

# Large Language Models: An Introduction

## Robert Haase

These slides can be reused under the terms of the [CC-BY4.0](#) license.

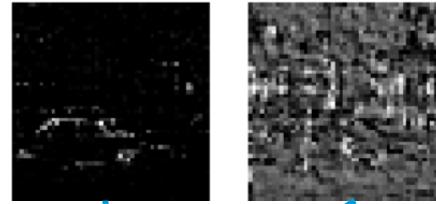
# Quiz

What is this part of a DNN typically called?

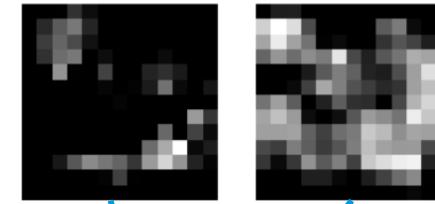
Layer 1 (256, 100, 100)



Layer 2 (512, 50, 50)



Layer 4 (2048, 13, 13)



400x400



convolutional

convolutional

conv.

conv.

conv.

- Beach wagon
- goldfish
- palace

Reducer



Increaser



Encoder

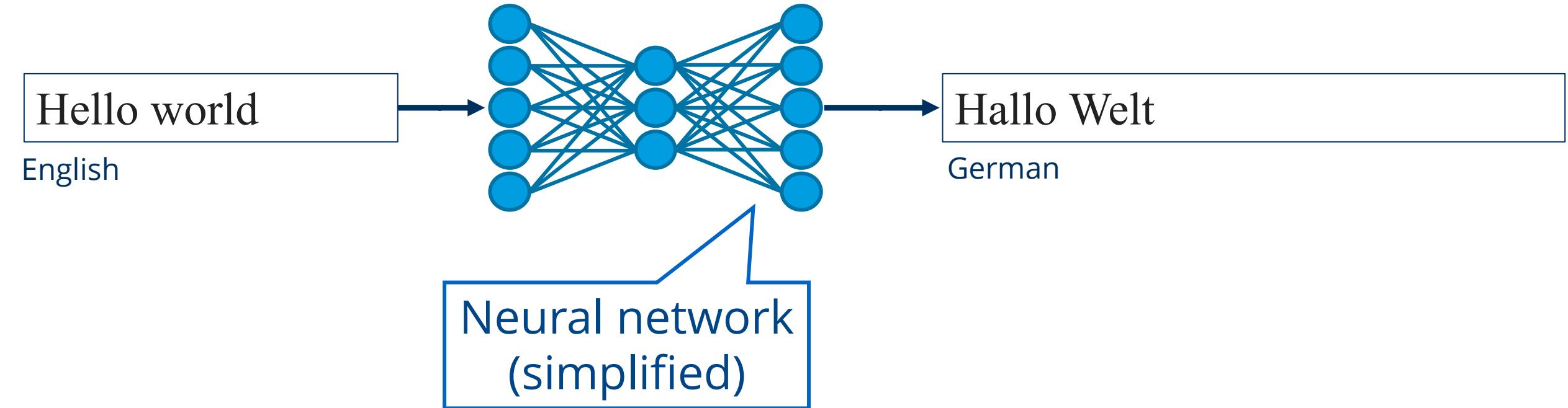


Decoder



# Large Language Models (LLMs)

Text-to-text, translation



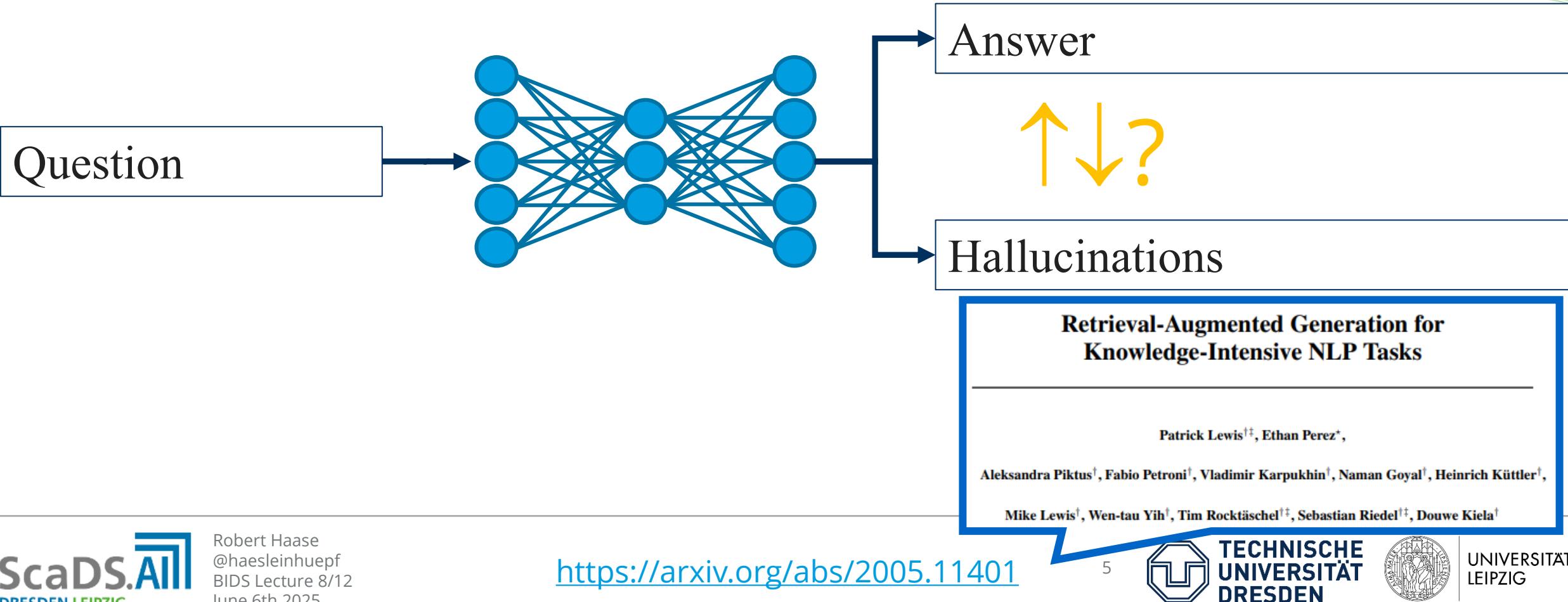
# Translating training materials

The figure shows four side-by-side Jupyter notebook interfaces, each displaying a different language version of a training material about multidimensional image stacks. Each notebook includes a flag of the target country in its header.

- Multidimensional image stacks:** A British flag header. The content covers handling multi-dimensional image data, three-dimensional image stacks, image slicing, and videos. It includes Python code for reading an MRI stack and creating orthogonal slices, followed by three grayscale MRI brain slices.
- Multidimensional Bildstapel:** A German flag header. The content is identical to the English version but translated into German. It includes the same code and image examples.
- Pilas de imágenes multidimensionales:** A Spanish flag header. The content is identical to the English version but translated into Spanish. It includes the same code and image examples.
- Piles d'images multidimensionnelles:** A French flag header. The content is identical to the English version but translated into French. It includes the same code and image examples.

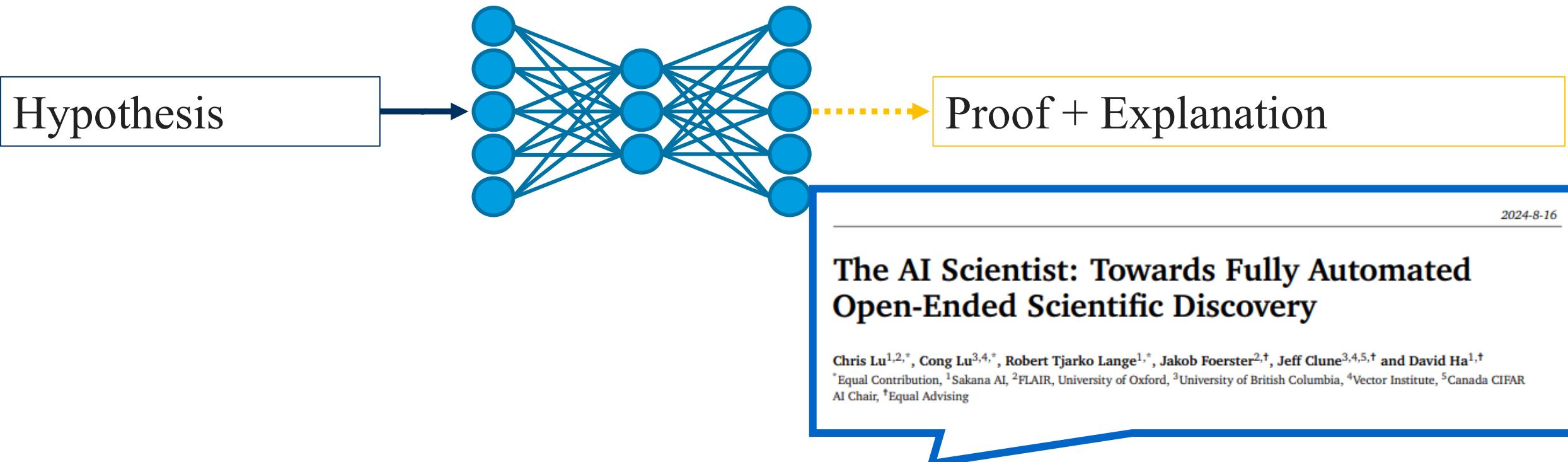
# Large Language Models (LLMs)

Text-to-text, translation, knowledge retrieval



# Large Language Models (LLMs)

Text-to-text, translation, reasoning



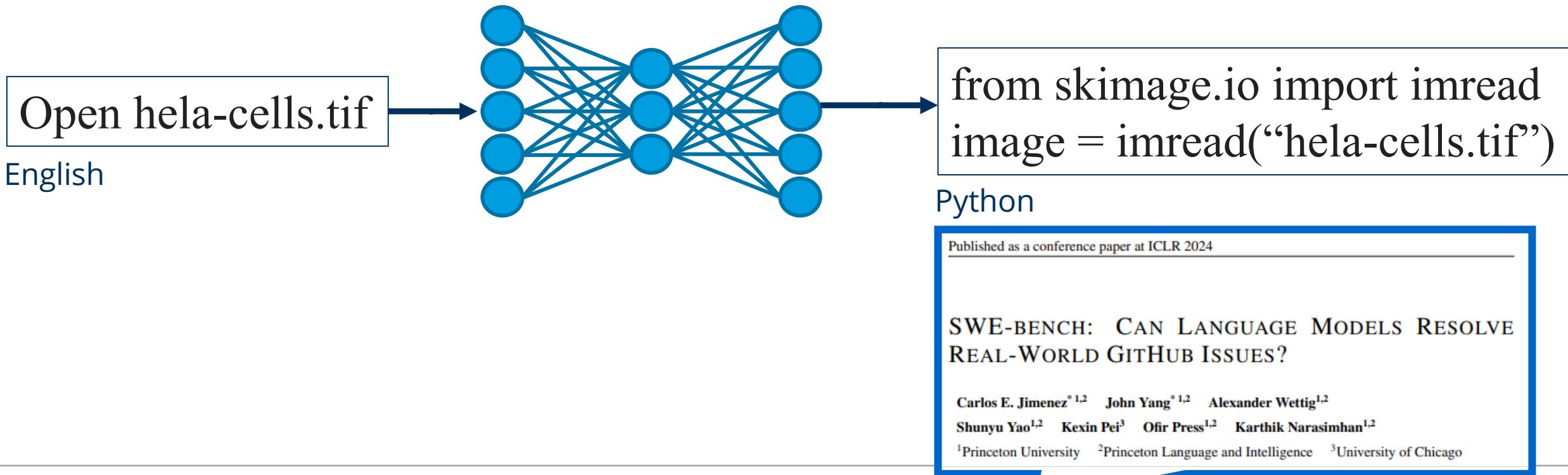
# Large Language Models (LLMs)

Text-to-text, translation, reasoning



# Large Language Models (LLMs)

Text-to-text, translation, code generation



# Large Language Models (LLMs)

Text-to-text, translation, code generation

## Issue

*data leak in GBDT due to warm start (This is about the non-histogram-based version of...*

## Codebase

<code>sklearn/</code>	<code>reqs.txt</code>
<code>examples/</code>	<code>setup.cfg</code>
<code>README.rst</code>	<code>setup.py</code>



## Language Model



### Generated PR

+20 -12

<code>sklearn</code>
<code>gradient_boosting.py</code>
<code>helper.py</code>
<code>utils</code>



## Unit Tests

Pre PR	Post PR	Tests
✗	✓	join_struct_col
✗	✓	vstack_struct_col
✗	✓	dstack_struct_col
✓	✓	matrix_transform
✓	✓	euclidean_diff

Published as a conference paper at ICLR 2024

## SWE-BENCH: CAN LANGUAGE MODELS RESOLVE REAL-WORLD GITHUB ISSUES?

Carlos E. Jimenez<sup>\* 1,2</sup> John Yang<sup>\* 1,2</sup> Alexander Wettig<sup>1,2</sup>  
Shunyu Yao<sup>1,2</sup> Kexin Pei<sup>3</sup> Ofir Press<sup>1,2</sup> Karthik Narasimhan<sup>1,2</sup>

<sup>1</sup>Princeton University <sup>2</sup>Princeton Language and Intelligence <sup>3</sup>University of Chicago

# Chatting with LLMs on GitHub/Lab

Goal: Transparency + knowledge exchange

haesleinhuepf opened last week ...

I'd like to have python code for classifying images. I think using the resnet50 model from torchvision. We can use this as example image:



...  
Before running the classification, make sure all necessary libraries are installed.  
git-bob try to do this.

Create sub-issue ▾   1

github-actions last week Contributor ...

This message was generated by [git-bob](#) (version: 0.19.1, model: claude-3-5-sonnet-20241022, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

I worked on implementing image classification using ResNet50 from torchvision for the microscope and cat image. I first saved the input image as microscope\_cat.jpg, then created a Jupyter notebook (image\_classification.ipynb) that implements the classification logic using the ResNet50 model. The notebook loads the image, preprocesses it, and uses the pre-trained model to predict what's in the image. The classification results were saved to a text file and I made sure to specify the required dependencies in requirements.txt for anyone who wants to run this code.

• [classification\\_results.txt](#) Contains the top predicted categories from the ResNet50 model for the input image  
• [image\\_classification.ipynb](#) Jupyter notebook implementing the image classification using ResNet50 model from torchvision  
• [requirements.txt](#) Lists the Python package dependencies needed to run the classification code



Top 5 predictions:  
sewing machine: 0.443  
toaster: 0.240  
iron: 0.132  
soap dispenser: 0.047  
vacuum: 0.033

### Image Classification using ResNet50

First, let's install and import the necessary libraries.

```
# pip install torch torchvision pillow requests
```

In [2]:

```
import torch
from torchvision import models, transforms
from PIL import Image
import requests
from io import BytesIO
```

Load and preprocess the image

```
# Define image transformations
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Load image
image = Image.open('microscope_cat.jpg')
input_tensor = preprocess(image)
input_batch = input_tensor.unsqueeze(0)
```

Load the pre-trained ResNet50 model and run inference

```
# Load model and weights
weights = models.ResNet50_Weights.IMAGENET1K_V1
model = models.resnet50(weights=weights)
model.eval()

# Run inference
with torch.no_grad():
    output = model(input_batch)

# Get class names
categories = weights.meta['categories']

# Get top 5 predictions
_, indices = torch.sort(output[0], descending=True)
probabilities = torch.nn.functional.softmax(output[0], dim=0)

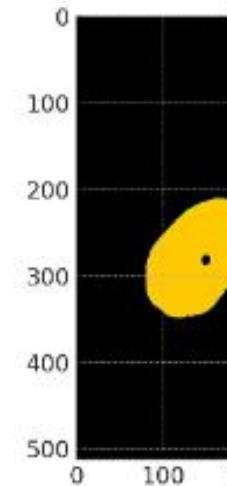
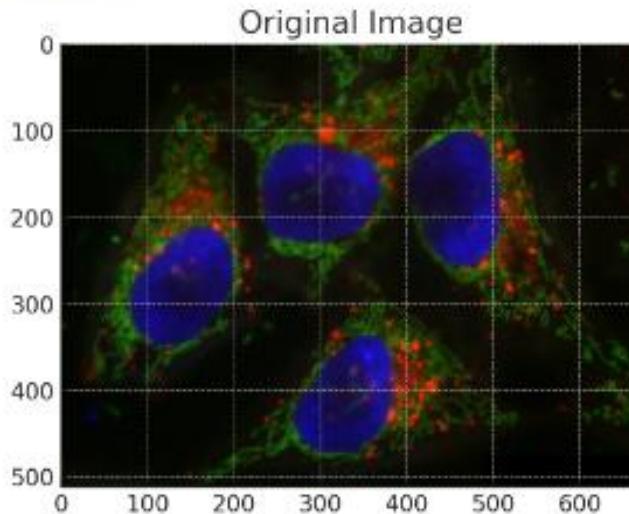
# Save results
with open('classification_results.txt', 'w') as f:
    f.write('Top 5 predictions:\n')
    for idx in indices[:5]:
        f.write(f'{categories[idx]}: {probabilities[idx].item():.3f}\n')
```

# Bio-image Analysis using Large-L

Analysing images in the browser, e.g. using ChatGPT

Prompt:  
“Please segment  
the nuclei in the  
blue channel of  
the image”

ChatGPT



The segmentation of the nuclei has been performed on the blue channel. The figure shows the segmented nuclei in different colors, each corresponding to the field of view. Each color represents a unique label assigned to the image. [→]



Analysis

```
from skimage import io, filters, measure, morphology
import matplotlib.pyplot as plt
import numpy as np

# Load the image
image = io.imread('/mnt/data/hela-cells-8bit.png')

# Since it's a color image, we'll just take the blue channel
blue_channel = image[:, :, 2]

# Threshold the image to get the binary image for nuclei
thresh = filters.threshold_otsu(blue_channel)
binary = blue_channel > thresh

# Remove small objects and perform morphological closing to merge touching nuclei
cleaned = morphology.remove_small_objects(binary, 50)
closed = morphology.closing(cleaned, morphology.disk(3))

# Label the image
label_image = measure.label(closed)

# Plot the original image and the binary image
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(image)
ax[0].set_title('Original Image')
ax[1].imshow(label_image, cmap='nipy_spectral')
ax[1].set_title('Segmented Nuclei')
plt.show()
```

# Why code generation using LLMs?

Output of LLMs is often criticized for

- Limited reproducibility
- Hard to evaluate (automatically)

Output of LLM-generated code is

- Reproducible
- Can be executed and evaluated (automatically)

Say a

Say a

Write Python code that says "a".



A.

