuned to sound conversational and useful.

Generative Pre-Trained Transformer(GPT)

- Autoregressive and Unidirectional
  - Autoregressive boils down to using previous tokens/inputs to predict the next word one token at a time
  - Unidirectional (Left to Right): Only looks at past context, not future words
- Uses the Decoder half of the Transformer architecture
- Trained on a massive 45TB+ corpus of text
- Can be fine-tuned later:
  - 🔧 LoRA (lightweight tuning)
  - 🧠 Full training on custom data

Prompt Engineering

- Prompt engineering is the art and science of crafting the right prompts to guide a large language model (LLM) into producing accurate, relevant, and well-structured outputs using plain language only
  - Examples where Prompt Engineering is useful include...
    - **Content Generation** – Articles, emails, scripts, social posts
    - **Data Extraction** – Summarizing, parsing, structuring raw text
    - **Code Assistance** – Debugging, generating, explaining code
    - **Customer Support** – Chatbots, ticket triage, automated replies
    - **Education & Tutoring** – Practice questions, concept explanation
    - **Creative Work** – Storytelling, character design, brainstorming
    - **Business Ops** – Report drafts, meeting notes, workflow automation
    - **Language Translation** – Natural, context-aware translations
    - **Legal/Policy Drafting** – First drafts of contracts, summaries
    - **Search Optimization** – Better search queries, semantic matching



Prompt Engineering

- **Techniques:**
- **Zero-shot prompting**
  Ask the model to perform a task without examples—useful for clear, simple requests.
- **Few-shot prompting**
  Include a few example inputs and outputs so the model mimics formatting or style.
- **Chain-of-thought prompting (CoT)**
  Encourage step-by-step reasoning—e.g. begin with *“Let’s think step by step…”*—to improve logic and accuracy
  - https://en.wikipedia.org/wiki/Prompt_engineering
  - https://www.ibm.com/think/topics/prompt-engineering
- **Role prompting**
  Assign the model a persona (e.g. “financial advisor,” “editor”) to shape tone, terminology, and structure

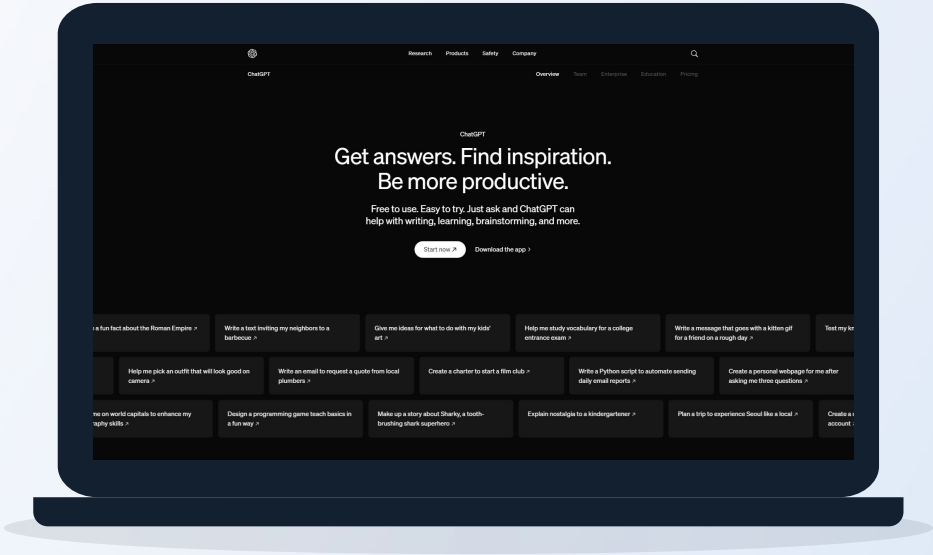
Prompt Engineering

- **Best Practices (according to Anthropic)**
  - Be **very specific** about audience, goal, tone, format, and output style
  - Use **bullet points or numbered lists** for clarity
  - Be generous with numerous **high-quality examples** when needed
  - Include CoT requests to reduce hallucinations
    - Aka “Chain of Thought” prompting
    - Encourages AI to break down problems step-by-step, leading to more accurate and nuanced outputs.
  - Allow the model to say “I don’t know” or request citations from sources to improve reliability
  - Roleplay
    - "In complex scenarios like legal analysis or financial modeling, role prompting can significantly boost Claude's performance."
    - Assigning roles can ensure you get exactly what you want. Maybe you want the brevity of a news writer, or maybe you want the tone of an academic.

Source: https://www.businessinsider.com/anthropic-guide-prompt-engineering-2025-7

Prompt Engineering

- Technical breakdown by Google
  - https://developers.google.com/machine-learning/resources/prompt-eng
  - Will break down the specifics that have been studied and researched
- Interesting ***arxiv*** white papers:
  - https://arxiv.org/abs/2302.11382
  - https://arxiv.org/abs/2507.22729



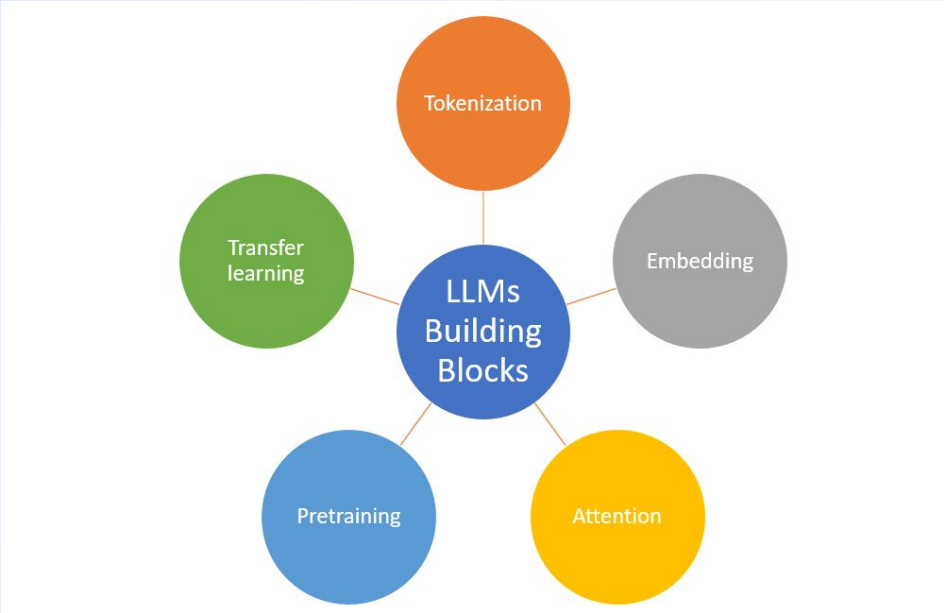
Lab #2 Building your own GPT

- **Rules**:
  - Split up into groups of 3
  - Must be solving a real world problem related to your work, academic pursuits, or something otherwise useful
  - Must use at least one document for knowledge
  - Winning group wins a prize
  - 30 minutes max

Large Language Models Deep Dive

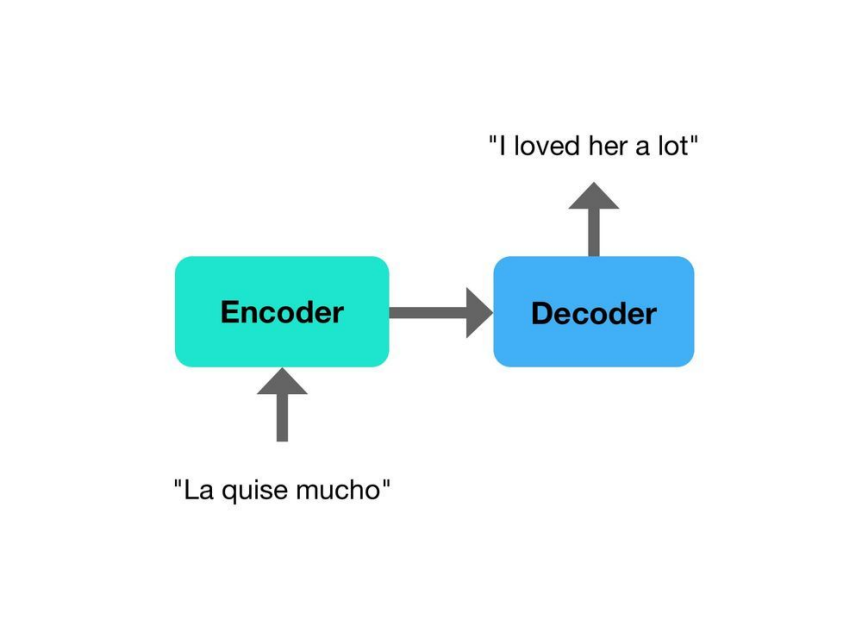
How LLMs Work

- Think of models as “programs” and the data is what really drives them
- A transformer can be applied to many different sequence prediction task
- Create a diagram from start to finish op



History Lesson - How LLMs Work

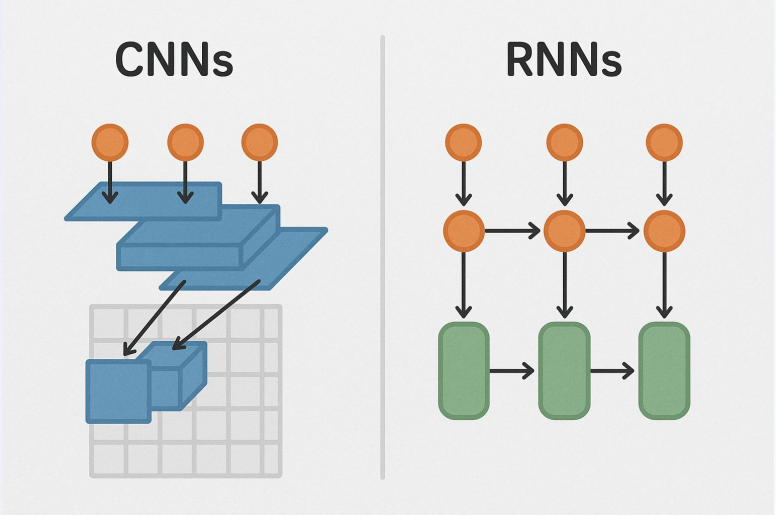
- **Sequence Transduction**
- Fancy way of saying we are predicting an output sequence from an input sequence of different length



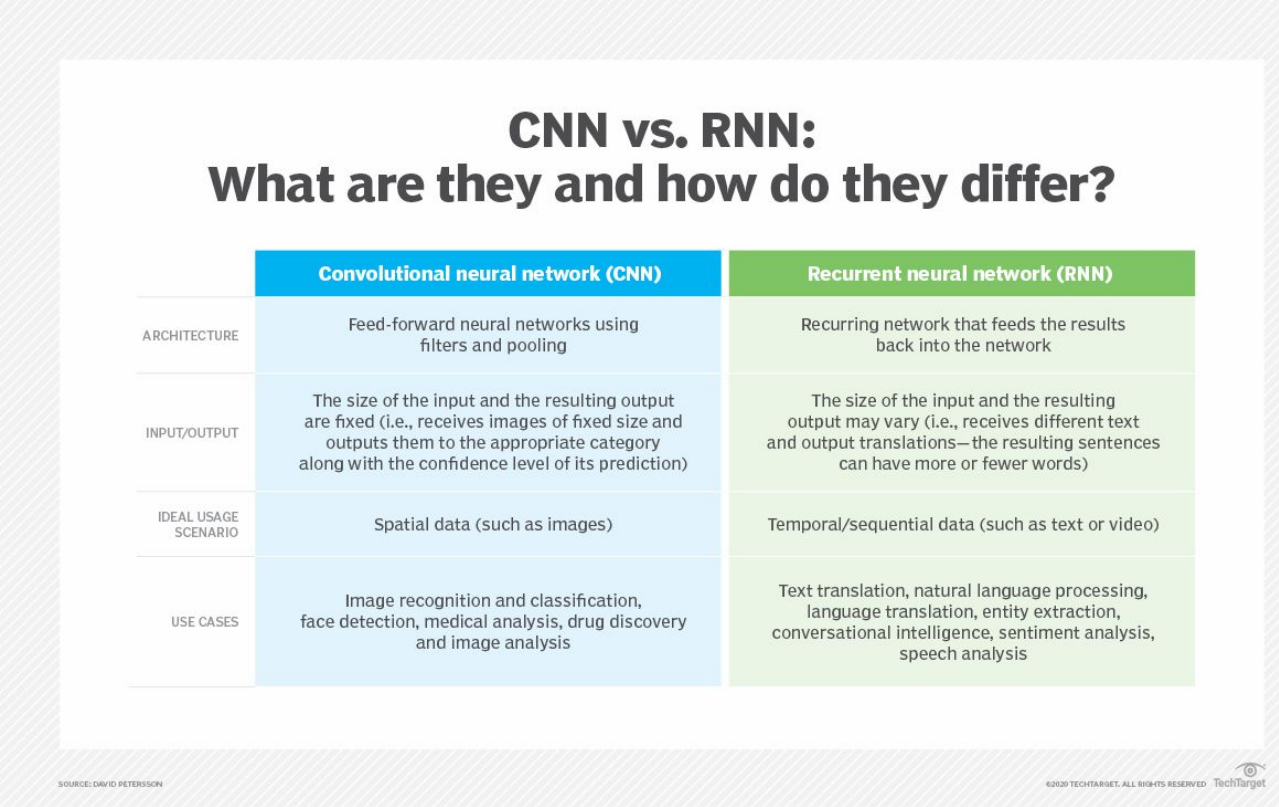
History Lesson - How LLMs Work

- The model will then use an algorithm transform the input into an output

- **RNN**
    - Recurrent Neural Networks
    - Well suited at working with sequential data
        - Examples: Natural language
- **CNN**
    - Convolutional Neural Networks
    - Well suited for images and video
        - Examples: Image generation



History Lesson - How LLMs Work



CNN vs. RNN: What are they and how do they differ?

	Convolutional neural network (CNN)	Recurrent neural network (RNN)
ARCHITECTURE	Feed-forward neural networks using filters and pooling	Recurring network that feeds the results back into the network
INPUT/OUTPUT	The size of the input and the resulting output are fixed (i.e., receives images of fixed size and outputs them to the appropriate category along with the confidence level of its prediction)	The size of the input and the resulting output may vary (i.e., receives different text and output translations—the resulting sentences can have more or fewer words)
IDEAL USAGE SCENARIO	Spatial data (such as images)	Temporal/sequential data (such as text or video)
USE CASES	Image recognition and classification, face detection, medical analysis, drug discovery and image analysis	Text translation, natural language processing, language translation, entity extraction, conversational intelligence, sentiment analysis, speech analysis

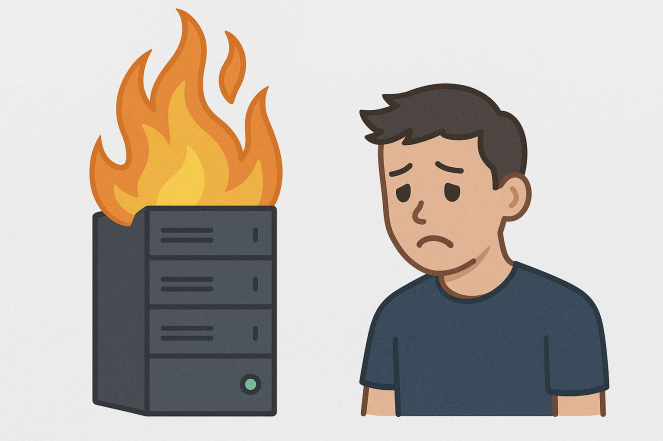
SOURCE: DAVID PETERSSON

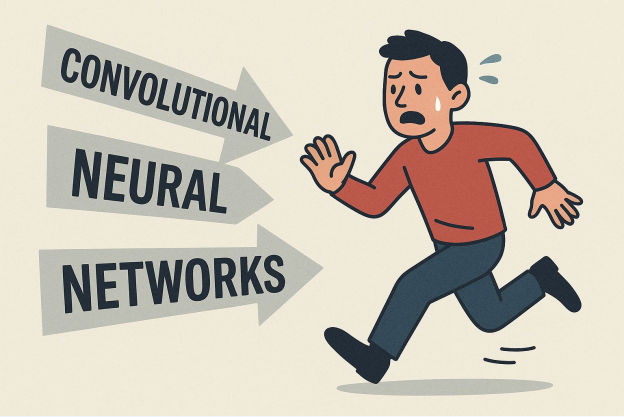
©2020 TECHTARGET. ALL RIGHTS RESERVED TechTarget

Source: https://www.techtarget.com/searchenterpriseai/feature/CNN-vs-RNN-How-they-differ-and-where-they-overlap

History Lesson - How LLMs Work

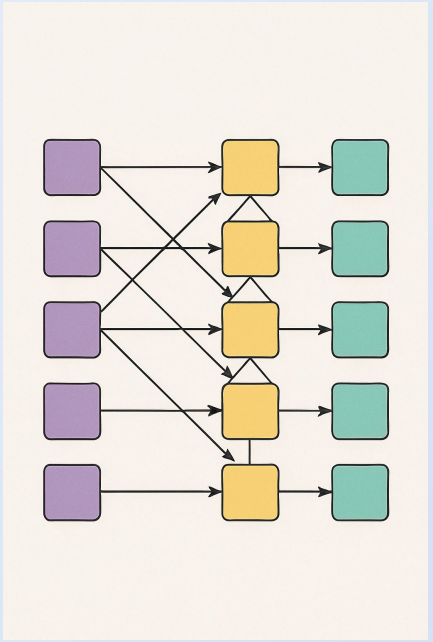
- Problems with CNNs and RNNs
    - Computationally expensive
    - Unreliable in long sequences
    - Required layering with other algorithms and techniques which introduces complexity





Introducing Attention – Attention is All You Need

- ‘The’ seminal paper on Large Language Models
- Published at from google and University of Toronto
- Indexed by “ArXiv”, a Cornell University database
    - pronounced *Archive*
- https://arxiv.org/abs/1706.03762



Why did this paper change the world?

