OpenAI

Discovering AI & Deep Learning

Hosted by

Global AI Community

EY
Building a better working world

Info Pills

Context understanding & Contacts

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Antimo Musone

Senior Manager & Digital Innovator of **EY Italy**

Microsoft **MVP** AI

Intel Innovator Program

Co-Founder of **Fifth Ingenium** & **ItaliaDotNet** Community

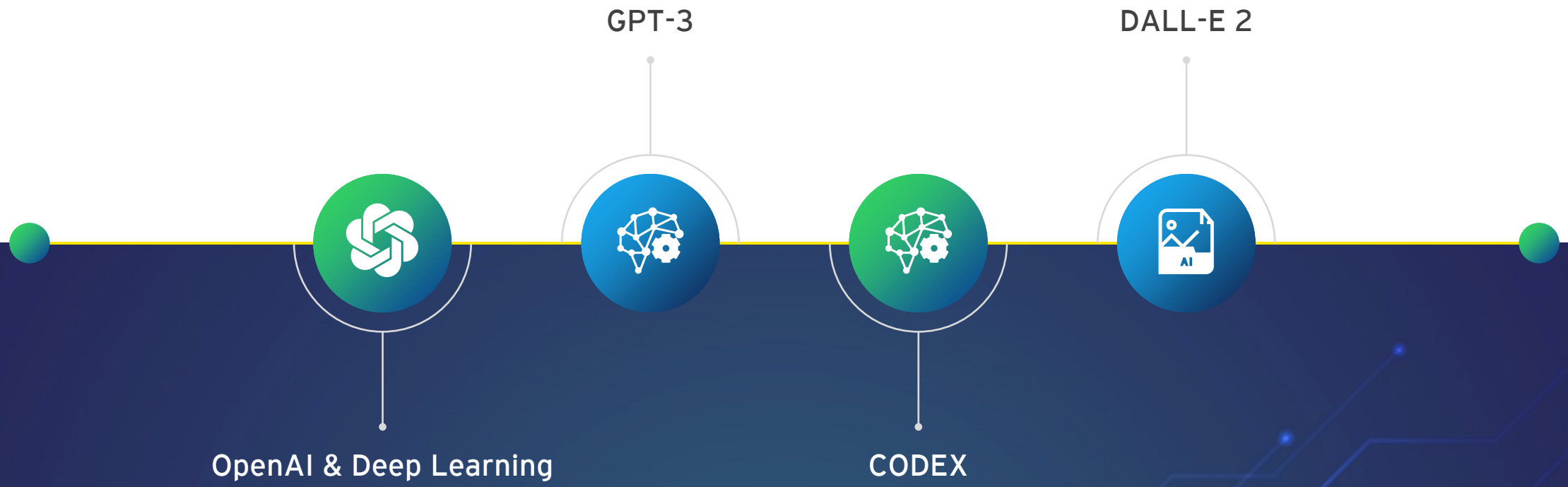
https://www.linkedin.com/in/antimo-musone/

antimo.musone@it.ey.com

https://twitter.com/AntimoMusone

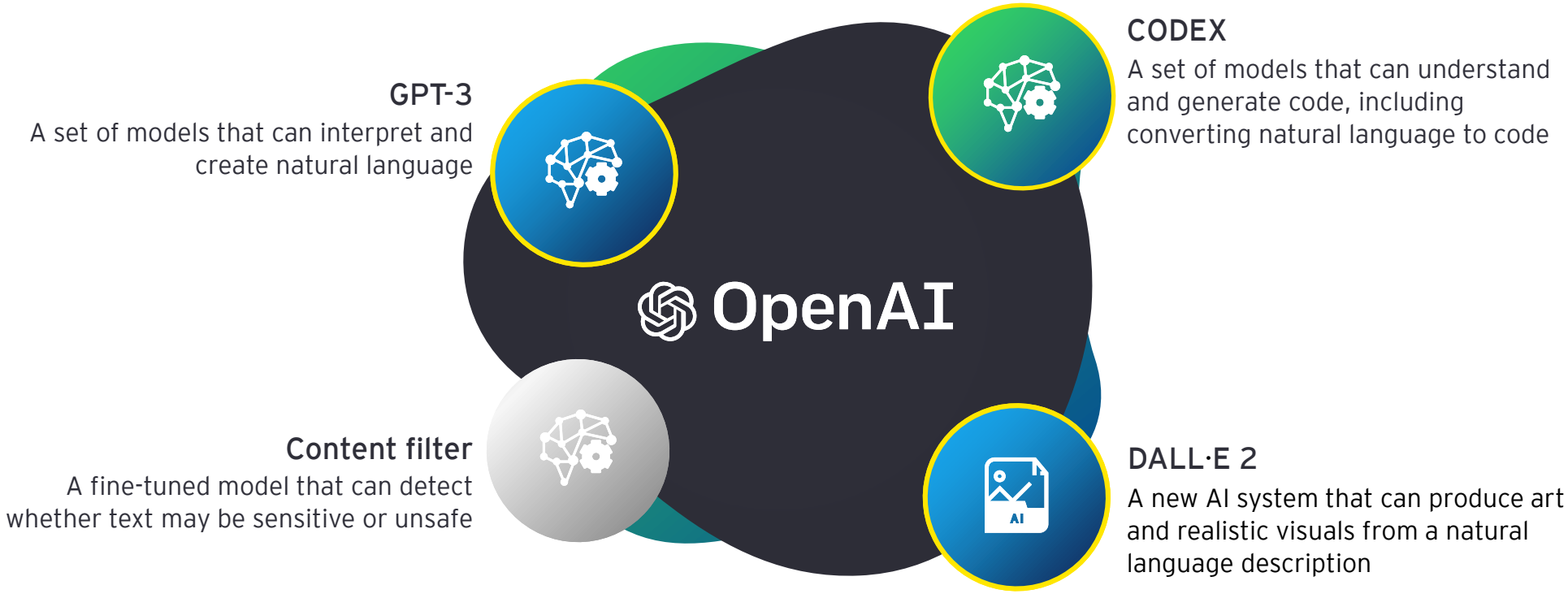
+39 345 3636642

Agenda
Main Contents



OpenAI & Deep Learning
Overview

Almost **any activity** that requires understanding or producing natural language or code can be made using the **OpenAI API**. OpenAI provides a range of models with varying degrees of power appropriate for various jobs as well as the option to fine-tune your own unique models. **Everything from content creation to semantic search and classification may be done using these models**



GPT-3
Offering Overview

GPT-3 models can comprehend and produce natural language. There are **four primary types with varying degrees** of **power appropriate** for **various purposes. The most capable model is Davinci, while the fastest is Ada**

LATEST MODEL	DESCRIPTION	MAX REQUEST	TRAINING DATA
`text-davinci-002`	Most capable GPT-3 model. Can do any task the other models can do, often with less context. In addition to responding to prompts, also supports inserting completions within text.	4,000 tokens	Up to Jun 2021
`text-curie-001`	Very capable, but faster and lower cost than Davinci.	2,048 tokens	Up to Oct 2019
`text-babbage-001`	Capable of straightforward tasks, very fast, and lower cost.	2,048 tokens	Up to Oct 2019
`text-ada-001`	Capable of very simple tasks, usually the fastest model in the GPT-3 series, and lowest cost.	2,048 tokens	Up to Oct 2019

Fonte: https://beta.openai.com/docs/models/gpt-3

GPT-3
Revolutionizing AI

OpenAI GPT-3 provides a general-purpose **"text in, text out" interface**, which **makes it possible to apply it to virtually any language task.** This is different from most other ML Models, which are designed for a single task

To use the GPT-3, you simply give it a **text prompt** and it will return a **text completion**, attempting to match the **context** or **pattern** you gave it. You can **"program"** it by crafting a description or writing just a few examples of what you'd like it to do.