< 2/2 >



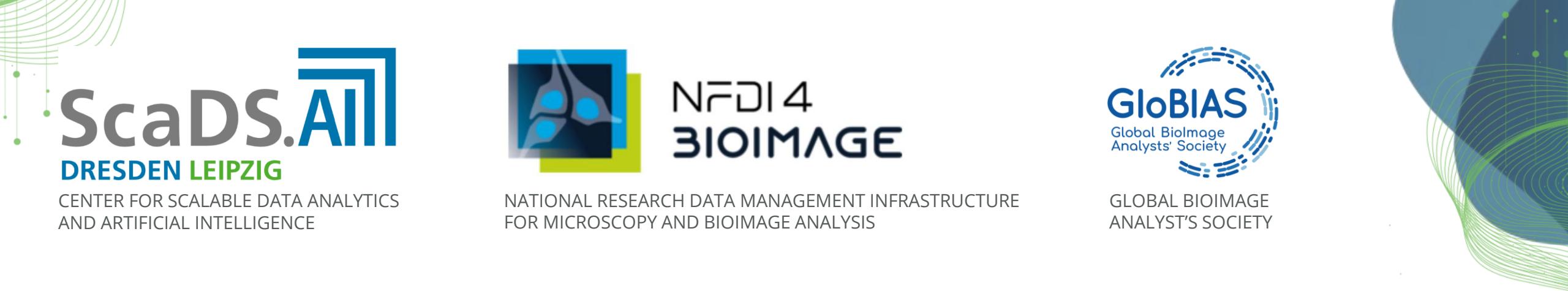
Certainly! Here's a simple Python code snippet that prints the letter "a":

python

```
print("a")
```

Copy code

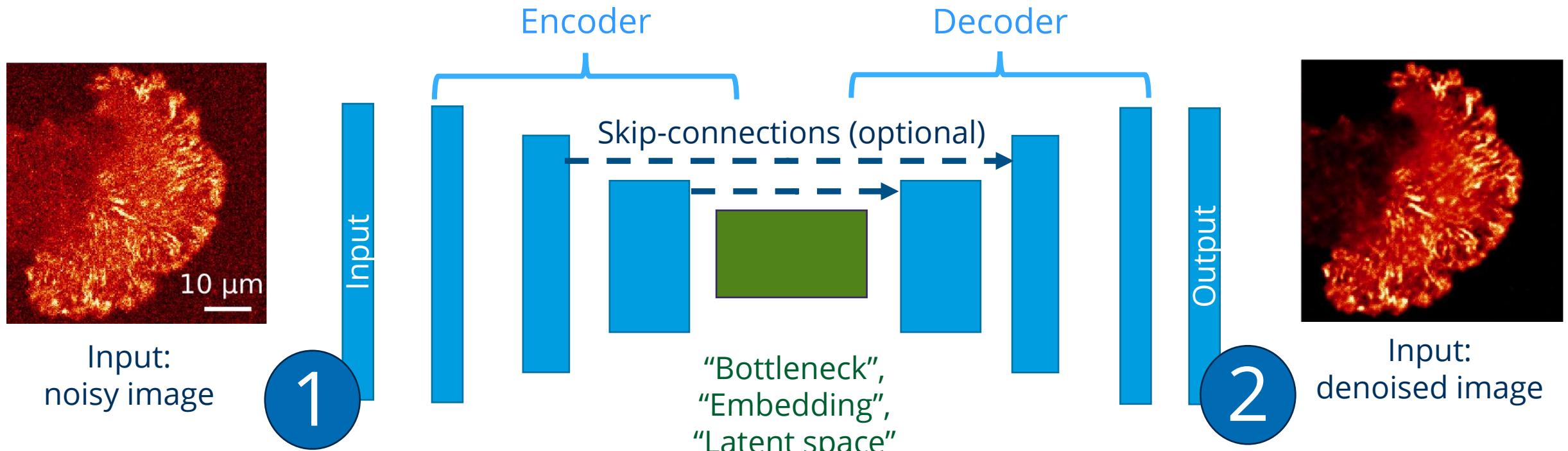
No matter how often you execute this code, the output is always guaranteed to be „a“.



# Transformer neural networks

# Recap: Encoder-Decoder Networks

Related: „Auto-encoder“, „Variational Auto-Encoder“, „U-Net“



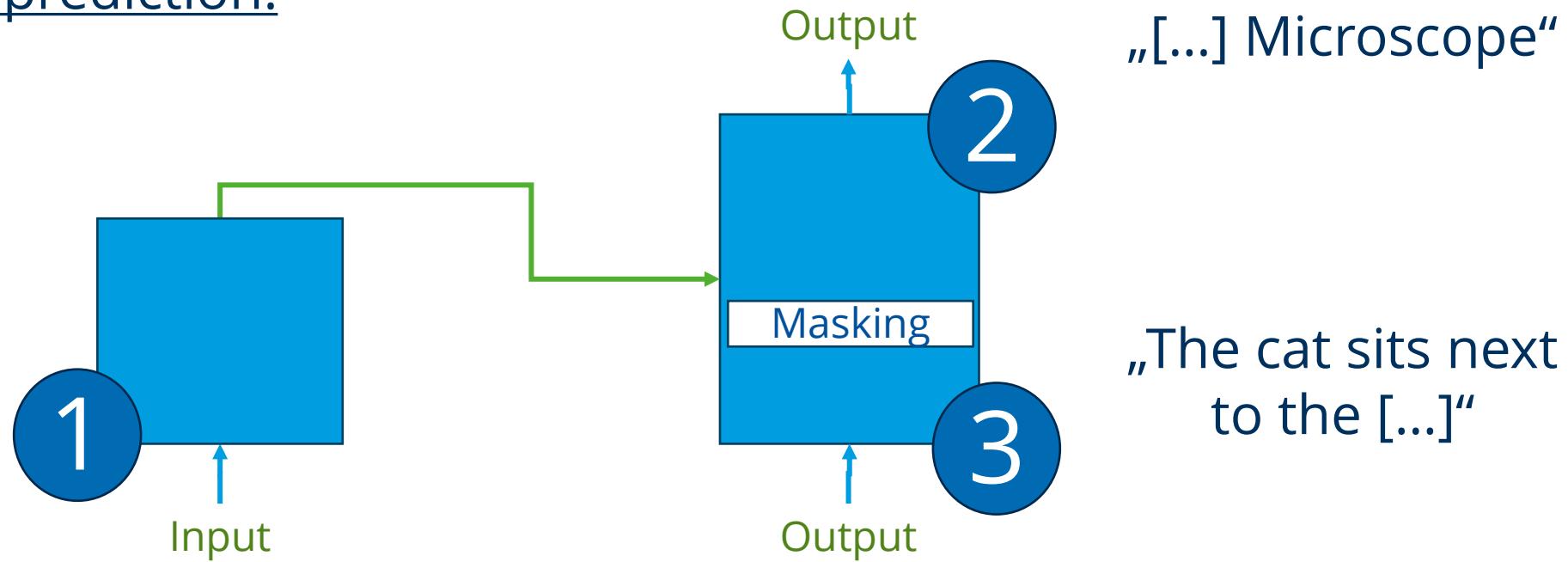
# Transformers

LLMs use the **transformer** neural network architecture

Related: Generative Pretrained Transformer (GPT)

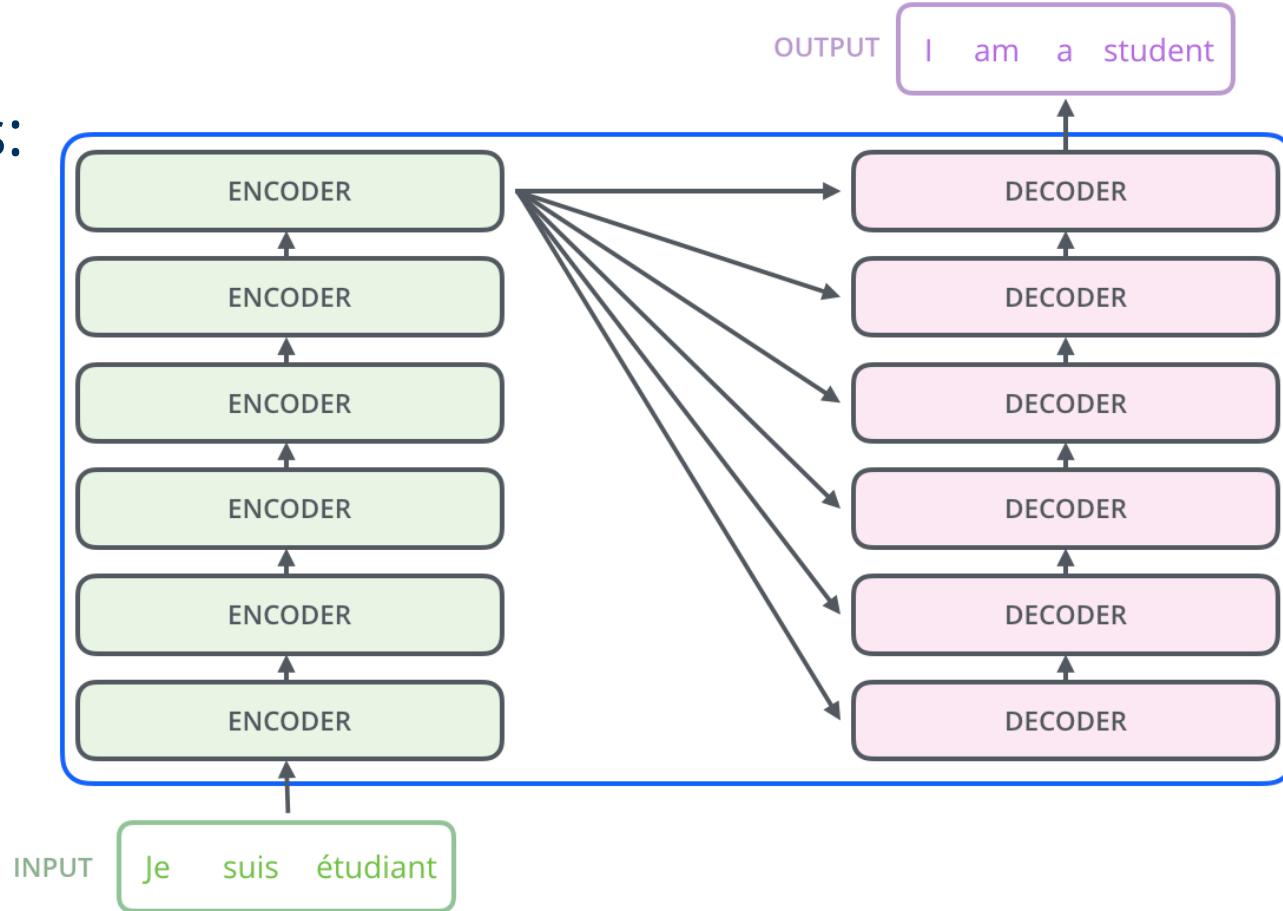
LLMs were originally developed for translation tasks  
and next-word prediction:

„Die Katze sitzt  
neben dem  
Mikroskop.“



# Generative Pretrained Transformer (GPT)

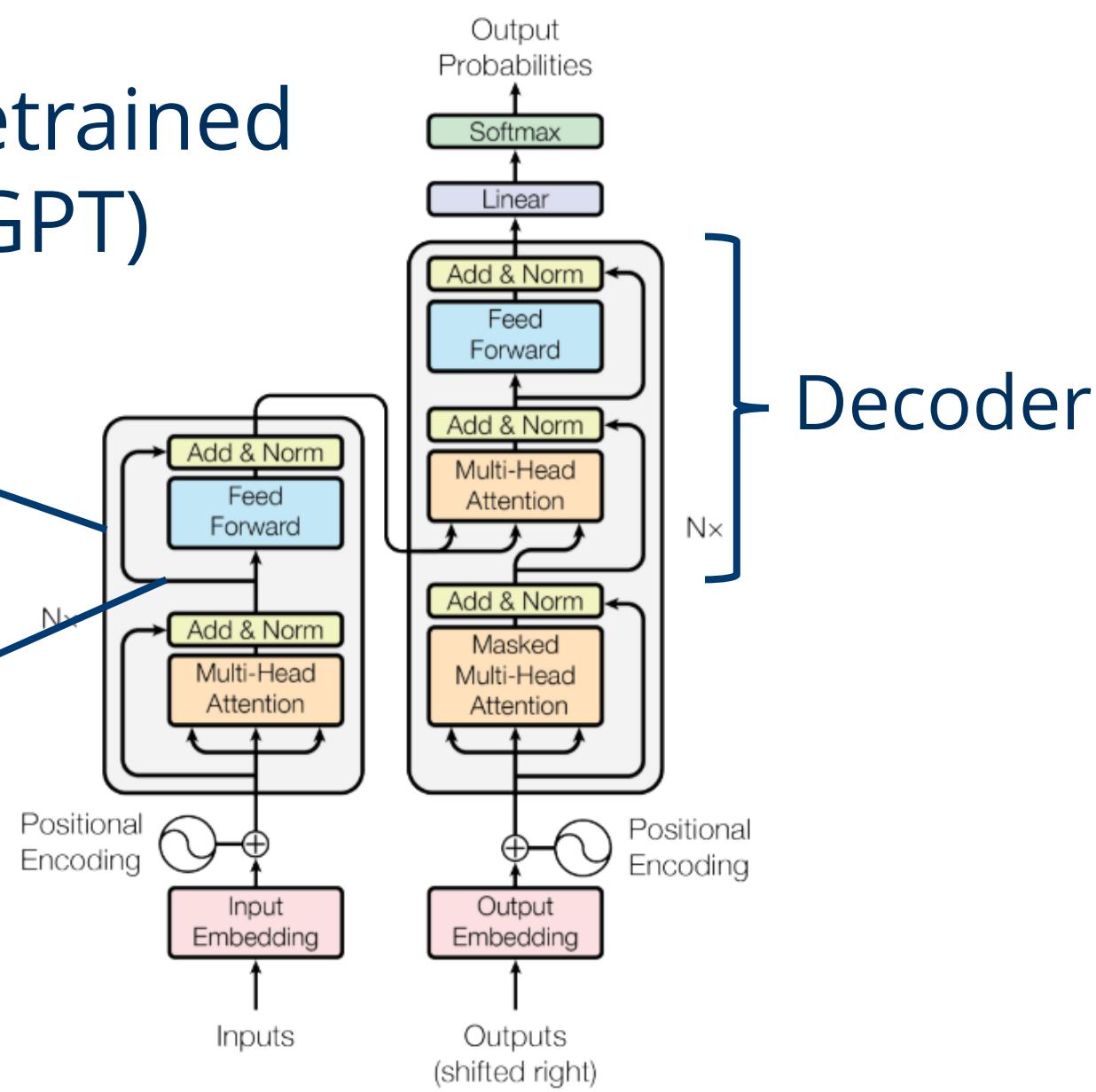
Stacks of encoders and decoders arranged like this:



# Generative Pretrained Transformer (GPT)

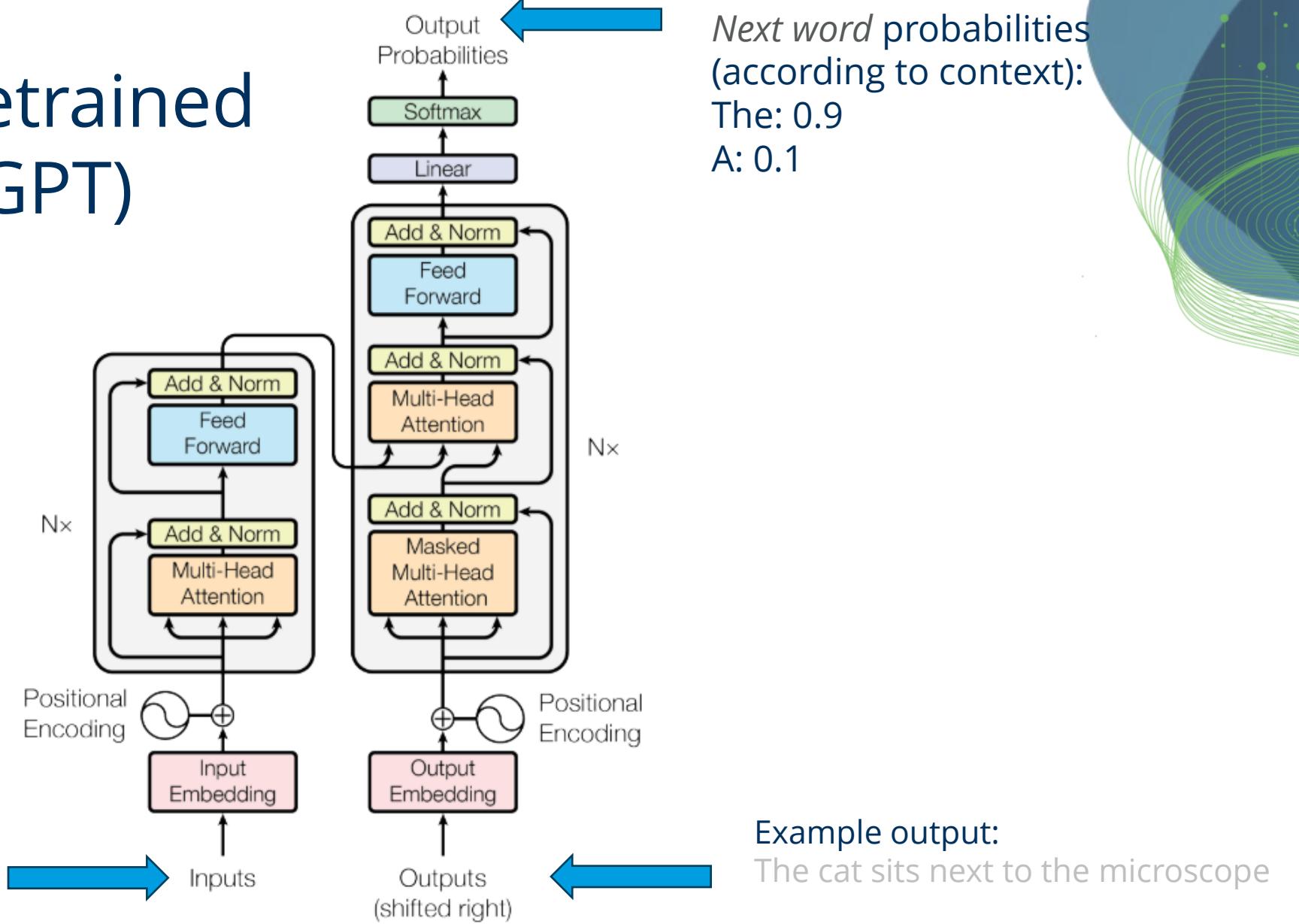
Encoder

Vectors  
and  
matrices



# Generative Pretrained Transformer (GPT)

Task: Translation

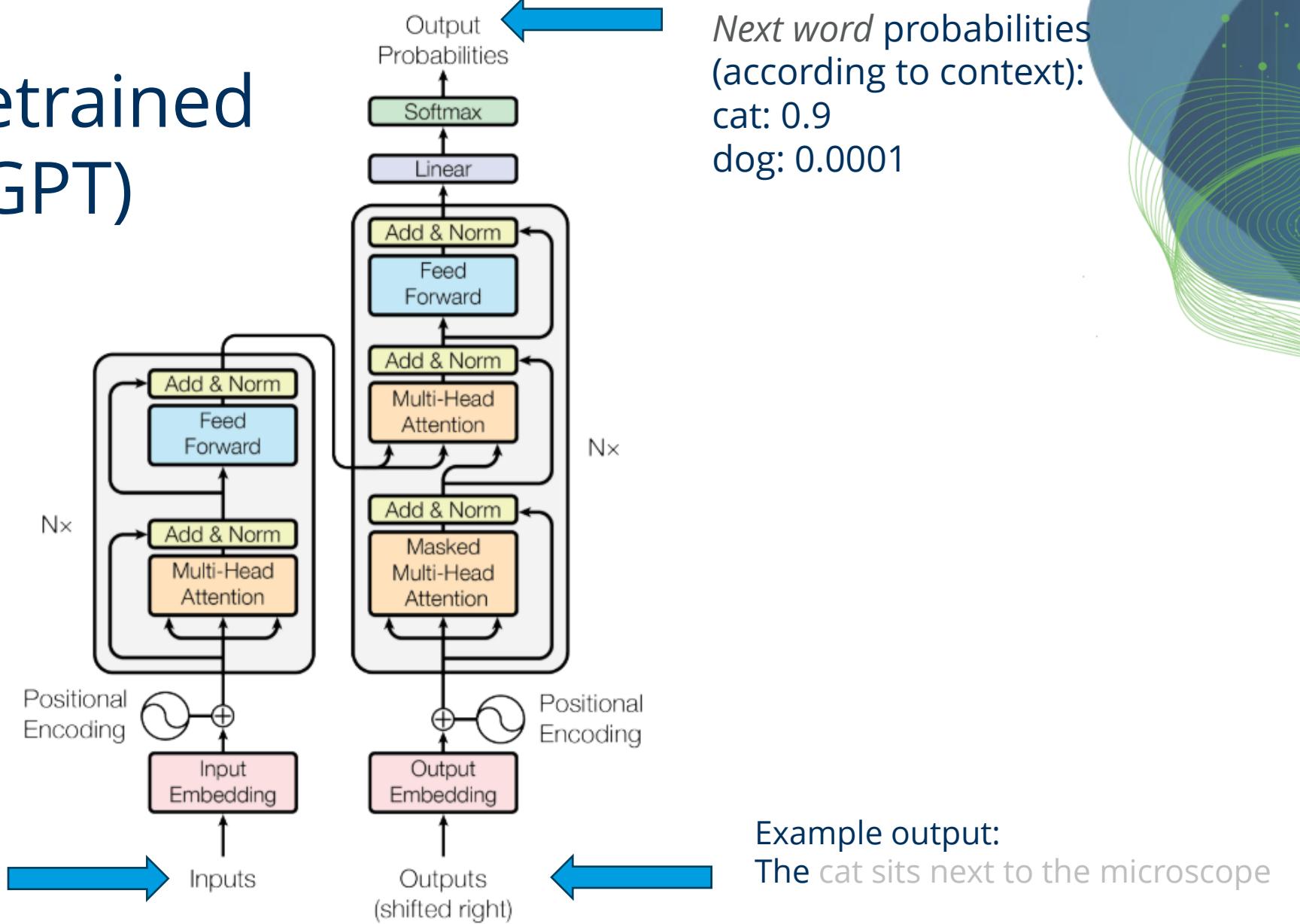


Example input:

Die Katze sitzt neben dem Mikroskop

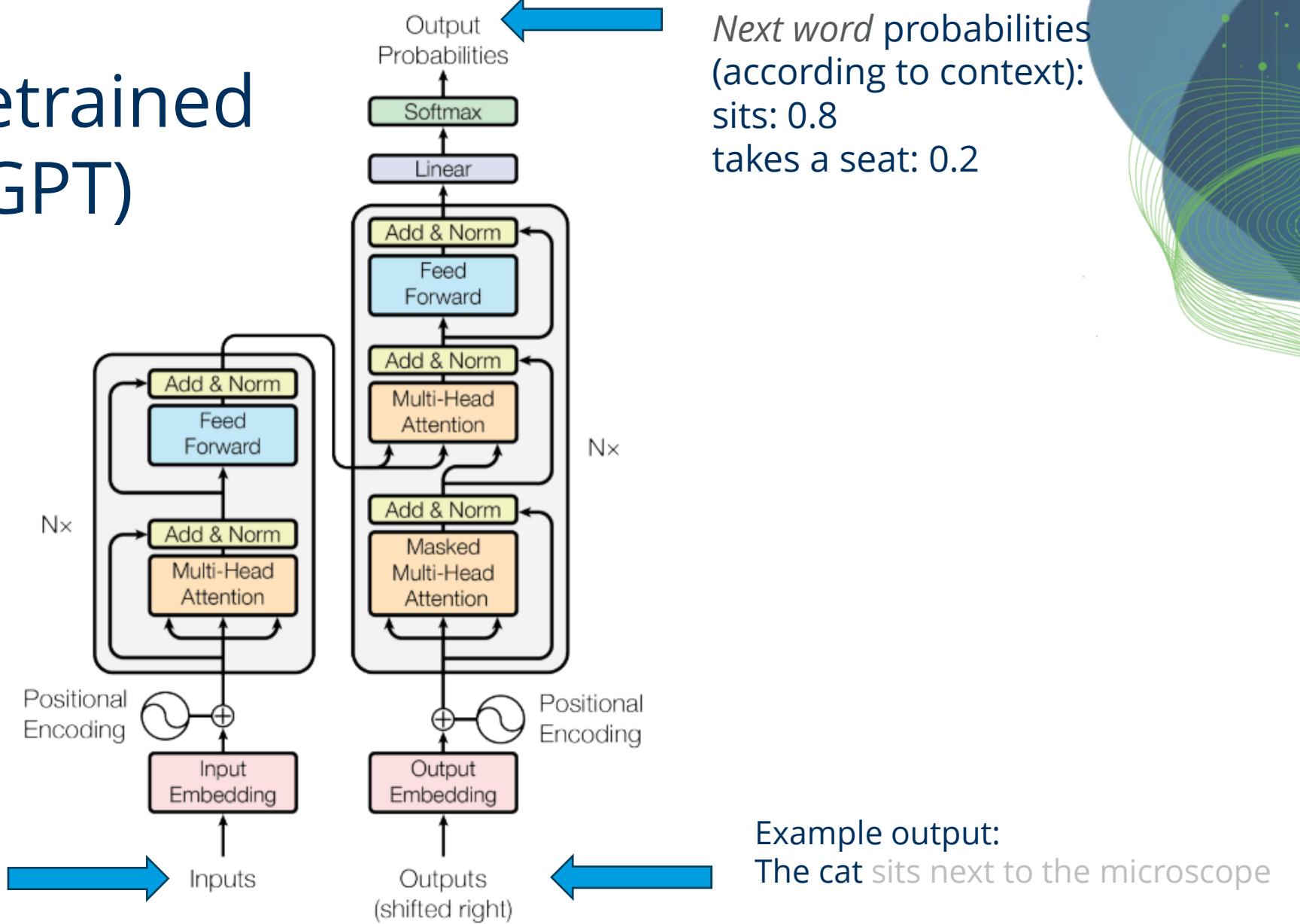
# Generative Pretrained Transformer (GPT)

Task: Translation



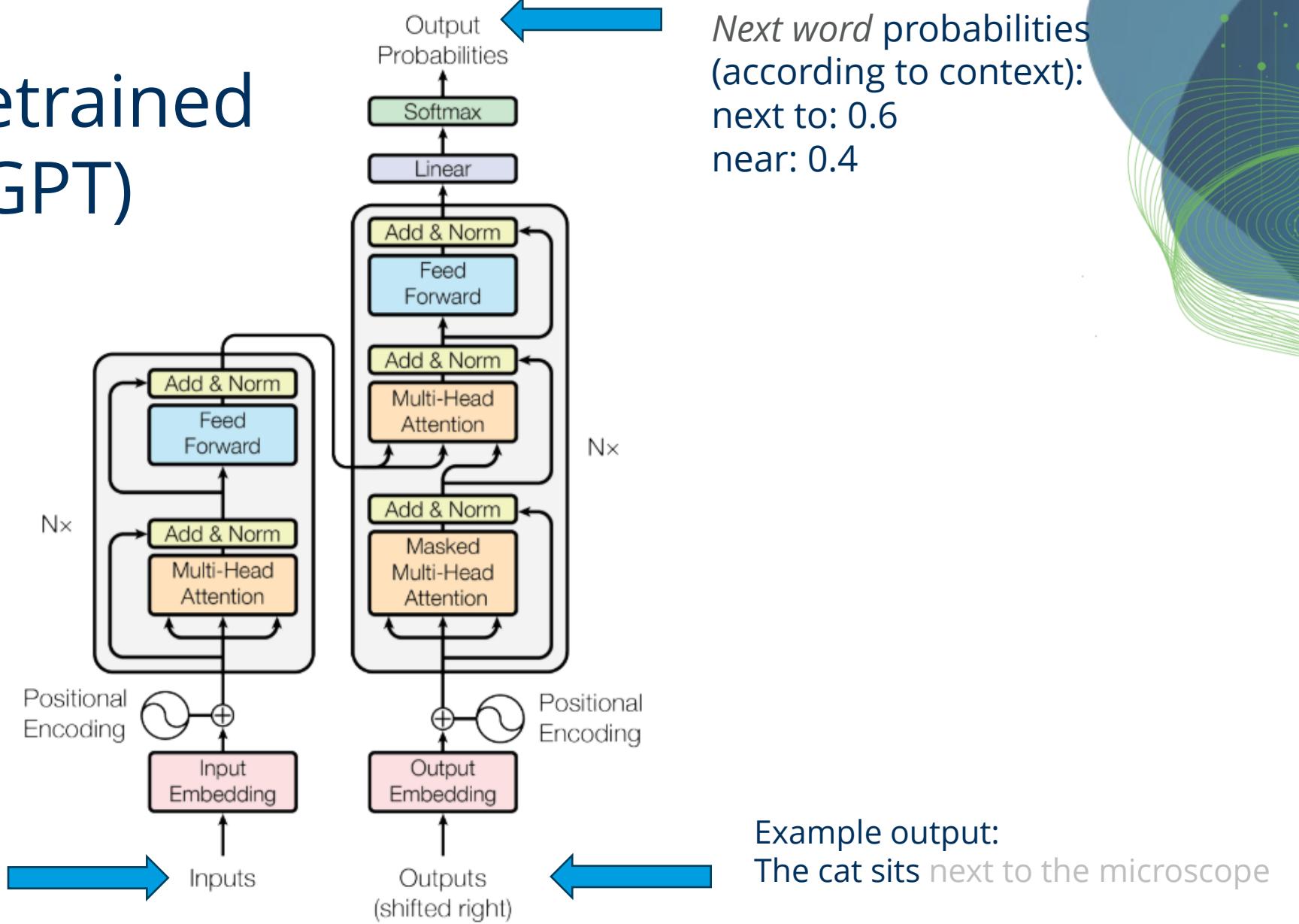
# Generative Pretrained Transformer (GPT)

Task: Translation



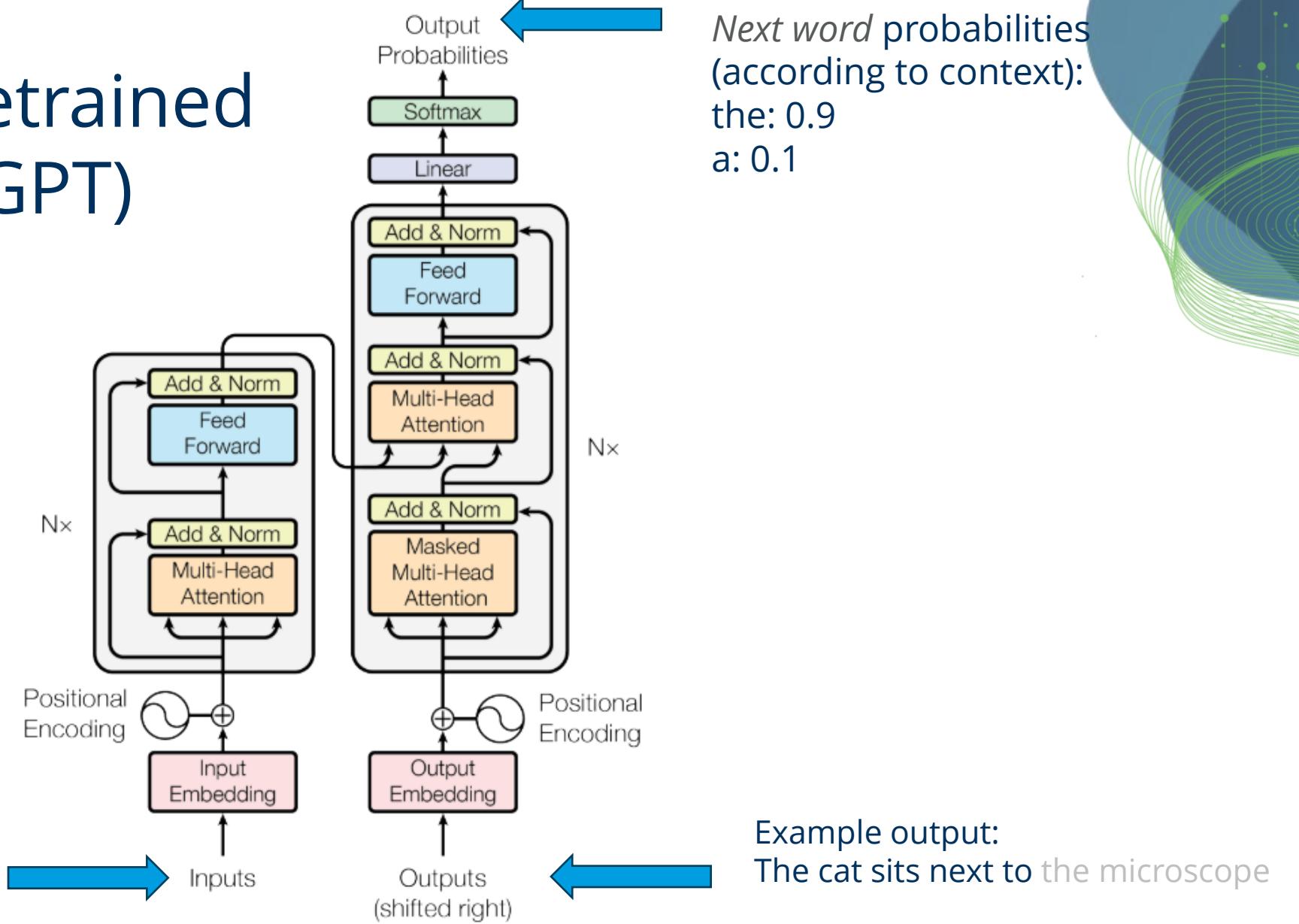
# Generative Pretrained Transformer (GPT)

Task: Translation



# Generative Pretrained Transformer (GPT)

Task: Translation

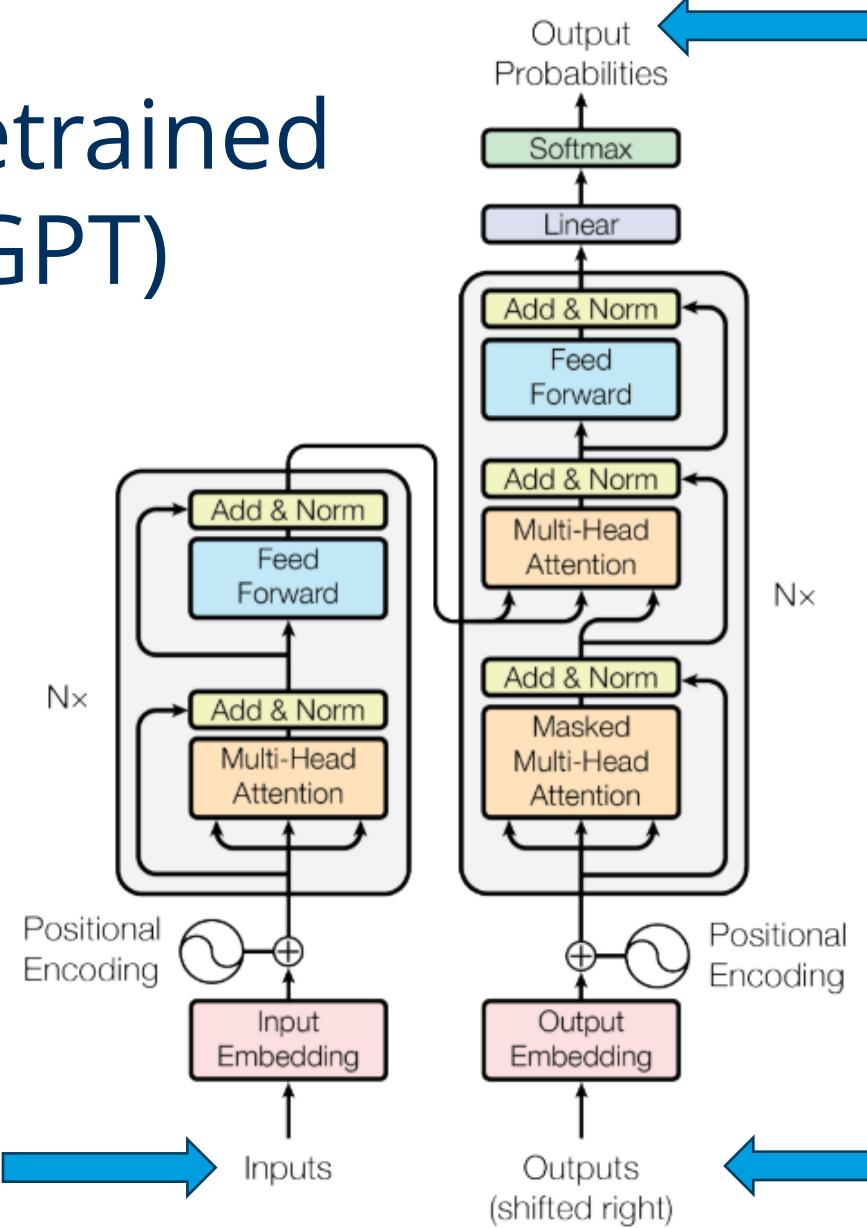


Example input:  
Die Katze sitzt neben dem *Mikroskop*

Source: Vaswani et al (2017)  
<https://arxiv.org/abs/1706.03762>

# Generative Pretrained Transformer (GPT)

Task: Translation



Example input:

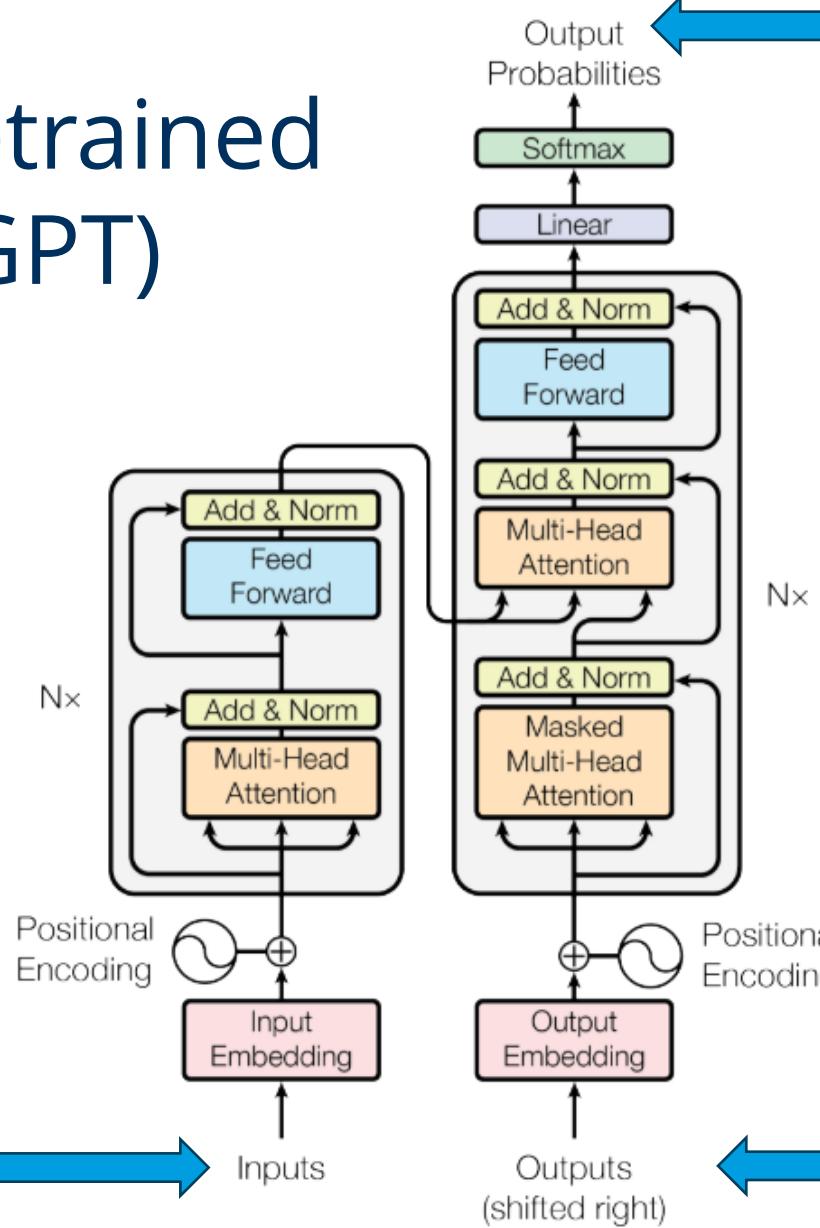
Die Katze sitzt neben dem *Mikroskop*

Next word probabilities  
(according to context):  
Heating: 0.9  
Food: 0.8  
Dog: 0.2  
Microscope: 0.01