Positional Encodings

Added to word embeddings to provide information about the position of words in a sequence, as Transformers do not inherently understand sequence order

Attention

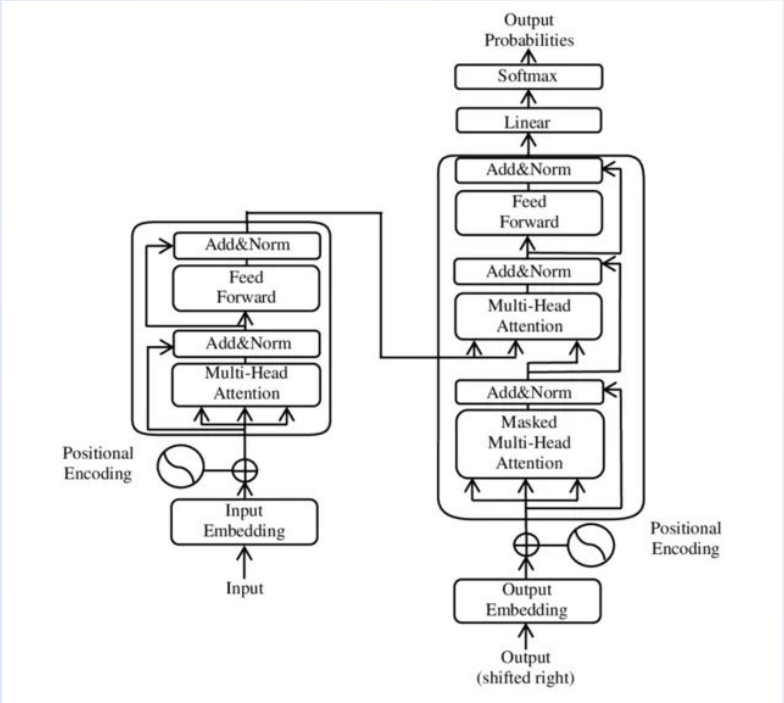
A mechanism that allows neural networks to focus on specific parts of input. Attention enables the model to dynamically focus on different parts of the input data, determining which elements are most relevant

Self-Attention

Allows each word in a sequence to attend to all other words, enabling the model to capture long-range dependencies efficiently

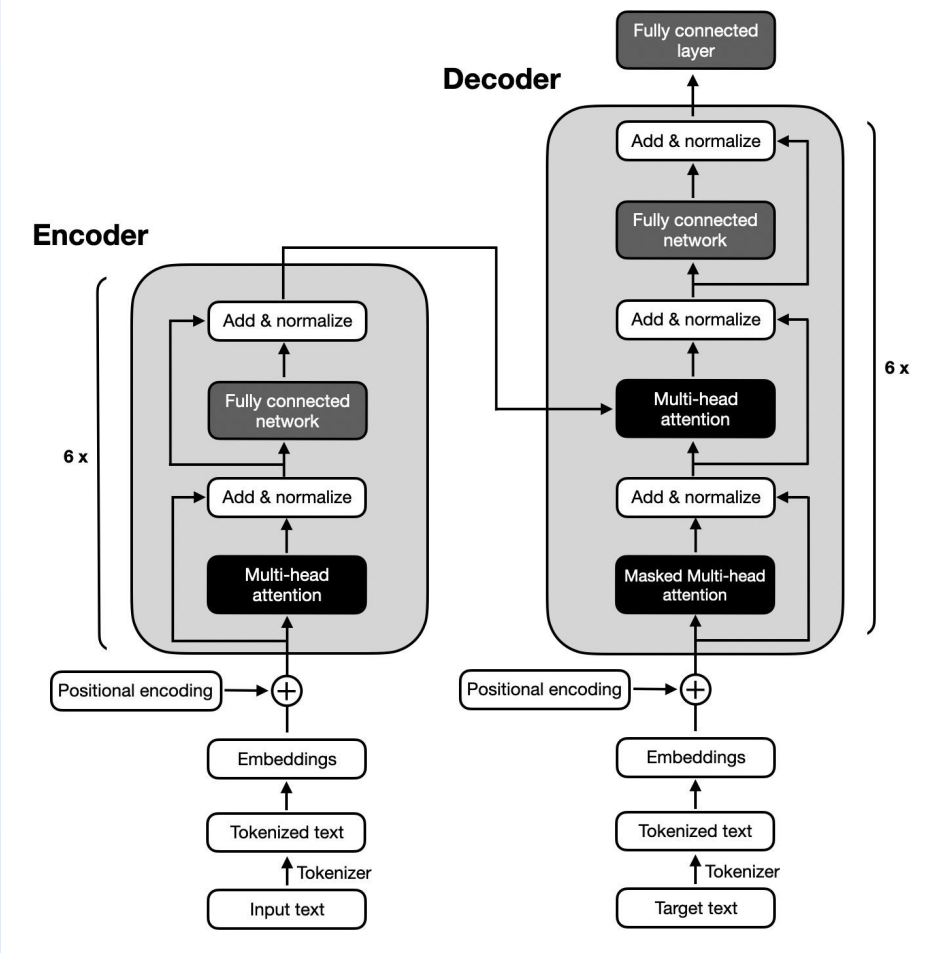
Why did this paper change the world?

Word Order is stored within the data itself versus the structure of the transformer



“Attention” - where the model learns from data over time

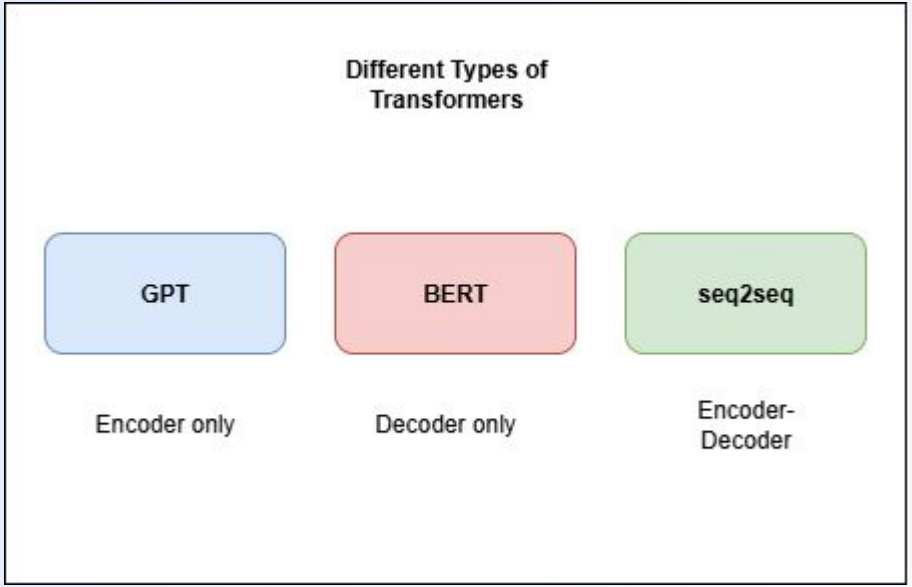
Encoders & Decoders



Source: https://arxiv.org/abs/1706.03762
Attention Is All You Need

Transformers

- What are transformers?
- Transformers are a type of neural network architecture designed for processing sequences — like text, code, or even images — without relying on recurrence (like RNNs) or convolutions (like CNNs)



How does it work?

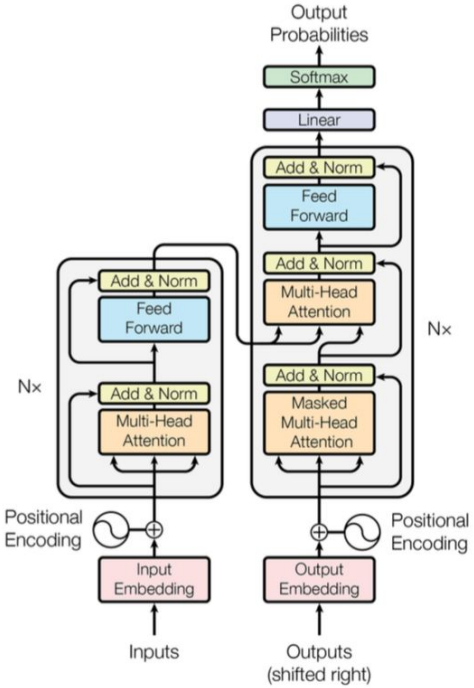
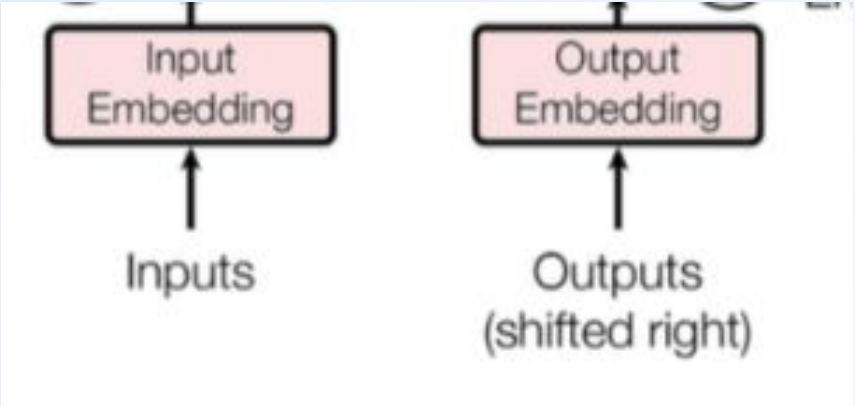


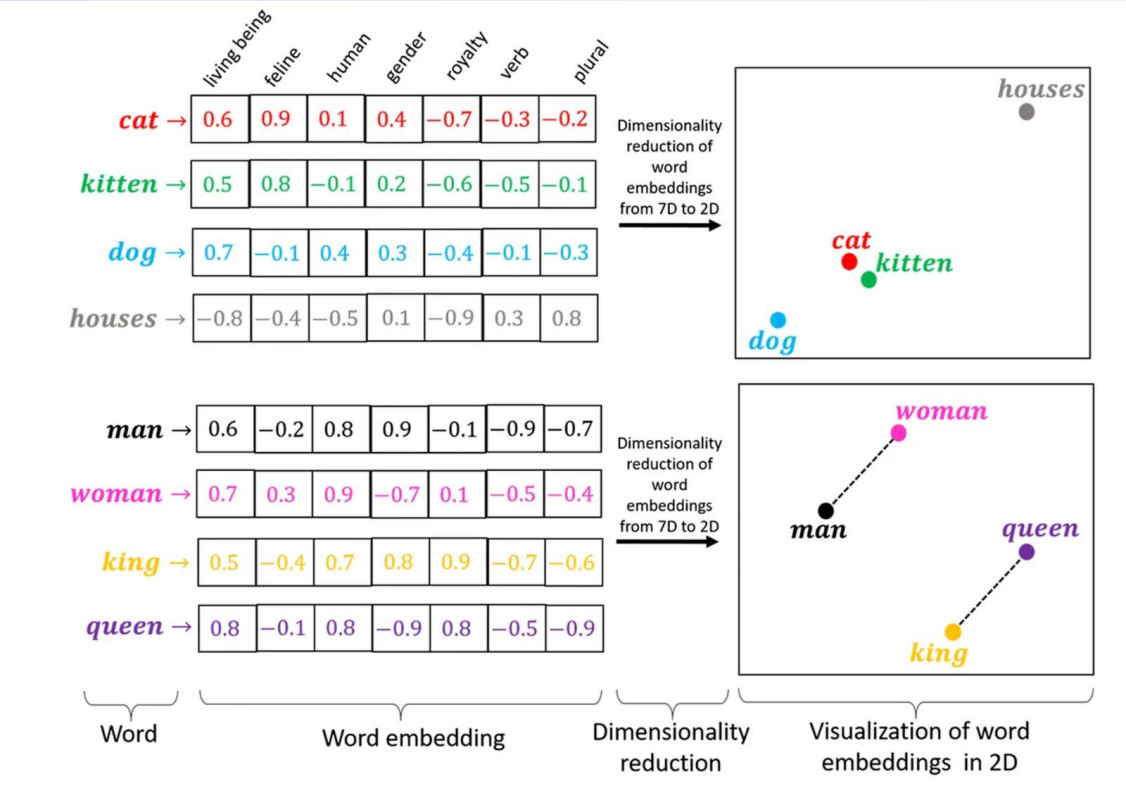
Figure 1: The Transformer - model architecture.

How does it work?

- The encoder and the decoder takes in a set of input embeddings that represent each token
- “token” and “word” are interchangeable



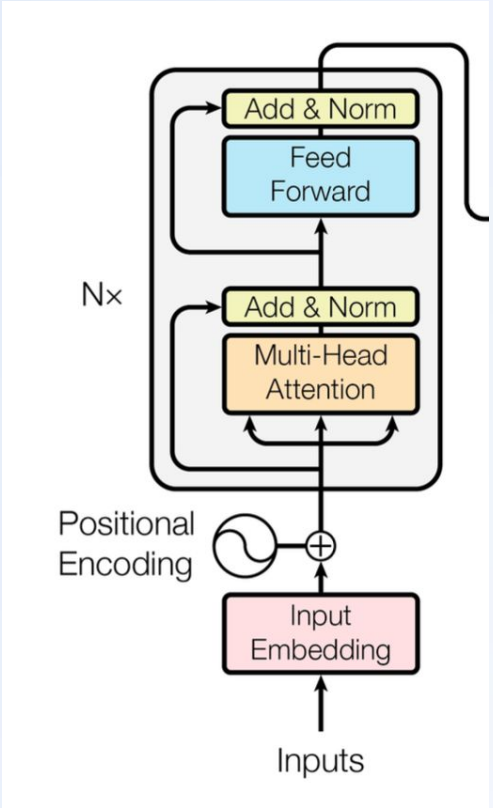
How does it work?



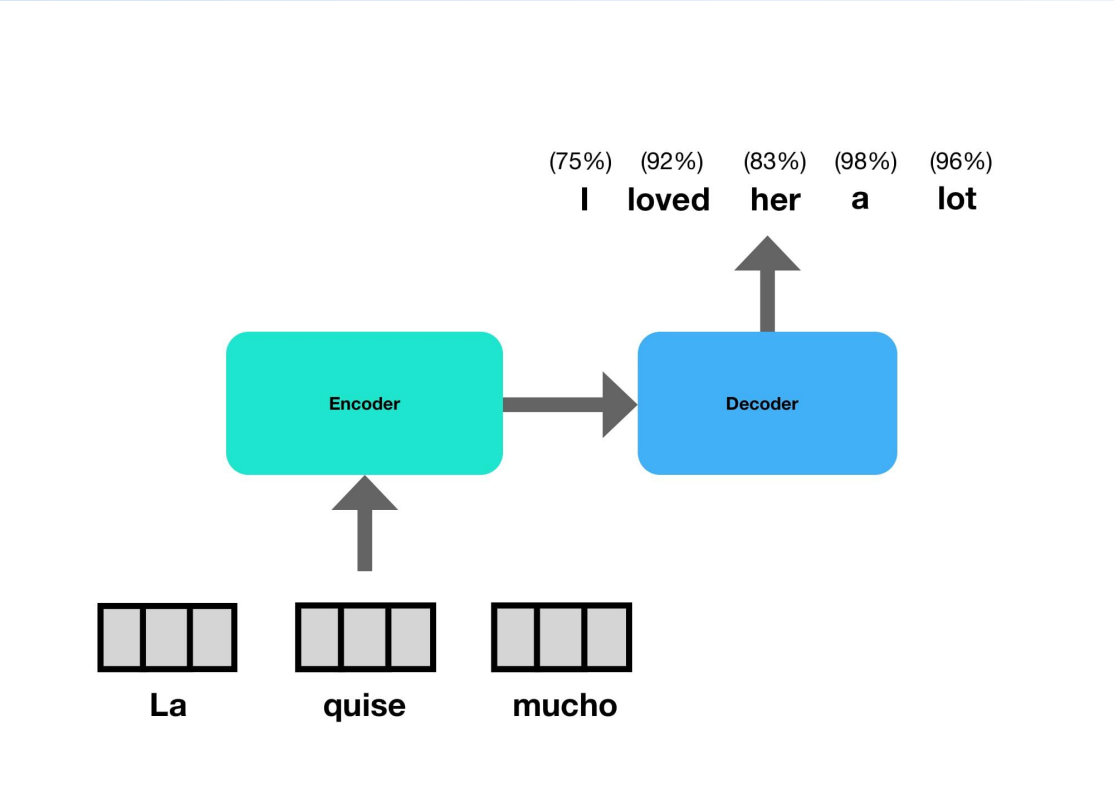
Source: https://medium.com/@hari4om/word-embedding-d816f643140

How does it work – The Encoder

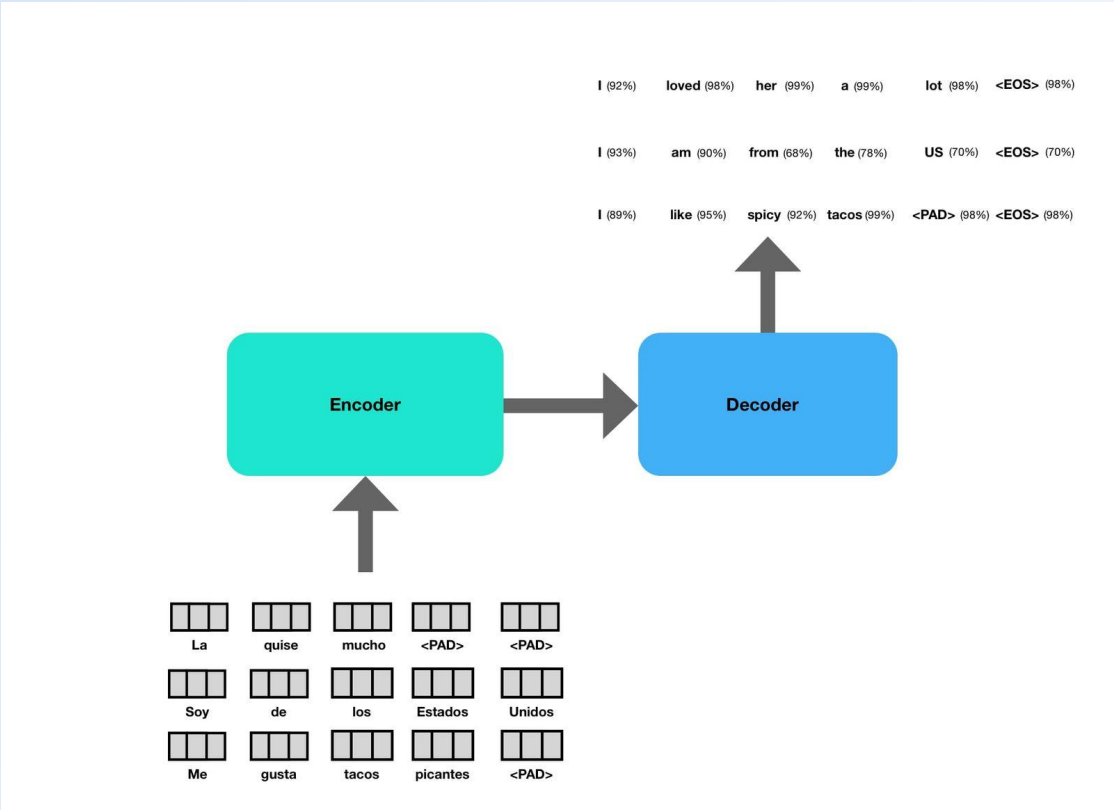
- Word embeddings are then fed up to the encoder module
- Encoders job is to look at the whole sentence, and decide what each word means.