GPT-3 is a "Language Model"

GPT-3 is a **2048-gram model**:

- Assigns probabilities to word sequences: $P(s) = P(w_1, \ldots, w_m)$
- Can factor as the product: $P(s) = \prod_{i=1\ldots m} P(w_i | w_1, \ldots, w_{i-1})$
- E.g. (n-1)-gram model: $P_{ngram}(s) = \prod_{i=1\ldots m} P(w_i | \mathbf{h}_i) = \prod_{i=1\ldots m} P(w_i | w_{i-n}, \ldots, w_{i-1})$

GPT-3 Training Data

- Trained on **499 Billion tokens**
- Would require 355 years and $4,600,000 train on cheapest GPU Cloud

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Key concepts
Prompt, Completion & Tokens

There are **three concepts** that are core to the API GPT-3: **prompt**, **completion**, and **tokens**. The "**prompt**" is text input to the API, and the "**completion**" is the text that the API generates based on the prompt

```
As Descartes said, I think therefore
```

```
As Descartes said, I think therefore I am
```

The API turns **text** into "**tokens**" before processing it, as a trick to be able to handle more text at once. The text prompt and generated completion must be below **2048 tokens** (roughly ~1500 words)

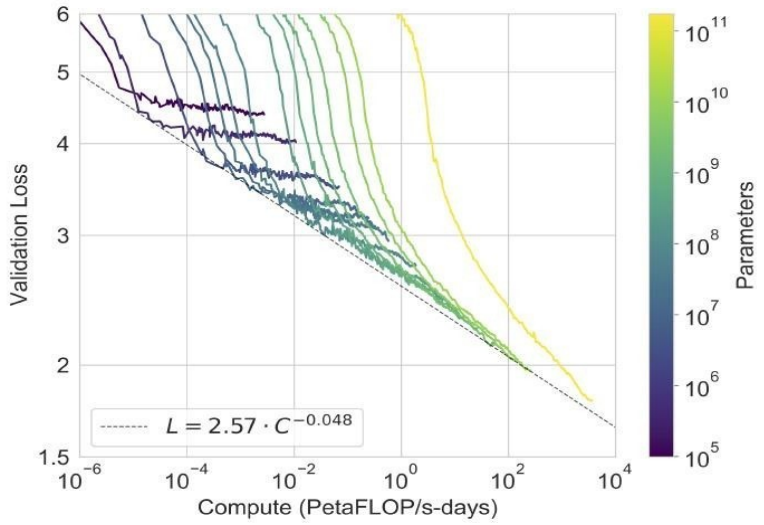
GPT-3 175B Model Architecture

- Input reversibly encoded as tokens with "Byte Pair Encoding" of UTF-8 bytes
- Context window of 2,048 tokens
- 96 transformer layers
- 96 self-attention heads, each 128 dimensional
- 12,288 units in bottleneck layer, 49,152 in feed forward layer
- Batch size of 3.2M samples
- Learning rate .6*10^-4

Microsoft-built AI Supercomputer

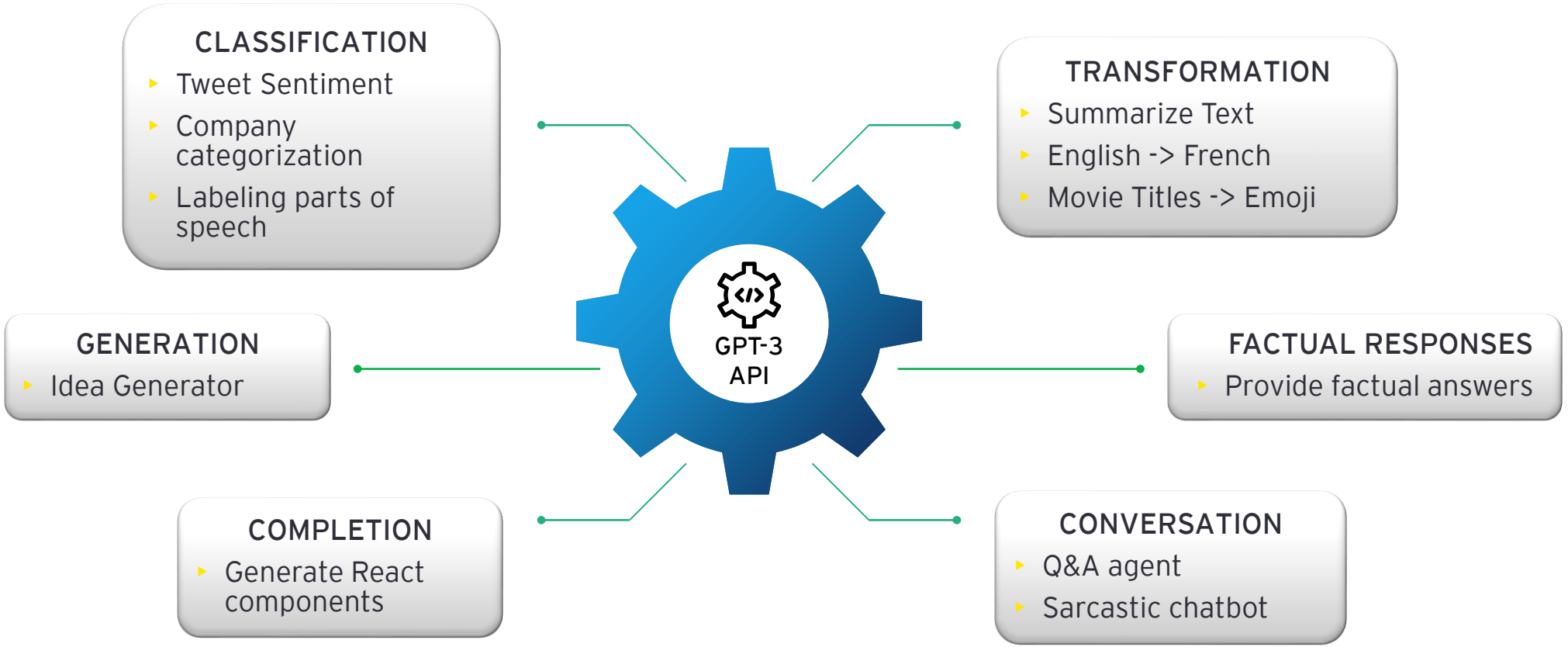
- NVIDIA V100 GPUs in a high-bandwidth cluster
- 285,000 CPU cores
- 10,000 GPUs
- 400 gigabits per second network connectivity for each GPU server
- Trained on cuDNN accelerated PyTorch models

GPT Validation Loss vs. Compute



Key concepts
Prompt Design

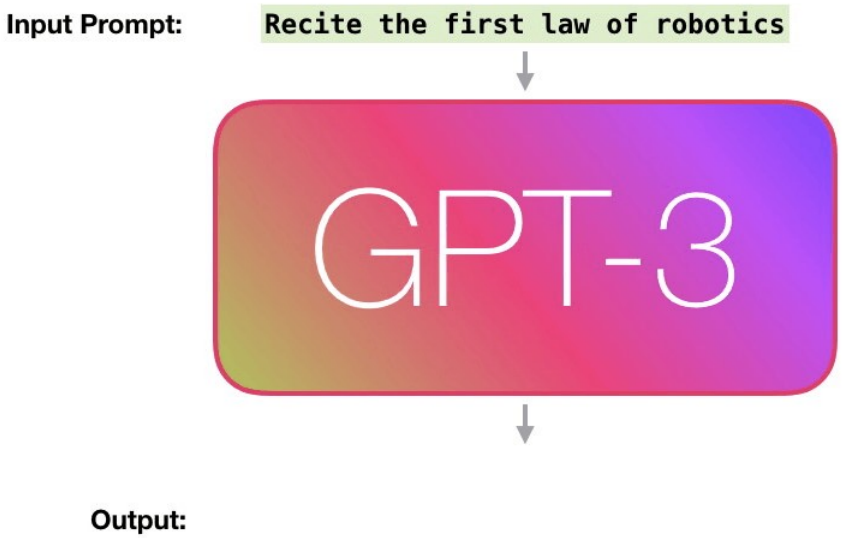
The **GPT-3 API** can execute a variety of different tasks. A **well-written prompt** provides enough information for the API to know what you are asking for and how it should respond

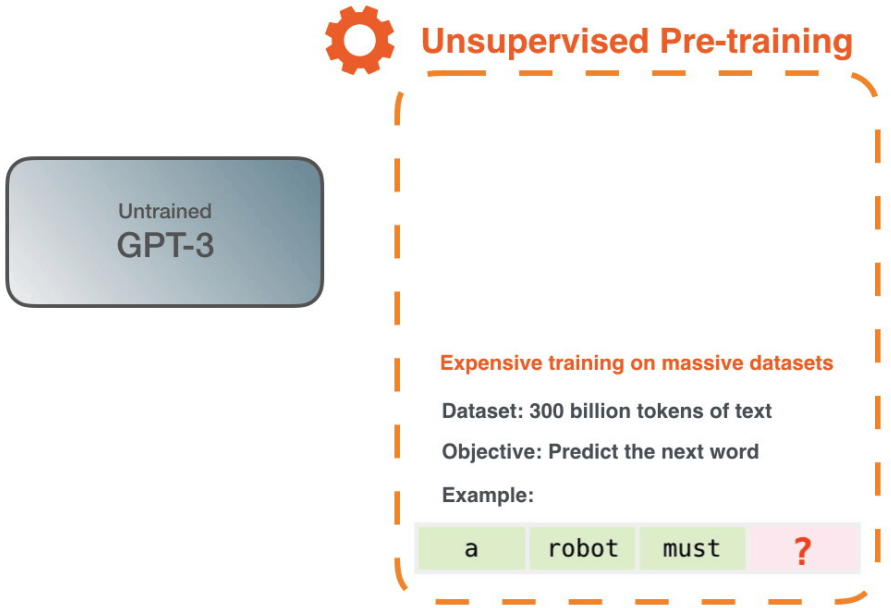


How GPT-3 Works
1/7

A **The output is generated from what the model "learned" during its training period** where it scanned vast amounts of text.

B **Training is the process of exposing the model to lots of text.** That process has been completed. All the experiments you see now are from that one trained model. It was estimated to cost 355 GPU years and cost $4.6m.





