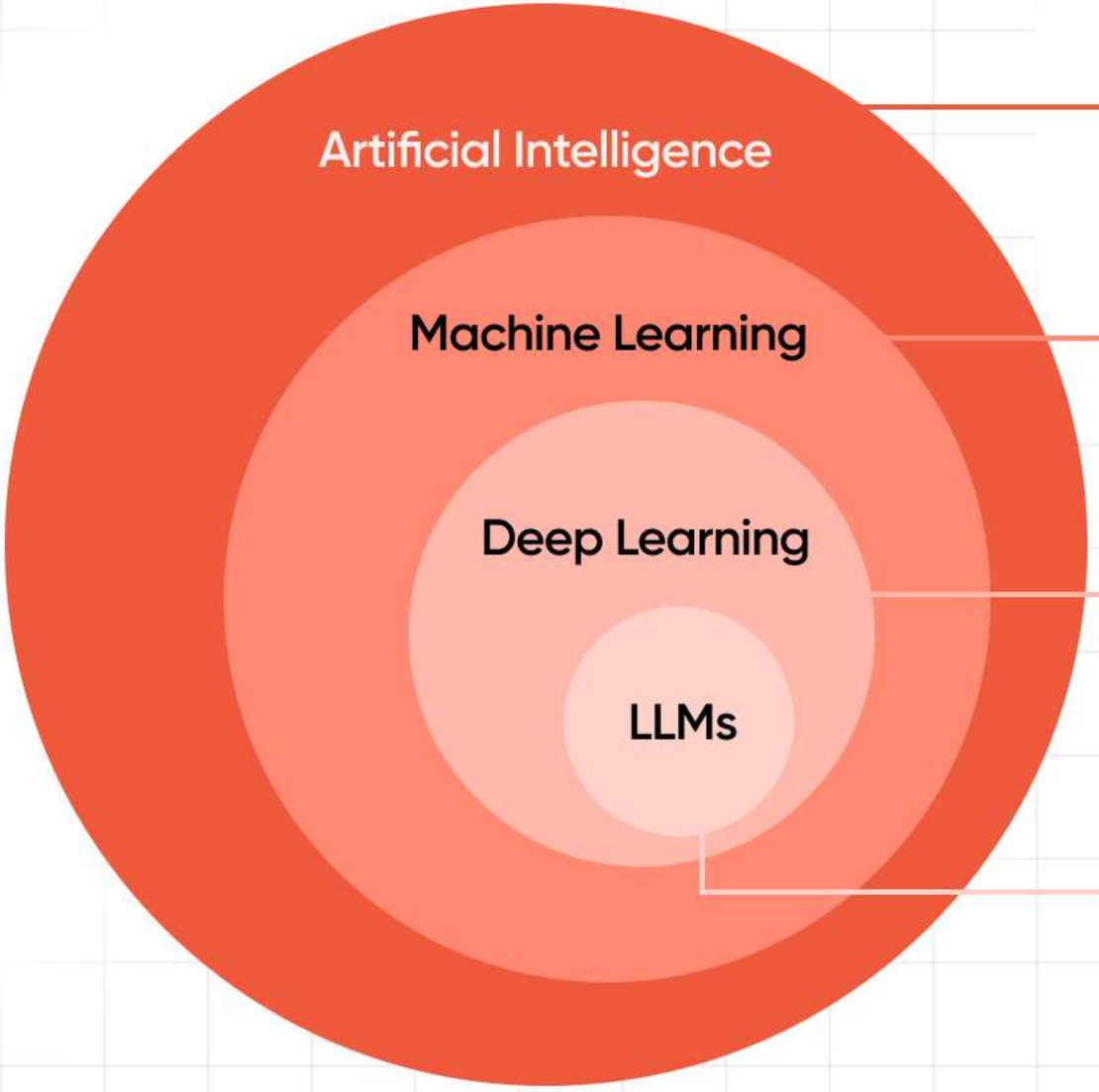
Understanding Large Language Models

100x

What are we going to cover?

1. What is AI? & where LLMs fit in the world of AI
2. Machine learning basics
3. Deep learning
4. What are LLMs
5. How do LLMs work

The AI Hierarchy



Intelligent Machines

Broadly Defined

Pattern Recognition

Learning general patterns from data

Neural Networks

Learning general patterns in **unstructured** data (i.e. images, text, audio, etc.)