Example output:  
The cat sits next to the microscope

# Generative Pretrained Transformer (GPT)

Task: Translation



Example input:

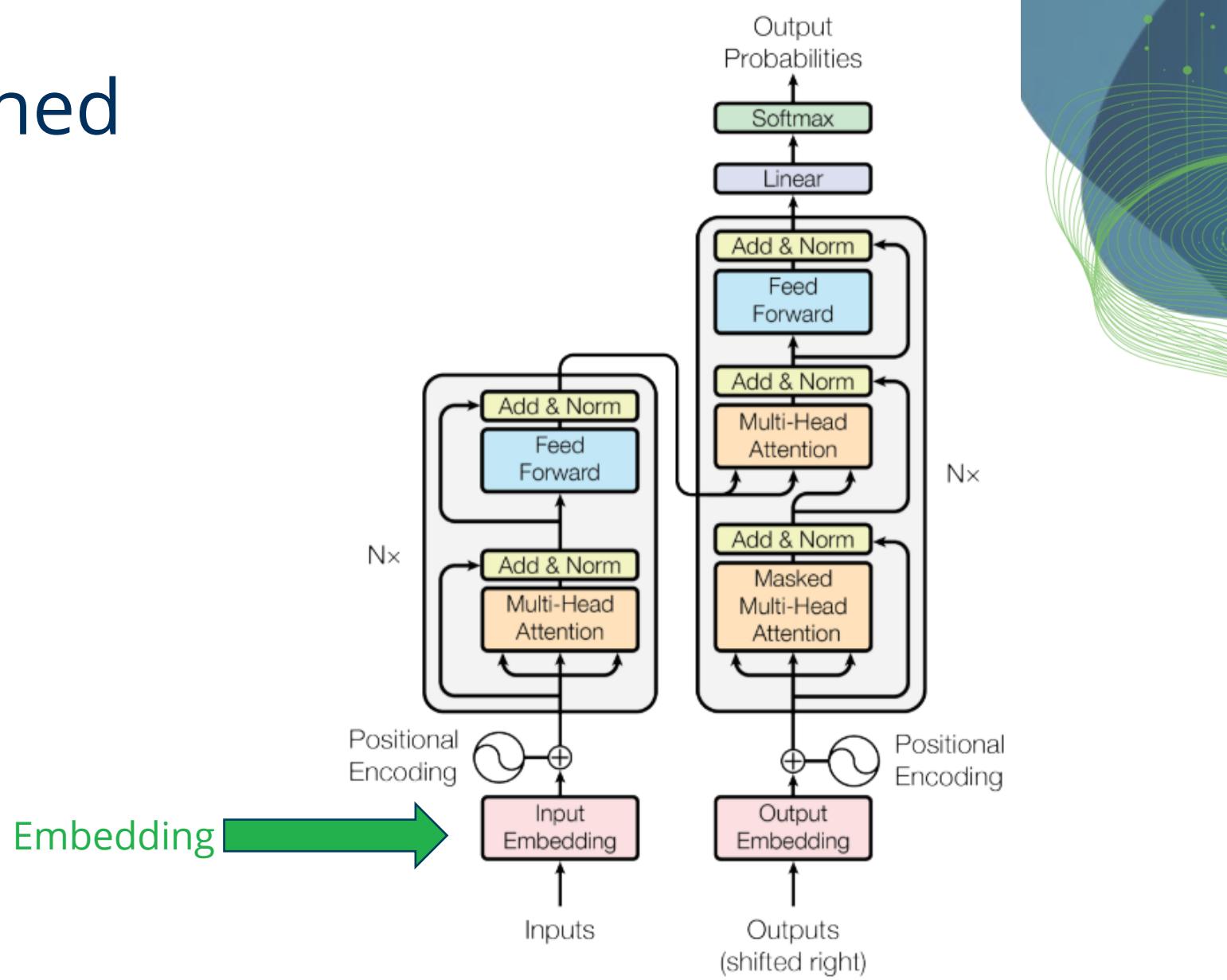
Die Katze sitzt neben dem *Mikroskop*

Next word probabilities  
(according to context):  
Heating: 0.9  
Food: 0.8  
Dog: 0.2  
**Microscope: 0.2**

Through fine-tuning the transformer, you can increase the probabilities of words.

Example output:  
The cat sits next to the microscope

# Generative Pretrained Transformer (GPT)



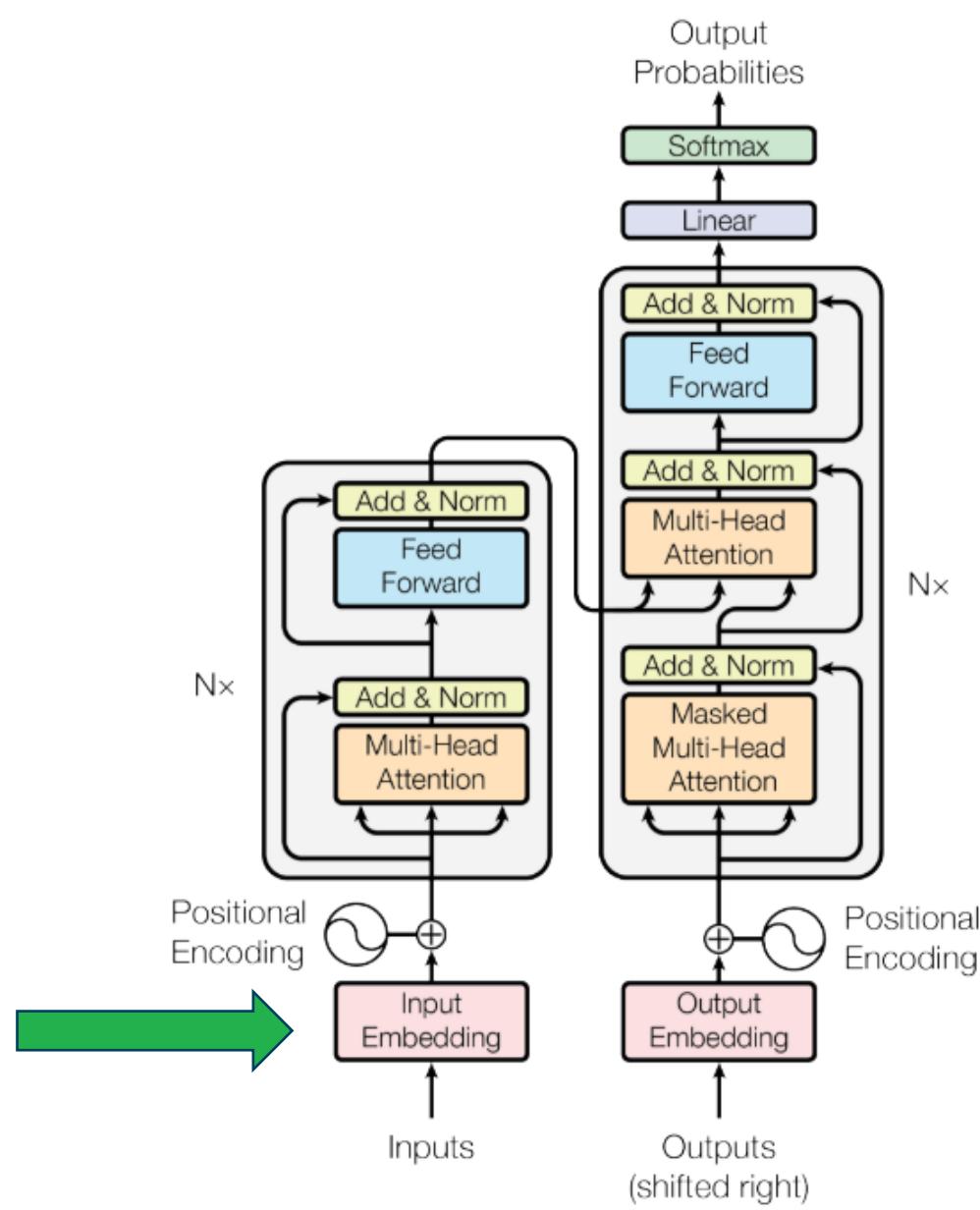
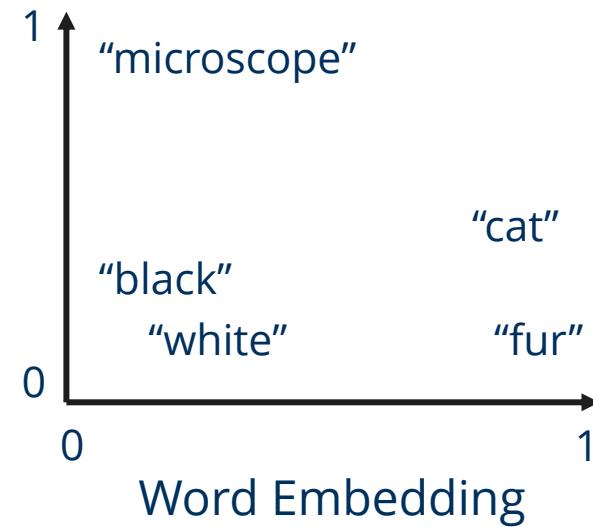
# Quiz

What is an **embedding**?

- a) in the context of image processing
- b) in the context of neural networks
- c) in general

# Generative Pretrained Transformer (GPT)

Words need to be converted into vectors to enable NNs to process them.

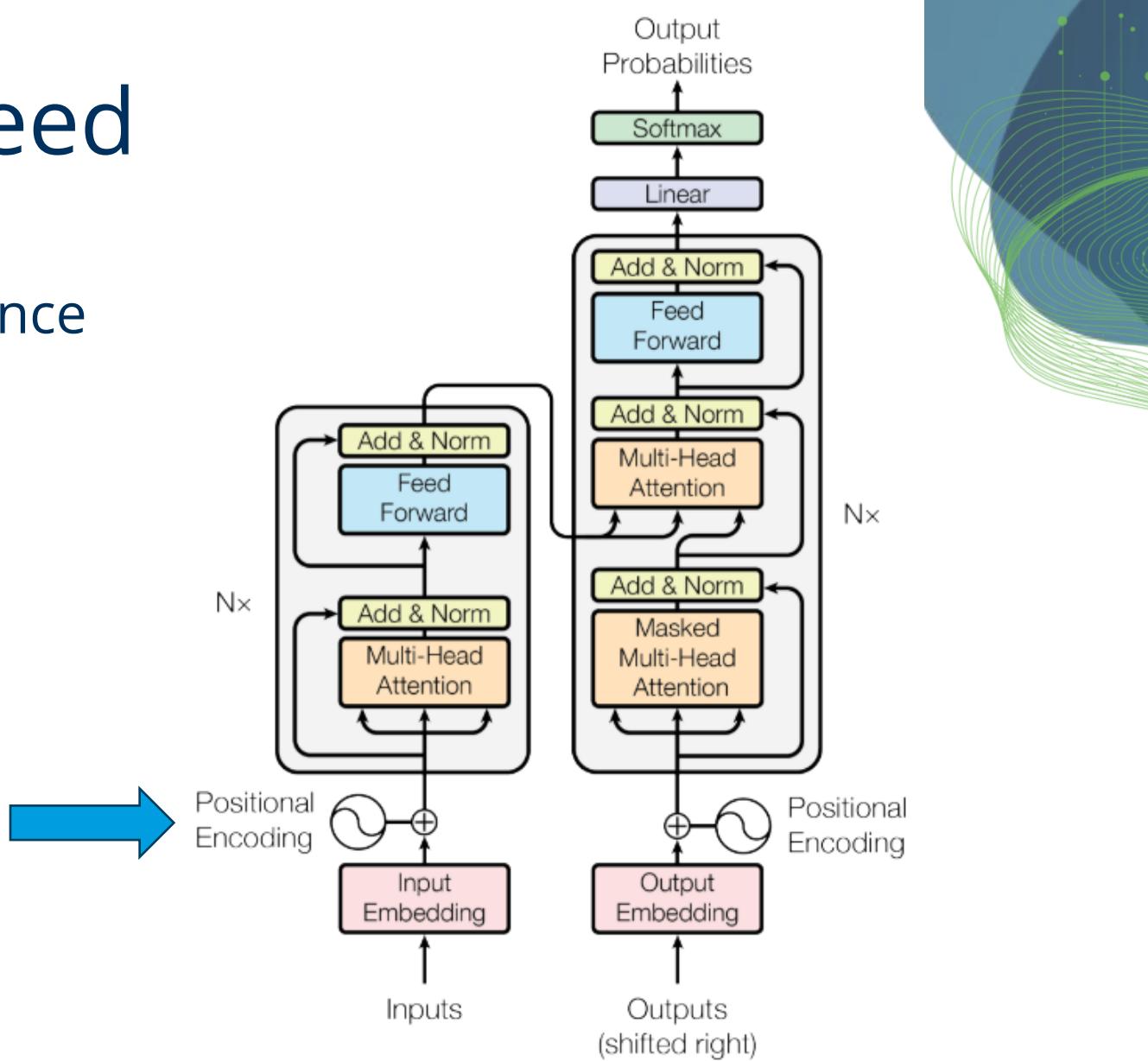


# Attention is all you need

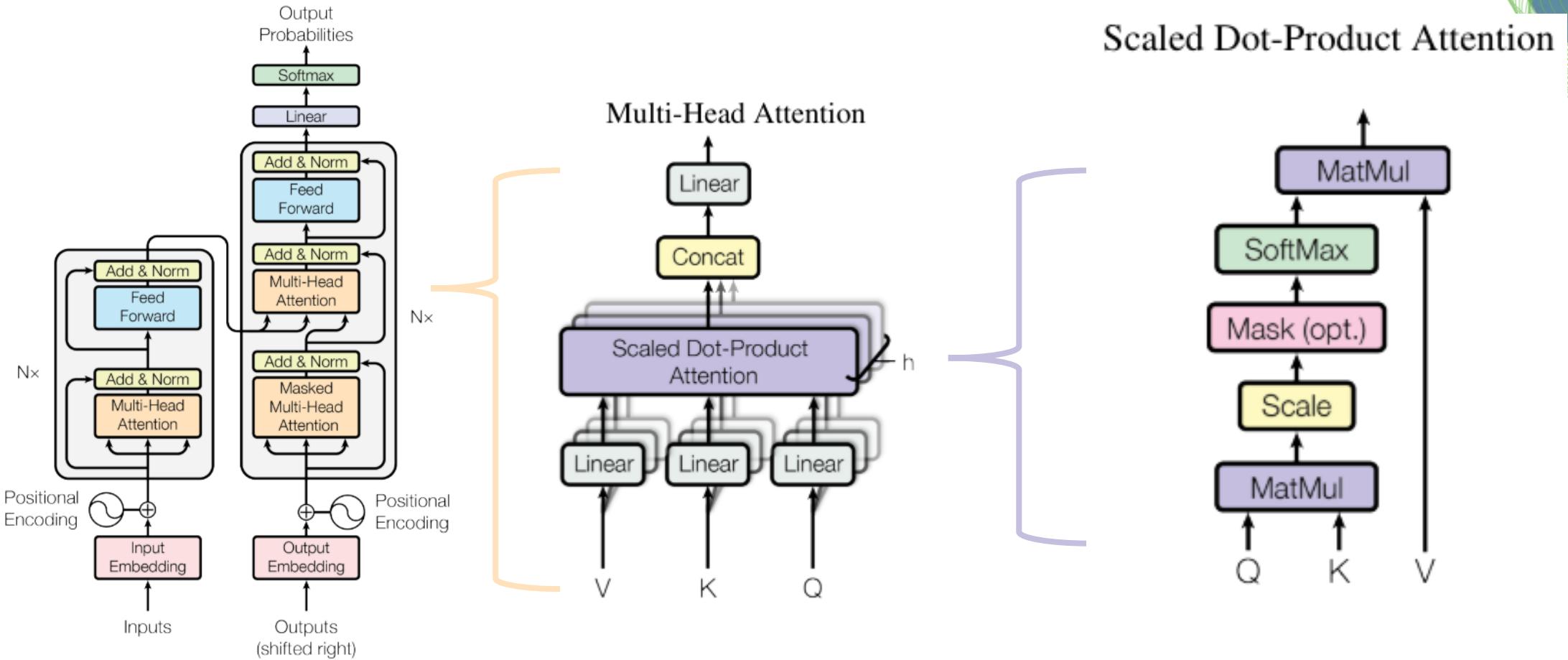
The position of the word in the sentence / context may have influence on its meaning. The position is encoded in a vector as well.

The cat sits next to a **microscope**.

Next to the **microscope** there is a cat.



# Attention is all you need



Source: Vaswani et al (2017)  
<https://arxiv.org/abs/1706.03762>

# Scaled dot-product attention

Attention score: How much related are two words?

Query: For which word are we calculating attention?