How GPT-3 Works
2/7

The dataset of 300 billion tokens of text is used to generate training examples for the model. For instance, these are three training examples generated from the one sentence at the top. You can see **how you can slide a window across all the text and make lots of examples**

Text: Second Law of Robotics: A robot must obey the orders given it by human beings

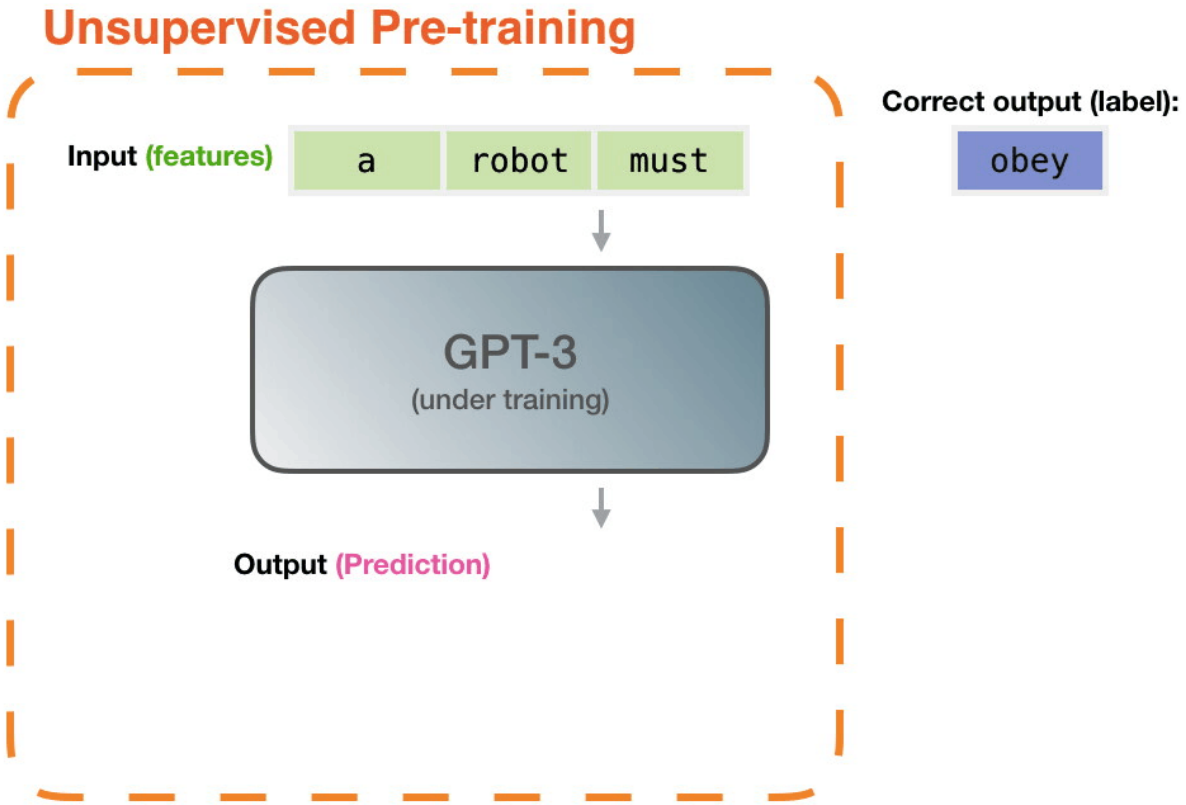
Generated training examples

Example #	Input (features)	Correct output (labels)
1	Second law of robotics :	a
2	Second law of robotics : a	robot
3	Second law of robotics : a robot	must
...		

How GPT-3 Works
3/7

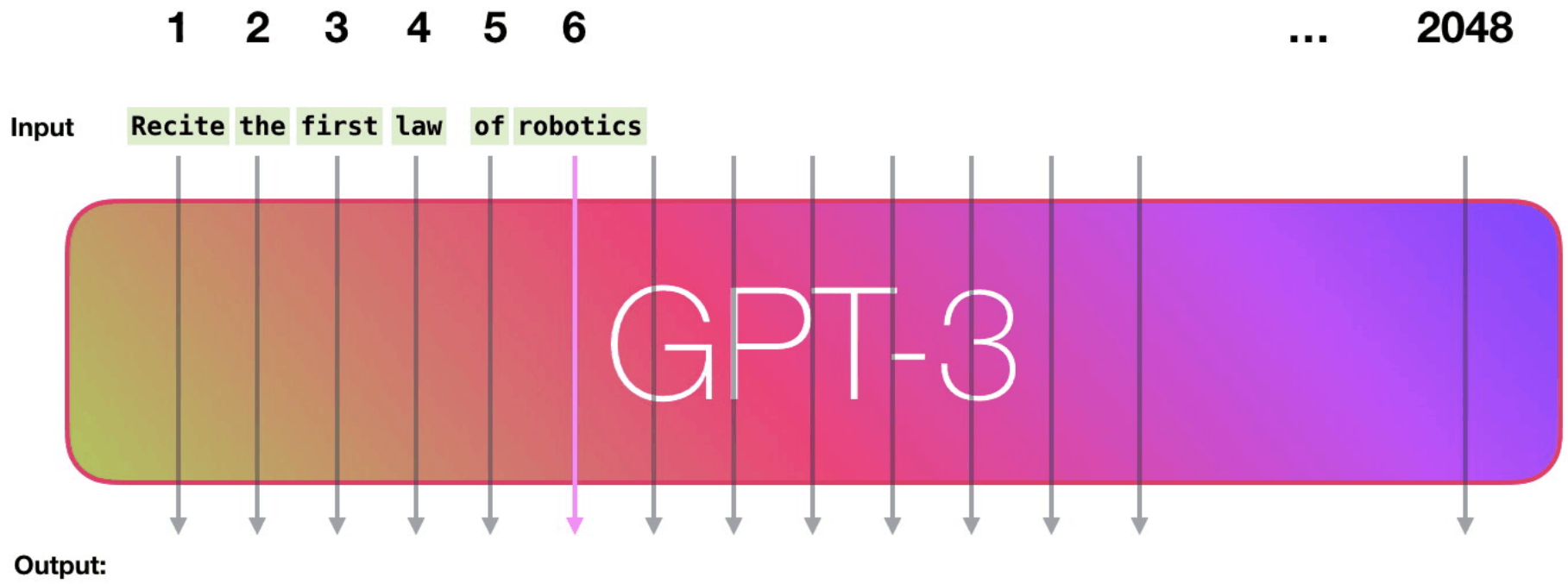
D

The model is presented with an example. We **only** show it the **features** and **ask** it to predict the **next word**. The model's prediction will be wrong. We **calculate** the **error in its prediction** and **update** the **model** so **next time** it makes a **better prediction**



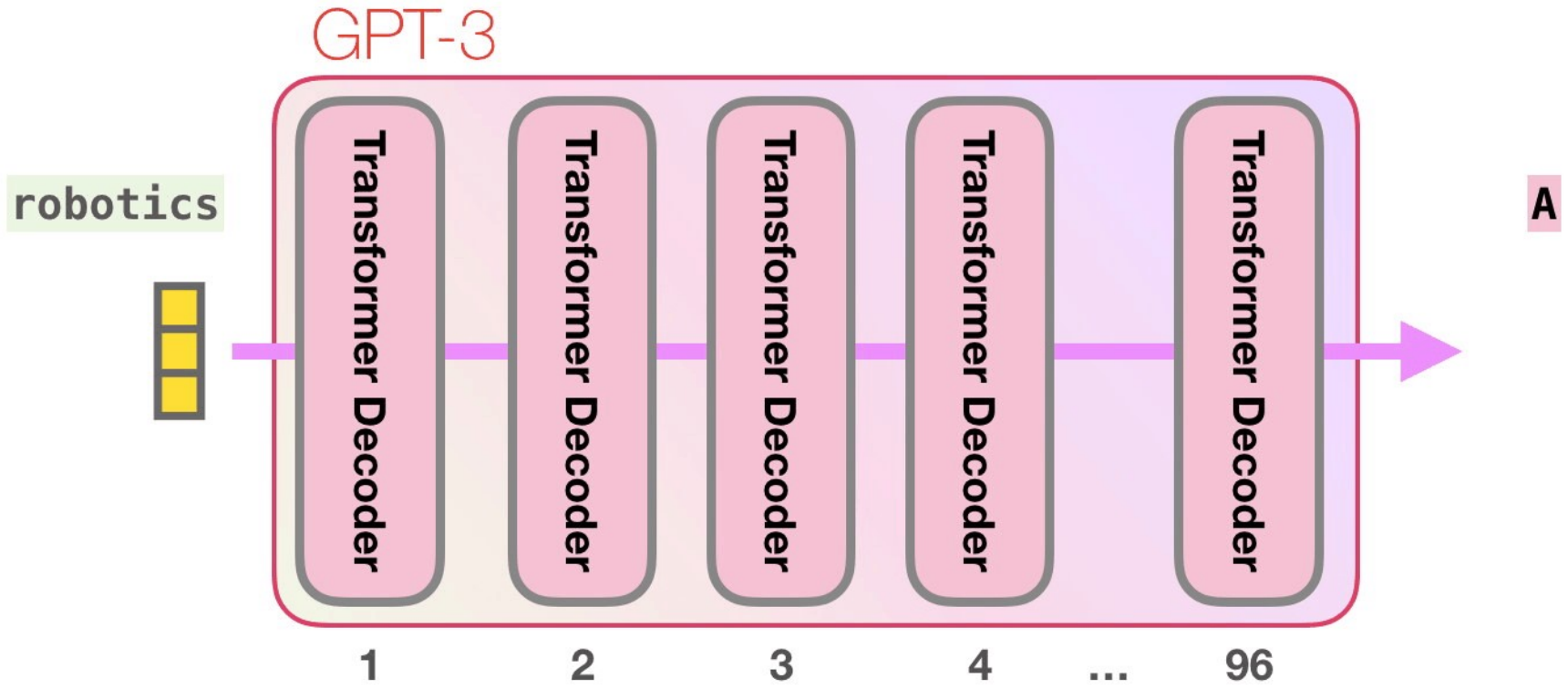
How GPT-3 Works
4/7

GPT-3 is 2048 tokens wide. That is its "**context window**". That means it has **2048 tracks** along which tokens are processed