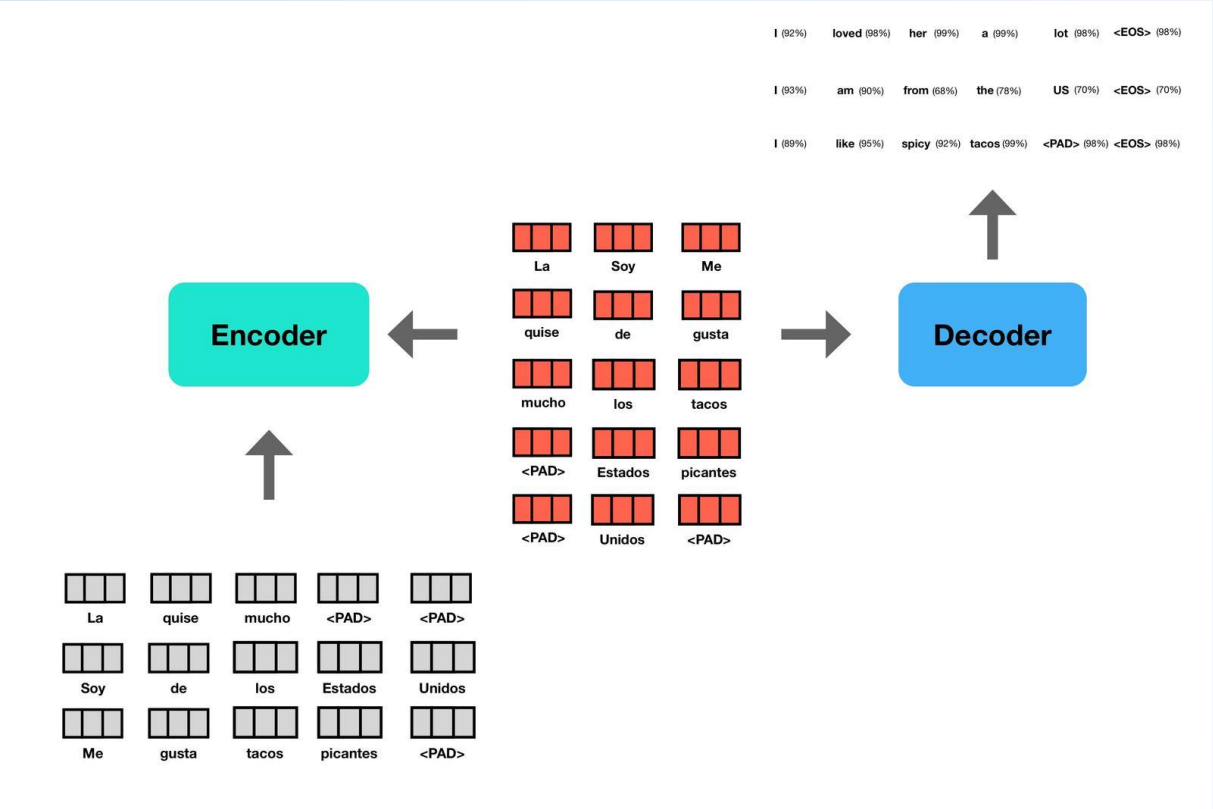
How does it work – The Encoder



How does it work – The Encoder

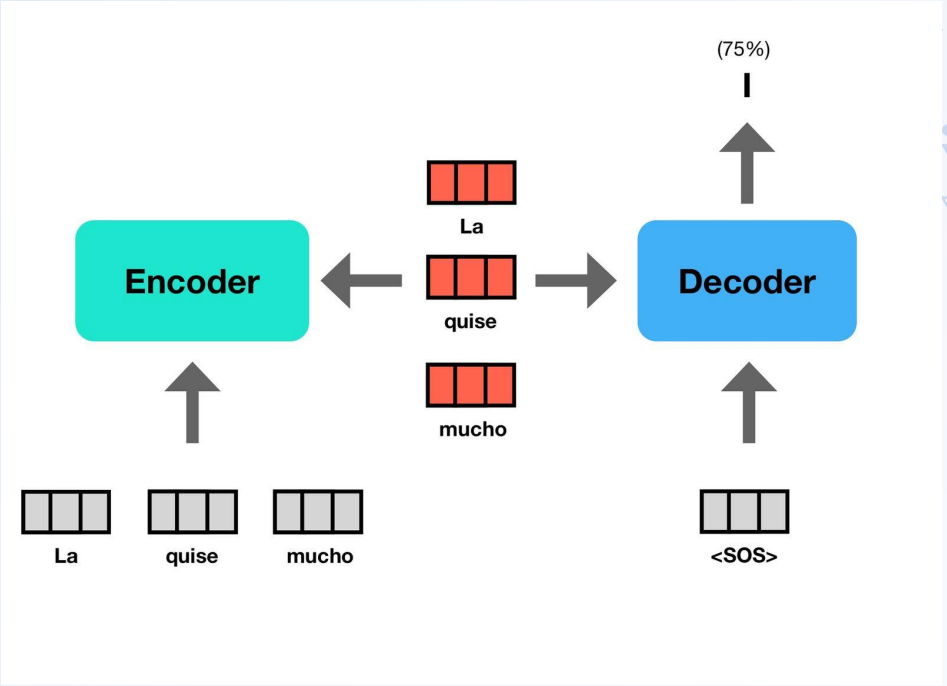


How does it work – The Encoder

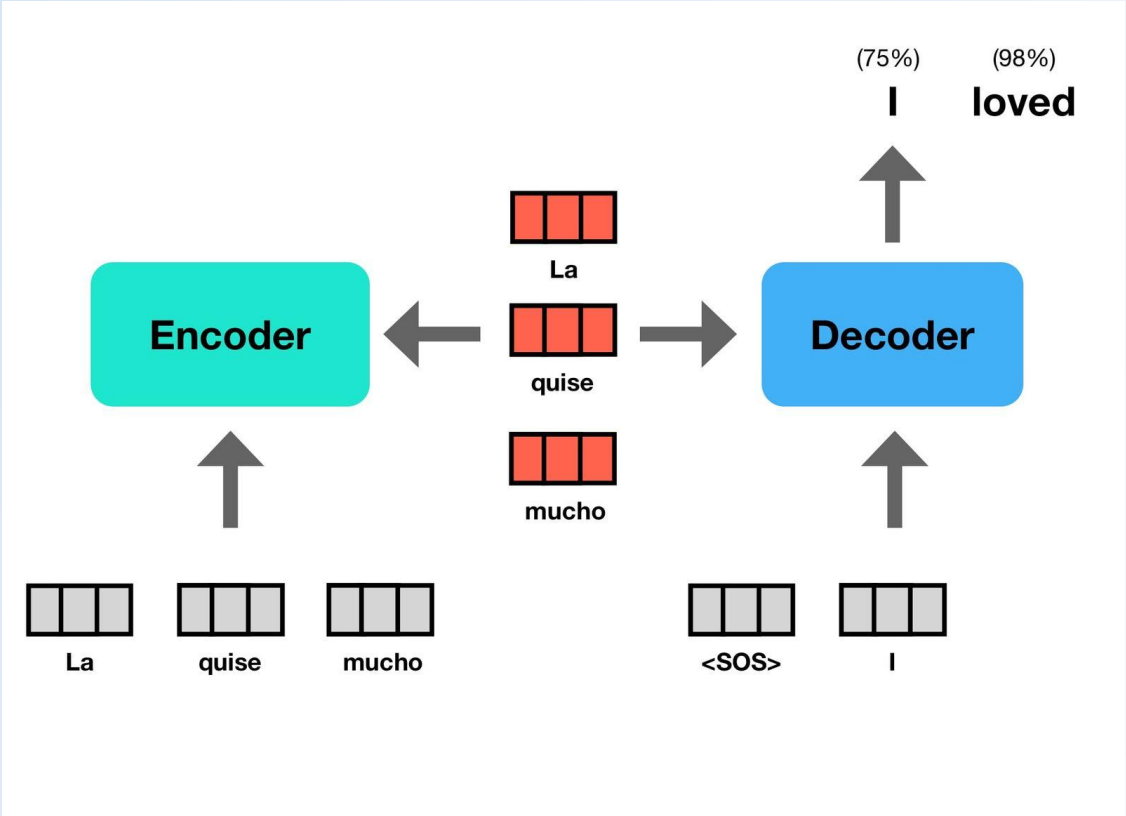


How does it work – The Decoder

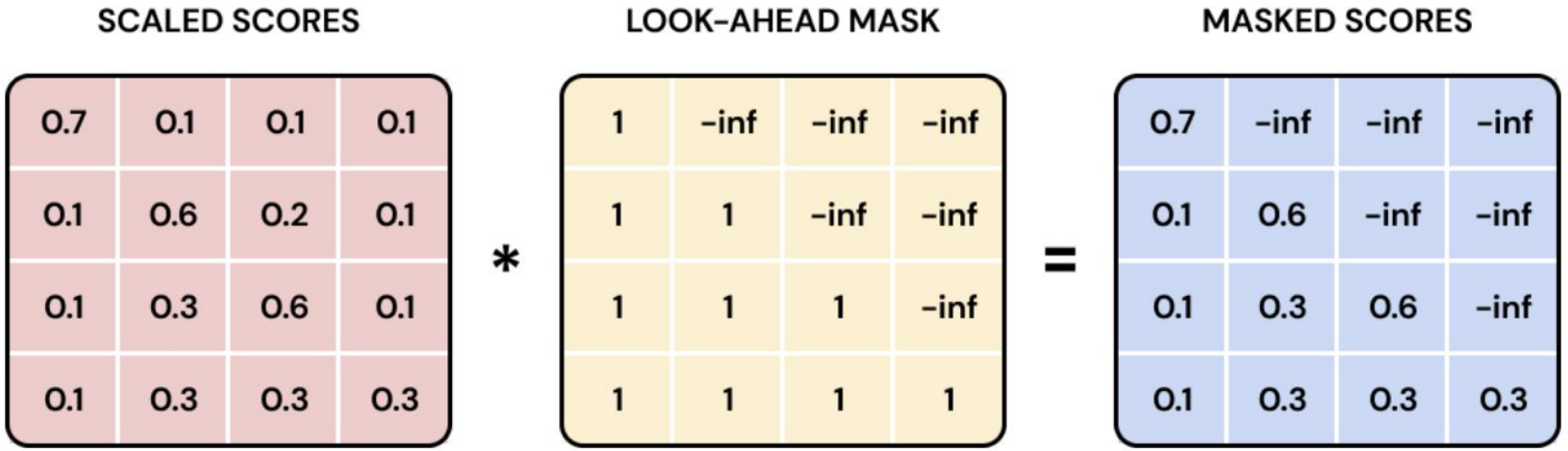
- Consumes processed representations and word embeddings
- Outputs probabilities for each word
- Each token is one input into the decoder



How does it work – The Decoder



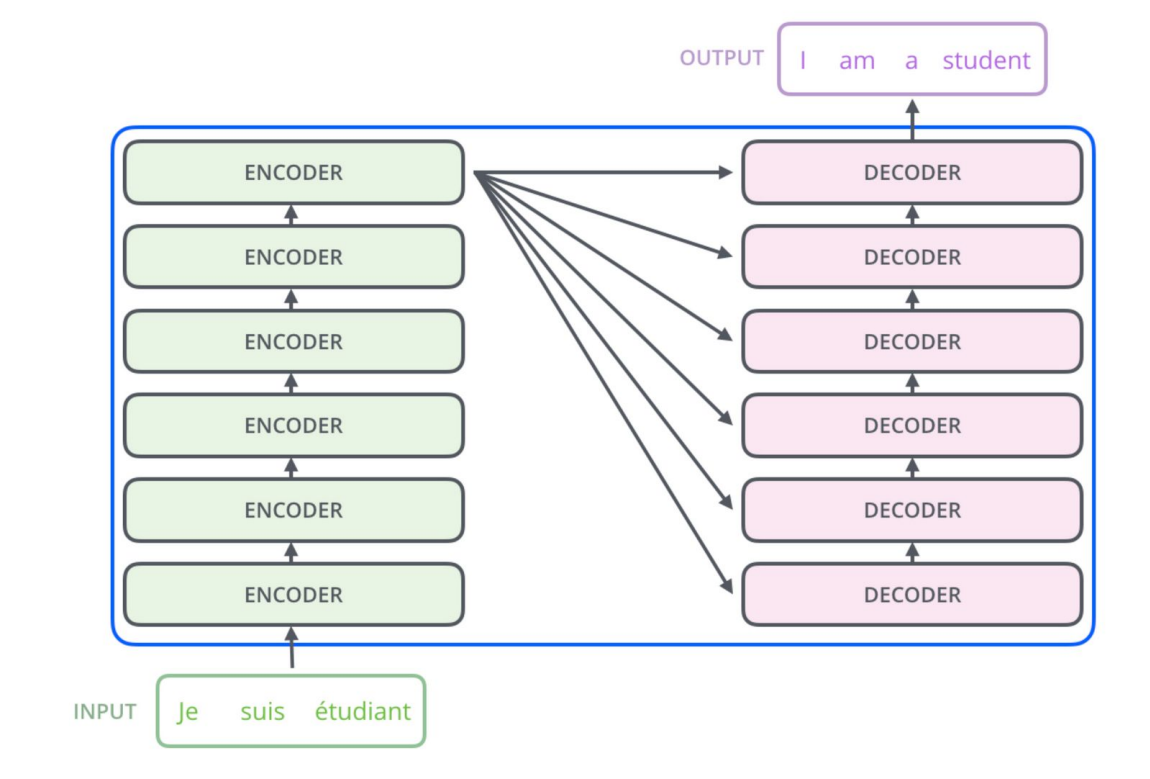
How does it work – The Decoder



A Look-Ahead Mask turns Multi-Head Attention into Masked Multi-Head Attention.

Source: https://www.revistek.com/posts/transformer-architecture

How does it work – The Decoder



How does it work?

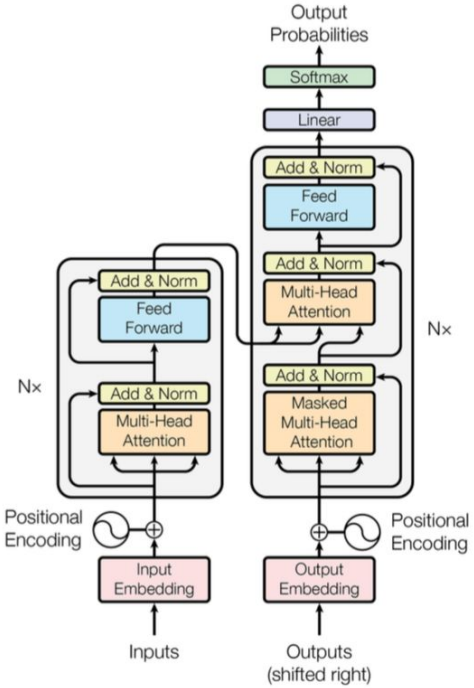


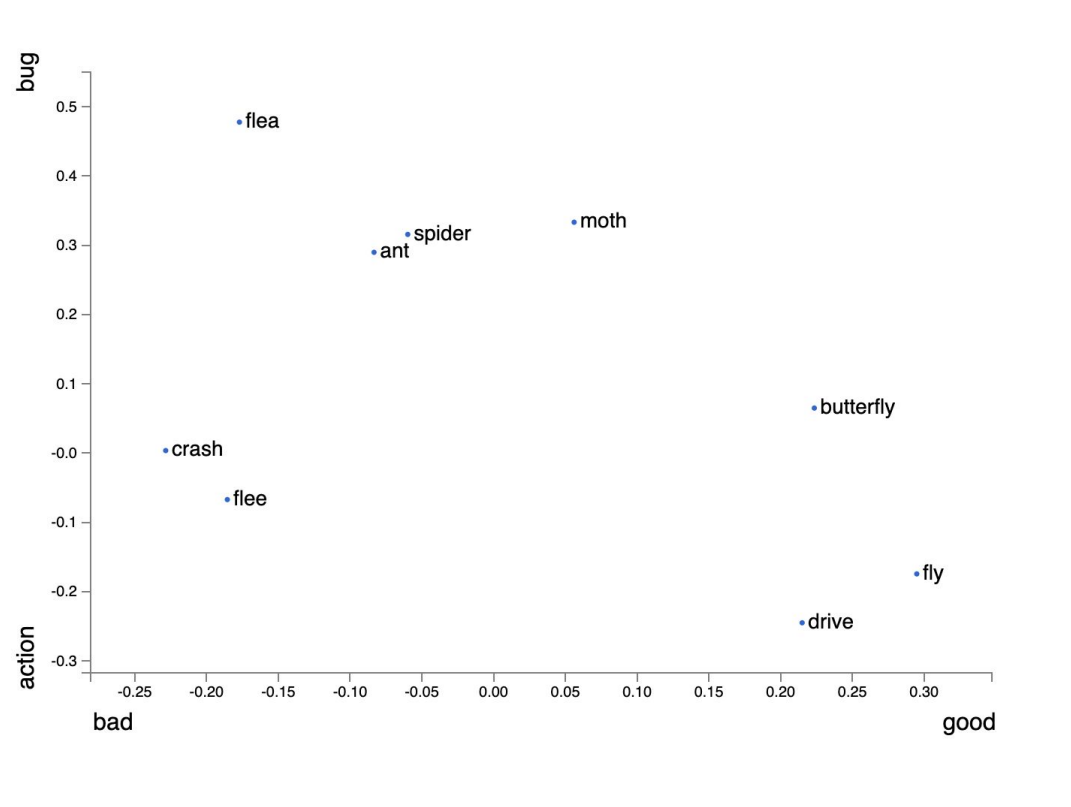
Figure 1: The Transformer - model architecture.

Attention

Why is it all we need again?

- In natural language words mean different things in different contexts
- Word embeddings hold a lot of information but context shifts
- Attention is used to solve this problem of disambiguation of words. It “looks” at different parts of the sentence, and shift the representation of words accordingly

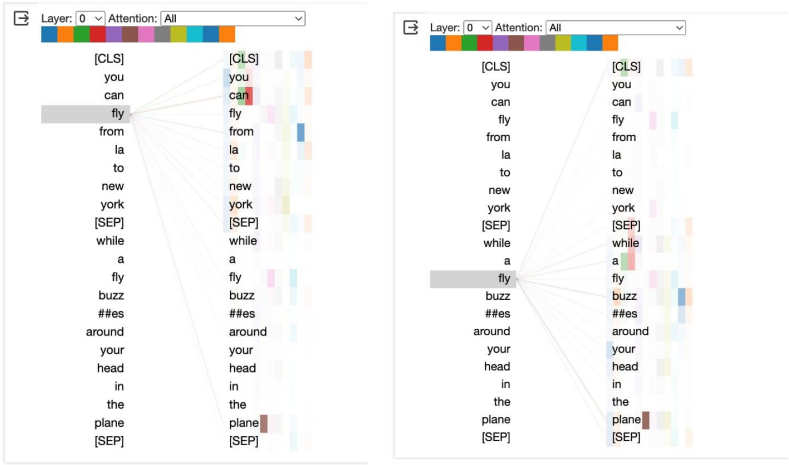
Attention


bug
action
bad
good
flea
spider
moth
ant
butterfly
crash
flee
fly
drive
0.5
0.4
0.3
0.2
0.1
-0.0
-0.1
-0.2
-0.3
-0.25
-0.20
-0.15
-0.10
-0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30

Attention

- First instance of “Fly” is a verb
  - Attends closely to the word “can” before it and “from” after it
  - Then the locations you are flying to
- Second instance of “fly” is more interested in “buzz” and “while a”
- Humans are quickly disambiguate a noun from a verb
  - E.g. “a fly” vs “can fly”

"You can fly from LA to New York
While a fly buzzes around your head in the plane"

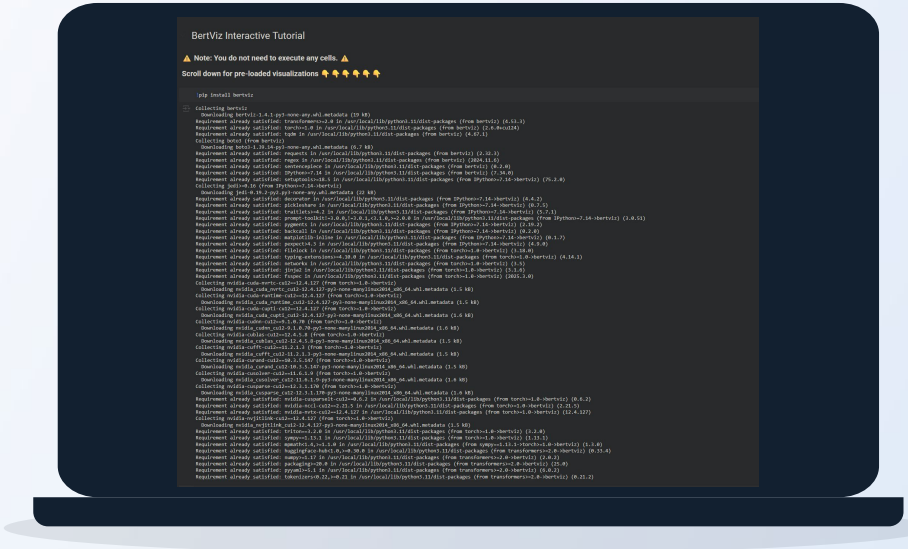


Tokens

- Basic units of data processed by an LLM
- These can be words, parts of words, or even single characters
  - Depends on the tokenization process that's been set up
- Text that is passed through the tokenizer gets encoded for the LLM
  - This is the vectorization process we will talk about in-depth later in the class
  - The vectors can be interpreted(to the human eye) as random numbers
- Encoding scheme depends on the LLM
- Encoded values will be transferred back to text after it goes through the Decoder.