Large Language Models

Learning to understand natural language (i.e. text)

100x

Machine Learning 101: Classification

100x

What is Machine Learning ?

Machine Learning is a subset of artificial intelligence (AI) that allows systems to learn and improve without being explicitly programmed.

What is the goal with Machine Learning ?

The goal of Machine Learning is to discover patterns in data.

Or more specifically, a pattern that describes the relationship between an input and an outcome.

Let's see it with a example

Let's take 2 music genres

Reggaeton

Reggaeton is a Latin urban genre known for its lively beats and danceable rhythms

Rhythm & Blues

R&B (Rhythm and Blues) is a genre rooted in African-American musical traditions, characterized by soulful vocals and a mix of upbeat and slower-paced songs.

Analyzing Music Genres

Imagine you have a collection of 20 songs, each characterized by two key metrics:

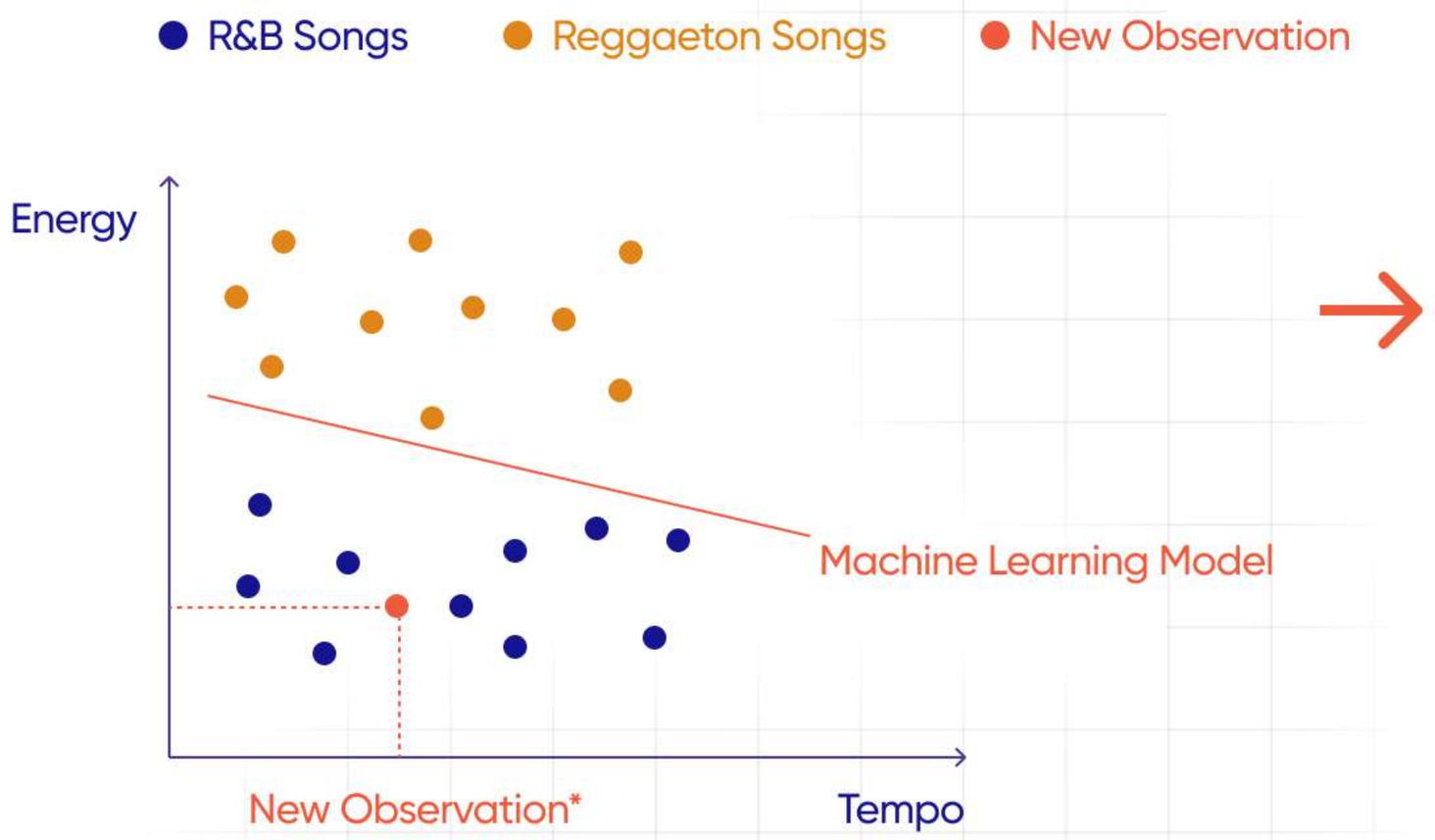
- **Tempo** (the speed of the song)
- **Energy** (the intensity or liveliness)