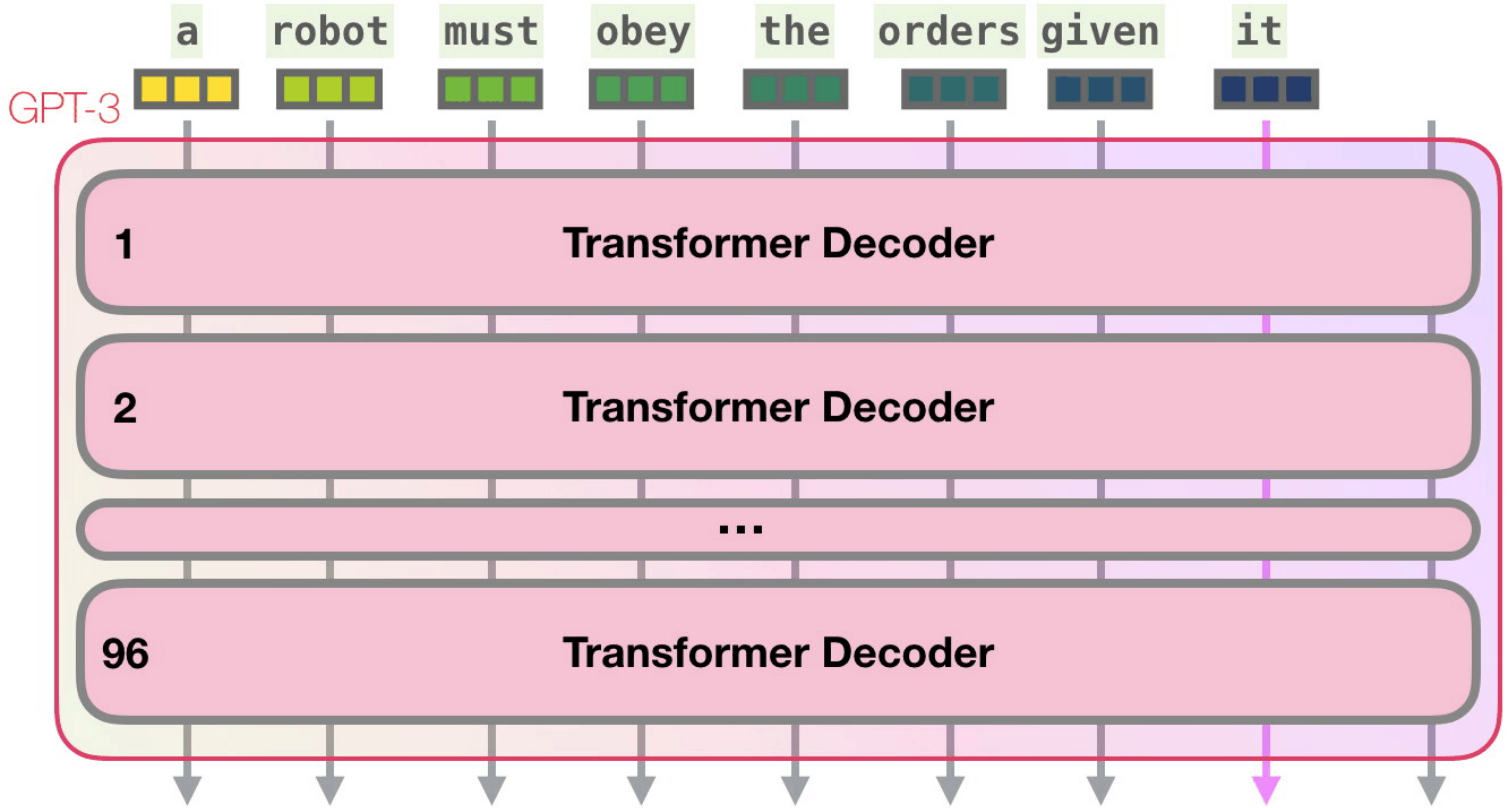
How GPT-3 Works
5/7

The **important calculations** of the **GPT-3** occur inside **its stack of 96 transformer decoder layers**. Each of these layers has its own **1.8B parameter** to make its calculations. This is a high-level view of that process:



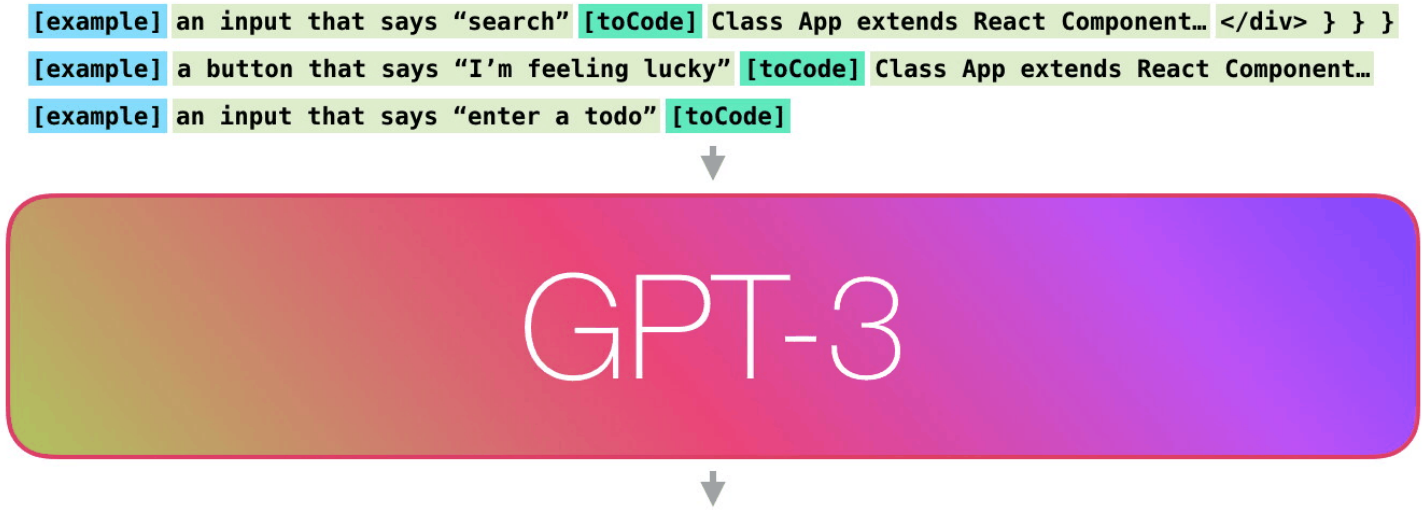
How GPT-3 Works
6/7

Notice how **every token flows through the entire layer stack**. We don't care about the output of the first words. When the **input** is **done**, we **start caring** about the **output**. We feed every word back into the model



How GPT-3 Works
7/7

In the **React code generation example**, the **description** would be the **input prompt in green**, in addition to a couple of examples of description → code, I believe. And the react code would be generated like the **pink tokens** here token after token. The **assumption** is that the **priming examples** and the **description** are appended as **input**, with specific tokens separating examples and the results. Then fed into the model



Tasks
DEMO 1/4

1 CLASSIFICATION

To create a **text classifier with** the **API** we provide a **description** of the task and a **few examples**.

```
This is a tweet sentiment classifier
Tweet: "I loved the new Batman movie!"
Sentiment: Positive
###
Tweet: "I hate it when my phone battery dies."
Sentiment: Negative
###
Tweet: "My day has been 👍"
Sentiment: Positive
###
Tweet: "This is the link to the article"
Sentiment: Neutral
###
Tweet: "This new music video blew my mind"
Sentiment:
```

- State what the prompt does at the art
- Use plain language to describe your inputs and outputs
- Use separators to define your examples
- Show the API how to respond to any case
- You can use text and emoji
- You need fewer examples for familiar tasks

2 GENERATION

One of the **most powerful** yet **simplest tasks** to **accomplish** with the **API** is generating **new ideas** and **versions** or **input**. You can give the API a **list of story ideas** and it will **add** to that **list** from just a **few examples**. It can **create business plans**, **character descriptions** and **marketing slogans** just by providing it a handful of examples.

```
Ideas involving education and virtual reality

1. Virtual Mars

Students get to explore Mars via virtual reality and go on missions to collect and catalog
what they see.

2.
```