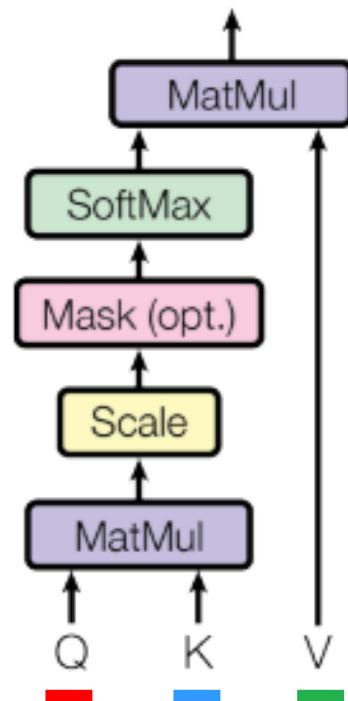
Key: To which word are we calculating attention

Value: Relevance of the query-key relationship

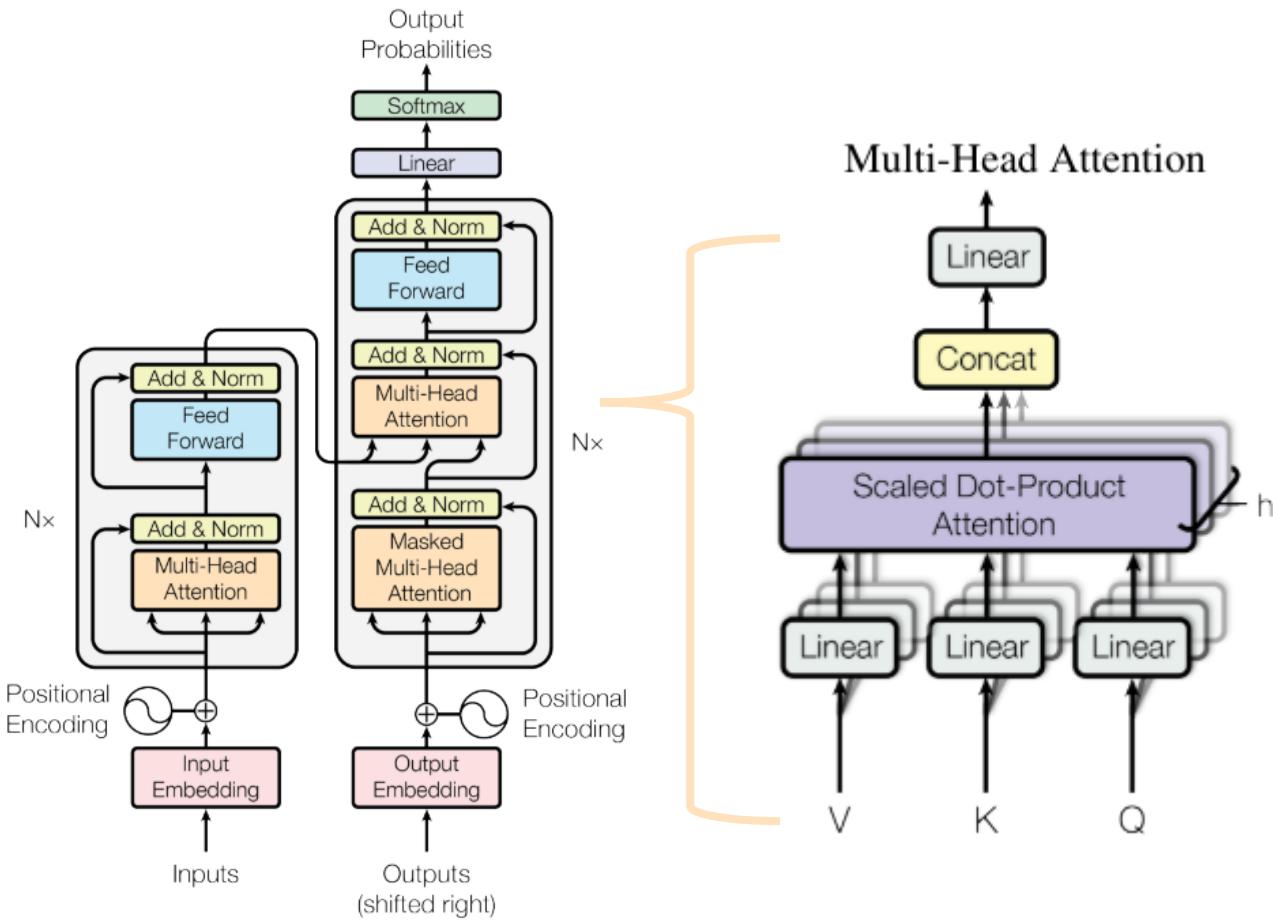
The cat is black and white.  
Relevance value: 0.1

The cat is meowing.  
Relevance value: 0.9

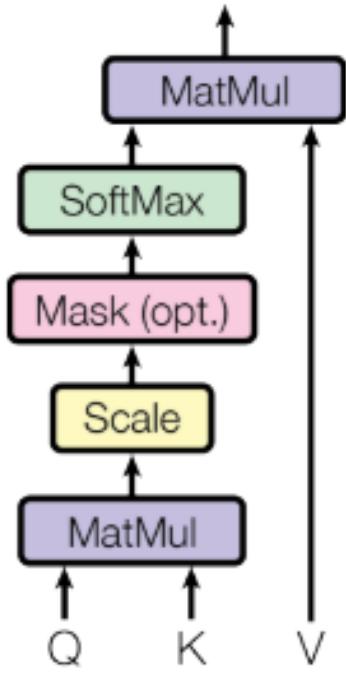
## Scaled Dot-Product Attention



# Attention is all you need

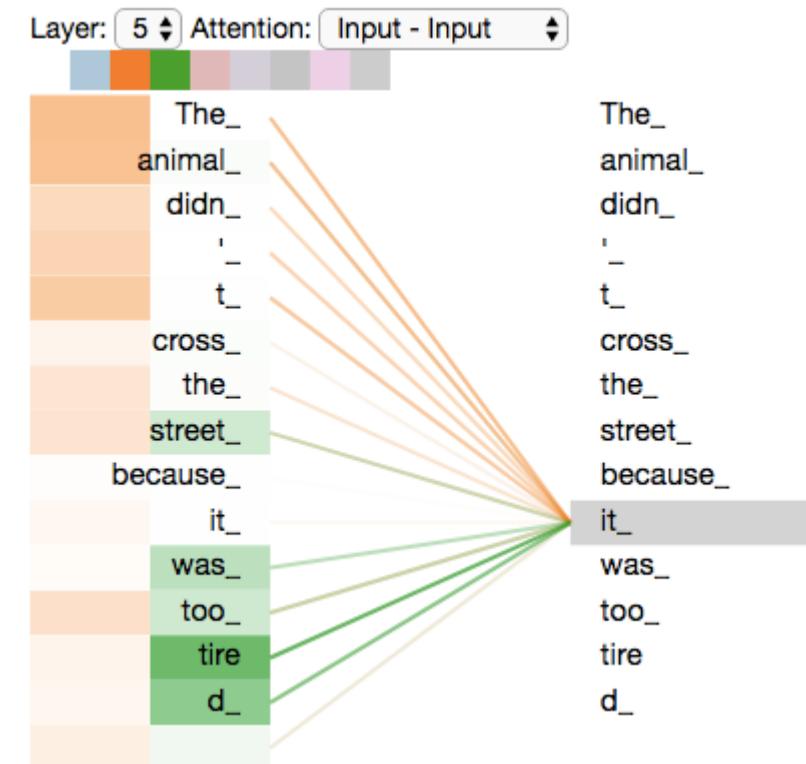
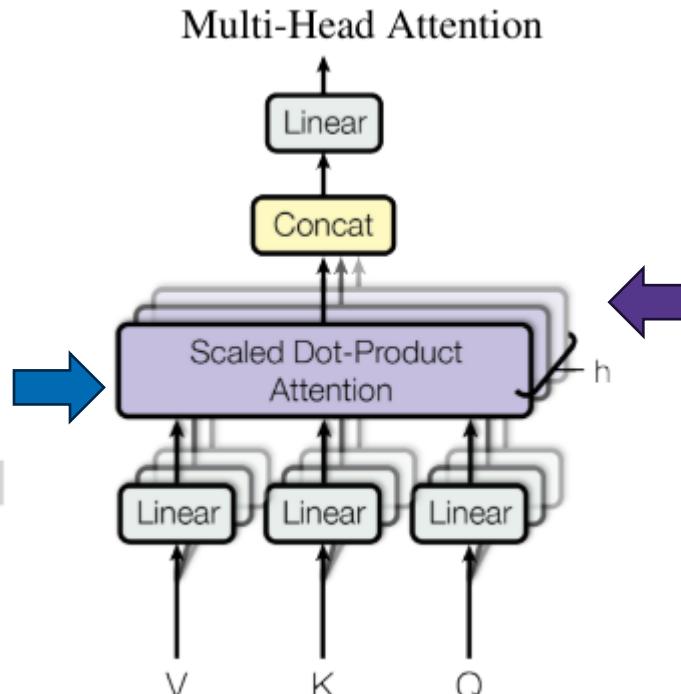
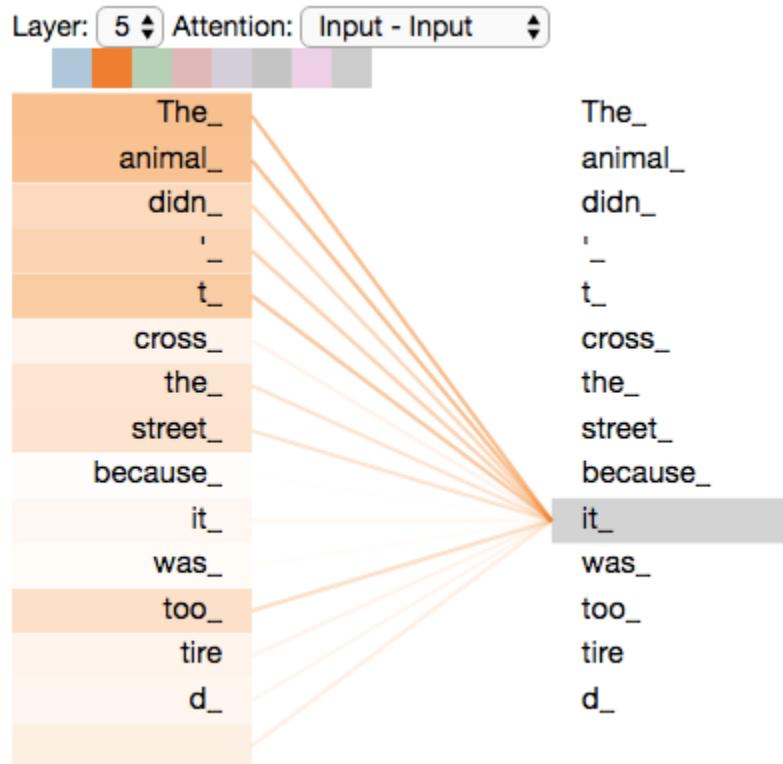