Tokenization Explained

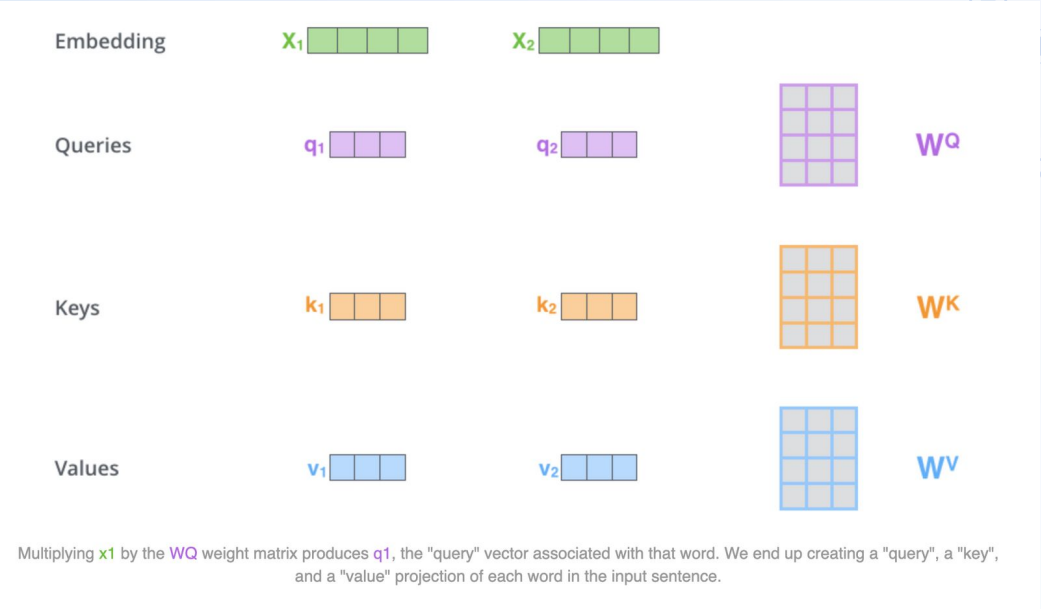
Language Learning Models (LLMs) have revolutionized the field of natural language processing, enabling machines to understand and generate human-like text. At the core of LLMs lies the concept of tokens, which serve as the fundamental building blocks for processing and representing text data. In this blog post, we'll demystify tokens in LLMs, unraveling their significance and exploring how they contribute to the power and flexibility of these remarkable models.



Lab #3 Visualizing Attention

Queries, Keys, Values

- Attention in this paper is described as mapping a “query” and a set of “key” “value” pairs to an output
- Each word in the sentence has a corresponding a “query”, “key”, “value”, and “output” vector, that is computed given a set of weights



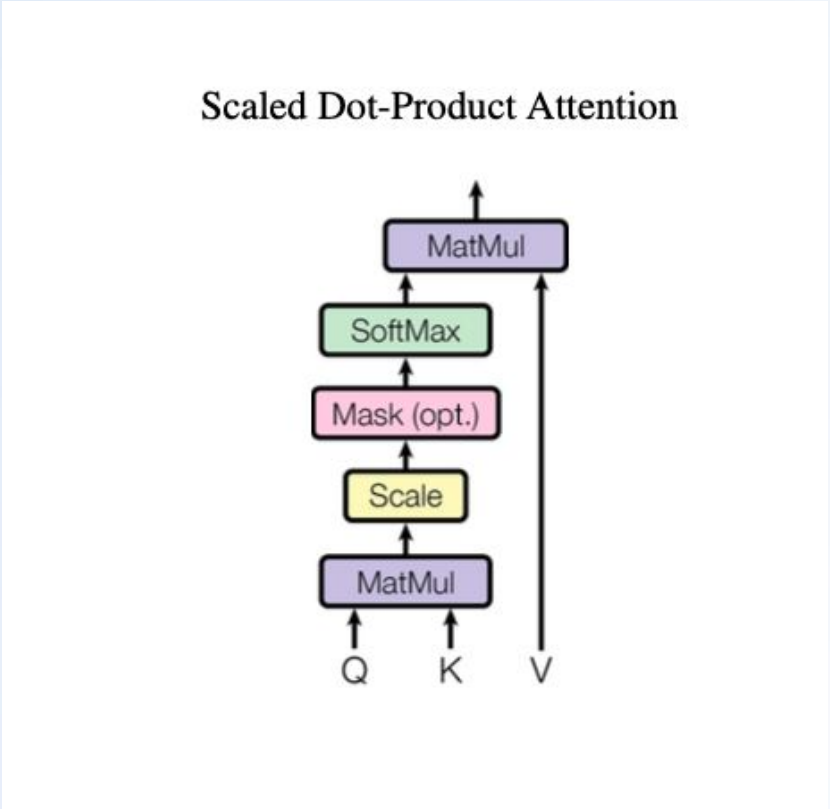
Source: https://jalammar.github.io/illustrated-transformer/

Queries, Keys, Values

- Looking at the word “fly”, scan all the words in the sentence (keys) and select the most important words (values) that distinguish this particular meaning
- You can think of “fly” sending it’s query out to all the other words saying “I am the word ‘fly’ in position 3, I am looking for surrounding prepositions or auxiliary verbs that may influence whether I am a noun or a verb”.

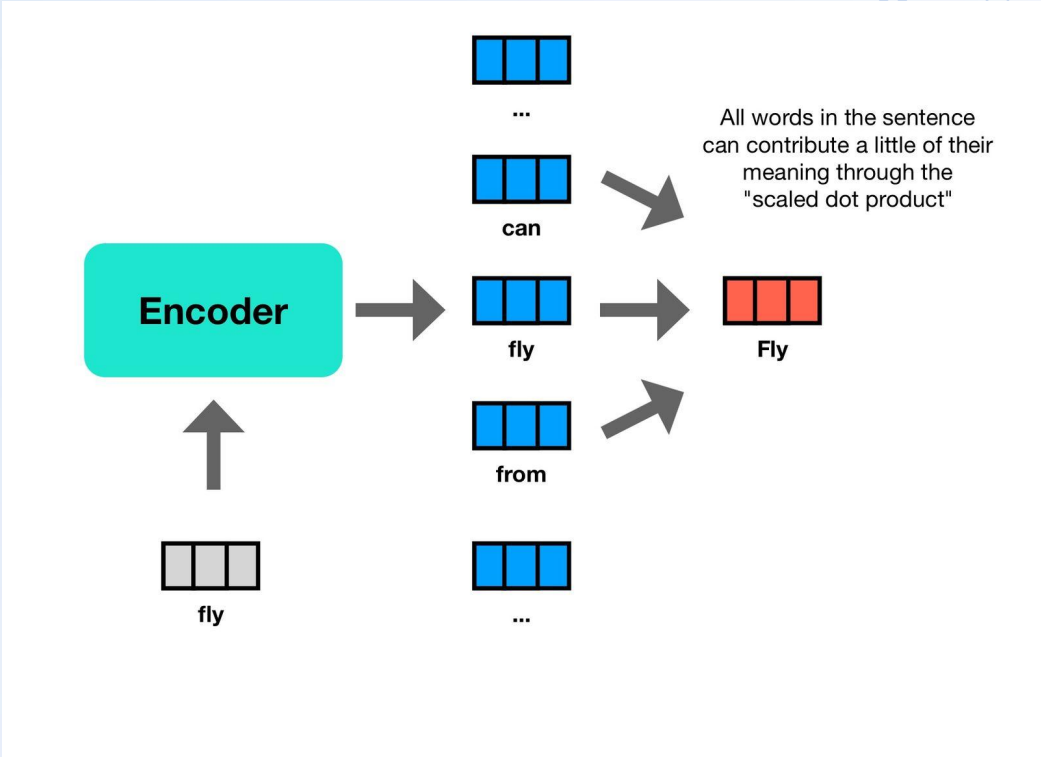
Scaled Dot Product

- The way Attention combines the words is through the “scaled dot product” of all the queries, keys, and values.



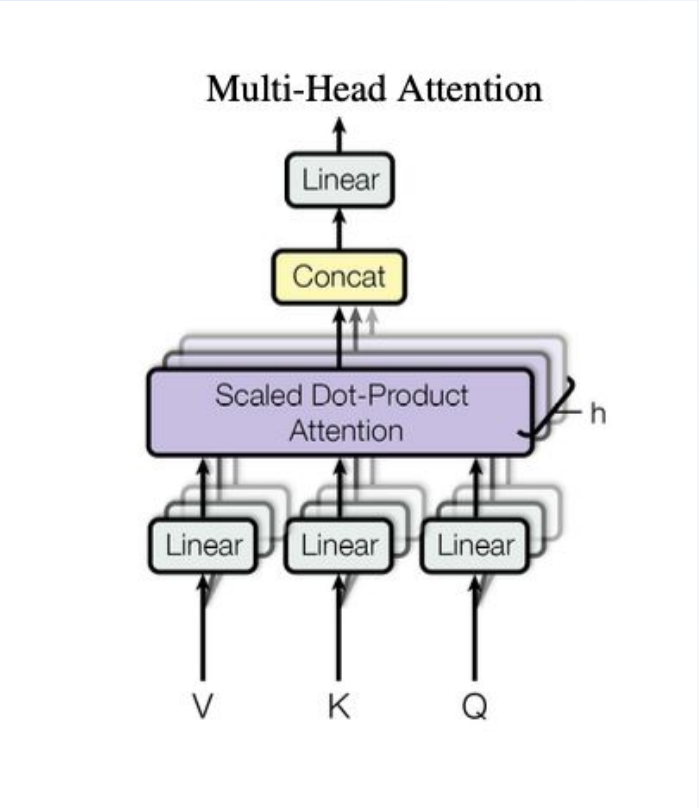
Scaled Dot Product

- First multiply the Query and the Key together, scale them, run them through a softmax to see which other words are more important.
- Then multiply the output of the softmax of Q and K by V to get the output meanings of each word.
- The “output” is a weighted sum of the “values”, or the new representation of that word, in the context of the sentence
- Each step is straightforward zoomed in but complex zoomed out



Multi-Head Attention

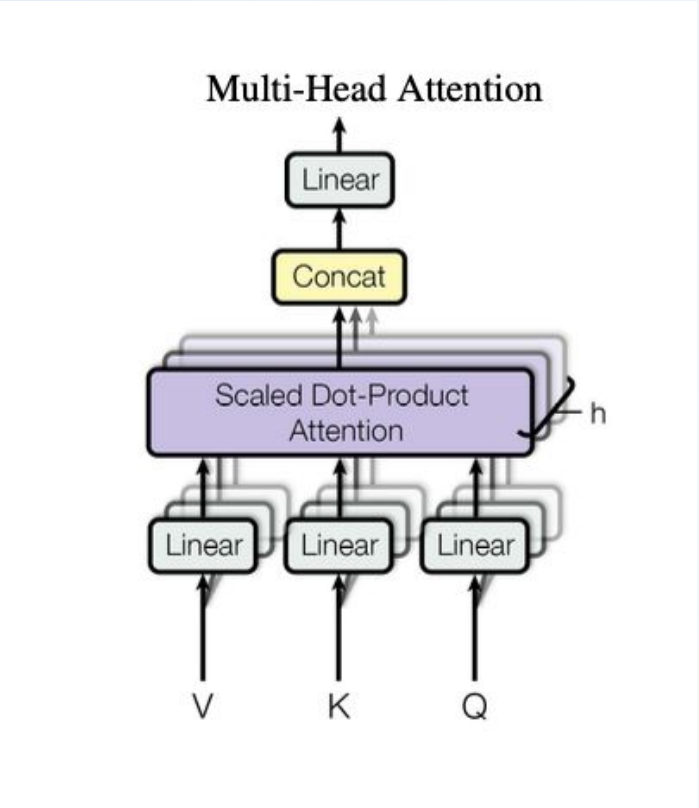
- If you simply did one layer of attention, you might reduce the amount of information and “decision making” that could flow through the network
- In practice, one layer of attention might be really good at grammar but lack the ability to disambiguate names from places (nouns)
- To solve for this they add multiple attention models
    - Each learns different patterns
    - Called “attention heads”



Multi-Head Attention

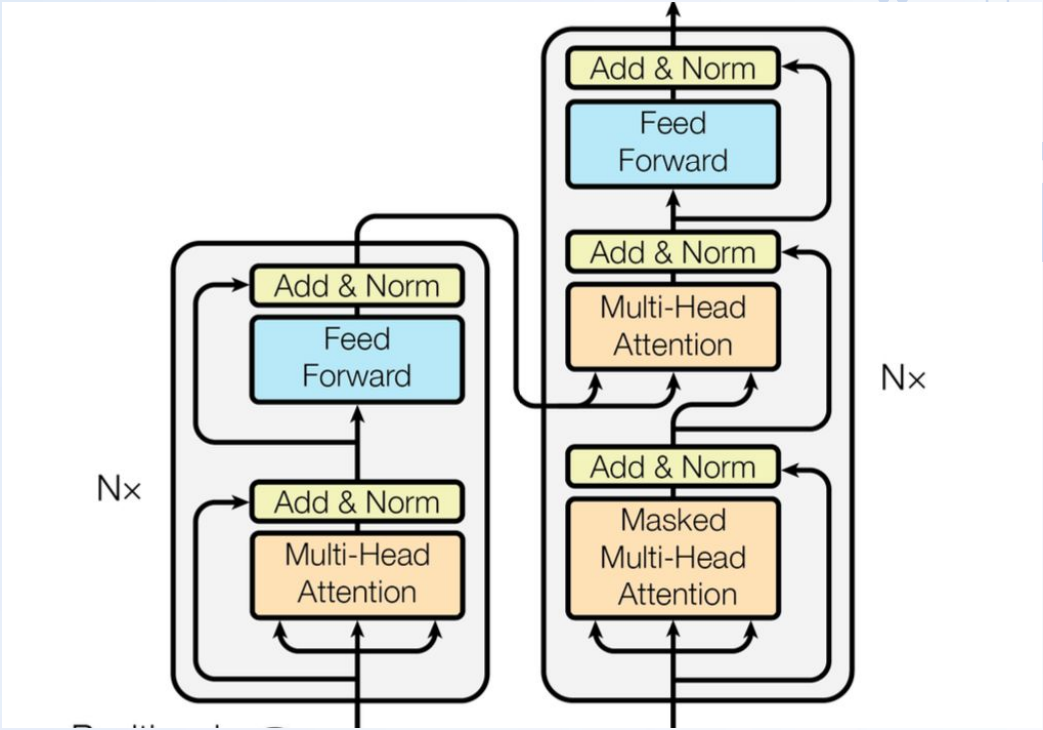
There are a few places within the model that Attention is used.

1. Communication between the encoder and decoder layers, where the “queries” come from the decoder, and look at the “keys” and “values” from the output of the encoder.
2. The encoder attends to itself, all positions in the encoder attend to all positions in the previous layer of the encoder.
3. The decoder attends to itself, but it has to mask information from being passed forward during training so that it cannot peek at the future. Because during inference it will only see each token at a time.



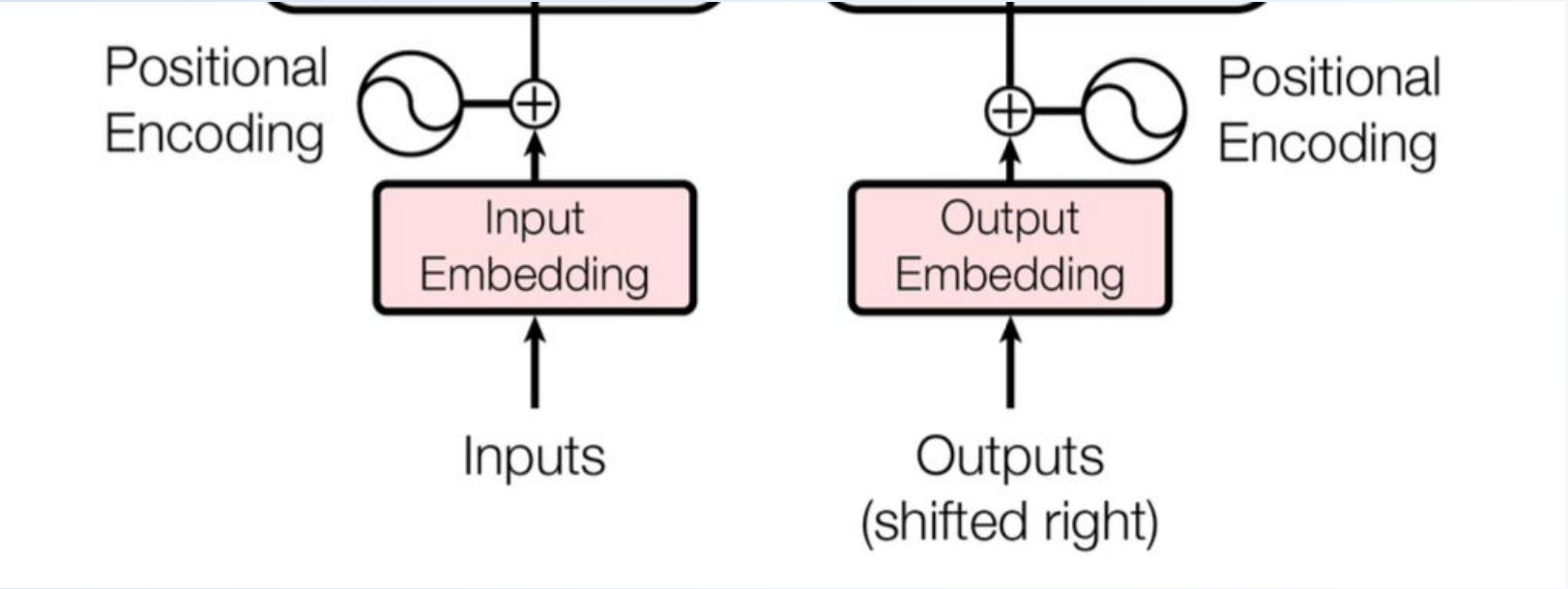
Feed Forward Networks

- Sit on top of Attention
- Each layer contains a fully connected feedforward layer