Tasks
DEMO 2/4

3 **CONVERSATION**

The API is extremely **adept** at **carrying on conversations with humans** and even with itself. With just a few lines of instruction, the **API can perform** as a **customer service chatbot** that intelligently answers questions without ever getting flustered or a wise-cracking conversation partner that **makes jokes** and **puns**. The **key** is to **tell** the API **how it should behave and then provide a few examples**

```
The following is a conversation with an AI assistant. The assistant is helpful, creative,
clever, and very friendly.
Human: Hello, who are you?
AI: I am an AI created by OpenAI. How can I help you today?
Human:
```

4 **COMPLETION**

While all prompts are forms of completion it can be **helpful** to **think of text completion as its own task in instances where you want the API to pick up where you left off**. Examples of this include helping you write a summary or writing lines of code where the API can infer what should come next

```
''''
import React from 'react';
const ThreeButtonComponent=()=>(
<div>
<p>Button One<p>
<button className="button-green" onClick={this.handleButtonClick}>Button One</button>
<p>Button Two</p>
<button className="button-green" onClick={this.handleButtonClick}>Button Two</button>
<p>Button Three</p>
<button className="button-green" onClick={this.handleButtonClick}>Button Three</button>
</div>
)
''''
import React from 'react';
const HeaderComponent=()=>(
```

Tasks
DEMO 3/4

5 TRANSFORMATION

The **API** is a **language model** that is **familiar with a variety of ways** that **words** and **characters** can be used to **express information**. This ranges from natural language text to computer code and languages other than English. The **API** is also able to **understand** the **content** on a level that allows it to **summarize**, **convert** and **express** it in **different ways**

TRANSLATION

```
English: I do not speak French.
French: Je ne parle pas français.
English: See you later!
French: À tout à l'heure!
English: Where is a good restaurant?
French: Où est un bon restaurant?
English: What rooms do you have available?
French: Quelles chambres avez-vous de disponible?
English:
```

CONVERSION

```
Back to Future: 👴👱🚗🕒
Batman: 👱🦇
Transformers: 🚗🤖
Wonder Woman: 👩👧👩🏾👩🏾👩🏾
Spider-Man: 🕸🕷🕸🕸🕷🕸
Winnie the Pooh: 🐻🐼🐻
The Godfather: 👨👩👧👩‍🦰🧑💥
Game of Thrones: 🏹🗡🗡🏹
Spider-Man:
```

SUMMARIZATION

```
My ten-year-old asked me what this passage means:
"""
A neutron star is the collapsed core of a massive supergiant star, which had a total mass
of between 10 and 25 solar masses, possibly more if the star was especially metal-rich.[1]
Neutron stars are the smallest and densest stellar objects, excluding black holes and
hypothetical white holes, quark stars, and strange stars.[2] Neutron stars have a radius
on the order of 10 kilometres (6.2 mi) and a mass of about 1.4 solar masses.[3] They
result from the supernova explosion of a massive star, combined with gravitational
collapse, that compresses the core past white dwarf star density to that of atomic nuclei.
"""
I rephrased it for him, in plain language a ten-year-old can understand:
"""
```

Tasks
DEMO 4/4

6 FACTUAL RESPONSES

The API has a lot of knowledge that it's learned from the data that it was been trained on. It also can provide responses that sound very real but are in fact made up. There are **two ways** to **limit** the **likelihood** of the **API making up an answer**

```
Q: Who is Batman?
A: Batman is a fictional comic book character.
###
Q: What is torsalplexity?
A: ?
###
Q: What is Devz9?
A: ?
###
Q: Who is George Lucas?
A: George Lucas is American film director and producer famous for creating Star Wars.
###
Q: What is the capital of California?
A: Sacramento.
###
Q: What orbits the Earth?
A: The Moon.
###
Q: Who is Fred Rickerson?
A: ?
###
```

01 Provide a ground truth for the API

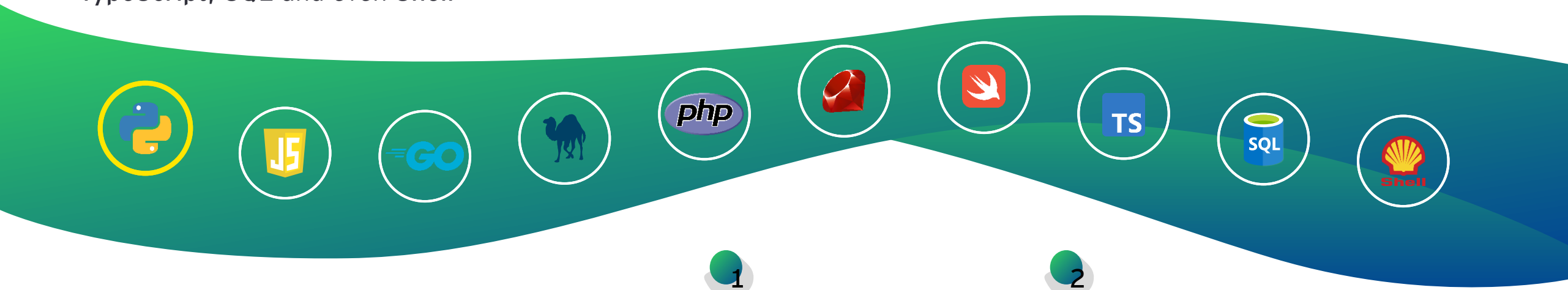
If you provide the API with a body of text to answer questions about (like a Wikipedia entry) it will be less likely to confabulate a response

02 Use a low probability and show the API how to say "I don't know"

If the API understands that in cases where it's less certain about a response that saying "I don't know" or some variation is appropriate, it will be less inclined to make up answers

CODEX
Offering Overview

The **Codex models** are **descendants** of our **GPT-3 models** that **can understand** and **generate code.** Their **training data contains** both **natural language** and billions of lines of **public code** from **GitHub.**
The language they **excel** in is Python, but they also know how to use **JavaScript, Go, Perl, PHP, Ruby, Swift, TypeScript, SQL** and even **Shell**



1

code-davinci-002
Most capable Codex model. Good at translating natural language to code

+ completing code
+ inserting completions within code

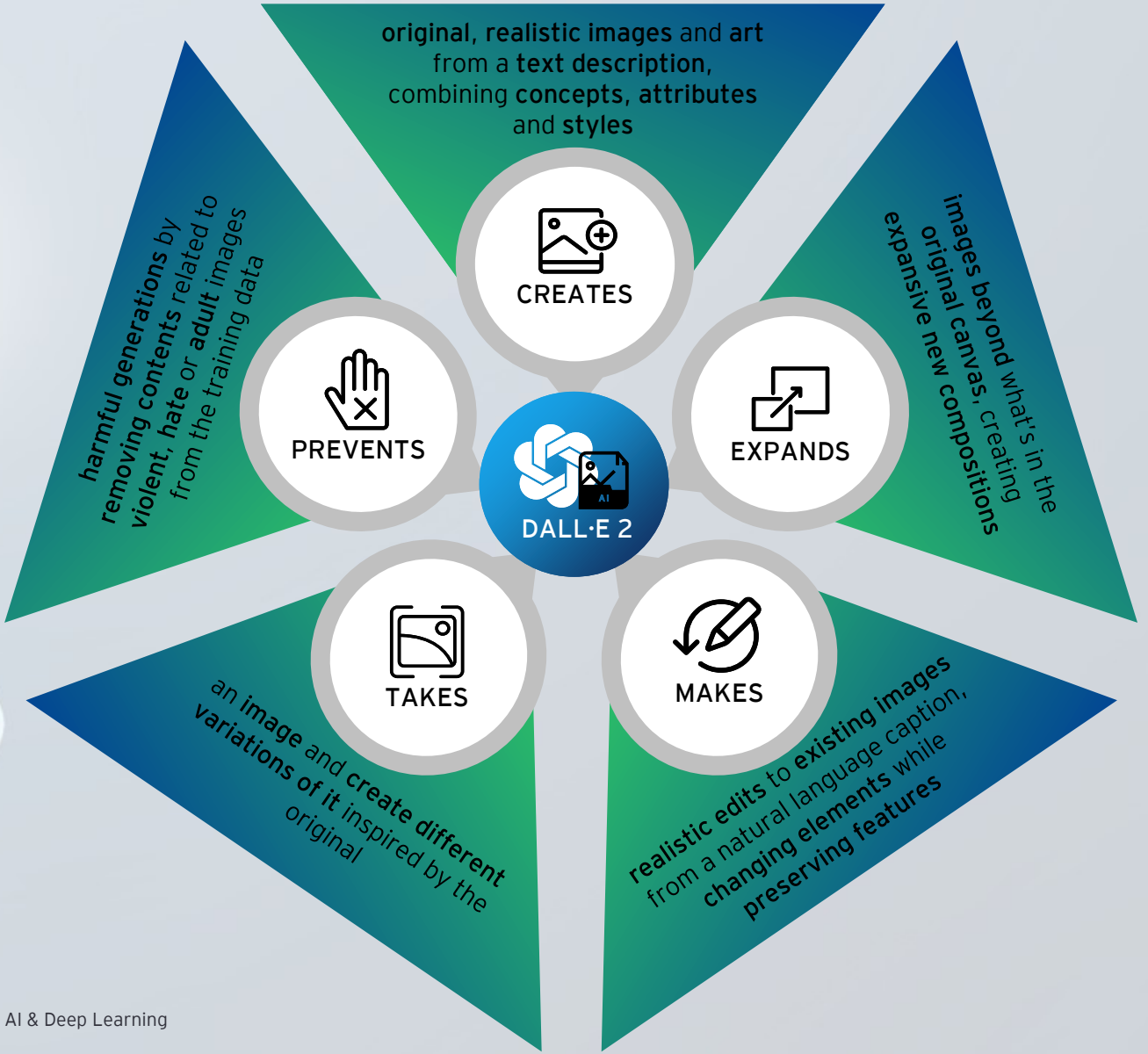
Max Request
8,000 tokens

2

code-cushman-001
Faster than Davinci Codex, preferable for real-time applications