### Scaled Dot-Product Attention

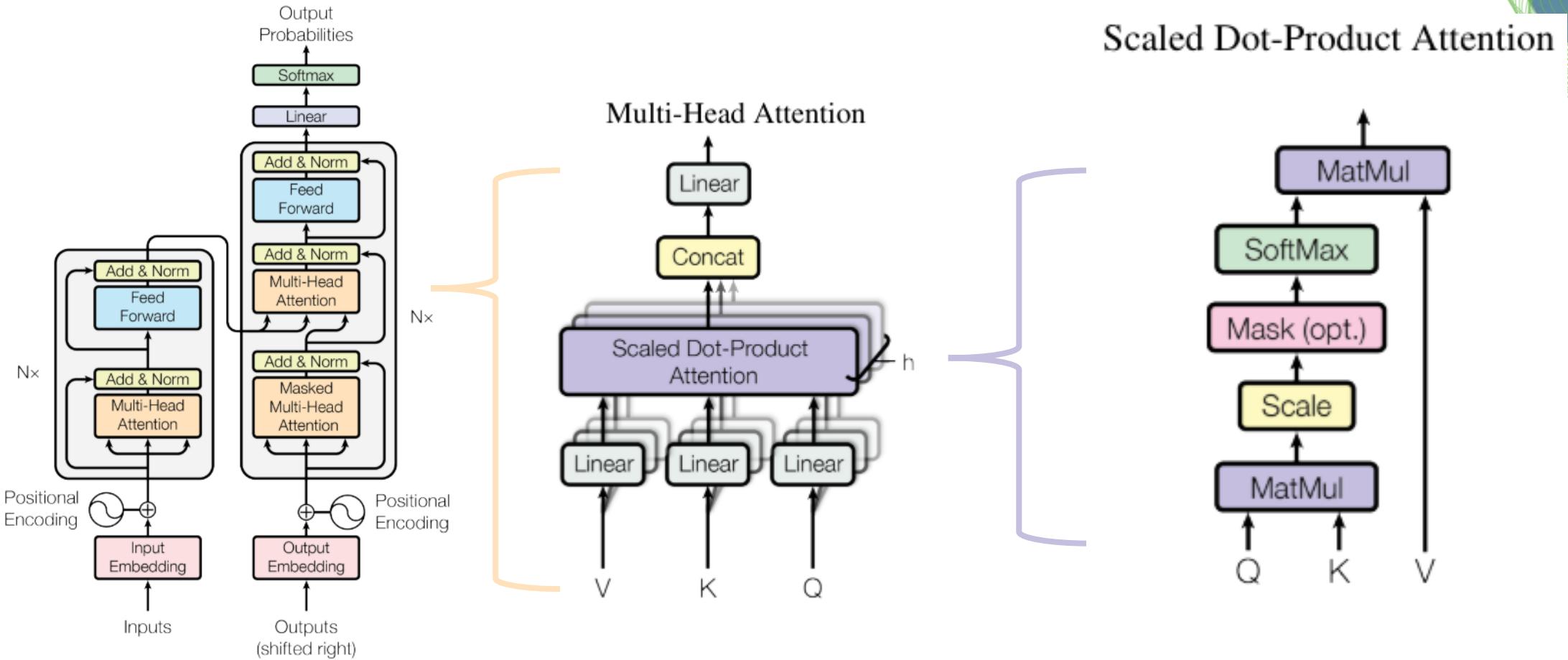


# Multi-head attentions

Multiple aspects represented by multiple attention heads



# Attention is all you need

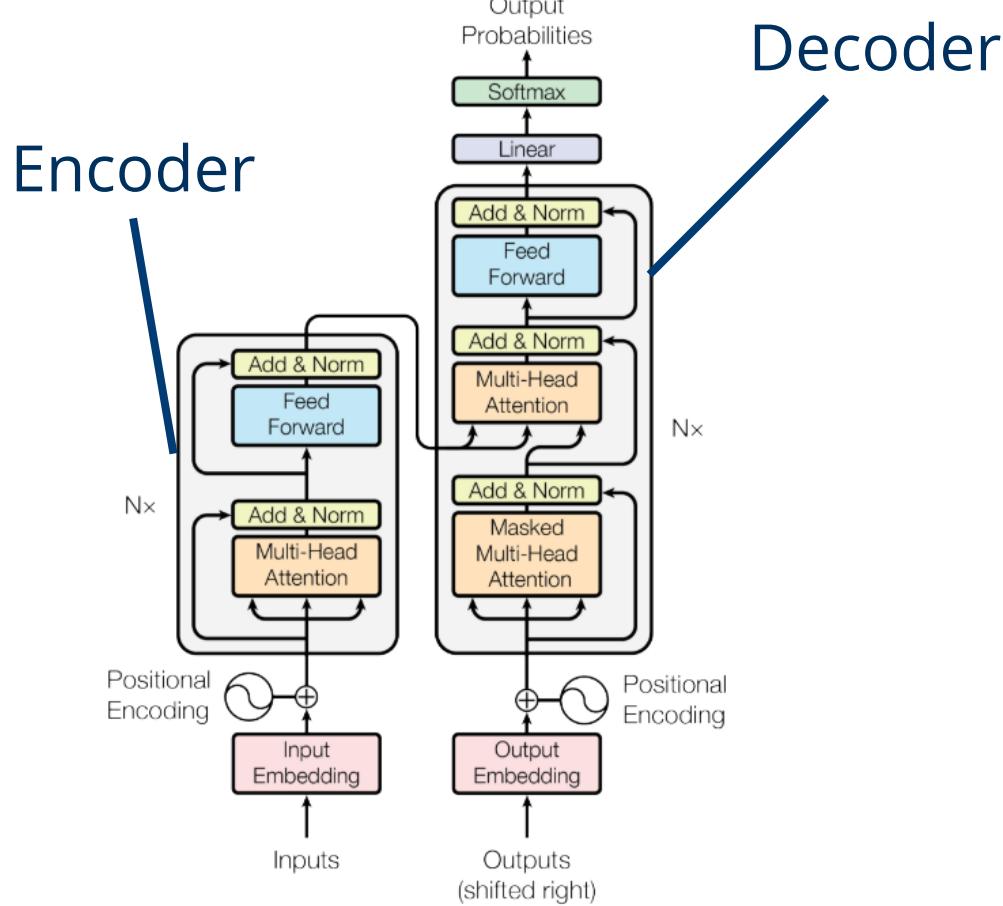


Source: Vaswani et al (2017)  
<https://arxiv.org/abs/1706.0376>

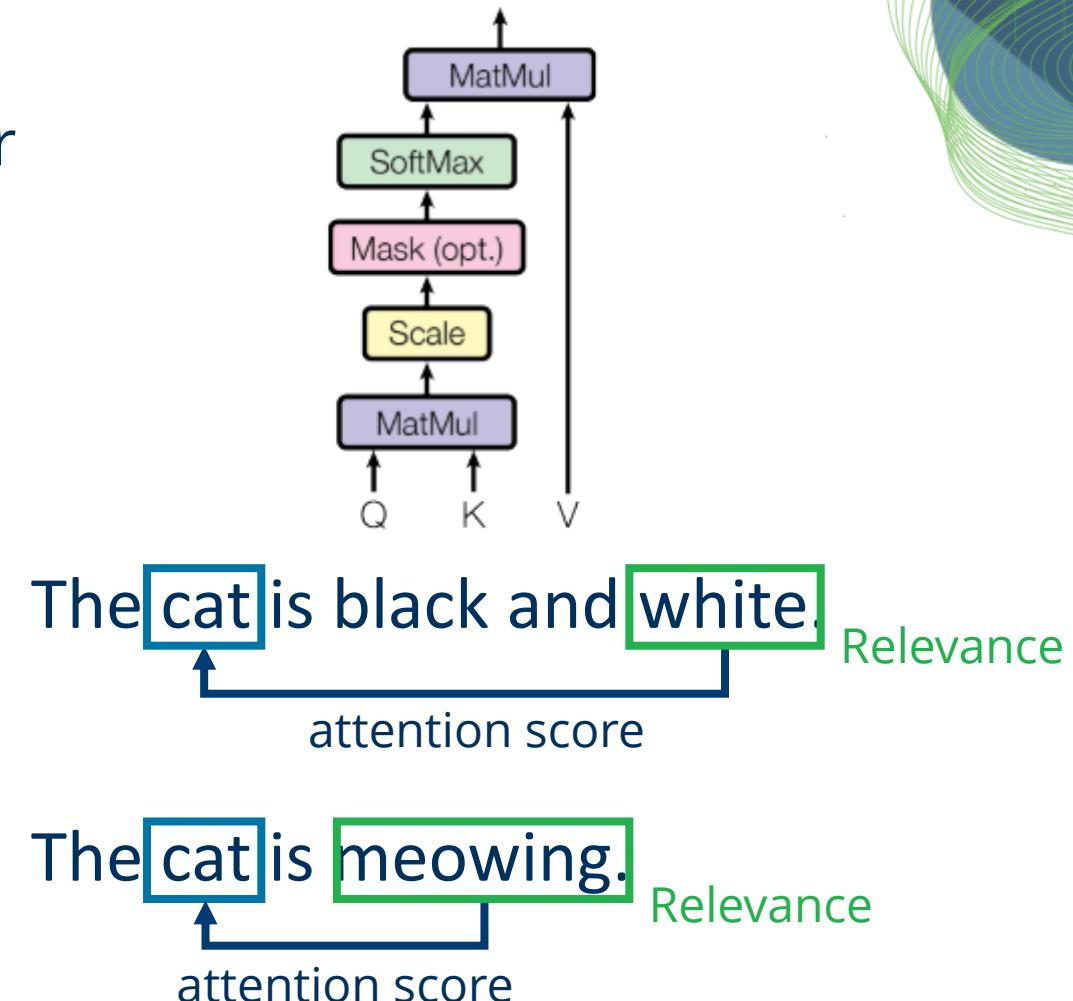
33

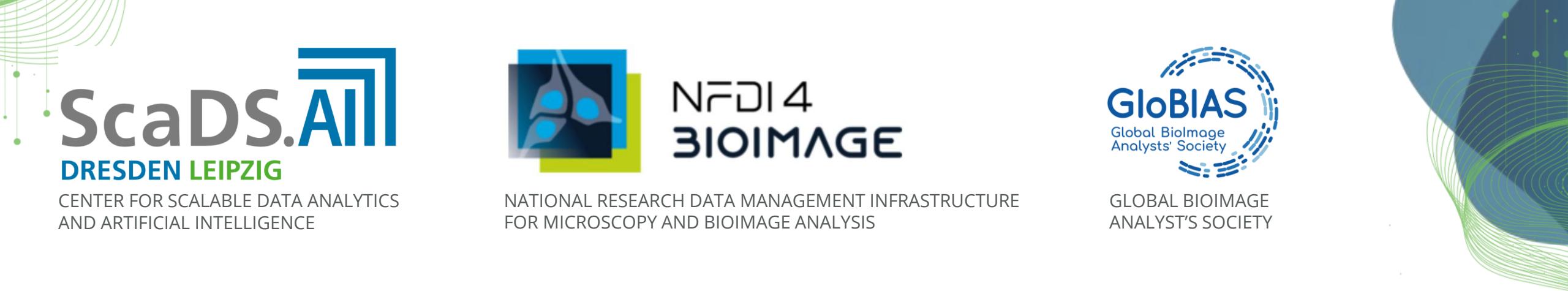
# Attention is all you need

## Summary



Scaled Dot-Product Attention



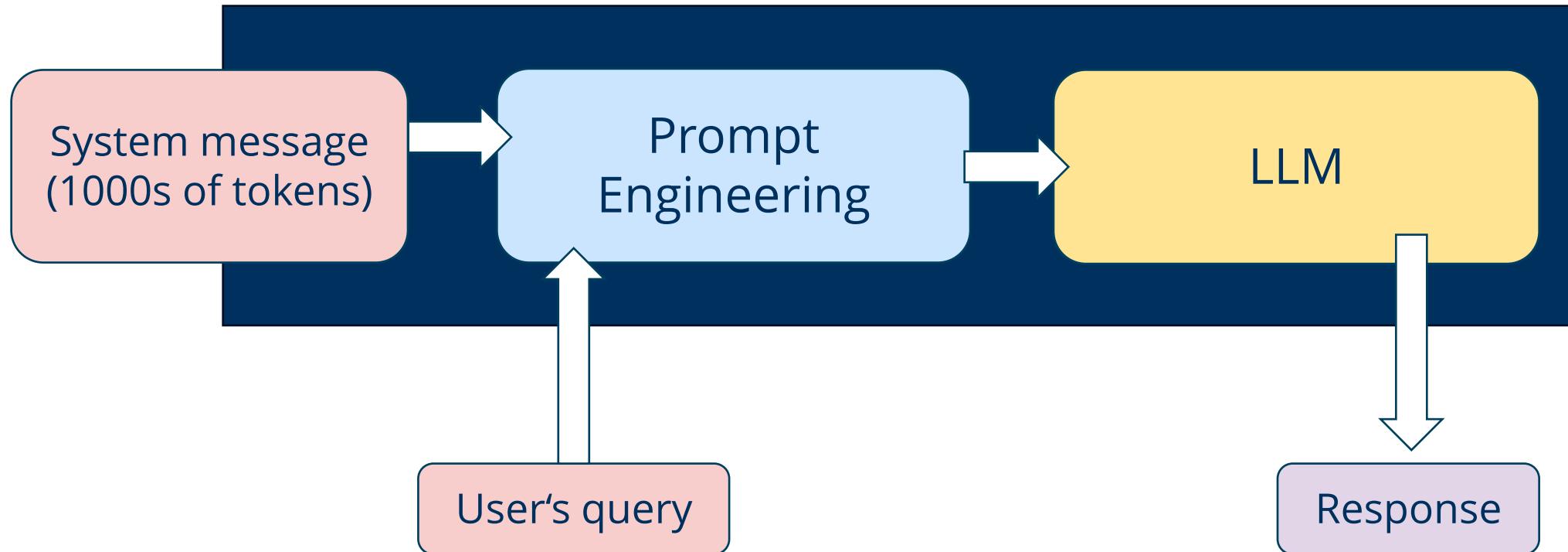


# Prompt Engineering

# Long-context prompting

Also known as *in-context learning*

Context with plenty of knowledge provided in system message



# Prompt Engineering

*Modify a prompt until it works.*

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

%%prompt

You are a professional academic consultant who gives advice to students **for** writing their thesis.

Give these hints:

- \* Submit the thesis registration form only **if** you're sure you can finish writing in 5 months.
- \* Start writing your thesis early, e.g. the fundamentals chapter can be written **from** day one.
- \* Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

Here are some important hints to consider while writing your thesis:

1. Evaluate your readiness: Before submitting the thesis registration form, carefully assess your commitment and availability. Ensure that you are confident in completing your thesis within the given timeframe of five months. This will prevent unnecessary stress and help you maintain a focused mindset throughout the writing process.
2. Begin early: Starting your thesis early **is** crucial. Begin by working on the fundamentals chapter, as it lays the foundation for your entire thesis. By initiating this chapter from day one, you will establish a good writing routine and make steady progress as you move forward.
3. Plan ahead: Create a realistic schedule to guide your writing process. Break down the work into smaller tasks, setting deadlines for each chapter or section. **This will** help you maintain a structured approach and make the workload more manageable. Be sure to allocate time for research, data analysis, and revisions.
4. Aim to finish ahead of time: Plan to complete your thesis writing at least one month before the official deadline. This buffer time will allow you to thoroughly review and revise your work before final submission. It will also provide an opportunity to seek feedback from professors or peers, improving the overall quality of your thesis.

# Prompt Engineering

*Modify a prompt until it works.*

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

`%%prompt`

You are a professional academic consultant who gives advice to students **for** writing their thesis.  
Keep it short.

Give these hints:

- \* Submit the thesis registration form only **if** you're sure you can finish writing in 5 months.
- \* Start writing your thesis early, e.g. the fundamentals chapter can be written **from** day one.
- \* Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

As a professional academic consultant, here are some helpful hints for writing your thesis:

1. Only submit the thesis registration form if you are confident about completing it within the given timeframe of 5 months.
2. Begin writing your thesis early to ensure a smooth process. For instance, you can start with the fundamentals chapter right from day one.
3. Plan your schedule effectively to finish writing your thesis at least a month before the submission deadline. This will allow ample time for reading, revising, and receiving feedback.

By following these recommendations, you'll have a better chance of successfully completing your thesis on time and with improved quality. Good luck!

# Prompt Engineering

*Modify a prompt until it works.*

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

%%prompt

You are a professional academic consultant who gives advice to students **for** writing their thesis.  
Keep it short.

Write like Yoda would.

Give these hints:

- \* Submit the thesis registration form only **if** you're sure you can finish writing in 5 months.
- \* Start writing your thesis early, e.g. the fundamentals chapter can be written **from** day one.
- \* Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

Hints for thesis writing, I have. Follow them, you must.

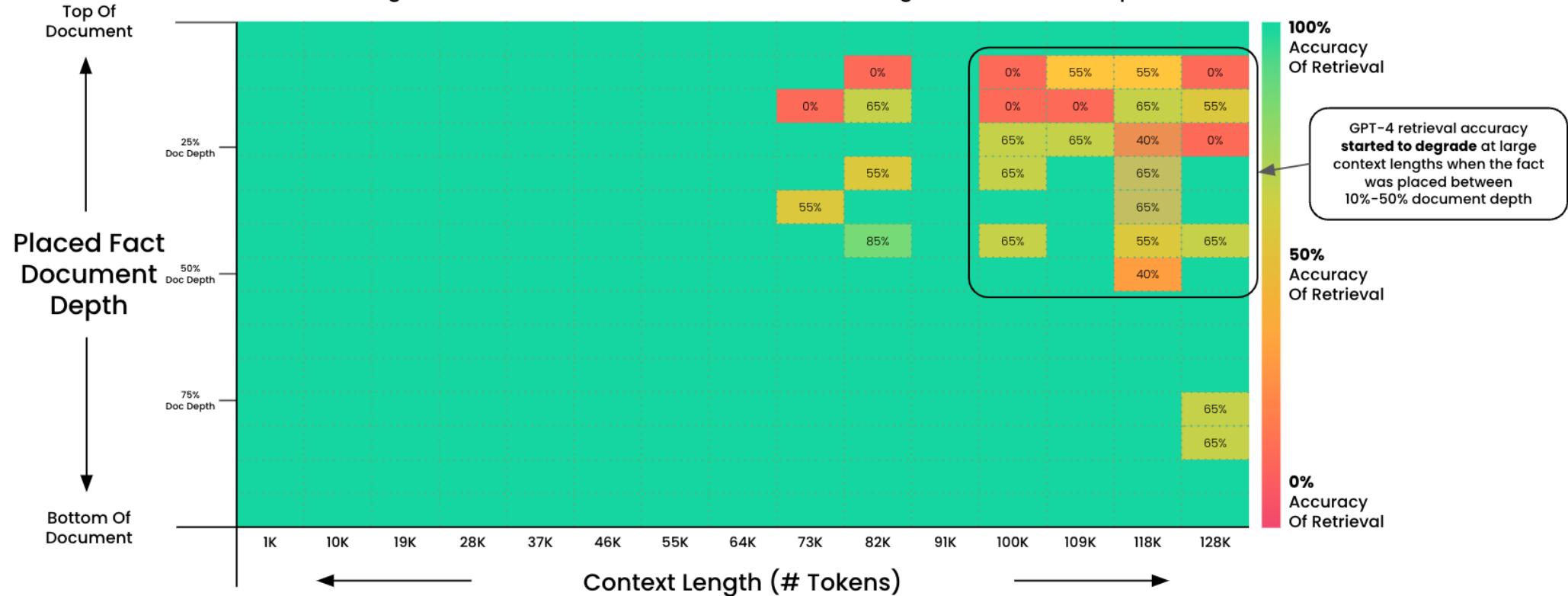
- Only submit the thesis registration form, if in 5 months, complete writing, you are sure.
- From day one, start writing your thesis, you must. Begin with the fundamentals chapter.
- A month before the deadline, finish writing, you should. Time to read and provide feedback, it will give you.

Follow these hints, you must. Successful thesis writing, you will achieve. Hmmmm.

# Prompt engineering

## Pressure Testing GPT-4 128K via "Needle In A HayStack"

Asking GPT-4 To Do Fact Retrieval Across Context Lengths & Document Depth



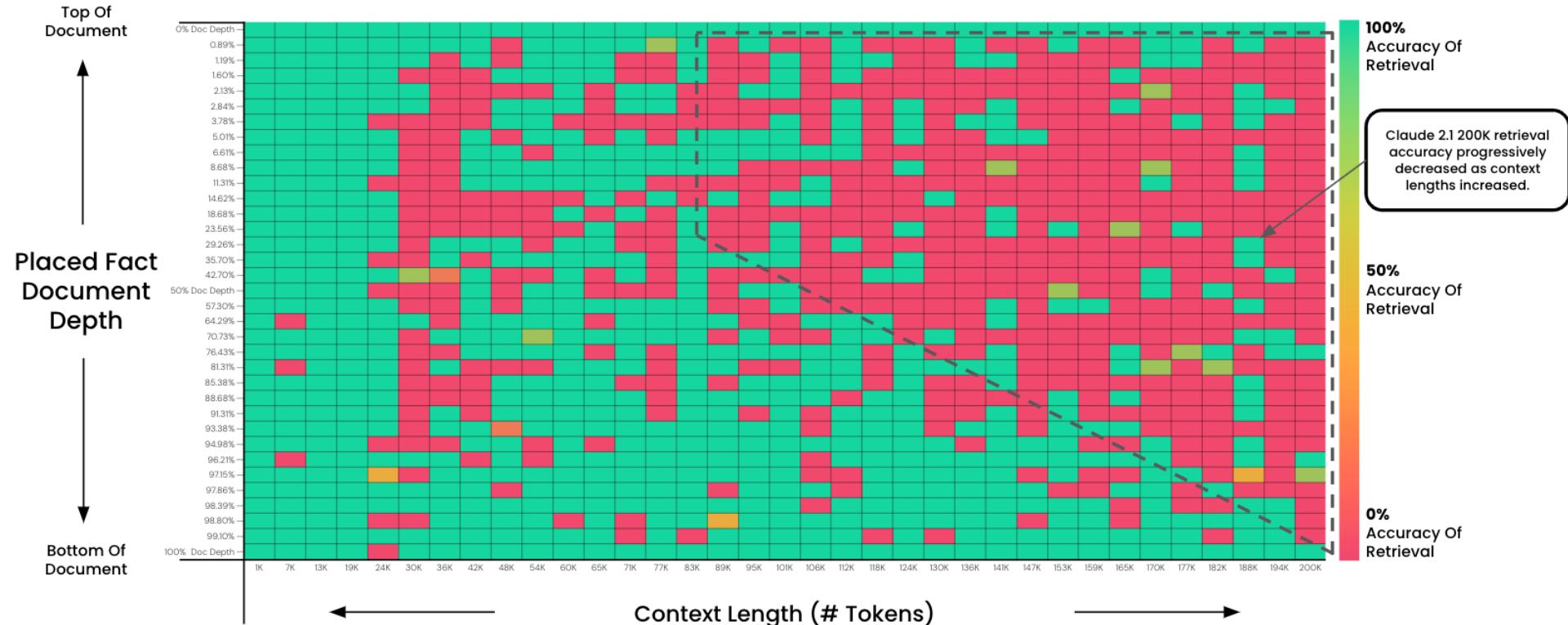
### Goal: Test GPT-4 Ability To Retrieve Information From Large Context Windows

A fact was placed within a document. GPT-4 (1106-preview) was then asked to retrieve it. The output was evaluated for accuracy. This test was run at 15 different document depths (top > bottom) and 15 different context lengths (1K > 128K tokens). 2x tests were run for larger contexts for a larger sample size.

# Prompt engineering

## Pressure Testing Claude-2.1 200K via "Needle In A HayStack"

Asking Claude 2.1 To Do Fact Retrieval Across Context Lengths & Document Depth



### Goal: Test Claude 2.1 Ability To Retrieve Information From Large Context Windows

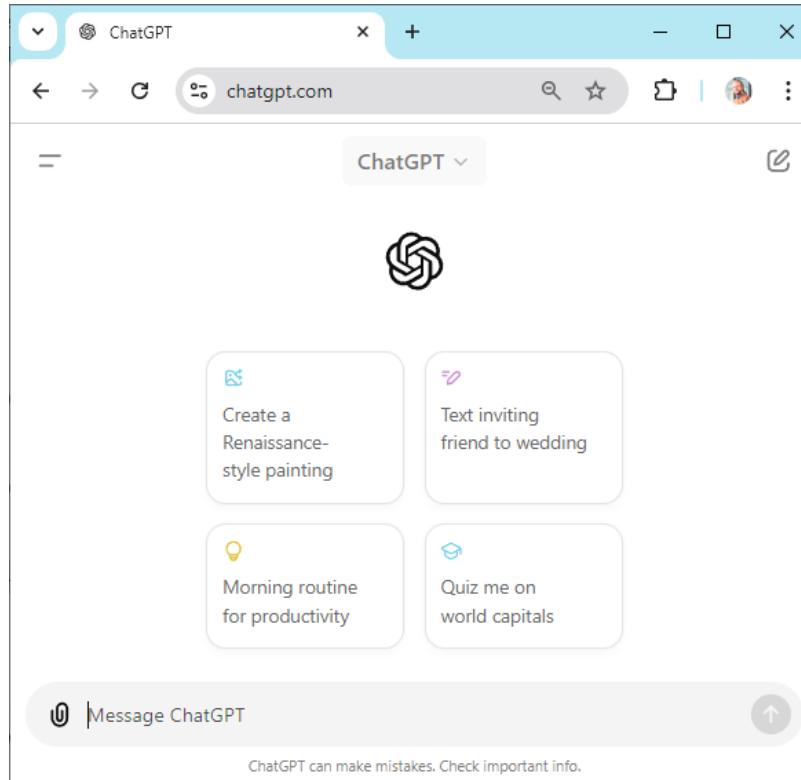
A fact was placed within a document. Claude 2.1 (200K) was then asked to retrieve it. The output was evaluated (with GPT-4) for accuracy. This test was run at 35 different document depths (top > bottom) and 35 different context lengths (1K > 200K tokens).