Additionally, each song is **labeled** with a genre—either Reggaeton or R&B.

- When we plot these songs on a graph with tempo on one axis and energy on the other, an interesting pattern emerges.
- Songs with high tempo and high energy tend to be Reggaeton, while those with lower tempo and lower energy are predominantly R&B.
- This observation aligns with our understanding of these genres, where Reggaeton is generally more upbeat and lively, and R&B is typically slower and more relaxed.

How it looks in practice

Classification Example: Predicting Music Genre



*New Observation:

Tempo: 30, Energy: 20

ML

Class	Probability
R&B	0.92
Reggaeton	0.08

Understanding Classification in Machine Learning

In Machine Learning, this scenario is an example of a **classification problem**. Here, the goal is to predict the **genre** of a song, which can only belong to one of a set number of categories—**Reggaeton** or **R&B**.
This differs from a **regression problem**, where the objective is to predict a continuous outcome, such as a **temperature or distance.**

In Machine Learning, we can "train" a model, often referred to as a **classifier**, using our labeled dataset—where we already know the genre of each song.
During training, the model learns to identify patterns in the data. Specifically, it finds the **boundary** or **line** that best separates the two genres, **Reggaeton** and **R&B**, based on their tempo and energy.

Why is this useful?

Once the model has learned this boundary, it can predict the genre of any new song. By simply measuring the tempo and energy of the song, the model can determine on which side of the boundary the song falls, and thus, predict whether it's a Reggaeton or R&B track. This approach is much more efficient and scalable than having a human manually label the genre of every song.

What happens when things get complicated ?