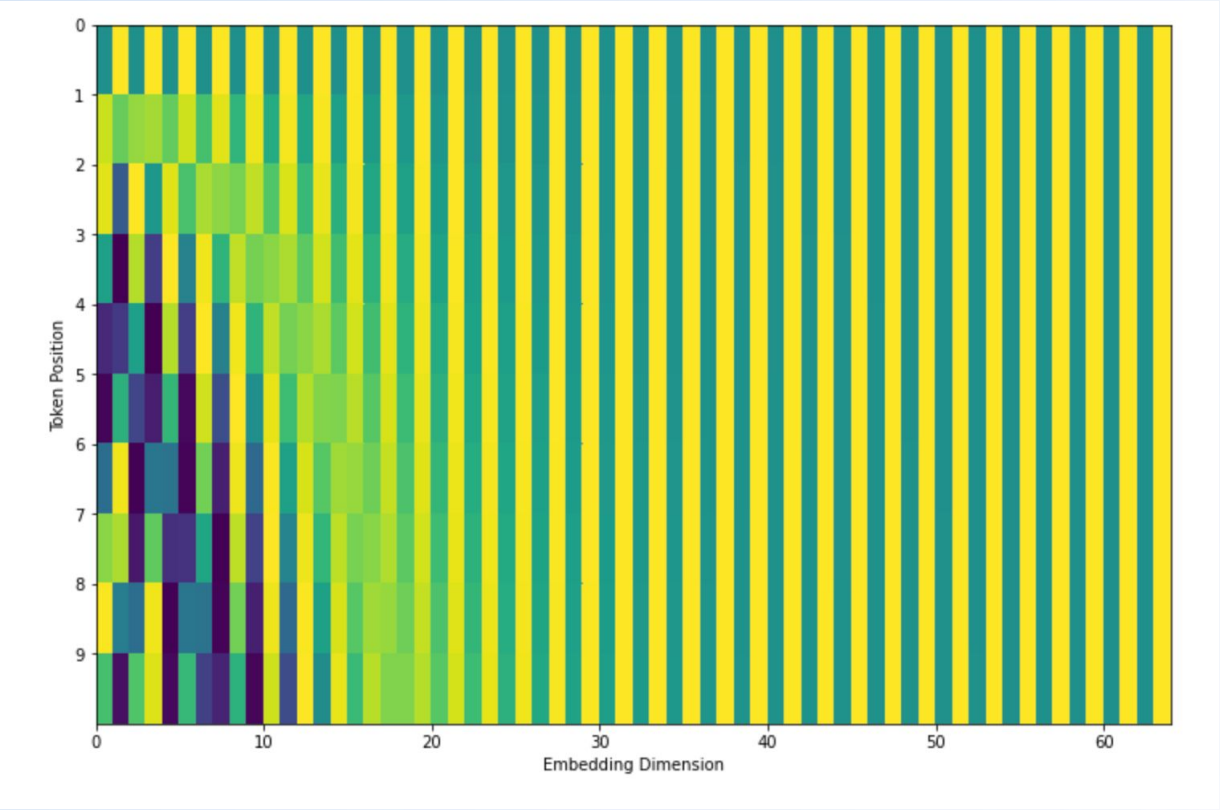


Positional Encoding

- Positional encoding added to each embedding
- No inherit order which allows us to process in parallel



Positional Encoding



Source: https://jalammar.github.io/illustrated-transformer/

Why is Self-Attention Efficient?

- This looks complicated–how is this more efficient than CNN or RNN? Takeaways:
- It can process everything in parallel (unlike RNNs)
- Computation is manageable since sequence length < model dimension in most cases
- CNNs need many layers to connect distant parts of a sequence, which adds cost.
- They mention a future idea: limit attention to nearby positions to make it even faster (not tested yet).
- Bonus: Self-attention might also be easier to interpret.

Summary on “Attention is All You Need”

Simplicity Can Enable Great Power

By removing recurrence and embracing attention as the sole computational primitive, the paper demonstrated that elegant, minimalist architectures can outperform complex, layered traditions — echoing the broader idea that clarity and focus can surpass brute complexity.

Understanding Over Sequence

The shift from rigid, sequential processing to global, context-aware attention reflects a deeper philosophical turn — that understanding language (or thought) hinges less on linear order and more on the relationships between ideas, wherever they occur.

Scalability as a Path to Intelligence

The architecture laid the groundwork for models that scale — not just in size, but in capability. It philosophically aligns with the belief that intelligence can emerge from systems designed to learn broadly, flexibly, and efficiently — not through handcrafted rules, but through universal mechanisms like attention.

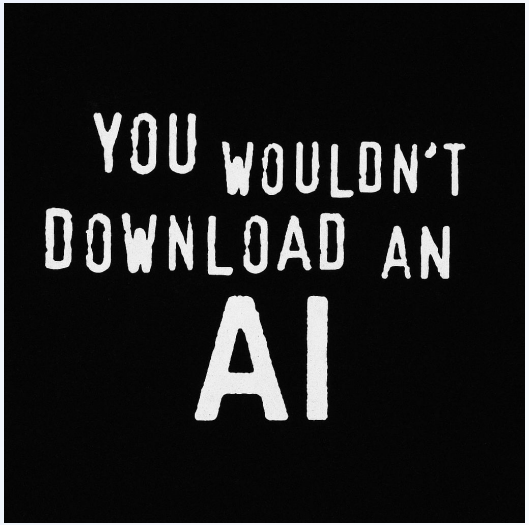
Models, Formats, and Hardware

Differences between Models

- Number of Parameters(tokens)
- Model Size
  - Mistral can greatly vary in size depending on use case
  - LLaMA has standard sizes such as 7B, 13B, 30B, etc
- Base Training Corpus
- Different tensor architecture
  - Standard Attention Model vs Sliding Window Attention
    - SAM will take the entire sequence of tokens when encoding, SWA will only take a window of tokens around the one it is focused on
- Standard vs instruct models
  - Instruct models are trained specifically with a corpus that includes instructions and responses. This in turn, alongside other modifications, makes it more useful for task-oriented applications

You wouldn’t download an AI?

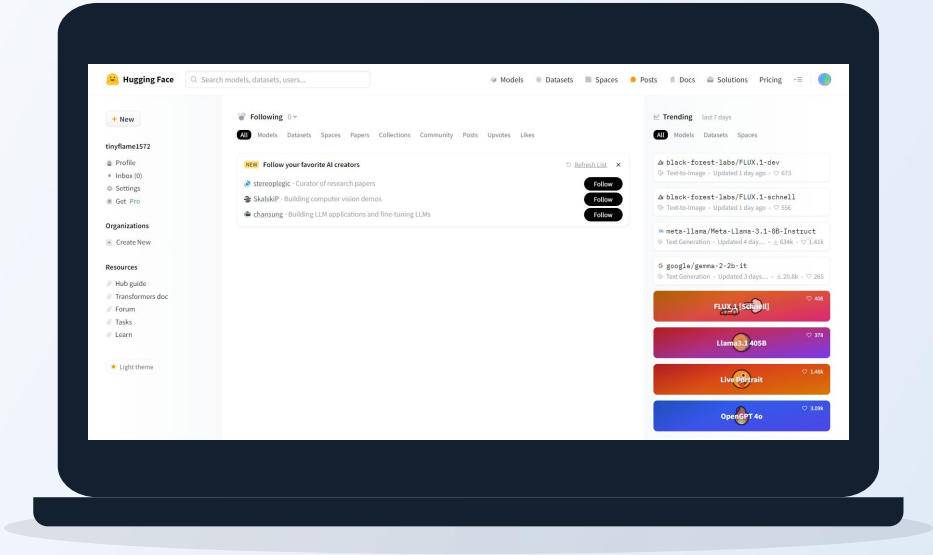
- Huggingface.co is the largest (easy to access) repo of LLMs on the internet
- Mistral, LLaMa, etc get published there
- Most LLM interfaces have an option to directly download the model from **Huggingface**
    - In this class, we use **LMStudio**
- Other LLMs, such as **Sonnet’s Claude**, can only be run from **Anthropic** or **AWS Bedrock**
    - Since **AWS** did not choose to give us money, we won’t be playing with this



What is Huggingface?

- Largest AI driven community relevant to machine learning engineers
- Central hub/landing page/community for machine learning and Artificial Intelligence
- Backend driven by Git
    - Instead of codebases it has machine learning models and datasets

Hugging Face



Live Demo Hugging Face

Problems with AI: False positives/negatives, hallucinations, etc

- Issues with the Training Data
  a. Incomplete data or too much data
  b. Too Focused or Unfocused
  c. Biased Data
- Semantic Gaps
- Lack of access to real-time data
- Incomplete or Noisy Training Data
- Too much data input

Problems with AI: False positives/negatives, hallucinations, etc

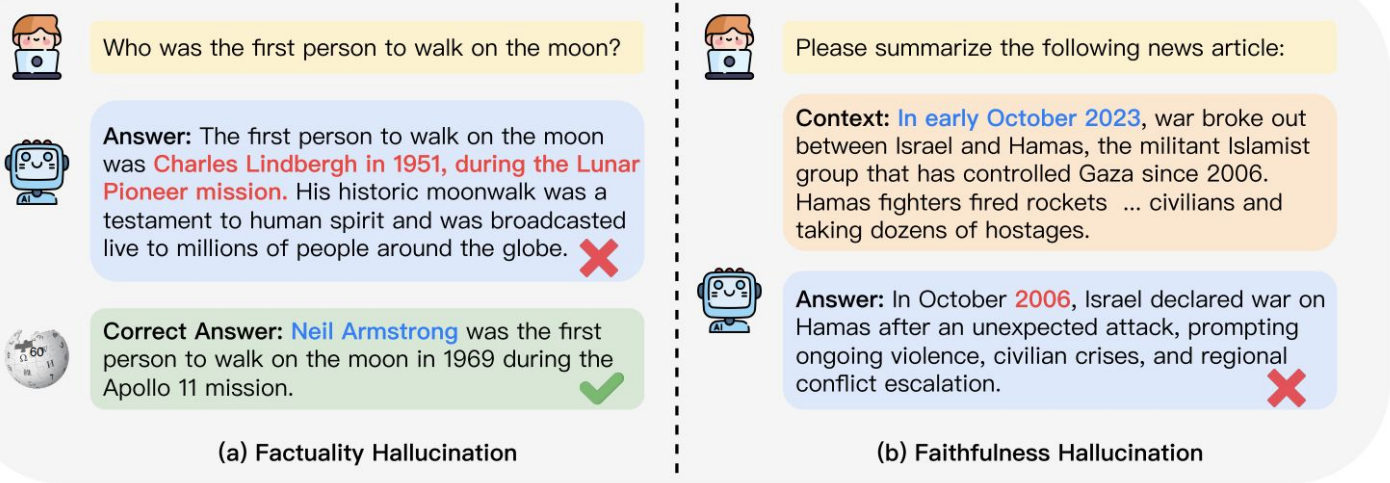


Figure 1: An intuitive example of LLM hallucination.

https://arxiv.org/pdf/2311.05232
A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions

What are Models?

- In machine learning, a model is a system that learns patterns from data and makes predictions or decisions based on what it has learned.
- Think of it like this:
  - A model is a function: it takes input (like text, images, or numbers) and produces output (like a classification, prediction, or answer).
  - It learns this function by being trained on examples — adjusting its internal parameters to reduce errors.

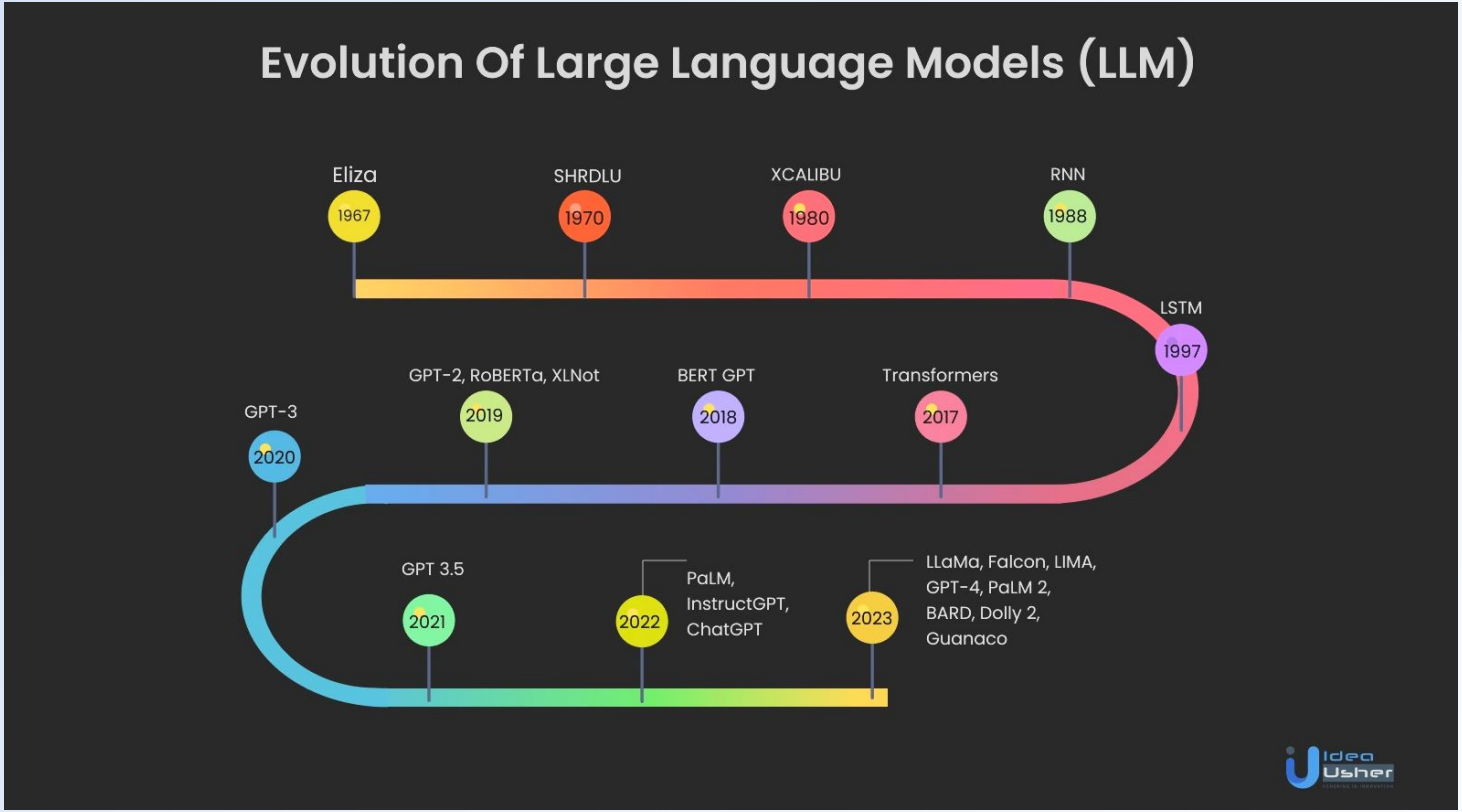
Everyday analogy:

- A model is like a recipe you've learned by practice.
- At first, you try cooking with instructions (training data). Over time, you adjust based on what works. Eventually, you can make the dish (predictions) from ingredients (input) without needing the instructions again.

What are Models?

Source: https://www.techradar.com/computing/artificial-intelligence/best-llms

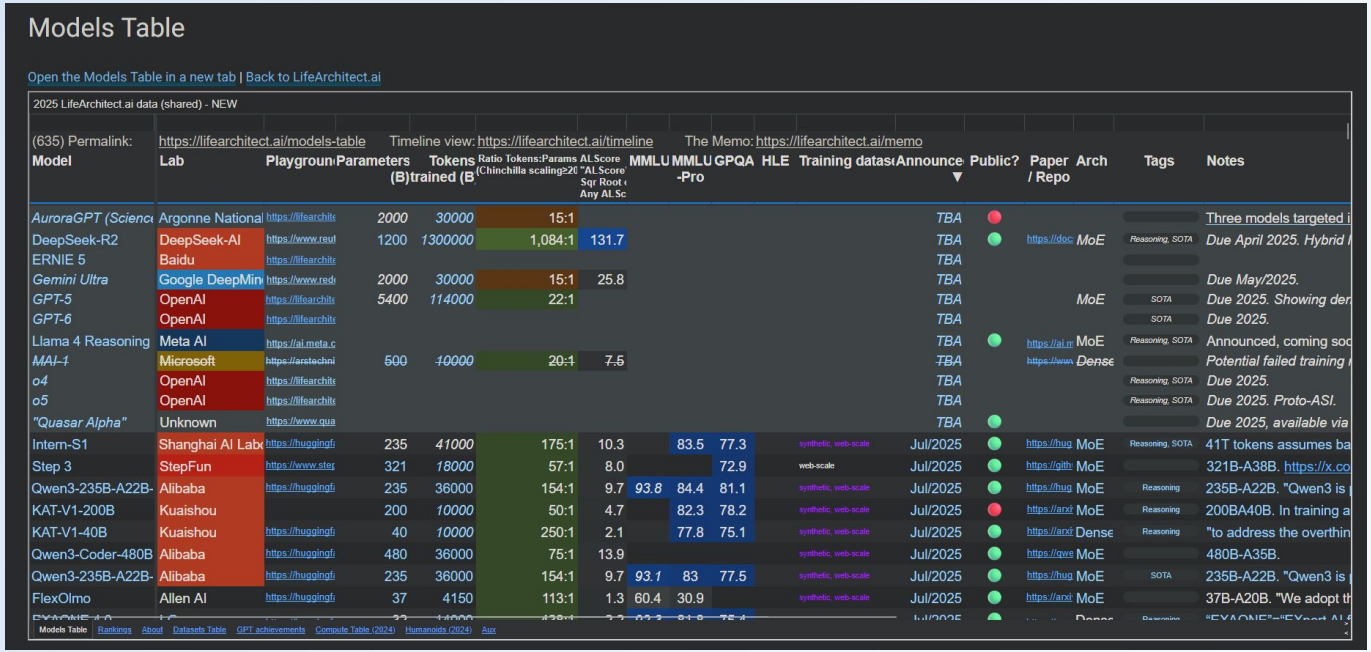
The Evolution Of Large Language Models



Source: https://ideausher.com/wp-content/uploads/2023/09/mid-image1-1.webp

Model Rankings

Rankings Visualized: https://lifearchitect.ai/models-table/



Models Table

Open the Models Table in a new tab | Back to LifeArchitect.ai

2025 LifeArchitect.ai data (shared) - NEW

(635) Permalink: https://lifearchitect.ai/models-table Timeline view: https://lifearchitect.ai/timeline The Memo: https://lifearchitect.ai/memo