Max Request
Up to 2,048 tokens

DALL·E 2
Offering Overview 1/2



DALL·E 2
Offering Overview 2/2

DALL-E 2 sets **a new bar for image generation and manipulation.** With only short text prompt, DALL-E 2 **can generate completely new images that combine distinct and unrelated objects in semantically plausible ways**, like the images below which were generated by entering the prompt "**a bowl of soup that is a portal to another dimension as digital art**"

Fonte: https://www.assemblyai.com/blog/how-dall-e-2-actually-works/

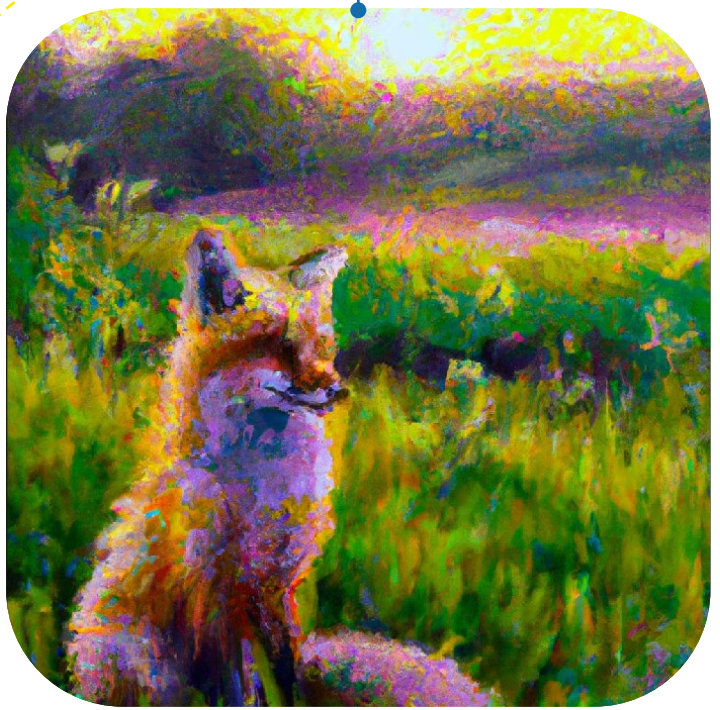
DALL·E 2
DALL·E 1 vs DALL·E 2

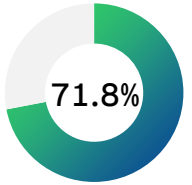
DALL·E 2 is preferred over DALL·E 1 for its caption matching and photorealism when evaluators were asked to compare 1,000 image generations from each model

DALL·E
January 2021

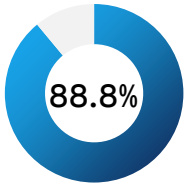
DALL·E 2
April 2022

TIMELINE





PREFFERRED FOR CAPTION MATCHING



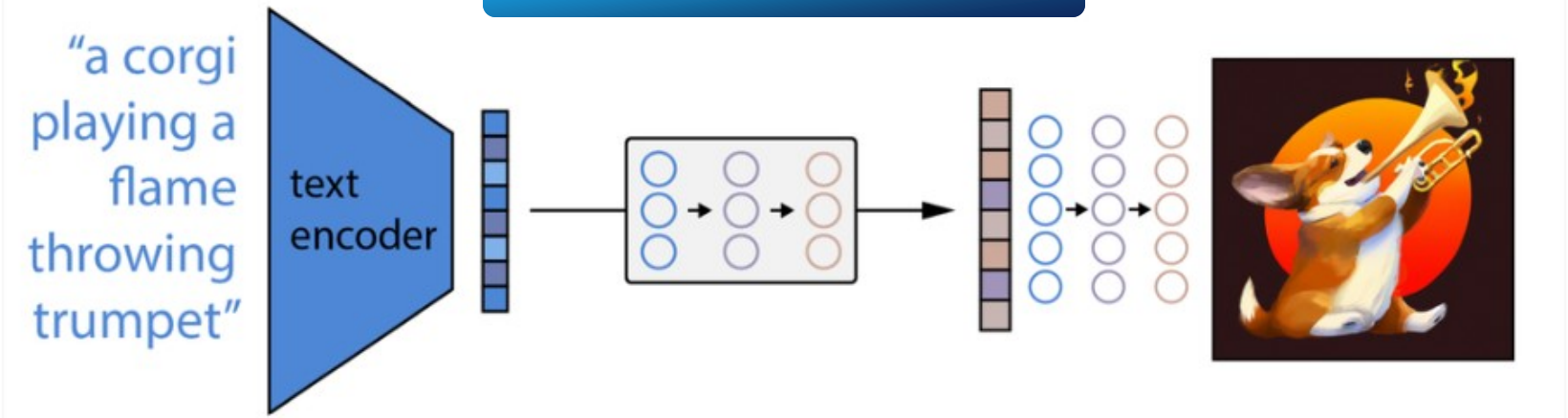
PREFERRED FOR PHOTOREALISM

Fonte: https://openai.com/dall-e-2//

How DALL·E 2 works
Bird's eye-view

DALL·E 2 has learned the **relationship between images** and the **text** used to describe them. It uses a **process** called "**diffusion**," which **starts** with a pattern of **random dots** and gradually alters that pattern towards an **image** when it **recognizes specific aspects** of that image, as described below

DIFFUSION PROCESS IN BRIEF



01

A **text prompt** is **input** into a **text encoder** that is trained to map the prompt to a representation space

02

A model called "the **prior**" maps the text encoding to a corresponding **image encoding** that **captures** the **semantic information** of the prompt contained in the text encoding

03

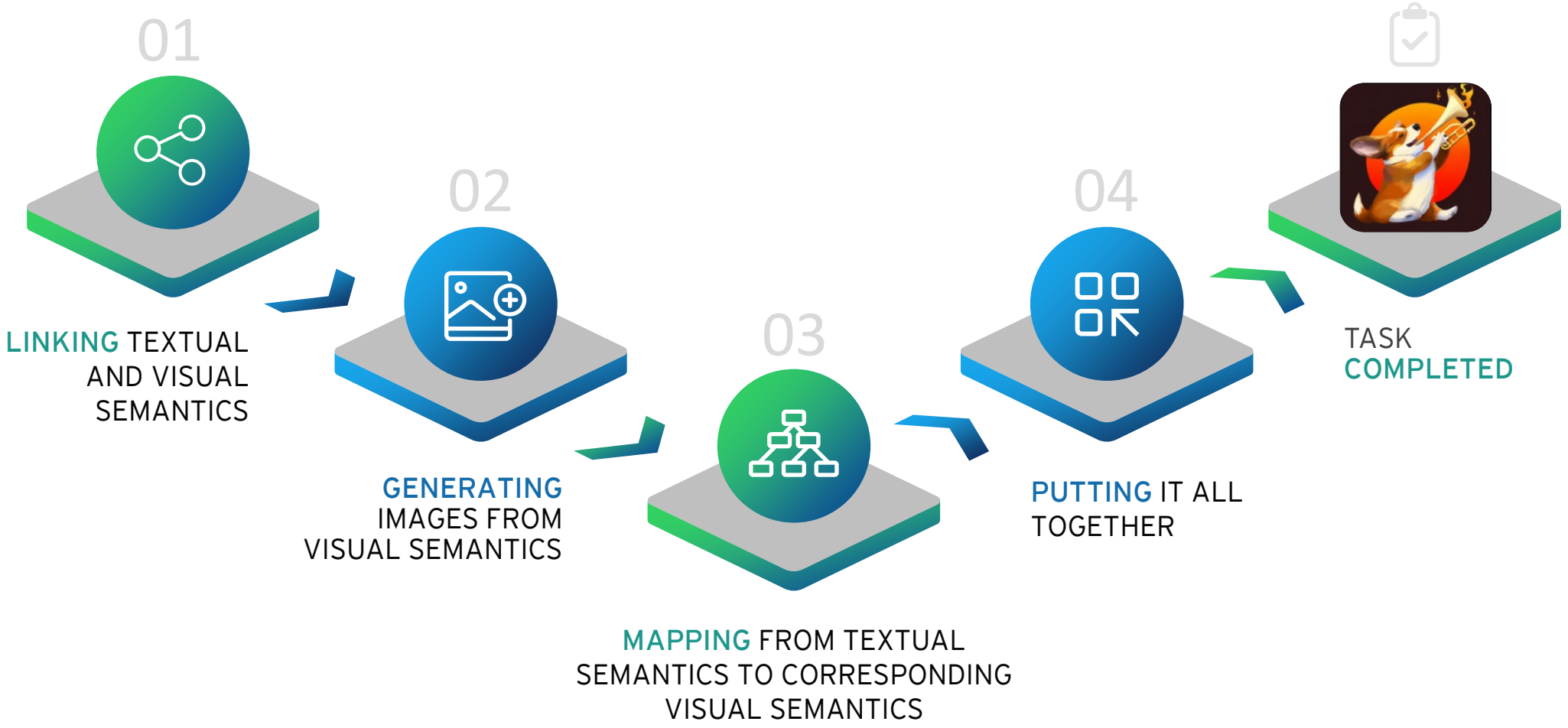
Finally, an **image decoder** stochastically generates an image which is a **visual manifestation** of this **semantic**