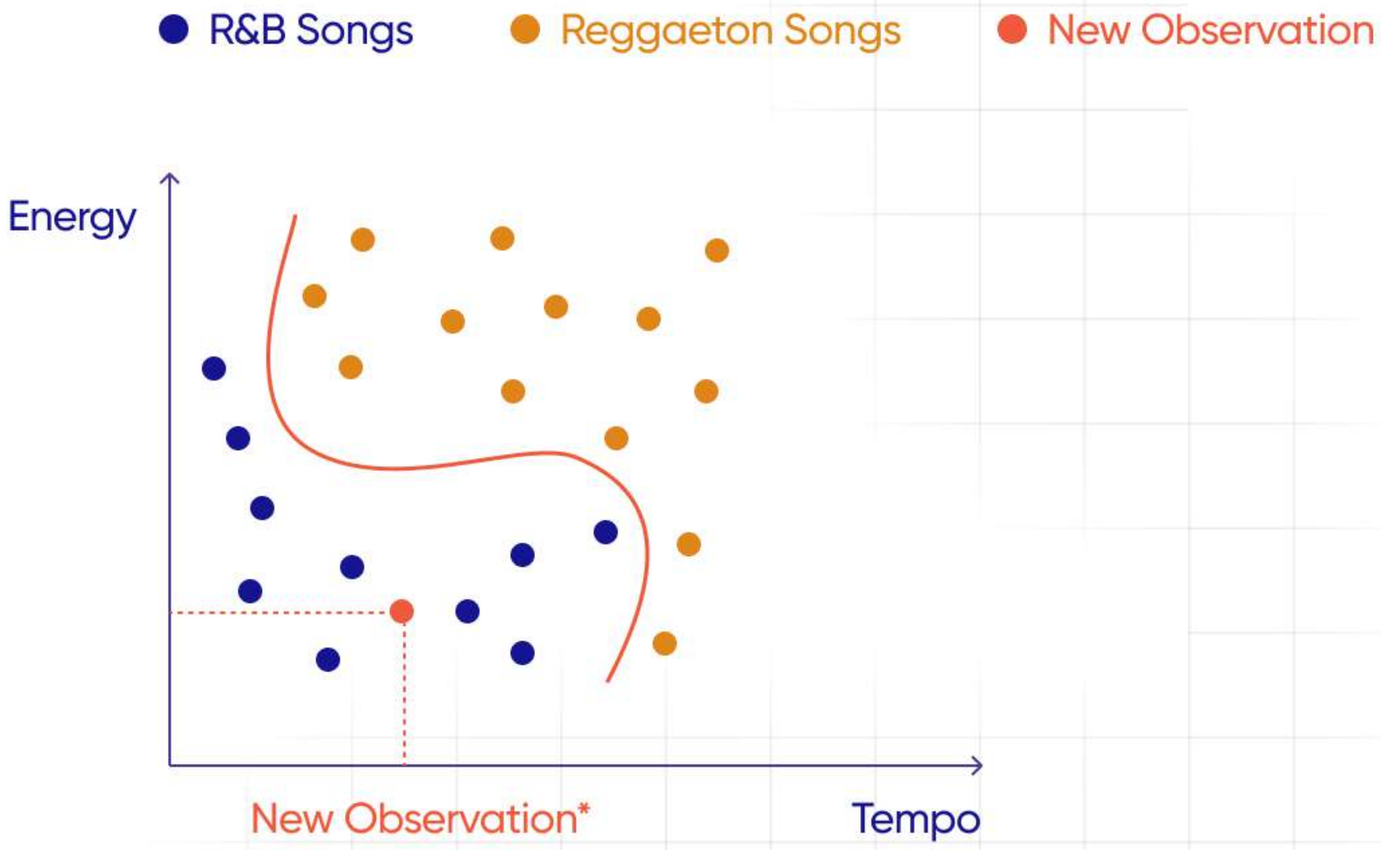
Real-world scenarios are often more complex than our simple example.

The boundary that separates different classes, such as genres, may not always be a straight line. The relationship between input features (like tempo and energy) and the outcome (the genre) can be more intricate. The boundary might be curved or even have a highly complex shape that isn't easy to visualize.

Complexity also increases when we move beyond just two input variables. In practice, we might have dozens, hundreds, or even thousands of features influencing the outcome. Similarly, we often deal with more than just two classes—there could be multiple genres to predict. The relationship between all these inputs and outputs is typically non-linear and very complex.

When Things Get Complicated

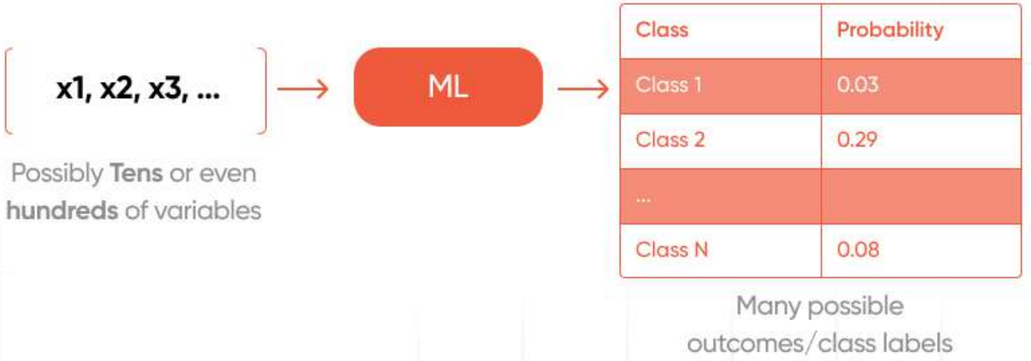
Classification Example: Non-linear relationships



Main Take-Away:

The more complicated the **input -> output relationship**, the more flexibility we need

Real World



Possibly **Tens** or even **hundreds** of variables

Many possible outcomes/class labels

100x

Even in our music example, there are more than two genres in reality, and more factors beyond tempo and energy play a role in determining the genre. The connections among these factors are likely far from straightforward.