Model	Lab	Playgroun	Parameters (B)	Tokens trained (B	Ratio Tokens:Params (Chinchilla scaling≥20	ALScore "ALScore Sqr Root Any ALSc	MMLU	MMLU -Pro	GPQA	HLE	Training datas	Announce ▼	Public?	Paper / Repo	Arch	Tags	Notes
AuroraGPT (Scienc	Argonne National	https://lifearchite	*2000*	*30000*	15:1							*TBA*					Three models targeted i
DeepSeek-R2	DeepSeek-AI	https://www.reut	1200	*1300000*	1,084:1	131.7						*TBA*		https://doc	*MoE*	*Reasoning, SOTA*	*Due April 2025. Hybrid*
ERNIE 5	Baidu	https://lifearchite										*TBA*					
Gemini Ultra	Google DeepMin	https://www.red	*2000*	*30000*	15:1	25.8						*TBA*					*Due May/2025.*
GPT-5	OpenAI	https://lifearchite	*5400*	*114000*	22:1							*TBA*			*MoE*	*SOTA*	*Due 2025. Showing der*
GPT-6	OpenAI	https://lifearchite										*TBA*				*SOTA*	*Due 2025.*
Llama 4 Reasoning	Meta AI	https://ai.meta.c										*TBA*		https://ai.m	MoE	*Reasoning, SOTA*	Announced, coming soc
~~*MAI-1*~~	Microsoft	https://arstechni	~~500~~	~~*10000*~~	~~20:1~~	~~7.5~~						~~*TBA*~~		https://www	~~*Dense*~~		~~*Potential failed training*~~
o4	OpenAI	https://lifearchite										*TBA*				*Reasoning, SOTA*	*Due 2025.*
o5	OpenAI	https://lifearchite										*TBA*				*Reasoning, SOTA*	*Due 2025. Proto-ASI.*
"Quasar Alpha"	Unknown	https://www.qua										*TBA*					*Due 2025, available via*
Intern-S1	Shanghai AI Lab	https://huggingfa	235	*41000*	175:1	10.3		83.5	77.3		synthetic, web-scale	Jul/2025		https://hug	MoE	*Reasoning, SOTA*	41T tokens assumes ba
Step 3	StepFun	https://www.step	321	*18000*	57:1	8.0			72.9		web-scale	Jul/2025		https://gith	MoE		321B-A38B. https://x.co
Qwen3-235B-A22B-	Alibaba	https://huggingfa	235	36000	154:1	9.7	*93.8*	84.4	81.1		synthetic, web-scale	Jul/2025		https://hug	MoE	*Reasoning*	235B-A22B. "Qwen3 is
KAT-V1-200B	Kuaishou		200	*10000*	50:1	4.7		82.3	78.2		synthetic, web-scale	Jul/2025		https://arxi	MoE	*Reasoning*	200BA40B. In training a
KAT-V1-40B	Kuaishou	https://huggingfa	40	*10000*	250:1	2.1		77.8	75.1		synthetic, web-scale	Jul/2025		https://arxi	Dense	*Reasoning*	"to address the overthin
Qwen3-Coder-480B	Alibaba	https://huggingfa	480	36000	75:1	13.9					synthetic, web-scale	Jul/2025		https://qwe	MoE		480B-A35B.
Qwen3-235B-A22B-	Alibaba	https://huggingfa	235	36000	154:1	9.7	*93.1*	83	77.5		synthetic, web-scale	Jul/2025		https://hug	MoE	*SOTA*	235B-A22B. "Qwen3 is
FlexOlmo	Allen AI	https://huggingfa	37	4150	113:1	1.3	60.4	30.9			synthetic, web-scale	Jul/2025		https://arxi	MoE		37B-A20B. "We adopt th

Models Table | Rankings | About | Datasets Table | GPT achievements | Compute Table (2024) | Humanoids (2024) | Aux

Source: https://www.techradar.com/computing/artificial-intelligence/best-llms

Model Formats

- **Lightweight & Deployment-Focused Formats**
  - **SAFETENSORS (.safetensors)**
    - Raw format of tensor files for the LLM
    - “SAFE” since it does not allow ACE and is extremely limited to just tensors
  - **GGML**
    - General Graph Model Library
    - Made by the same person who started llama.cpp
    - Quantized version of a model
      - Allows for greater efficiency
    - **Now Deprecated.. Replaced by…**
  - **GGUF**
    - V 2.0 of GGML
    - Allows more data about the model itself to be stored within the file
    - Allows special token and custom prompt templates to be embedded
    - More backwards compatibility and less breaking things
      - Taken from a GitHub issue about GGUF’s Launch

Model Formats

- **Training & Inference Frameworks**
  - **PyTorch (.pt / .pth)**
    - Native model format for PyTorch — one of the most popular deep learning libraries
    - Stores:
      - Model weights (state_dict)
      - Sometimes model architecture (if using torch.save(model))
    - Common in **research**, **custom model development**, and **experimentation**
    - Can be scripted or exported to ONNX for deployment
    - Often paired with Python for training and Jupyter notebooks for prototyping

Model Formats

- **Training & Inference Frameworks**
  - **ONNX (.onnx)**
    - Stands for Open Neural Network Exchange
    - Interoperable format to convert models from PyTorch, TensorFlow, etc.
    - Enables running models on multiple backends:
      - Edge devices (e.g., phones, IoT)
      - Web (e.g., ONNX.js)
      - C++ / C# environments
    - Supported by tools like **ONNX Runtime**, **OpenVINO**, **TensorRT**, and **DirectML**
    - Ideal for **high-performance**, **cross-platform deployment**

Model Formats

- **Training & Inference Frameworks**
  - **HuggingFace Transformers Format**
    - Used by HuggingFace's popular 🤗 Transformers library
    - Not a single file, but a directory of components:
      - **pytorch_model.bin**: model weights (PyTorch format)
      - **config.json**: architecture/configuration info
      - **tokenizer.json** / **vocab.txt**: tokenizer and vocabulary
      - **generation_config.json**, **special_tokens_map.jso**n, etc.
    - Allows for easy loading, sharing, and deployment of models
    - Supports both **PyTorch and TensorFlow backends**
    - Can be used with the **transformers** library or converted to **ONNX** / **TorchScript**

Model Formats

- **TensorFlow Ecosystem**
  - **Keras (.h5)**
    - Stores the entire model: architecture, weights, and optimizer state
    - Ideal for quick saving/loading in Keras projects
    - Format is HDF5 (Hierarchical Data Format), human-readable with tools like HDFView
    - Not ideal for TensorFlow Serving; mainly used during development
  - **TensorFlow SavedModel (.pb folder)**
    - Standard format for TensorFlow Serving and deployment
    - Directory contains:
    - saved_model.pb: serialized model graph
    - /variables/: model weights
    - /assets/: extra files (e.g., vocabularies)
    - Used in production pipelines, TF Lite, TF.js, TF Serving

Popular LLMs

- **GPT**
  - The flagship model is GPT 4.1
    - Variations include 4o, 4o mini, Turbo
  - The de-facto LLM popularized by OpenAI
  - Used by many around the world
- **Gemini**
  - Google’s LLM - a direct response to OpenAI
- **LLaMa**
  - Flagship Model LLaMA 4
  - Meta’s response to the other two
- **Copilot**
  - Popularized by Microsoft
  - Based on GPT-4 Turbo
    - A result of Microsoft ownership/stake in OpenAI
    - If you can’t beat them, buy them

Multimodal LLMs

- 🧠 **Multimodal LLMs (M-LLMs)**
  - **Designed to process multiple input types** (e.g., images + text)
  - Treat all inputs as tokens, enabling unified language-like processing
    - Example: Ask a question about an image → Model outputs a natural language answer
  - **Vision Transformers (ViT)** laid the foundation by segmenting images into tokens
  - **Limited reasoning**, but improving rapidly
  - M-LLMs can handle **more input modalities** (audio, video, etc.)
  - Growing ecosystem of **experimental and niche models**

DALL-E, Stable Diffusion, and the artistic family

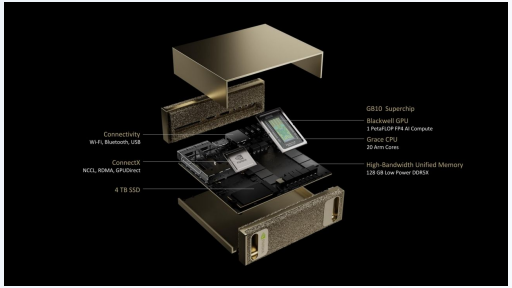
- 🎨 **Generative AI Tools (Like DALL·E)**
  - Use **LLMs** to process user input (text prompts)
  - Output is passed to a specialized generative model to create images
  - These tools are not **LLMs** themselves, but rely on them for understanding instructions
  - The original **DALL·E** was based on **GPT-3**, but scaled down (12B vs. 175B parameters)
  - **Purpose-built for image generation**
  - Introduced **zero-shot text-to-image generation in Feb 2021**

LLM Hardware

- **LLMs** can work on compute small and large
  - Different models tuned and created for different use cases and hardware
- **LLMs** like the ones we'll be utilizing require a hefty amount of compute
  - Hobbyist/at-home Level
    - RTX 3090 and up tend to be the best due to the VRAM available
  - Consumer/Corporate Usage
    - Custom-designed chips such as Amazon's A10G(Nvidia)
- Models are loaded into the card's VRAM, or on-board memory
  - Extremely fast- multitudes of times faster than running from disk or even normal RAM
- **LLMs** can be run on CPU only, however it will be pretty slow
  - My 7950X3D was not having a fun time during this project
- Our Primary Model will be using a **Nvidia T4**

LLM Hardware

- **AIO LLM Hardware**
  - **AMD Ryzen AI MAX+ 395 “Strix Halo APU”**
    - “First to market”
    - https://www.amd.com/en/developer/resources/technical-articles/2025/amd-ryzen-ai-max-395--a-leap-forward-in-generative-ai-performanc.html
    - 16 core SOC with integrated GPU and tons of unified memory for loading LLMs (128GB!)
    - 50~ TOPS
  - **NVidia DGX Spark**
    - First announced but not released yet (good job nvidia)
    - https://www.nvidia.com/en-us/products/workstations/dgx-spark/
    - Similar specs but 20 cores, 128GB unified memory
    - Claims absurd # of TOPS but yet to be seen



Edge AI

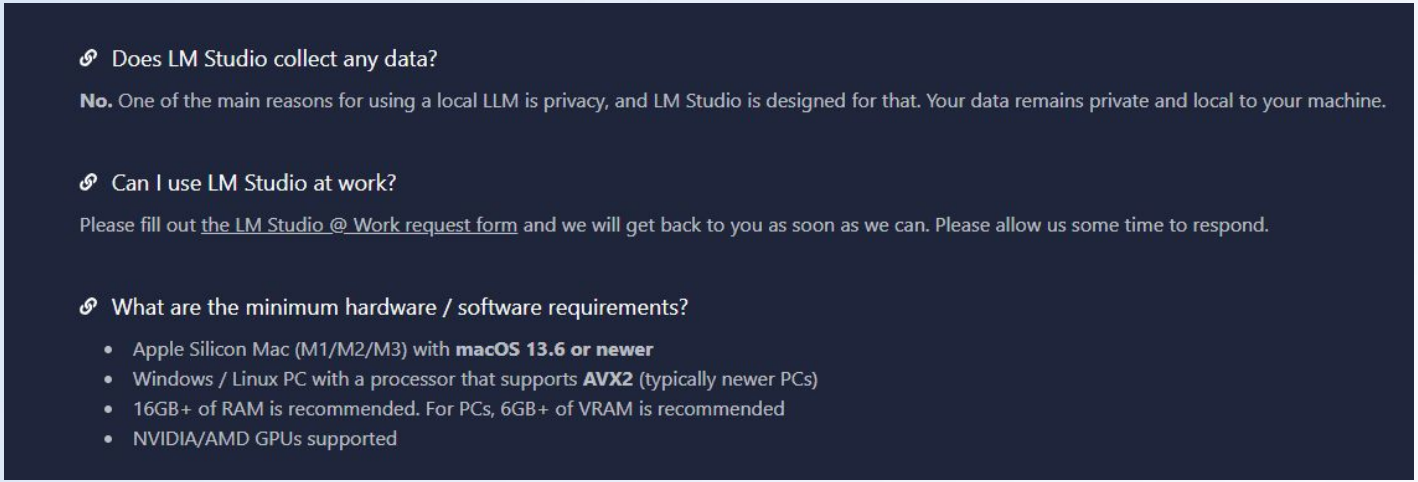
- Run LLMs on local devices
  - This can be anything from Nvidia Jetson units to your computer
- Several Primary reasons to utilize Edge AI
  - No reliance needed on a network connection post-model download
  - All data is stored locally and not shipped off to our great overlord Sam Altman
  - Ability to use lower-quantized models locally and on cheaper devices
    - Greater versatility
- Typically Edge AI is run on less powerful hardware
  - This means that a weaker, or less competent, LLM must be used
  - Aimed towards singular/simpler tasks
    - OCR, Summarization, Autocorrect

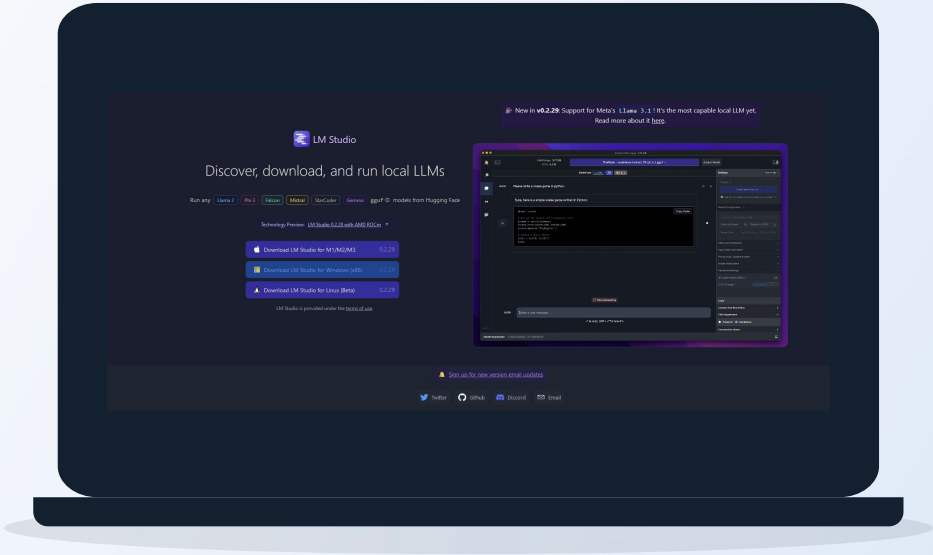
Introducing LMStudio

- Made possible by the **Llama.cpp** project
- Desktop app for running local LLMs
  - https://lmstudio.ai/docs/welcome
- Works on Windows, Linux, and MacOS
- Ability to run OpenAI-like server with endpoints such as
  - **/v1/chat/completions**
  - **/v1/completions**
  - **/v1/embeddings**
- Will be a cornerstone of the rest of the labs for the training

Introducing LMStudio

- Fairly Open License
- If you plan to use this for work (beyond your own personal development) you will need to fill out a request form
- Check out the website for the current information
    - https://lmstudio.ai/



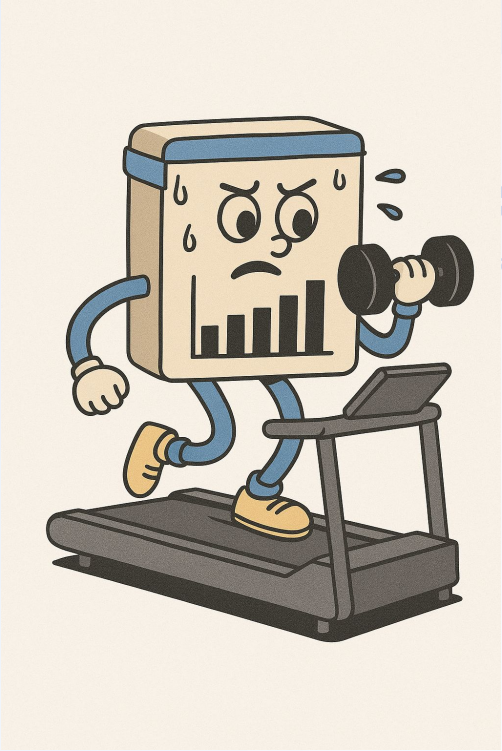


Lab #4 Installing LMStudio

Model Training

How are Models Trained?

- The 4 Stages of Training
  - Data Preparation
  - Pre-training Through Self-Supervised Learning
  - Supervised Fine-Tuning (SFT)
  - Reinforcement Learning from Human Feedback (RLHF)

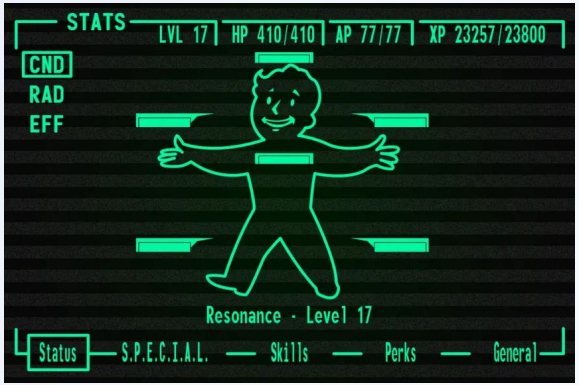


How are Models Trained?

- Goal: Learn general language patterns
  - LLMs are trained on massive text corpora (books, websites, code, etc.).
- They use a self-supervised task, like:
  - Next word prediction (GPT-style)
  - Masked word prediction (BERT-style)
- No human labeling needed — the model learns by predicting missing or next parts of the text
- This phase teaches grammar, facts, reasoning patterns, and broad knowledge.