Document Depths followed a sigmoid distribution

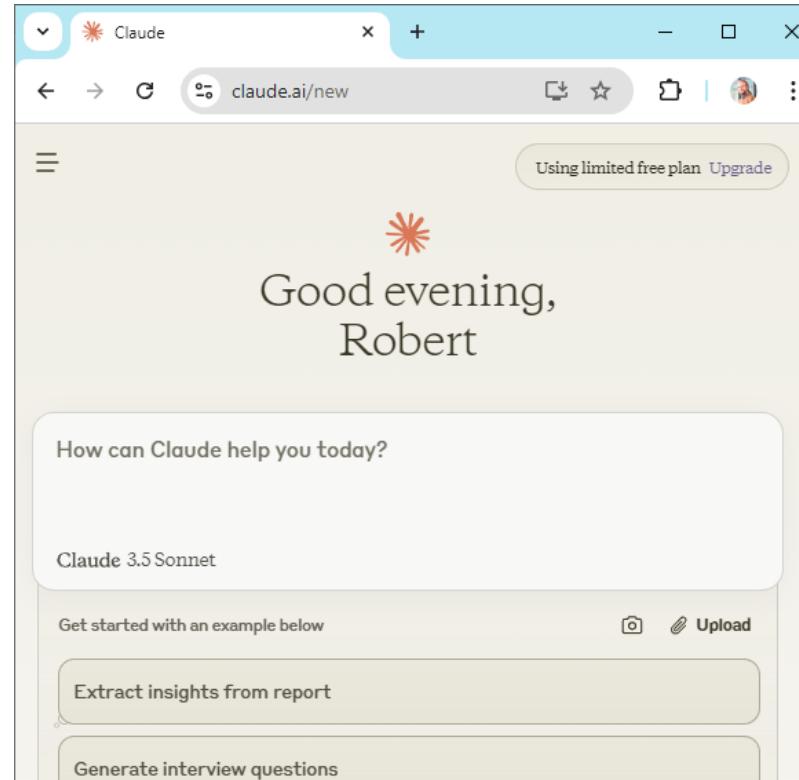
# LLMs are everywhere

# LLMs are everywhere

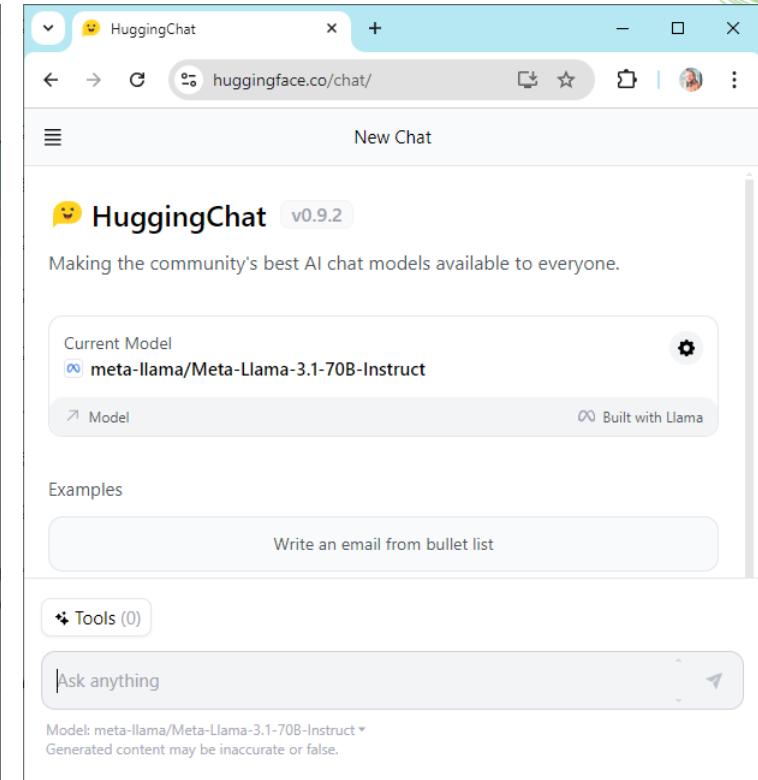
ChatGPT



Claude

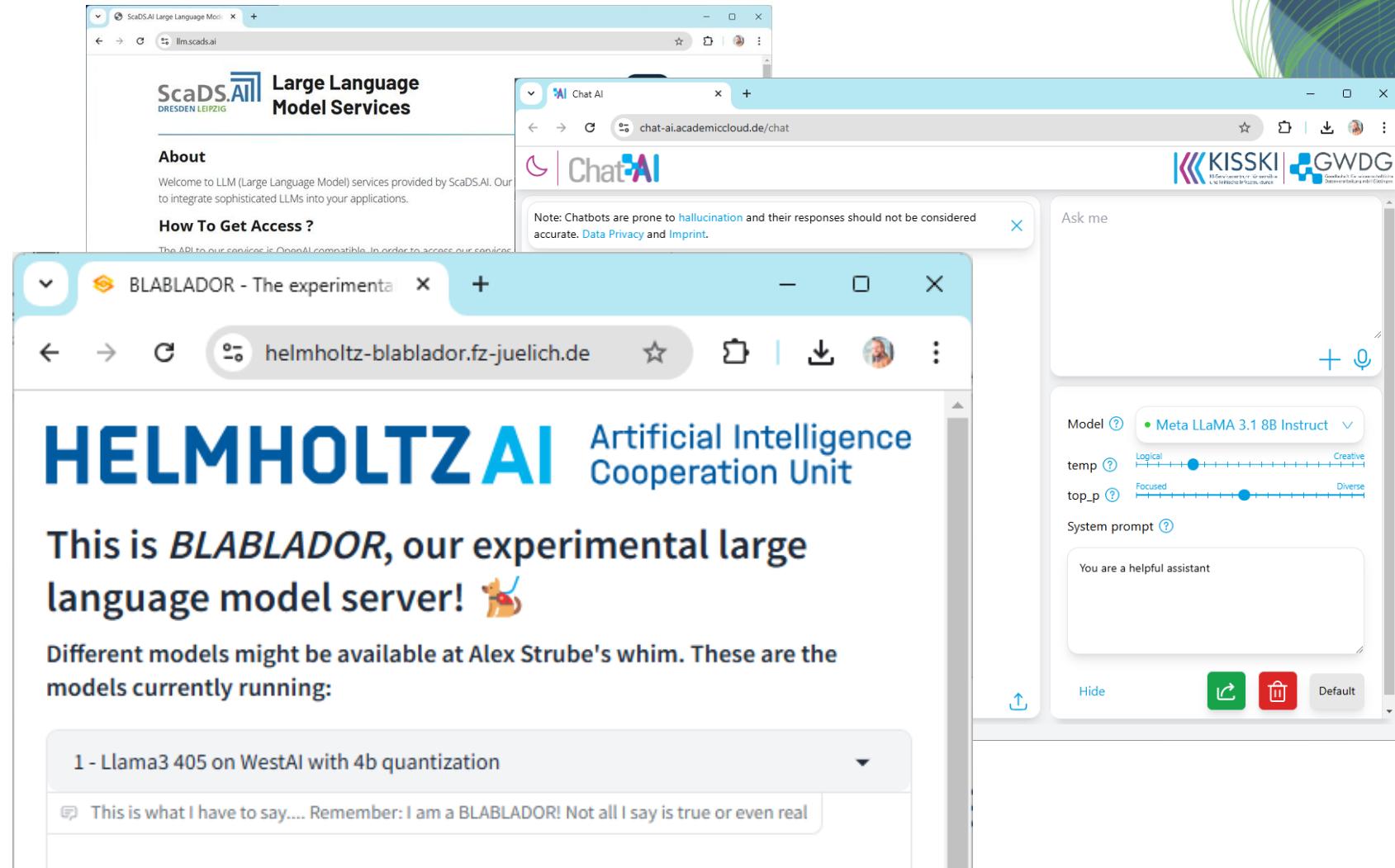


Open models via  
Huggingface



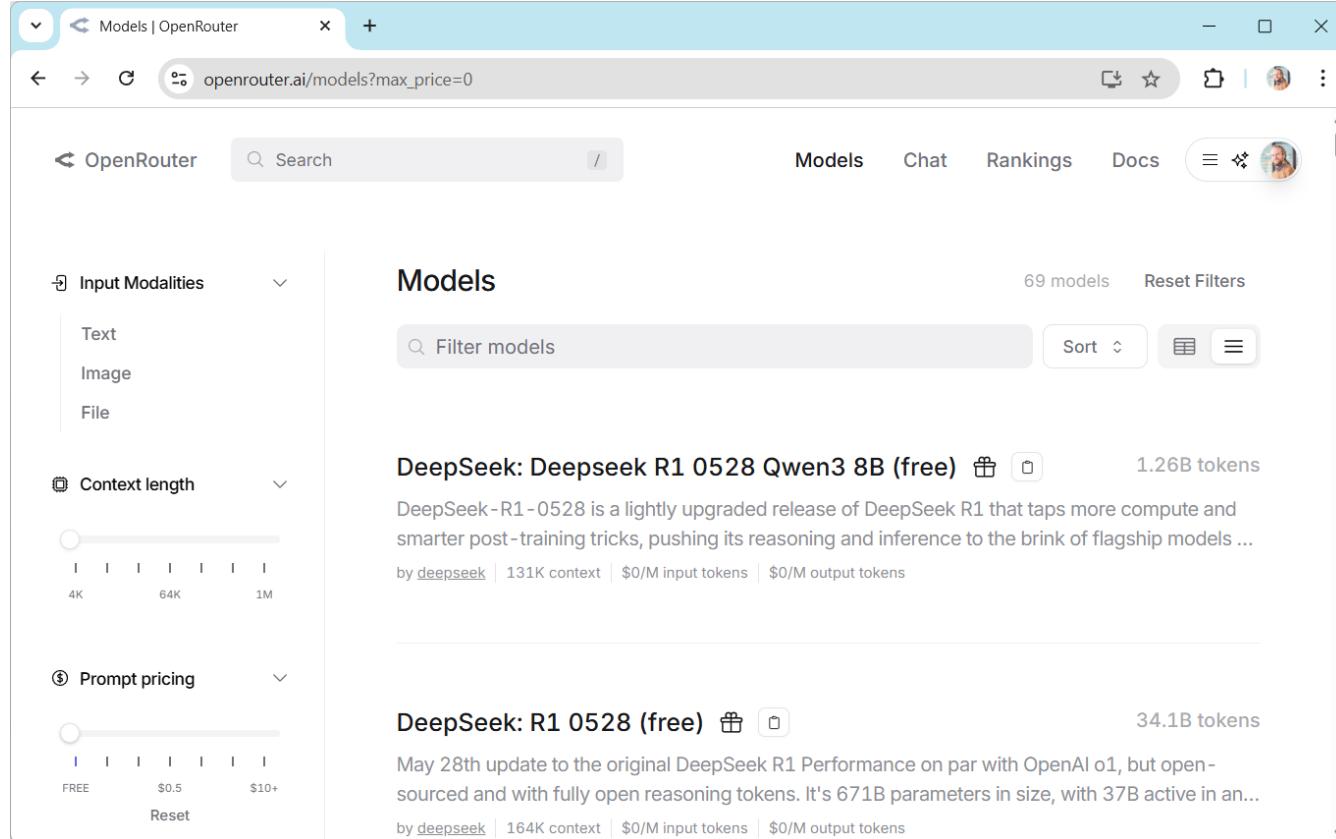
# Institutional AI Infrastructure

- Many institutes have a local LLM Server
  - Great for teaching
  - Privacy preserving
- Cross-institutional infrastructure, in Germany:
  - KISSKI ChatAI
  - Helmholtz Blablador



# Commercial AI Infrastructure

Github Models, OpenRouter, Huggingface,...



The screenshot shows the OpenRouter AI Model Marketplace. The interface includes a sidebar with filters for Input Modalities (Text, Image, File), Context length (4K, 64K, 1M), and Prompt pricing (FREE, \$0.5, \$10+). The main area displays a list of models. The first item is "DeepSeek: Deepseek R1 0528 Qwen3 8B (free)" with 1.26B tokens. The second item is "DeepSeek: R1 0528 (free)" with 34.1B tokens.

Models | OpenRouter

openrouter.ai/models?max\_price=0

Models Chat Rankings Docs

Input Modalities: Text, Image, File

Context length: 4K, 64K, 1M

Prompt pricing: FREE, \$0.5, \$10+

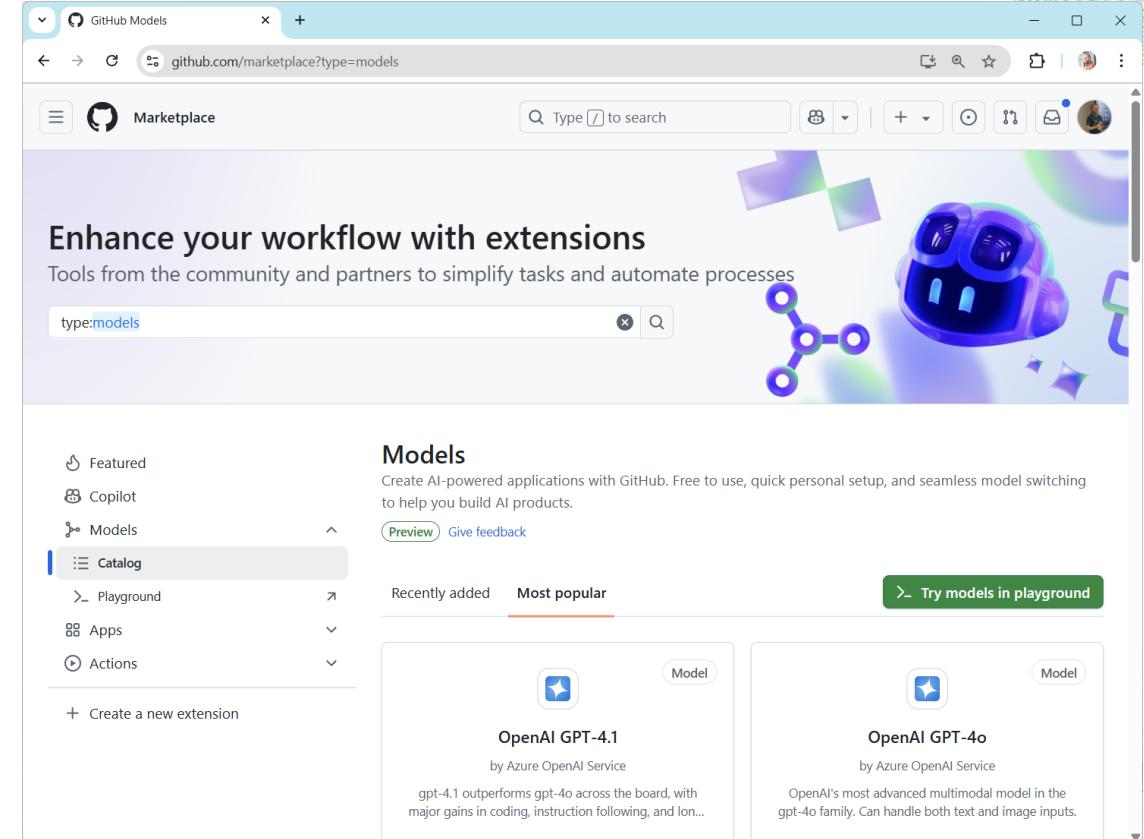
Models

69 models Reset Filters

Filter models Sort

DeepSeek: Deepseek R1 0528 Qwen3 8B (free) 1.26B tokens  
DeepSeek-R1-0528 is a lightly upgraded release of DeepSeek R1 that taps more compute and smarter post-training tricks, pushing its reasoning and inference to the brink of flagship models ... by deepseek | 131K context | \$0/M input tokens | \$0/M output tokens

DeepSeek: R1 0528 (free) 34.1B tokens  
May 28th update to the original DeepSeek R1 Performance on par with OpenAI o1, but open-sourced and with fully open reasoning tokens. It's 671B parameters in size, with 37B active in an... by deepseek | 164K context | \$0/M input tokens | \$0/M output tokens



The screenshot shows the GitHub Marketplace section for AI models. It features a search bar with "type:models". The sidebar includes categories like Featured, Copilot, Models (which is selected), Catalog, Playground, Apps, and Actions. The main area shows a catalog of AI models, with "OpenAI GPT-4.1" and "OpenAI GPT-4o" listed as examples.

GitHub Models

github.com/marketplace?type=models

Marketplace

Type / to search

Enhance your workflow with extensions

Tools from the community and partners to simplify tasks and automate processes

type:models

Models

Create AI-powered applications with GitHub. Free to use, quick personal setup, and seamless model switching to help you build AI products.

Preview Give feedback

Recently added Most popular

Try models in playground

OpenAI GPT-4.1

by Azure OpenAI Service

gpt-4.1 outperforms gpt-4o across the board, with major gains in coding, instruction following, and lon...

OpenAI GPT-4o

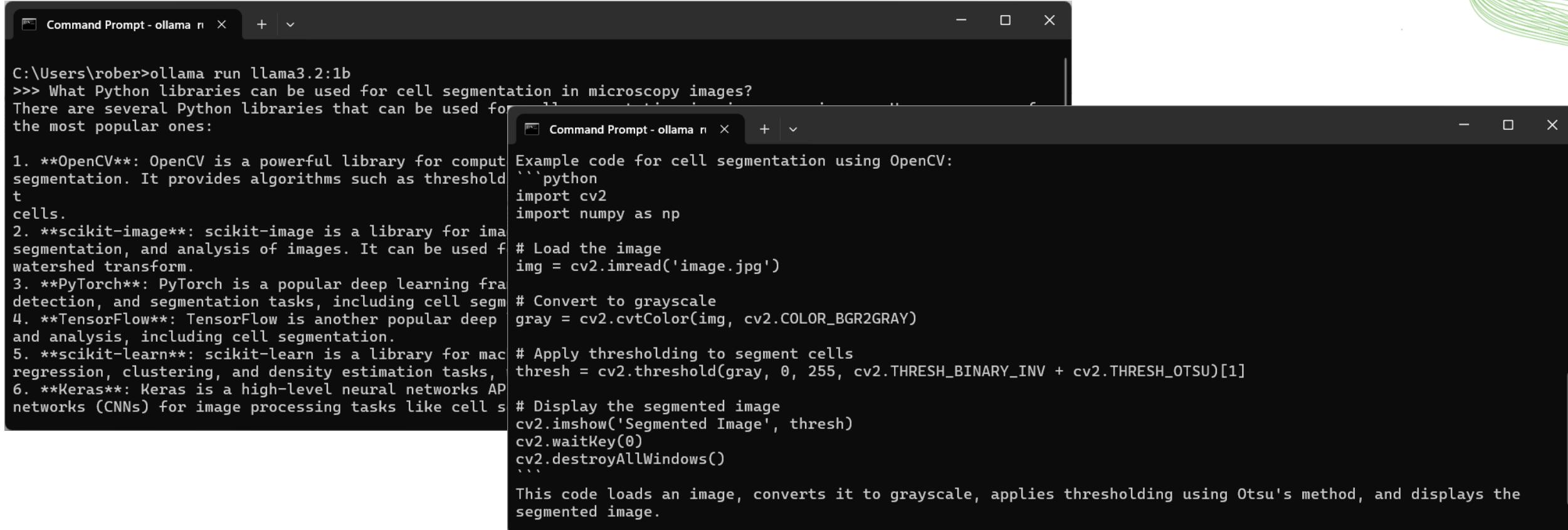
by Azure OpenAI Service

OpenAI's most advanced multimodal model in the gpt-4o family. Can handle both text and image inputs.

45

# Local AI infrastructure

## For Small Language Models only



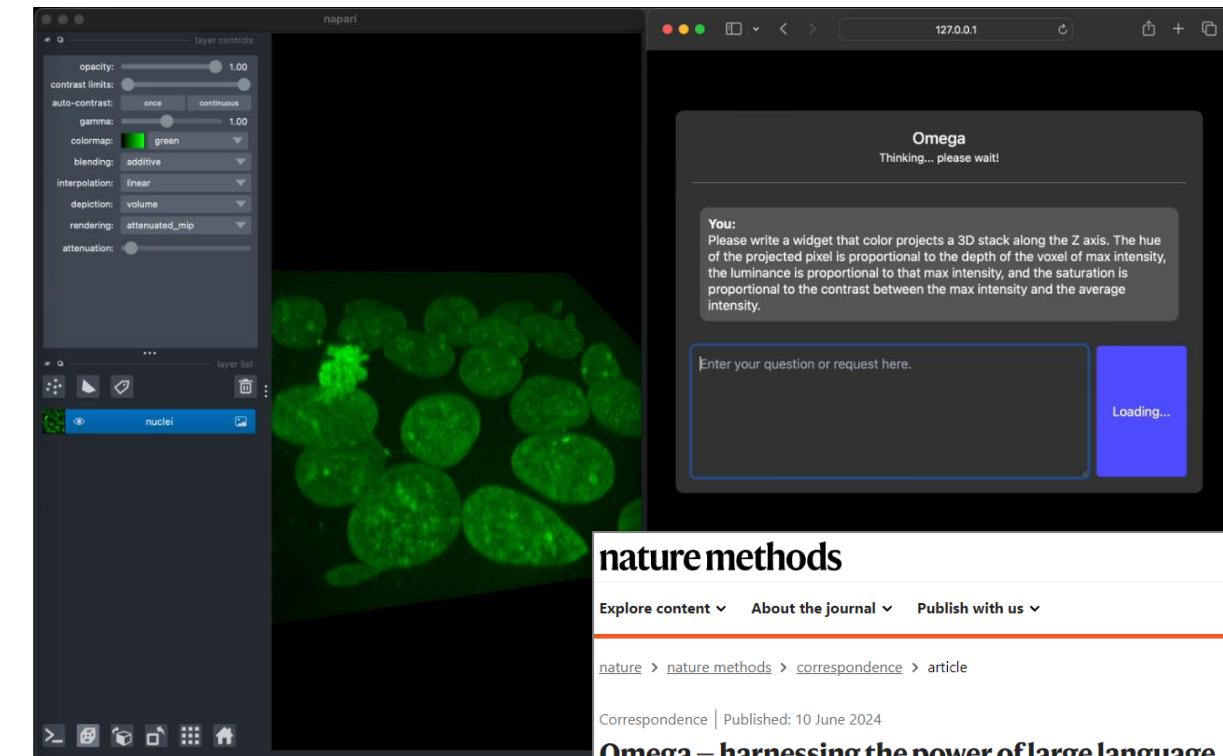
The image shows two overlapping command prompt windows. The left window displays a list of Python libraries for cell segmentation, generated by the AI model. The right window shows example code for cell segmentation using OpenCV, also generated by the AI model.

C:\Users\rober>ollama run llama3.2:1b  
>>> What Python libraries can be used for cell segmentation in microscopy images?  
There are several Python libraries that can be used for cell segmentation. Some of the most popular ones:  
1. \*\*OpenCV\*\*: OpenCV is a powerful library for computer vision and image processing. It provides algorithms such as thresholding, feature detection, and segmentation tasks, including cell segmentation.  
2. \*\*scikit-image\*\*: scikit-image is a library for image processing, analysis, and visualization. It can be used for watershed transform.  
3. \*\*PyTorch\*\*: PyTorch is a popular deep learning framework for detection, and segmentation tasks, including cell segmentation.  
4. \*\*TensorFlow\*\*: TensorFlow is another popular deep learning framework for detection, and segmentation, including cell segmentation.  
5. \*\*scikit-learn\*\*: scikit-learn is a library for machine learning, regression, clustering, and density estimation tasks.  
6. \*\*Keras\*\*: Keras is a high-level neural networks API built on top of TensorFlow. It is used for image processing tasks like cell segmentation.