The key takeaway here is that as the relationship between inputs and outputs becomes more complex, we require more advanced and powerful Machine Learning models to accurately learn and predict these relationships. Typically, as the number of inputs and classes increases, so does the complexity of the model needed.

What if the input is an Image?

Now, let's shift to a slightly different problem, but we'll apply the same mental model we've discussed. This time, our input is an **image**—for example, a cute picture of a cat inside a bag

For our outcome, we have three possible labels: **tiger**, **cat**, and **fox**.
Imagine we're trying to protect a herd of sheep and want to sound an alarm if we detect a tiger, but not if we see a cat or a fox.

As we learned earlier, this is another example of a **classification task** because the output can only belong to one of a few fixed categories. So, just like before, we could use labeled data (in this case, images that have already been categorized as tiger, cat, or fox) to train a Machine Learning model to make these classifications for us.

What if the input is an Image? From Pixels to Predictions

Challenge

High dimensionality

Classification

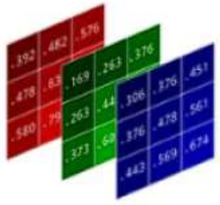
Is it a **tiger**, a **cat**, or a **fox?**

→ ML →

Class	Probability
Dog	0.03
Cat	**0.29**
Bird	0.01

=

=



= 224 x 224 x 3 = **150,528** Pixels (!!)

Complex relationship between raw pixels and image content

100x

Some problems we would face

However, in this new scenario, we encounter two significant challenges.

First, even a small, low-resolution image of 224x224 pixels contains over **150,000 individual pixels** (224x224x3, accounting for the three color channels: red, green, and blue). Previously, we were discussing models handling a few hundred or, at most, a thousand input variables. But now, we're dealing with at least **150,000 input variables**.

Second, consider the relationship between these raw pixels and the image's class label (tiger, fox, or cat). This relationship is incredibly complex from a Machine Learning perspective. While our human brains are naturally skilled at distinguishing between a tiger, fox, and cat, a Machine Learning model doesn't have this inherent ability.

What if the input is text?

We'll explore a more complex type of **input-output relationship**: the connection between a sentence and its sentiment. **Sentiment** refers to the underlying emotion or opinion expressed by a sentence, which we typically categorize as either positive or negative.

Let's break down the problem:

- **Input**: Our input is a sequence of words, which together form a sentence.
- **Output**: The output is the sentiment of that sentence, which can be one of two possible labels: **positive** or **negative**.

This task is a type of **classification problem**, where our goal is to correctly identify the sentiment of the sentence based on the words it contains.

In the previous example, we discussed how humans can easily interpret the relationship between images and their labels. But what about **sentences** and their **sentiments?**

- Can we train a Machine Learning model to understand and classify sentiment as we do?

Before diving into the answer, let's address a challenge: unlike images, where the input data is already in numeric form (pixels), words don't naturally convert to numbers. This adds a layer of complexity when working with textual data.

So, how do we transform words into something a Machine Learning model can understand?

The key lies in **word embeddings**. Without going into too much detail, you should know that word embeddings are a way to represent **words as numeric vectors** that capture their meaning—both in terms of **semantics** (what the word means) and **syntax** (how the word is used in a sentence). These embeddings can either be learned during the training of the Machine Learning model or pre-trained using separate algorithms.

Typically, word embeddings consist of several variables—ranging from tens to thousands—per word, effectively turning a **sentence into a sequence of numeric inputs**.

To sum up: What's important to take away here is that we can transform a sentence into a sequence of word embeddings, which contain rich semantic and syntactic information. These numeric representations can then be used as input for a Machine Learning model to predict sentiment.