Fonte: https://www.assemblyai.com/blog/how-dall-e-2-actually-works/

How DALL·E 2 works

Detailed Look - Step by Step

Let's get started by looking at how **DALL·E 2** learns to link related textual and visual abstractions. Going into more detail, the **process** is divided into **four main steps** leading to **task completion**



How DALL·E 2 works

Step 1 – Linking textual and visual semantics

The **link between textual semantics and their visual representations** in DALL-E 2 is learned by another OpenAI model called **CLIP** (**C**ontrastive **L**anguage-**I**mage **P**re-training). CLIP is important to DALL-E 2 because **it is what ultimately determines how semantically-related** a natural language snippet is to a visual concept, which is critical for text-conditional image generation

CLIP TRAINING PROCESS PRINCIPLES

1. **Generate** the image and text encoding of each of the image-caption pairs
2. **Calculate** the cosine similarity of each (image, text) embedding pair
3. Iteratively **minimize** the cosine similarity between incorrect image-caption pairs, and maximize the cosine similarity between the correct image-caption pairs

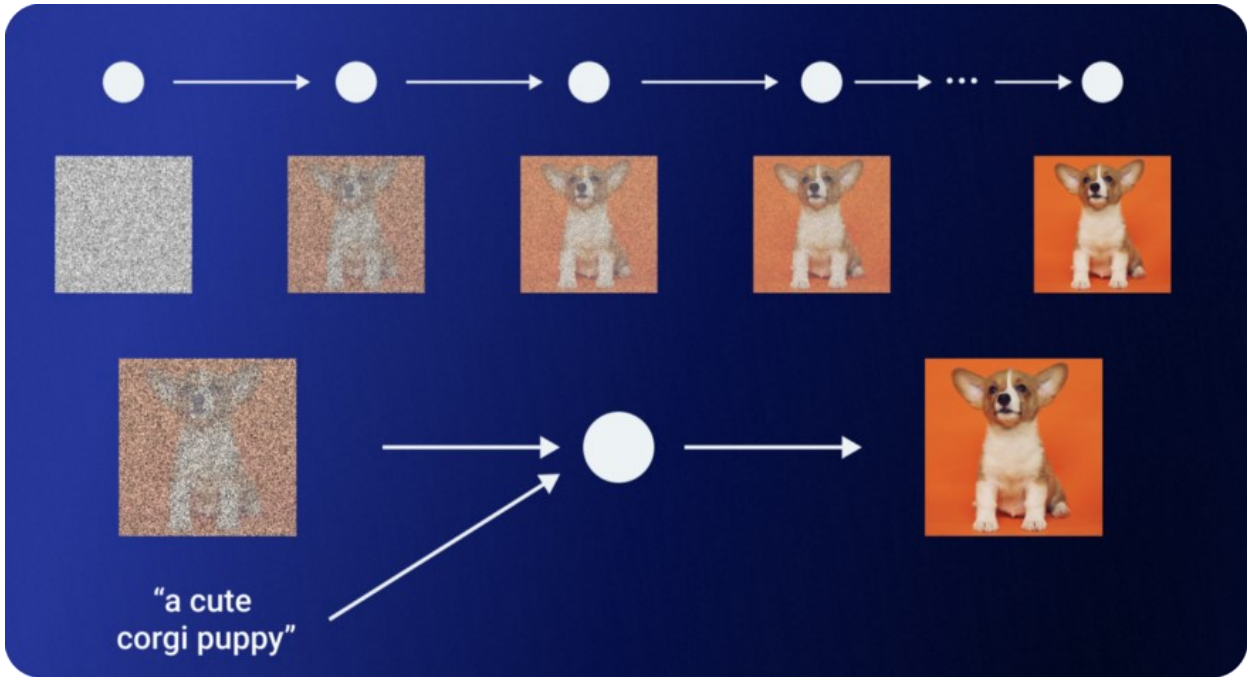
Fonte: https://www.assemblyai.com/blog/how-dall-e-2-actually-works/

How DALL·E 2 works

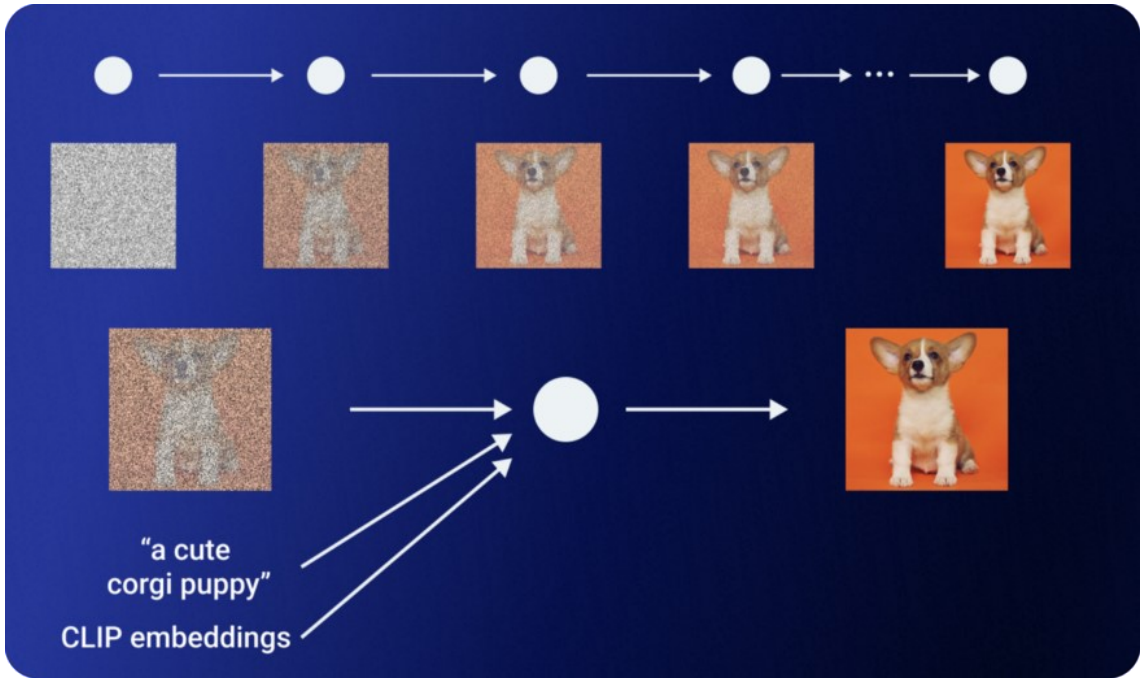
Step 2 – Generating images from visual semantics 1/2

After training, the CLIP model is frozen, and DALL-E 2 moves on to its next task - learning to reverse the image encoding mapping that CLIP just learned. OpenAI employs a modified version of another one of its previous models, **GLIDE** (**G**uided **L**anguage to **I**mage **D**iffusion for **G**eneration and **E**diting), to perform this image generation. The GLIDE model **learns** to **invert** the **image encoding process** in order to **stochastically decode CLIP image embeddings**