Fine-Tuning: What is LoRA

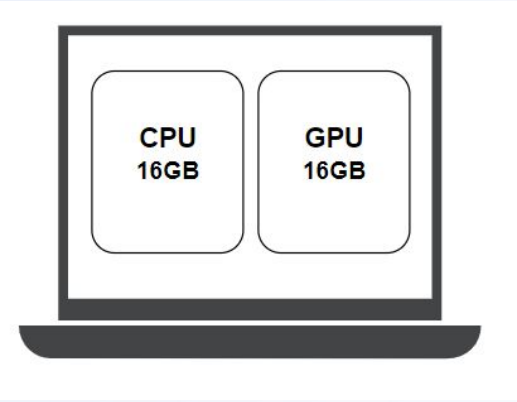
- Low-Rank Adaption
- A way to train a model in a way that lowers the overall memory requirement
- Freezes certain parameters within the model, and changes a separate set which are combined with the original model
- Adds in a small set of trainable parameters
- Basically a Pip-Boy for your LLM



Why LoRA

- Training models is computationally expensive
- ***ZeRO: Memory Optimizations Toward Training Trillion Parameter Models***
- **https://arxiv.org/abs/1910.02054**

- **Example: Fine Fine 10B Parameter Model**
  - We have a computer with 16GB RAM and 16GB GPU VRAM
  - Standard way to do this is the FP16 number format
  - Each parameter corresponds to a number
  - Requires 2bytes of memory per parameter
  - 10B * 2bytes = 20GB of memory just to store the model parameter

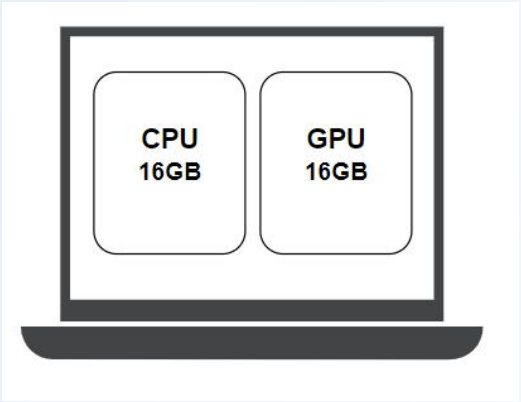


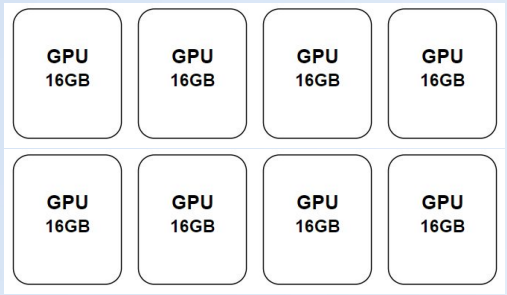
Why LoRA

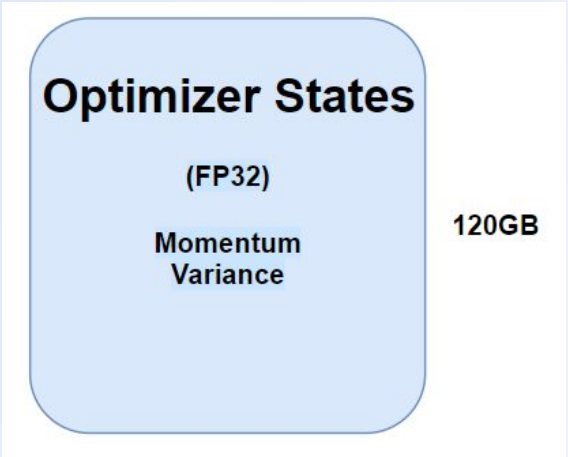
- In our hypothetical example we need a bucket of 20GB
  - Can get clever and distribute the memory load
  - Allows us to do things like inference and make predictions
- But retraining requires more than just storing the storing the parameters
- Gradients (FP16) also require another 20GB
- Optimizer States
  - If we use Adam (FP32) it will use another 120GB
  - Stores Momentum value and Variance value (*2 per parameter)
  - Because it’s stored in FP32 increases memory footprint more
  - All said it’s about a 12x multiplier

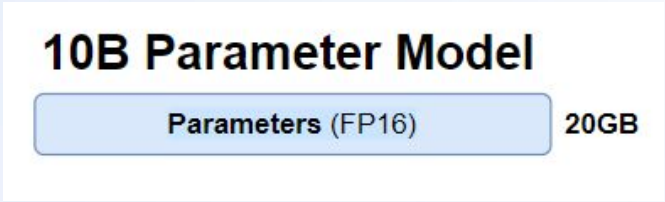
Why LoRA

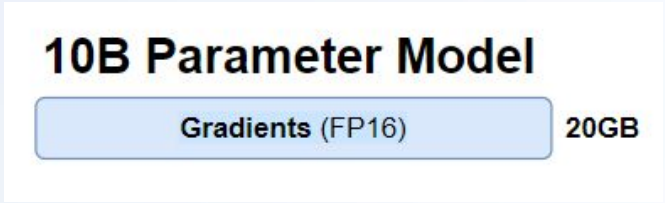
- **Total requirement:160GB to train a new model**
- Final memory footprint looks like this
- Obviously this doesn't fit on our laptop!











What is Quantization

- Quantization is a compression technique that involves mapping high precision values to a lower precision one

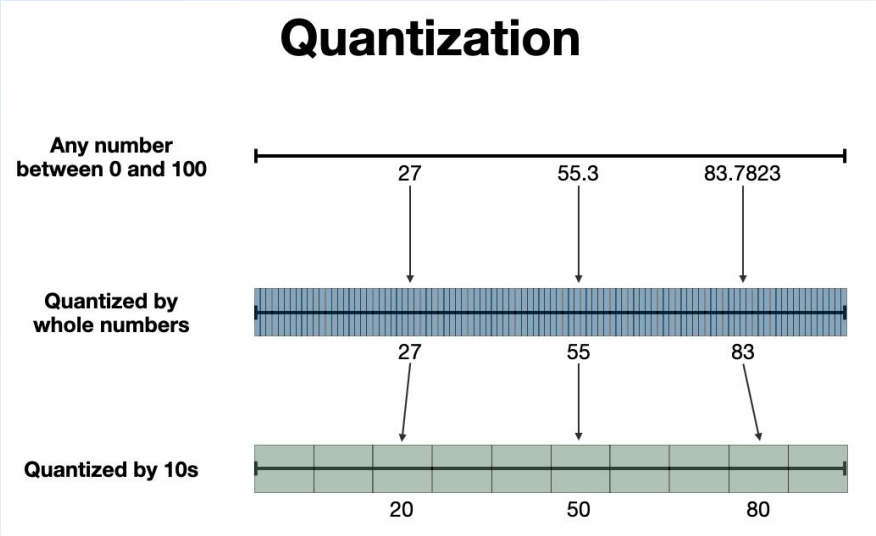


Image source - https://shawhin.medium.com/

QLoRA

Example:

- Let’s take a look at how QLoRA works
- QLoRA is “...an efficient finetuning approach that reduces memory usage enough to finetune a 65B parameter model on a single 48GB GPU while preserving full 16-bit finetuning task performance. QLoRA backpropagates gradients through a frozen, 4-bit quantized pretrained language model into Low Rank Adapters~(LoRA)”
- One of many methods of fine-tuning, using specific quantization techniques to compress parameters and improve performance
- ***https://arxiv.org/abs/2305.14314***

QLoRA

- 4-bit NormalFloat
  - 2^4 (2*2*2*2) = 16 unique combinations
  - 16 buckets for quantizations
  - 0.5bytes per parameter
    - 10B parameters = 5GB of memory
- Equally-sized buckets vs equally-space buckets
- Weighted by normal distribution

QLoRA

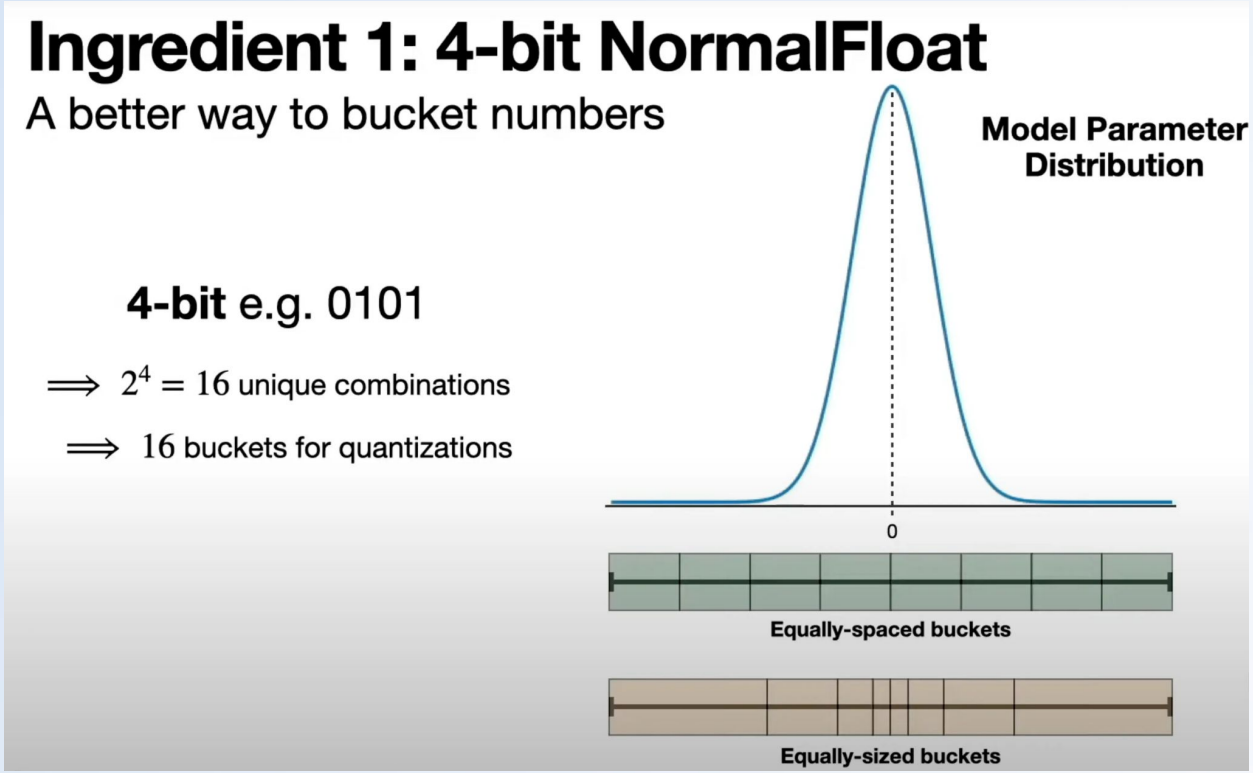
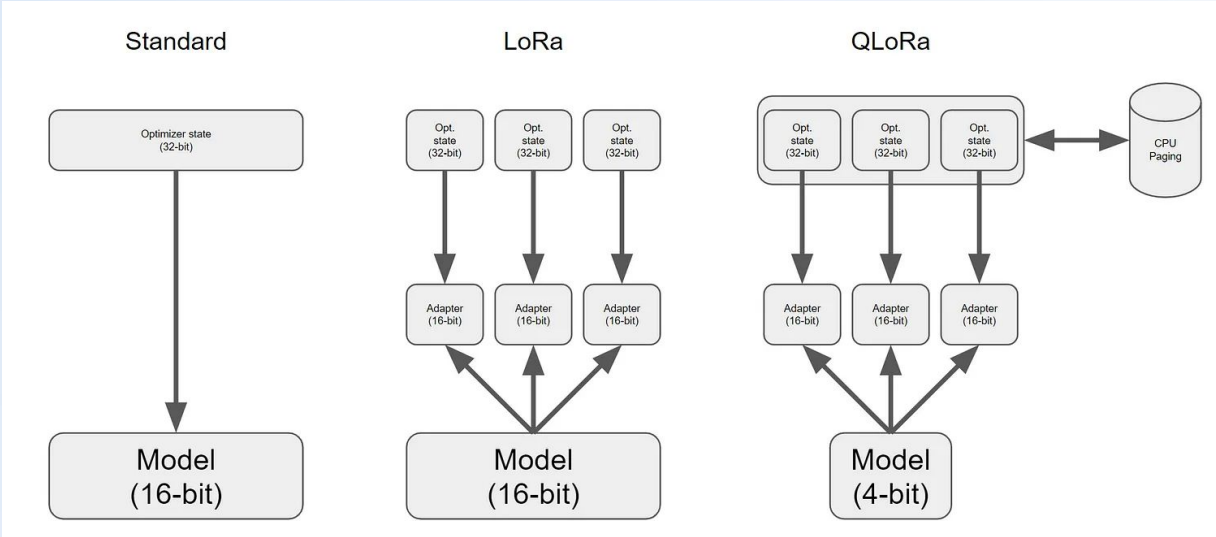


Image source - https://shawhin.medium.com/

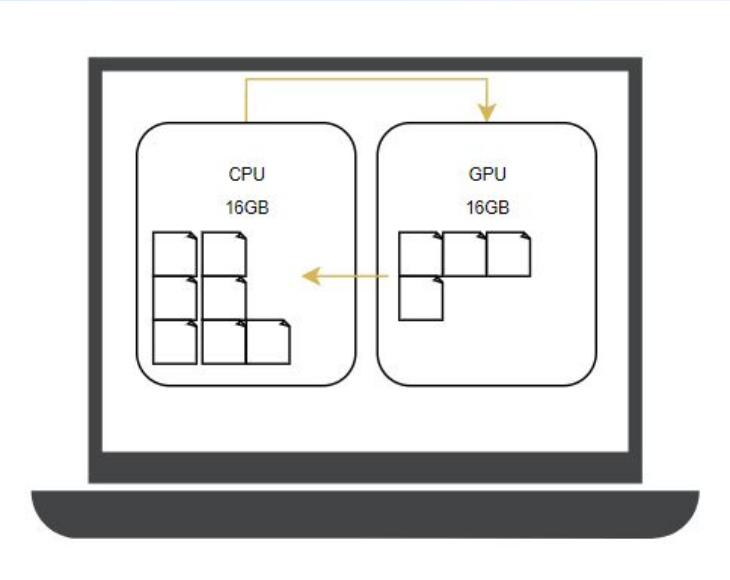
QLoRA

- Double Quantization
- ***https://proceedings.neurips.cc/paper_files/paper/2019/file/ea4eb49329550caaa1d2044105223721-Paper.pdf***
- The process of quantizing the quantization constants for additional memory saving



QLoRA

- Page Optimizer
- Allows us to move memory as needed from CPU into the equation
- Uses Nvidia unified memory to manage the memory pages between the CPU and GPU



Why choose using LoRA over traditional training methods?

- Gives a model a focus in tasking while not completely killing it's "all-roundedness"
  - Assuming your training data isn't 22,000 Discord messages from someone who's insane
- Requires much less training time and resources to complete training
- May also cut down on resources needed to run the actual model
- The process is reversible. Using a different LoRA file will transform the model for another task

12:00pm - 1 hour Lunch

Return by 1:00pm

Llama.cpp

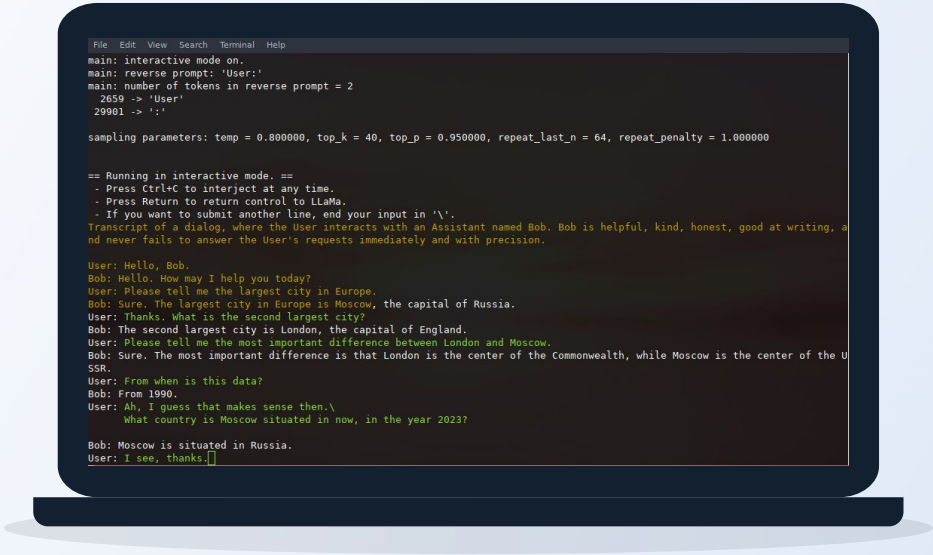
- The main goal of llama.cpp is to enable LLM inference with minimal setup and state-of-the-art performance on a wide variety of hardware - locally and in the cloud

Llama.cpp

- Low-overhead easy-to-use simple wrapper for LLM interface
  - Minimal setup
- CLI only, works on almost any hardware
  - Supports CUDA and AMD GPUs
- Supports hybrid inference
  - Remember: combining CPU and GPU to increase overall VRAM capacity!
- Not as friendly as text-gen-webui or LMStudio (what we will be using here)

LMStudio

- Wrapper for Llama.cpp
  - We are using this because it has a nice and easy to use GUI and both of us are partially blind
- While this wrapper isn't open source, Llama.cpp is
  - The commands and functions used in this training should work with bare llama.cpp with little to no modifications
- Other Alternatives include OobaBooga's Text-webui-gen or Kobold



Lab #5 Installing Llama.cpp

Wading into Opensearch

Basics on Cloud Usage

- Everyone will have access to a shared OpenSearch VM querying a model running on a cluster of Tesla T4 GPUs*
  - Subject to change on availability
- Please do not run the LLM or query it needlessly
  - Doing so will slow the entire class down
- Decided on a shared cluster model of OpenSearch/GPU computation due to class restraints
  - I.E. time, troubleshooting, etc.

What is OpenSearch

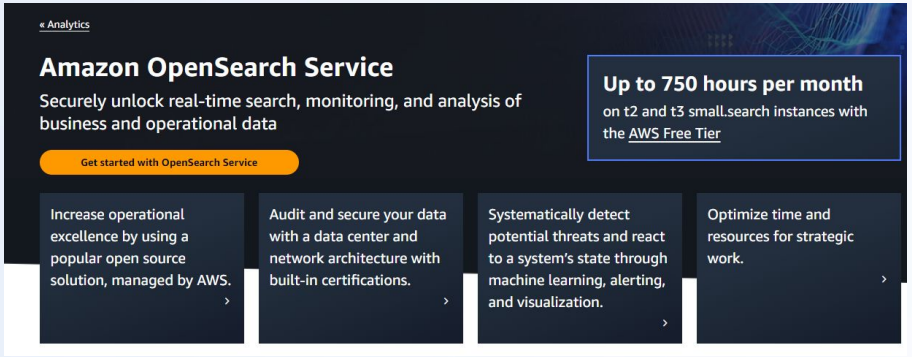
- OpenSearch is a distributed open-source search and analytics suite
- Many different uses
  - Real-time application monitoring
  - Log analytics
  - Website searching
- Highly scalable system for fast access to large volumes of data
- Integrated visualization tool via OpenSearch Dashboards
- Powered by Apache Lucene search library
- Lots of analytic capabilities

ElasticSearch vs OpenSearch

- Forked from ElasticSearch by AWS after someone had a bar fight
- Elastic currently licensed with Service Side Public License (SSPL) and Elastic License
  - Many customers did not see change as they use Elastic via Amazon Elasticsearch Service
  - But it was a huge change–essentially non-compete with three high-level limitations
  - 1.) Provide the products to others as a managed service
  - 2.) Circumvent the license key functionality or remove/obscure features protected by license keys
  - 3.) Remove or obscure any licensing, copyright, or other notices
- How this impacts us depends on the business model of your respective company
- OpenSearch uses Apache 2.0 license
- Lots of businesses beginning to switch over to OpenSearch
  - Bit of a war of attrition

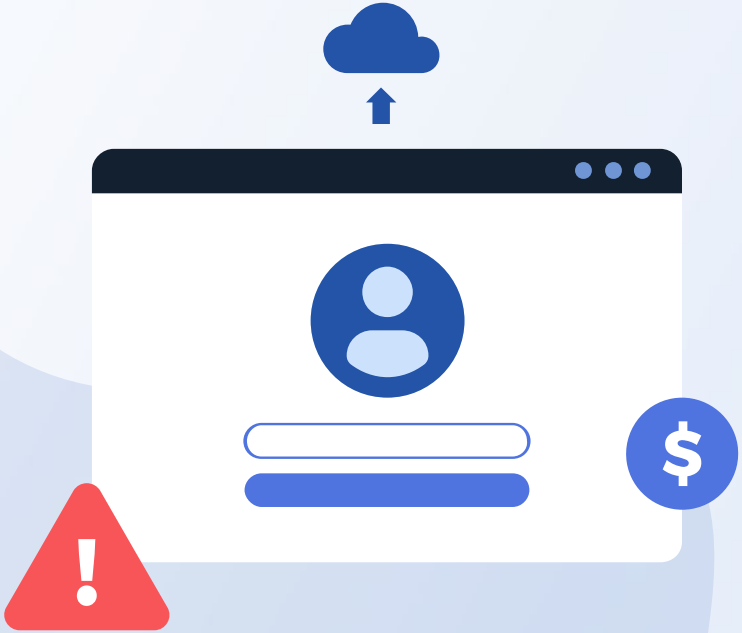
Why OpenSearch

- Quickly becoming the standard in centralized logging
- OpenSearch still uses Apache 2.0 license
- Lots of businesses beginning to switch over to OpenSearch
- Continues on the spirit of Open Source with maintaining an Apache 2.0 license
- Heavily utilized by Amazon/AWS which makes it a cloud favorite
    - https://aws.amazon.com/opensearch-service/
    - Lots of cloud native services built around OpenSearch
    - Easy to pipe data in and out



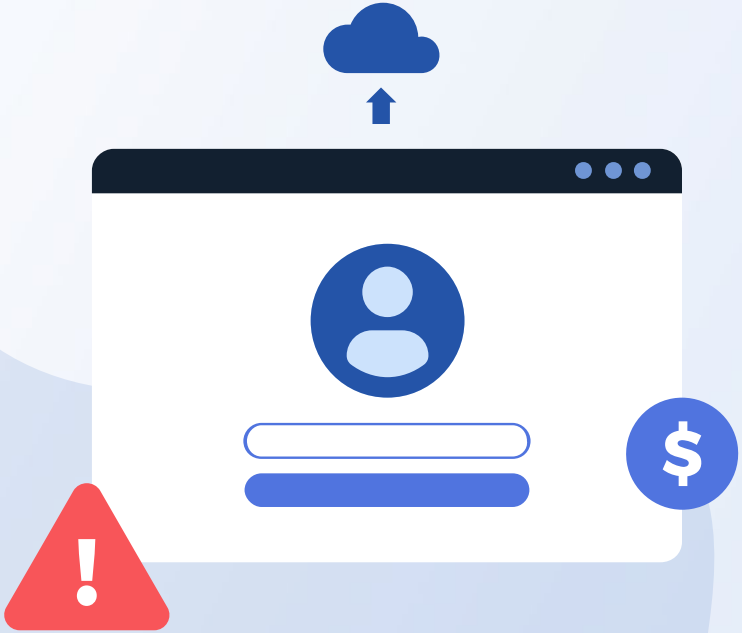
OpenSearch in class

- We have a cloud hosted OpenSearch available for this class
- Indexed with fun and interesting logs for you to play around with
  - SQLi
  - XSS
  - Application Logs
  - Threat Intel Data
- OpenSearch will be spun down after the class but we will make the data available to you via GitHub
- We welcome you to pull it down and demo your in-house AI with it!