What if the input is text? Understanding Sentiment

Challenge

Converting text to numbers

Class	Probability
Positive	**0.98**
Negative	0.02

Word embeddings: Numerical representations of words

Complex relationship between words and meaning

100x

Deep Learning

100x

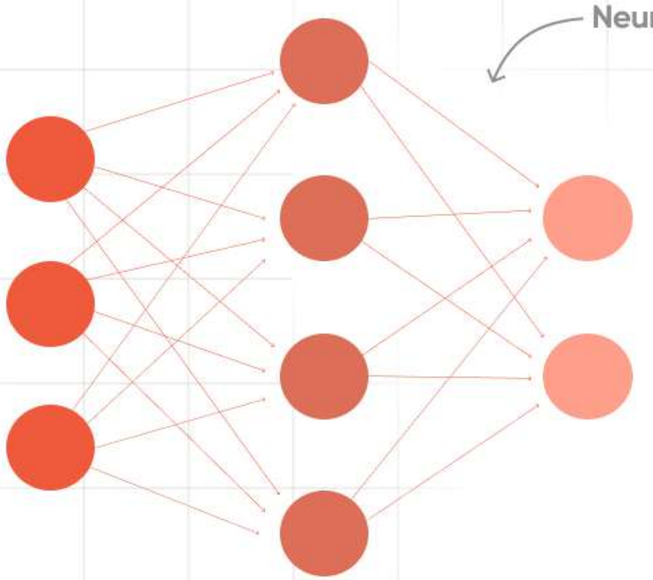
When do we use Deep Learning?

We've discussed that when the relationship between input and output is **highly complex**, or when the number of input or output variables is **large** (as in the case of our image and language examples), we require models that are not only more flexible but also more **powerful**. A simple linear model, or anything similar, won't be sufficient to solve complex tasks like visual recognition or sentiment classification.

This is where **neural networks**—the foundation of Deep Learning—come into play.

Enter Deep Learning

Input
(Image or Text)



Class	Probability
Dog	0.03
Cat	**0.29**
Bird	0.01

Output

Neural networks: Inspired by the human brain

Deep learning: Neural networks with many layers

Can model highly complex, non-linear relationships

Neural Networks: The Power Behind Deep Learning

Neural networks are some of the most powerful Machine Learning models available today, capable of modeling extremely complex relationships. They are the driving force behind the ability to learn and predict at a massive scale.

Although neural networks are loosely inspired by the **human brain**, the actual similarities are often overstated. The basic architecture of a neural network is straightforward: it consists of a **series of connected layers**, known as "**neurons**," that process an input signal to predict an outcome.

You can imagine neural networks as multiple layers of linear regression models stacked together, but with one crucial difference: non-linear functions are applied between the layers. This addition of non-linearities allows the network to capture and model highly complex, non-linear relationships.

Large Language Models: The Big Picture

Key Points

LLMs: Neural networks with billions of parameters

Examples: GPT-3 (175B), GPT-4 (1T+), BERT, T5

1. Trained on massive amounts of text data
2. Can generate human-like text
3. Perform various language tasks

What does LLM mean ?

"**Large**" refers to the size of the model, specifically the number of neurons, or parameters, in the neural network. These parameters are the building blocks that enable the model to learn from data. While there's no strict threshold, models with over 1 billion parameters are typically considered "**large**."

Now that we know what makes a model large, let's focus on the second part: "**Language Model.**"

A language model is designed to tackle a fundamental task: predicting the next word in a sequence of words, such as in a sentence or paragraph. Let's break this down as a Machine Learning problem.

Imagine we want to teach a model to predict the next word in any given sentence. This task is similar to the sentiment classification problem we discussed earlier. The input to the neural network is a sequence of words, and the output is the predicted next word.

Just like in sentiment classification, this is a classification task. However, instead of choosing between just two or a few categories (like positive or negative sentiment), the model now has to choose from thousands of possible words—let's say around 50,000. This is the essence of language modeling: learning to predict the next word in a sequence.

Language Modeling

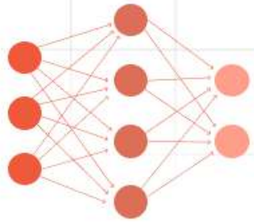
The Core Task: Predict the next word in a sequence

The cat likes to sleep in the _____ → What **word** comes next ?

Can we frame this as a ML Problem ? Yes, it's a classification task

The cat likes to sleep in the

Input



Neural Network (LLM)

Word	Probability
ability	0.002
bag	0.071
box	**0.085**
...	...
zebra	0.001

Output

Feeding the Beast: LLM Training Data

Massive datasets: Hundreds of billions of words
Diverse sources: Web pages, books, articles, code, etc.

Example of creating training samples:
"The quick brown fox" → "The quick brown [MASK]"

Large Language Model: Data and Self-Supervised Learning

The internet is filled with vast amounts of text, ranging from books and research papers to tweets and code snippets. We can compile a massive dataset from these sources without the need for manual labeling. Why? Because in this task, the next word in each sentence acts as the label itself. This approach is known as **self-supervised learning**.