GLIDE includes the text caption in the generation process



DALL·E 2's modified GLIDE adds the caption CLIP embeddings



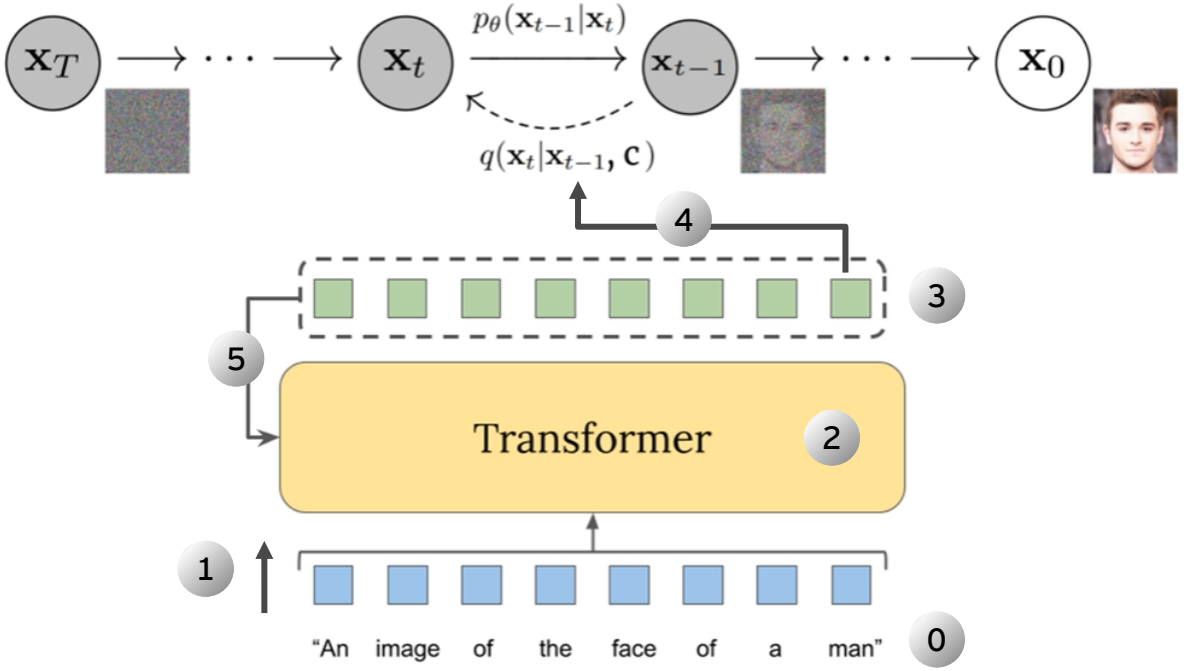
Fonte: https://medium.com/augmented-startups/how-does-dall-e-2-work-e6d492a2667f

How DALL·E 2 works

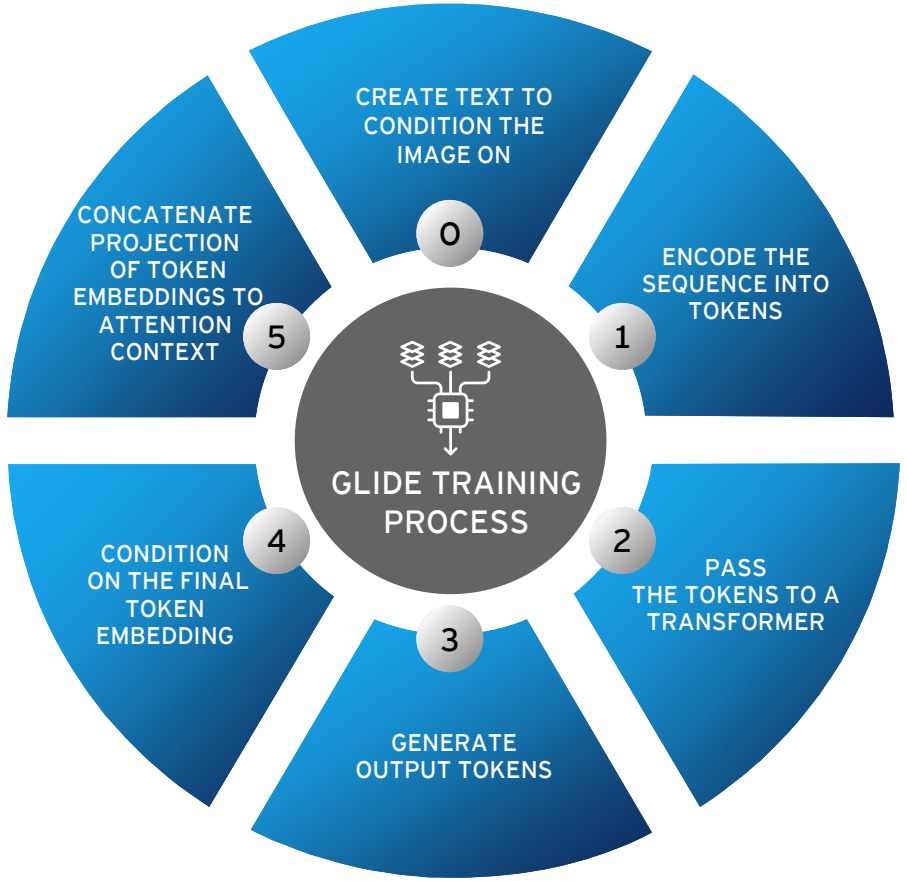
Step 2 – Generating images from visual semantics 2/2

By adding more textual data to the training process and generating text-conditional images as a result, **GLIDE expands** the **fundamental idea** of **Diffusion Models**. Let's examine the **training process for GLIDE**:

Adding textual conditioning to the diffusion process



Fonte: https://www.assemblyai.com/blog/how-dall-e-2-actually-works/



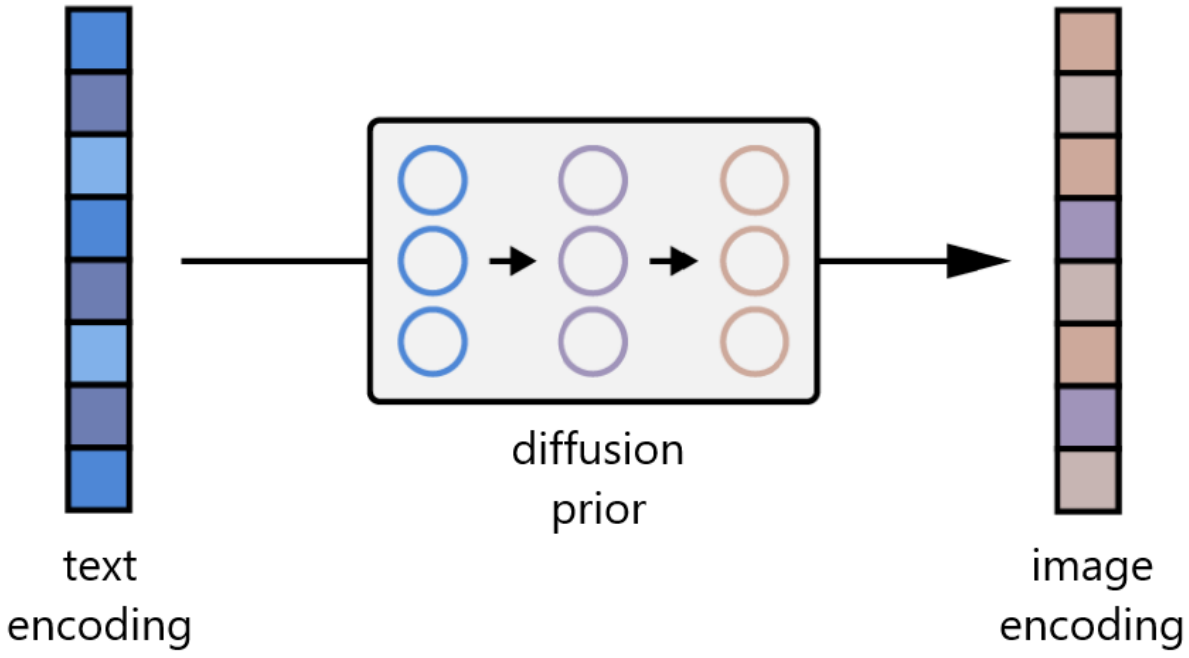
How DALL·E 2 works

Step 3 - Mapping from textual semantics to corresponding visual semantics

While the modified-GLIDE model successfully creates images that reflect the semantics captured by image encodings, **how do we inject the text conditioning information from our prompt into the image generation process? DALL·E 2 uses** the **Diffusion Prior** (DP) in order **to map from the text encodings of image captions to the image encodings of their corresponding images.** The DP consists of a **decoder-only Transformer** and operates, with a causal attention mask, in an **ordered sequence** (as described below)

DP ORDERED SEQUENCE

1. The tokenized text/caption
2. The CLIP text encodings of these tokens
3. An encoding for the diffusion timestep
4. The noised image passed through the CLIP image encoder
5. Final encoding whose output from Transformer is used to predict the unnoised CLIP image encoding



Fonte: https://www.assemblyai.com/blog/how-dall-e-2-actually-works/

How DALL·E 2 works

Step 4 - Putting it all together

At this point, **we have all DALLE 2's functional components** and need only to **chain them together** for **text-conditional image generation**, as described below

1 MAPPING (CLIP)

First, the CLIP text encoder maps the image description into the **representation space**

2 MAPPING (DP)

Next, the Diffusion Prior (DP) maps from the CLIP text encoding to a **corresponding CLIP image encoding**

3 GENERATING & CONVEYING

Finally, the modified-GLIDE generation model maps from the representation space into the image space via reverse-Diffusion, **generating one of many possible images that conveys the semantic information** within the input caption

THANK YOU FOR THE ATTENTION

The better the question. The better the answer.
The better the world works.

EY
Building a better working world

Q&A

Let's talk together

Building a better working world