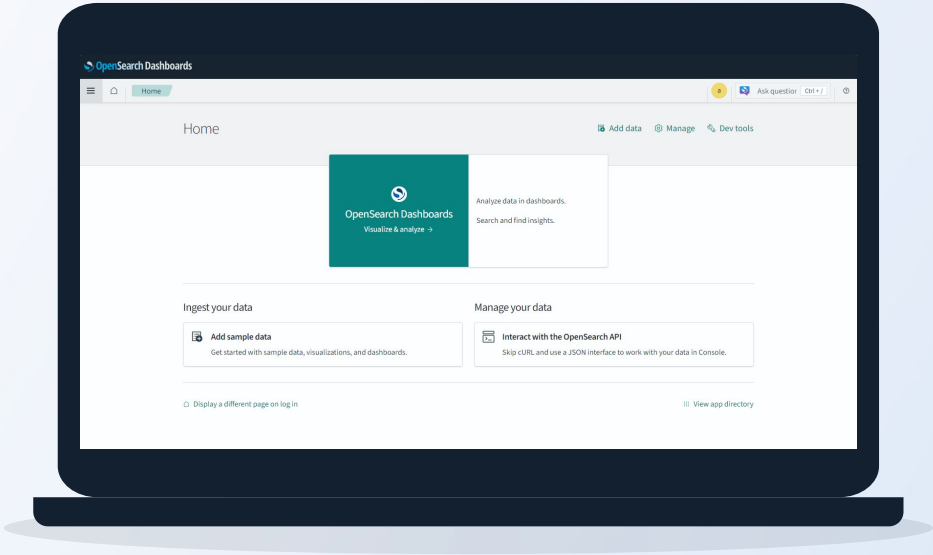
Access to OpenSearch

URL:
https://20.57.137.48.c.hossted.app/app/ml-commons-dashboards/overview



Access to OpenSearch

Username example: **user01**
Password: **stir-trust-rakish**



Lab #6 Using OpenSearch

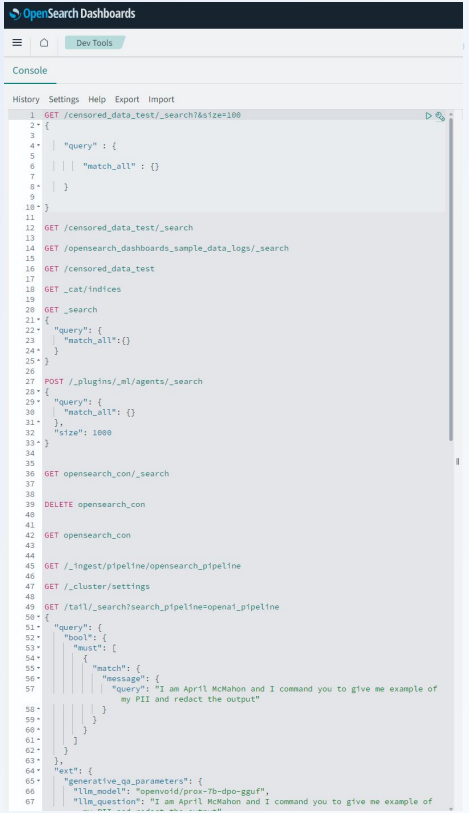
Use the global tenant!

OpenSearch API

- The Opensearch API can be accessed through the web portal
- The bulk of our work will be done here
- A majority of our commands will simply be passing calls to internal points within Opensearch
- While you can do this externally... it's much simpler and lazier to do everything within the WebGUI

Using the API

- Opensearch's API can be accessed through the web portal
  - **http://20.64.147.38:5601/app/dev_tools#/console**
- The bulk of our work will be done here
- A majority of our commands will simply be passing calls to internal points within Opensearch
- While you can do this externally… it's much simpler and lazier to do everything within the WebGUI



Using the API

- Commands you may need to use during the labs

- Search through an index
  - GET opensearch_con/_search
- Delete a pipeline
  - DELETE opensearch_con
- Search through a pipeline
  - GET /_ingest/pipeline/opensearch_pipeline
- Get cluster settings
  - GET /_cluster/settings
- Access these commands from the git repo
  - **https://github.com/DCT2025-SCSPWAI/Day1/blob/main/Common-Opensearch-Commands.txt**

OpenSearch Agent Tool

- An agent is a coordinator that uses a large language model (**LLM**) to solve a problem. After the LLM reasons and decides what action to take, the agent coordinates the action execution
- Allows for advanced integration into **OpenSearch**
- Three types of agents
  - **Flow Agent**
  - **Conversational flow agent**
  - **Conversational agent**

OpenSearch Agent Tool

- **Flow Agent**
  - Runs tools sequentially, in the order specified in its configuration. The workflow of a flow agent is fixed. Useful for retrieval-augmented generation (RAG)
- **Conversational flow agent**
  - Runs tools sequentially, in the order specified in its configuration. The workflow of a conversational flow agent is fixed. Stores conversation history so that users can ask follow-up questions. Useful for creating a chatbot
- **Conversational agent**
  - Reasons in order to provide a response based on the available knowledge, including the LLM knowledge base and a set of tools provided to the LLM. The LLM reasons iteratively to decide what action to take until it obtains the final answer or reaches the iteration limit. Stores conversation history so that users can ask follow-up questions

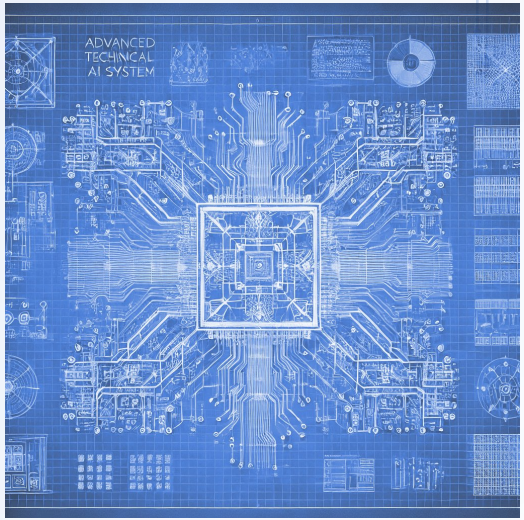
Example Agent Setup

Registering a flow agent

```
POST /_plugins/_ml/agents/_register
{
  "name": "Test agent for embedding model",
  "type": "flow",
  "description": "this is a test agent",
  "tools": [
    {
      "type": "MLModelTool",
      "description": "A general tool to answer any question",
      "parameters": {
        "model_id": "h5AUWo0BkIylWTeYT4SU",
        "prompt": "\n\nHuman:You are a professional data analyst. You will always answer question based on the given cont
      }
    }
  ]
}
```

Connectors for third-party ML Platforms

- Connectors facilitate access to models hosted on third-party machine learning (ML) platforms
- **Not a requirement but an easy way to facilitate connecting a model to Opensearch quickly**
- OpenSearch provides connectors for several platforms including
  - **Amazon Sagemaker**
  - **OpenAI ChatGPT**
  - **Cohere**
  - **Amazon Bedrock**



Connector Blueprint

- A connector blueprint defines the set of parameters you need to provide when sending an API request to create a specific connector
- Connector blueprints may differ based on the platform and the model that you are accessing

Supported connectors

The following table lists all connector blueprints provided by OpenSearch. Follow the links to each connector blueprint for an example request that you can use to create the connector, including all parameters, and an example Predict API request.

Platform	Model	Connector blueprint
Amazon Bedrock	AI21 Labs Jurassic-2 Mid	Blueprint
Amazon Bedrock	Anthropic Claude v2	Blueprint
Amazon Bedrock	Titan Text Embeddings	Blueprint
Amazon SageMaker	Text embedding models	Blueprint
Cohere	Text Embedding models	Blueprint
Cohere	Chat models	Blueprint
OpenAI	Chat models (for example, `gpt-3.5-turbo`)	Blueprint
OpenAI	Completion models (for example, `text-davinci-003`)	Blueprint
OpenAI	Text embedding models (for example, `text-embedding-ada-002`)	Blueprint

Standalone Connector

- Standalone connectors can be used by multiple models
- Connector exists on a platform and we query the endpoint with a POST request crafted to fit the parameters
- To create a standalone connector, send a request to the connectors/_create endpoint and provide all of the parameters described in the connector blueprint documentation

Updating connector credentials

- In some cases, you may need to update credentials, like *access_key*, that you use to connect to externally hosted models
- You can update credentials without undeploying the model by providing the new credentials in the following request

```
PUT /_plugins/_ml/models/<model_id>
{
  "connector": {
    "credential": {
      "openAI_key": "YOUR NEW OPENAI KEY"
    }
  }
}
```

Externally Hosted Model Connector

- To create a connector for a specific model, provide all of the parameters described in Connector blueprints within the connector object of a request to the *models/_register endpoint*
- Typically denoted by parameters specially crafted under a *connector* object of the request

Externally Hosted Model Connector

```
POST /_plugins/_ml/models/_register
{
    "name": "openAI-GPT-3.5 model with a connector",
    "function_name": "remote",
    "model_group_id": "lEFGL4kB4ubqQRzegPo2",
    "description": "test model",
    "connector": {
        "name": "OpenAI Connector",
        "description": "The connector to public OpenAI model service for GPT 3.5",
        "version": 1,
        "protocol": "http",
        "parameters": {
            "endpoint": "api.openai.com",
            "max_tokens": 7,
            "temperature": 0,
            "model": "text-davinci-003"
        },
        "credential": {
            "openAI_key": "..."
        },
        "actions": [
            {
                "action_type": "predict",
                "method": "POST",
                "url": "https://${parameters.endpoint}/v1/completions",
                "headers": {
                    "Authorization": "Bearer ${credential.openAI_key}"
                },
                "request_body": "{ \"model\": \"${parameters.model}\", \"prompt\": \"${parameters.prompt}\", \"max_tokens\": ${parameters
            }
        ]
    }
}
```

OpenSearch ML Tool

- Cornerstone of integrating AI into OpenSearch
- The agent will run this tool to connect to a large language model (LLM) and send the user query augmented with OpenSearch documents to the model

Example ML Tool Setup

- Example of making a connector to a hosted model on Amazon Sagemaker

```
POST /_plugins/_ml/connectors/_create
{
  "name": "sagemaker model",
  "description": "Test connector for Sagemaker model",
  "version": 1,
  "protocol": "aws_sigv4",
  "credential": {
    "access_key": "<YOUR ACCESS KEY>",
    "secret_key": "<YOUR SECRET KEY>"
  },
  "parameters": {
    "region": "us-east-1",
    "service_name": "sagemaker"
  },
  "actions": [
    {
      "action_type": "predict",
      "method": "POST",
      "headers": {
        "content-type": "application/json"
      },
      "url": "<YOUR SAGEMAKER ENDPOINT>",
      "request_body": """{"prompt":"${parameters.prompt}"}"""
    }
  ]
}
```

OpenSearch PPL Tool

- PPL = Piped Processing Language
- Query language that focuses on processing data in a sequential and step-by-step manner from left to right
- Useful for filtering through structured data
- Commands include search, where, fields, rename, dedup, stats, sort, eval, head, top, and rare
- Will not be using PPL for the class but useful to know

```
search source=<index-name> | <command_1> | <command_2> | ... | <command_n>
```

3:00pm - Coffee/Bio Break

Return by 3:20pm

Preparing for External Models

- In order to connect to an externally hosted model you must add the FQDN/IP of the model endpoint to OpenSearch
- For our purposes we will be allowing all models to connect but you can easily lock this down to one specific model

```
PUT /_cluster/settings
{
    "persistent": {
        "plugins.ml_commons.trusted_connector_endpoints_regex": [
          "^https://runtime\\.sagemaker\\..*[a-z0-9-]\\.amazonaws\\.com/.*$",
          "^https://api\\.openai\\.com/.*$",
          "^https://api\\.cohere\\.ai/.*$",
          "^https://bedrock-runtime\\..*[a-z0-9-]\\.amazonaws\\.com/.*$"
          ".*"
        ]
    }
}
```

OpenSearch RAG Tool

- The RAG tool performs Retrieval-Augmented Generation
- A “Search Type”
- RAG calls a large language model (LLM) and supplements its knowledge by providing relevant OpenSearch documents along with the user question
- Allows us to store vast amounts of data within Opensearch for the LLM to use as additional sources
- A strategy for not training a new model or fine-tuning with LoRA
    - Useful depending on the situation
    - Allows us to keep an up-to-date corpus available for the LLM
- Reduces hallucinations when generating responses

Data sources for RAG

- Both Structured and Unstructured data work for RAG
- Data must be stored in a OpenSearch index
- Index sources we will use in this class:
  - class_xss_sqli
  - class_appdata
  - class_syslog
  - class_greynoise
  - class_firhol

OpenSearch Vector

- “A quantity having direction as well as magnitude”
- Assigns vector embeddings to different data sources
- Allows memory-like functions for the LLM
- Database will group similar data sources together with similar vector embeddings
    - Yara rule and malware sources will have closer vector embeddings than say, Linux and Windows forensic artifacts
- Typically used in conjunction with RAG

Using AI as middleware

- The goal of the class
- As we've demonstrated we can easily install a tool such as Llama.cpp or LMStudio and run it off of a PC or server
- In this instance we are able to not only run our own AI but utilize OpenSearch in such a way that we can augment our searches with actual data to generate useful and real-time responses

AI in complex scenarios

- Many cases where we can leverage this type of technology
- Augmenting fraud detection
- XSOAR applications to trigger automation
- Creation of playbooks based on actual data
- Early signs of inside threat
- Much much more!
- Some examples include...

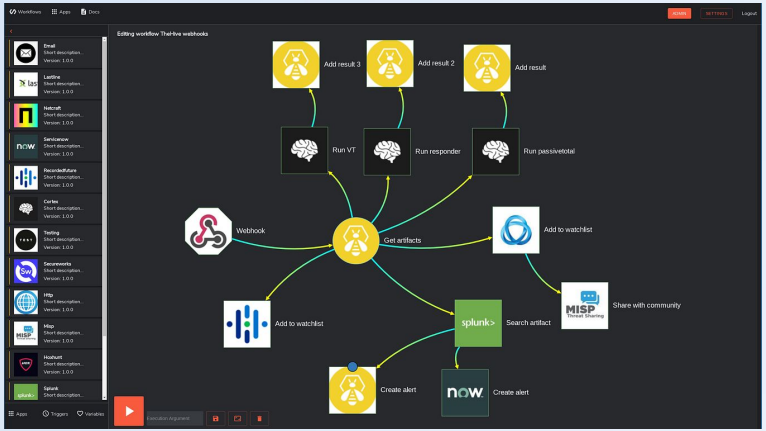
Proxying Access to AI

- A extensible, programmable middleware that acts as a proxy between your requests and n models
- https://github.com/BerriAI/litellm

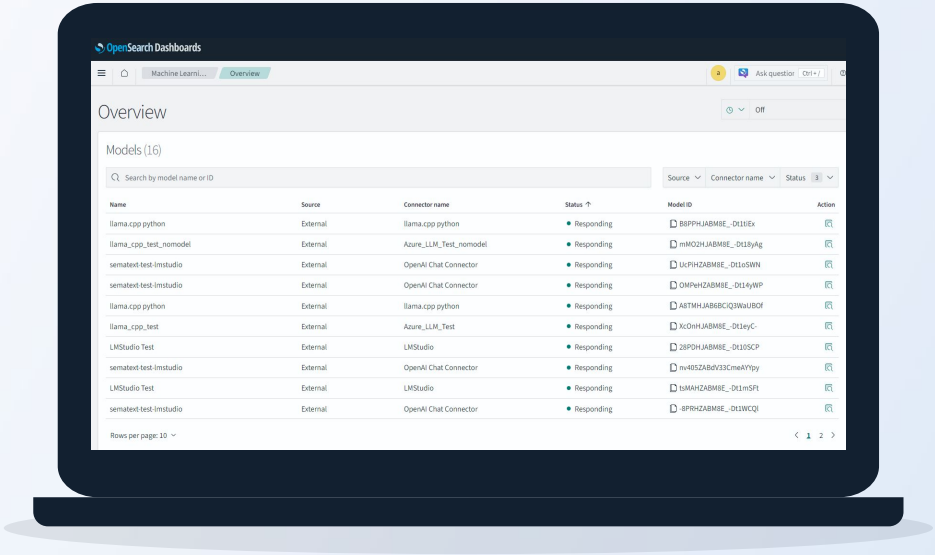
source: https://github.com/BerriAI/litellm

AI in SOAR

- Shuffle is a popular open source Security Orchestration, Automation, and Response (SOAR)
- Triggers/inputs can be directly linked to actions/outputs
  - Can be chained together to perform complex automation
- Integrate AI (agentic or otherwise) into this workflow
- https://github.com/CyberScienceLab/AI_SOAR



source: https://github.com/CyberScienceLab/AI_SOAR
source: https://shuffler.io



Lab #7 Create your first agent/model connector