To summarize: Our goal is to train a neural network—specifically, a Large Language Model (LLM)—to predict the next word in any given sequence. It doesn't matter whether the sequence is short or long, written in German, English, or any other language, or if it's a poem, a mathematical formula, or even a snippet of code. All of these are just different sequences found in our training data.

With a sufficiently large neural network and enough data, the LLM becomes highly proficient at predicting the next word.

Will it be perfect? **No**, because language is often **ambiguous**, and **multiple words** could logically follow any given sequence. However, the model will excel at choosing one of the words that are both **syntactically** and **semantically** appropriate.

Massive Training Data

We can create vast amounts of sequences for training a language model

● Context ● Next Word ● Ignored

The cat likes to sleep in the

The cat likes to sleep in the

The cat likes to sleep in the

The cat likes to sleep in the

The cat likes to sleep in the

We do the same with much **longer sequences.** For example:

A language model is a probability distribution over sequence of words. [...] Given any sequence of words, the model predicts the next ....

or also with **code:**

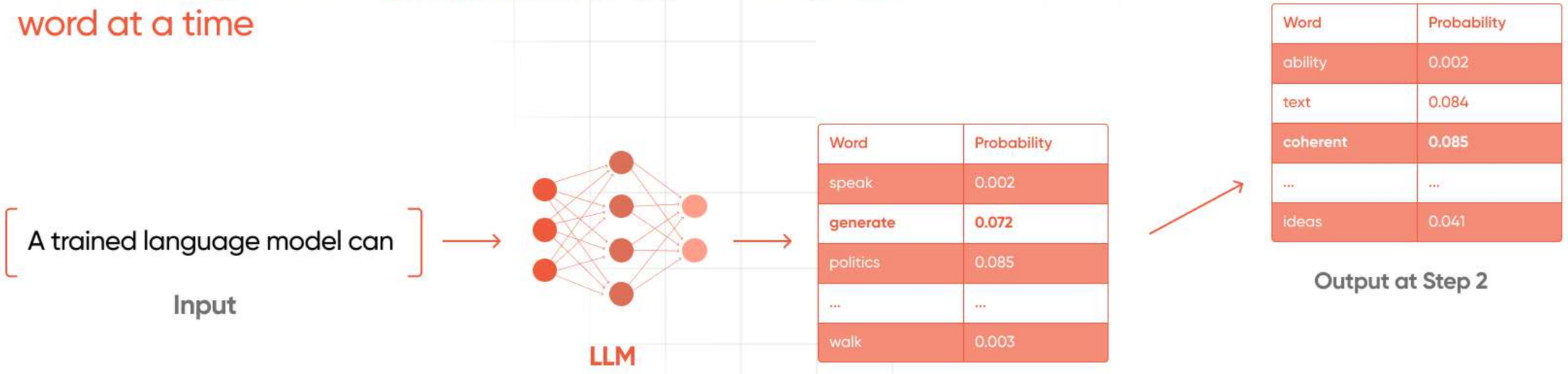
```
def square(number):
    """Calculates the square of a number"""
    return number**2
```

And as a result - the model becomes incredibly good at predicting the next word in any sequence

100x

Natural language generation

After training: We can generate text by predicting one word at a time



LLMs are an example of what's called **"Generative AI"**

Generation: Repeatedly predicting the next word

Sampling strategies for creativity and diversity

What does Generative Pre-trained Transfomer (GPT) mean?

Generative

Means **"next word prediction."**

As just described.

Pre-trained

The LLM is pretrained on massive amounts of text from the internet and other sources.

See next slide.

Transfomer

The neural network architecture used (introduced in 2017).

Building an LLM: A Three-Step Process

1. Pre-training

Massive amounts of data from the internet + books + etc.

Question: What is the problem with that?

Answer: We get a model that can babble on about anything, but it's probably not aligned with what we want it to do.

2. Instruction Fine-tuning

Teaching the model to respond to instructions.

Model learns to respond to instructions.

→ Helps alignment

3. Reinforcement Learning from Human Feedback

Similar purpose to instruction tuning.

Helps produce output that is closer to what humans want or like.

"Alignment" is a hugely important research topic

LLM Capabilities

Ability	Explanation
Why can an LLM perform **Text Summarization**?	Ability probably learned during **pre-training**
Why can an LLM perform **Question Answering?**	**Knowledge** acquired in pre-training, responds nicely due to fine-tuning
Why does a LLM sometimes answer wrong or even make stuff up?	Let's discuss this next...
why can an LLM perform summarization of a longer piece of text?	**Knowledge and ability** acquired in pre-training, responds nicely due to fine-tuning, Training data - research papers etc

Truthfulness

Problem	Idea	Solution
LLMs are trained to generate human-like text, **not true text**. Nothing indicates truthfulness to LLMs.	We need to **"ground"** them in reality, so that they don't make stuff up. In fact, we know everything to solve this.	Include the relevant knowledge in the **context** of the LLM.

Ask Me Anything: Question Answering

100x

Bingchat - example of a search-based LLM workflow

Who is Colombia's current president?

Bing performs a ***web search***

The **top search results** are extracted

Results are **stitched into the prompt** and sent to LLM

Prompt

Instruction: Answer the following question given some context.

Question: **Who is Colombia's current president?**

A web search found the following relevant context:
Wikipedia: Colombia is a country in South [...]
NYT: The recent election in Colombia [...]

Answer:

LLM

The current president in Colombia is Gustavo Francisco Petro Urrego.

Emerging LLM Abilities

Zero-shot learning, Few-shot learning, Chain-of-thought reasoning

Zero-Shot Prompting

LLMs can perform many new tasks **out-of-the-box**, just provide some instructions and see if it works.

See example on the right.

User

Translate from German to English but use only words that start with an "f".

German: Die Katze schläft gerne in der Box.
English:

LLM

Feline friend finds fluffy fortress.

Few-Shot Learning

Providing **examples** helps the LLM understand and follow your task.

This is especially helpful to ensure a specific **output format.**

User

Standardize different currency formats.
Examples:

- Sandwich EUR 8.9 -> €8.90
- 14.50 pounds Burger-> £14.00
- 4$ for a Milkshake -> $4.00
- Toppings are 99 cents each -> €0.99
- Steak: 24.99 USD ->

LLM

$24.99

Chain-of-Thought Prompting

Ask the model to solve complex tasks **step by step**.

Why does this work?
It gives the model a **working memory**, similar to humans.

User

Who won the World Cup in the year before Lionel Messi was born? Think step by step.

LLM

Lionel Messi was born on June 24, 1987. The World Cup that took place before his birth was the 1986 World Cup. The winner of the 1986 FIFA World Cup was Argentina.

Art of prompting

Elements of a Prompt

A prompt is composed with the following components:

- Instruction
- Context
- Input Data
- Output Indicator

```
Classify the text into neutral,
negative or positive

Text: I think the food was okay.

Sentiment:
```

Prompt Guide

Promptguide: https://www.promptingguide.ai/introduction/settings

Techniques

Prompting Techniques

Prompt Engineering helps to effectively design and improve prompts to get better results on different tasks with LLMs.

While the previous basic examples were fun, in this section we cover more advanced prompting engineering techniques that allow us to achieve more complex tasks and improve reliability and performance of LLMs.

- Zero-shot Prompting
- Few-shot Prompting
- Chain-of-Thought Prompting
- Meta Prompting
- Self-Consistency
- Generate Knowledge Prompting
- Prompt Chaining
- Tree of Thoughts
- Retrieval Augmented Generation
- Automatic Reasoning and Tool-use
- Automatic Prompt Engineer
- Active-Prompt
- Directional Stimulus Prompting
- Program-Aided Language Models
- ReAct
- Reflexion
- Multimodal CoT
- Graph Prompting

100x

Bonus resource: Emergent Abilities of LLMs

Research paper: https://arxiv.org/pdf/2206.07682

Blog: https://www.quantamagazine.org/the-unpredictable-abilities-emerging-from-large-ai-models-20230316/

ARTIFICIAL INTELLIGENCE

The Unpredictable Abilities Emerging From Large AI Models

59 |

Large language models like ChatGPT are now big enough that they've started to display startling, unpredictable behaviors.

100x