Example code for cell segmentation using OpenCV:  
```python  
import cv2  
import numpy as np  
  
# Load the image  
img = cv2.imread('image.jpg')  
  
# Convert to grayscale  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
# Apply thresholding to segment cells  
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)[1]  
  
# Display the segmented image  
cv2.imshow('Segmented Image', thresh)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
```  
This code loads an image, converts it to grayscale, applies thresholding using Otsu's method, and displays the segmented image.

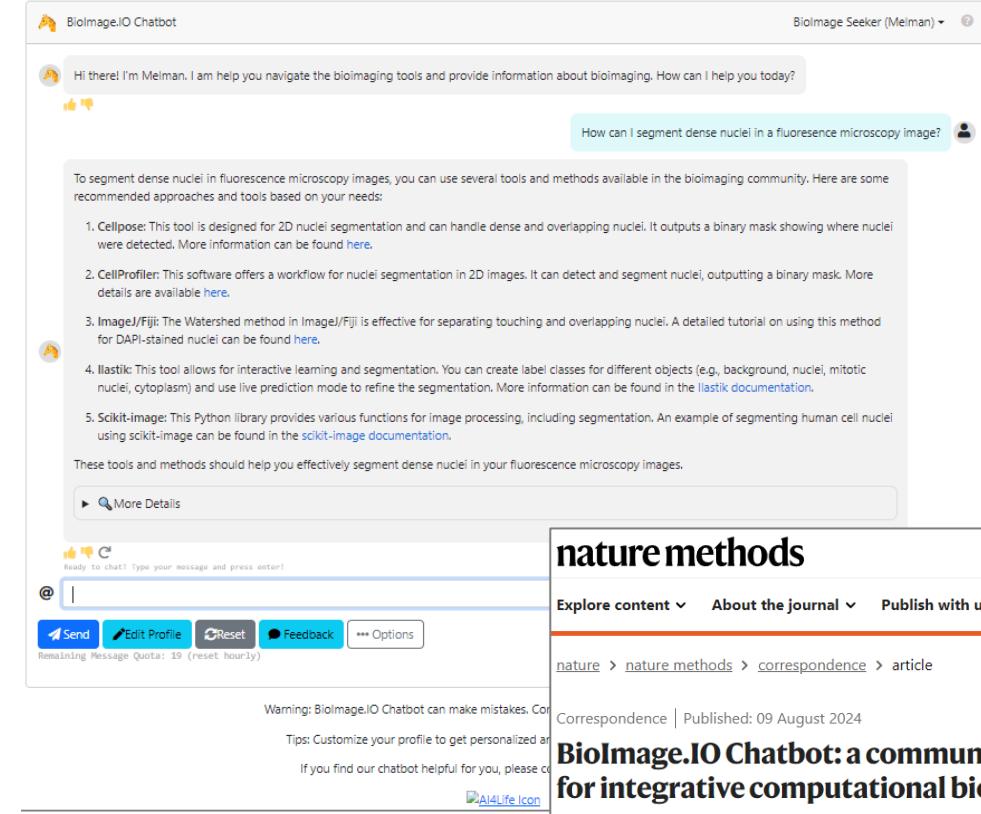
# LLMs are everywhere

Napari-chatGPT / Omega



<https://github.com/royerlab/napari-chatgpt>  
<https://www.nature.com/articles/s41592-024-02310-w>

## Bioimage-io ChatBot



<https://bioimage.io/chat/>  
<https://www.nature.com/articles/s41592-024-02370-y>

# LLMs are everywhere

aider

```
macbook$ aider demo.py
Added demo.py to the chat
Using git repo: .git

demo.py> add a name param to the `greeting` function. add all the types.

I will update the greeting function to accept a name parameter and include type
hints for the parameter and the return type.

demo.py
<<<<< ORIGINAL
def greeting():
    print("Hello, world!")

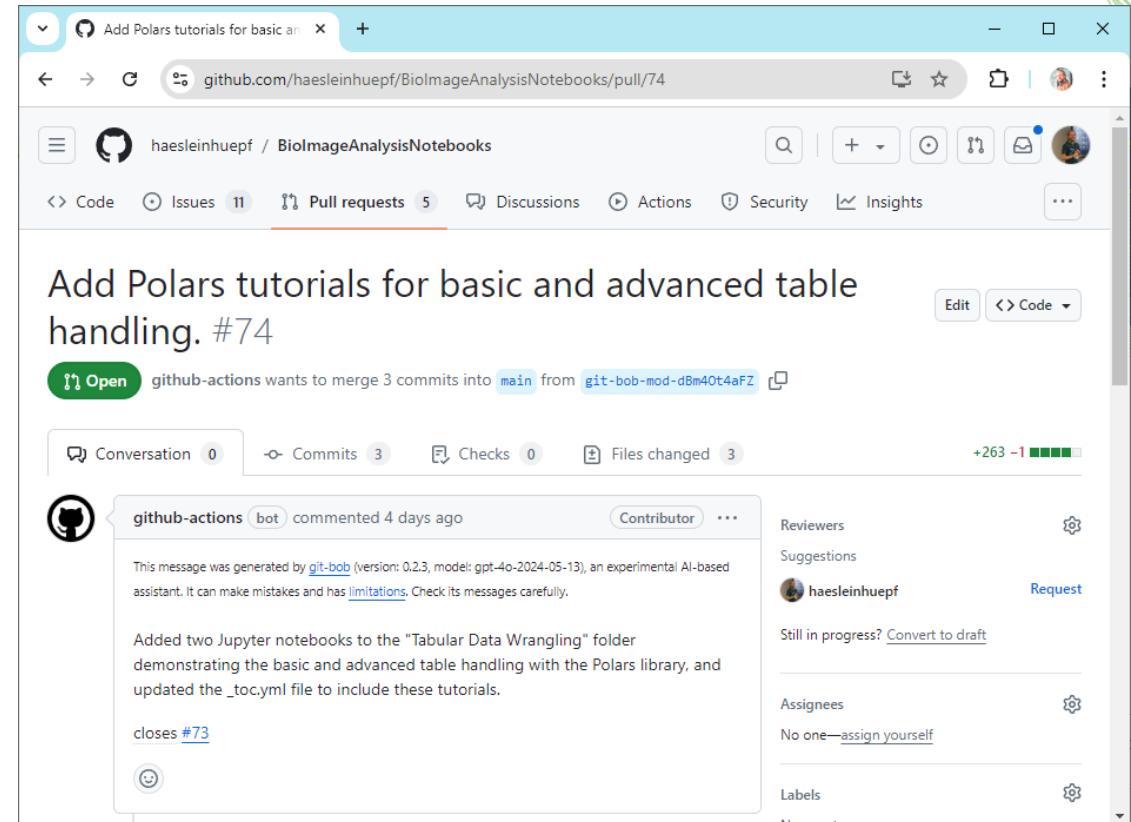
=====

def greeting(name: str) -> None:
    print(f"Hello, {name}!")
>>>>> UPDATED

Applied edit to demo.py
Commit 9237455 aider: Updated greeting function to accept a name parameter and
include type hints.

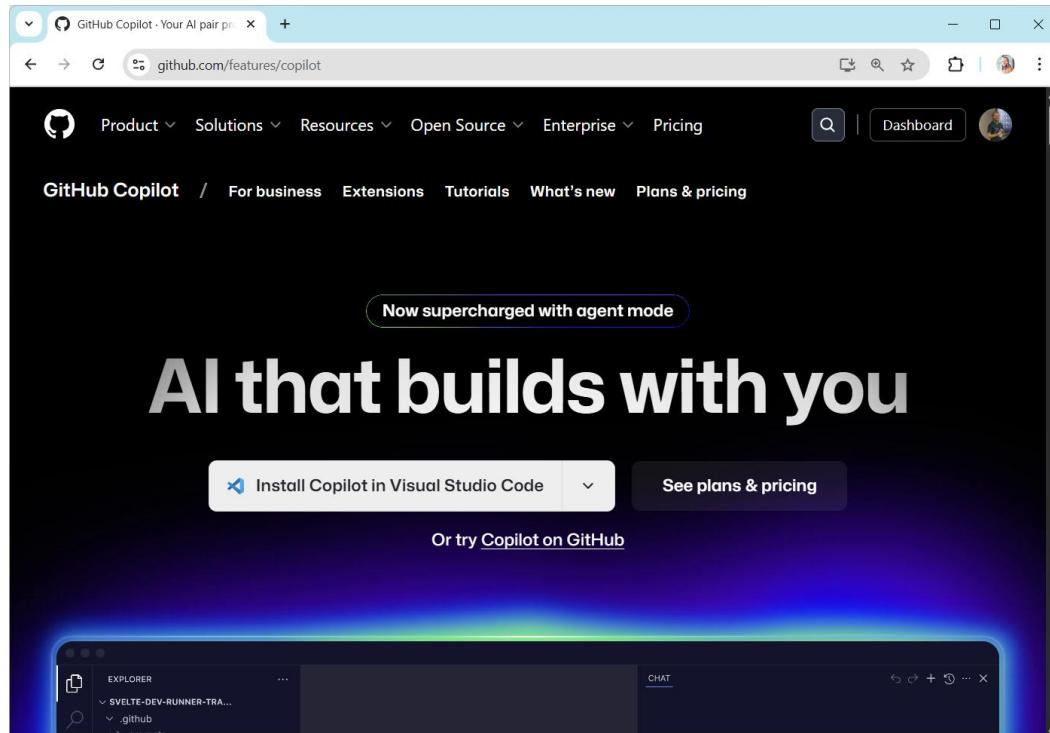
demo.py>
```

git-bob



# LLMs are everywhere

## Github Copilot



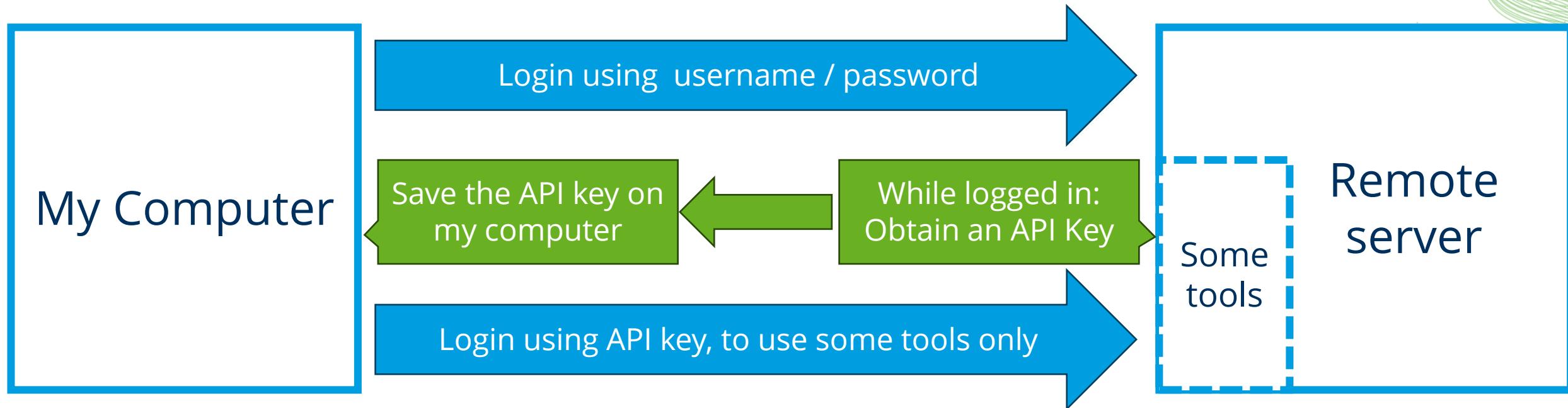
## Cursor

A screenshot of the Cursor AI interface. On the left, there's a file tree for "BIOIMAGEANALYSISNOTEBOOKS" containing various Jupyter notebooks and Python files like "my\_library.py". In the main editor area, there's a Python script with a function definition for "square". A tooltip or completion suggestion is shown over the "square" function, providing a docstring and parameters information. The code completion interface includes buttons for "Accept", "Reject", and "Follow-up instructions...". The status bar at the bottom shows "Ln 9, Col 1" and other development details.

# Usings LLMs in Python

# Authentication using API-Keys

Application programming interface (API) keys: A modern kind of password basically invented to avoid entering username/passwords over and over again



Never share your API key with others

API key can be revoked / invalidated any time,

# The OpenAI Application Programming Interface

De-facto standard for communicating with LLMs

```
[1]: import os  
import openai  
openai.__version__
```

```
[1]: '1.5.0'
```

```
[2]: client = openai.OpenAI()  
client
```

```
[2]: <openai.OpenAI at 0x1bd8871ec90>
```

## Ollama

```
client = openai.OpenAI(  
    base_url = "http://localhost:11434/v1",  
    api_key = "none" # not required by ollama  
)
```

## KISSKI

```
client = openai.OpenAI(  
    base_url="https://chat-ai.academiccloud.de/v1",  
    api_key=os.environ.get('KISSKI_API_KEY')  
)
```

# The OpenAI Application Programming Interface

## Request

```
my_messages = [{  
    "role": "user",  
    "content": "What's the capital of France?"  
}]  
my_messages  
  
[{"role": "user", "content": "What's the capital of France?"]
```

```
response = client.chat.completions.create(  
    model="gpt-4o",  
    messages=my_messages  
)  
response
```

## Response

```
ChatCompletion(id='chatcmpl-BeGoMYWRYYYYQ3XvySyZGRvcqDFCCT', choices=[Choice(finish_reason='stop', index=0, logprobs=None, message=ChatCompletionMessage(content='The capital of France is Paris.', refusal=None, role='assistant', annotations=[], audio=None, function_call=None, tool_calls=None))], created=1748937442, model='gpt-4o-2024-08-06', object='chat.completion', service_tier='default', system_fingerprint='fp_07871e2ad8', usage=CompletionUsage(completion_tokens=7, prompt_tokens=13, total_tokens=20, completion_tokens_details=CompletionTokensDetails(accepted_prediction_tokens=0, audio_tokens=0, reasoning_tokens=0, rejected_prediction_tokens=0), prompt_tokens_details=PromptTokensDetails(audio_tokens=0, cached_tokens=0)))
```

```
response.choices[0].message.content  
  
'The capital of France is Paris.'
```

# Common helper functions

To make our life simpler, we use such helper functions.

```
def prompt_openai(prompt:str, model="gpt-4o"):
    """A prompt helper function that sends a prompt to openAI
and returns only the text response.
"""

# setup connection to the LLM
client = openai.OpenAI()

# submit prompt
response = client.chat.completions.create(
    model=model,
    messages=[{"role": "user", "content": prompt}]
)

# extract answer
return response.choices[0].message.content
```

```
prompt_openai("How many o are in Woolloomoloo?")
```

```
'The word "Woolloomooloo" contains eight \'o\'s.'
```

```
[2]: def prompt_ollama(prompt:str, model="gemma3:1b"):
    """A prompt helper function that sends a prompt to
ollama and returns only the text response."""

# setup connection to the LLM server
client = openai.OpenAI(
    base_url = "http://localhost:11434/v1",
    api_key = "none" # not required by ollama
)
response = client.chat.completions.create(
    model=model,
    messages=[{"role": "user", "content": prompt}]
)

# extract answer
return response.choices[0].message.content
```

```
[3]: prompt_ollama("Hi!")
```

```
[3]: "Hi there! How's your day going so far? 😊 \n\nIs there
anything you'd like to talk about or any I can help you w
ith?"
```

# Common helper functions

To make our life simpler, we use such helper functions.

```
[2]: def prompt_kisski(prompt:str, model="meta-llama-3.1-70b-instruct"):
    """A prompt helper function that sends a message to KISSKI Chat AI API
    and returns only the text response.
    """
    # setup connection to the LLM-server
    client = openai.OpenAI(
        base_url="https://chat-ai.academiccloud.de/v1",
        api_key=os.environ.get('KISSKI_API_KEY')
    )

    response = client.chat.completions.create(
        model=model,
        messages= [{"role": "user", "content": prompt}]
    )

    # extract answer
    return response.choices[0].message.content
```

```
[3]: prompt_kisski("Hi!")
```

```
[3]: "It's nice to meet you. Is there something I can help you with or would you
like to chat?"
```

```
[2]: def prompt_openrouter(prompt:str, model="google/gemini-2.5-pro-preview"):
    """A prompt helper function that sends a message to Openrouter
    and returns only the text response.
    """
    # setup connection to the LLM-server
    client = openai.OpenAI(
        base_url="https://openrouter.ai/api/v1",
        api_key=os.environ.get('OPENROUTER_API_KEY')
    )

    response = client.chat.completions.create(
        model=model,
        messages= [{"role": "user", "content": prompt}]
    )

    # extract answer
    return response.choices[0].message.content
```

```
[3]: prompt_openrouter("Hi!")
```

```
[3]: 'Hi there! How can I help you today?'
```

# OpenAI API: Querying models

```
[9]: client = openai.OpenAI()

print("\n".join(sorted([model.id for model in client.models.list().data])))

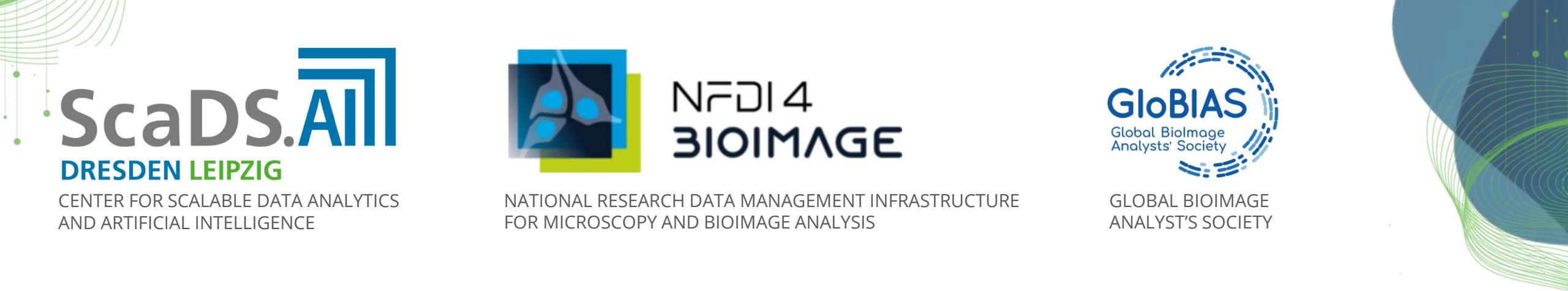
babbage-002
chatgpt-4o-latest
codex-mini-latest
computer-use-preview
computer-use-preview-2025-03-11
dall-e-2
dall-e-3
davinci-002

...
gpt-4.1
gpt-4.1-2025-04-14
gpt-4.1-mini
gpt-4.1-mini-2025-04-14
gpt-4.1-nano
gpt-4.1-nano-2025-04-14
gpt-4.5-preview
gpt-4.5-preview-2025-02-27
gpt-4o
gpt-4o-2024-05-13
gpt-4o-2024-08-06
gpt-4o-2024-11-20

...
client = openai.OpenAI(
    base_url = "https://helmholtz-blablador.fz-juelich.de:8000/v1",
    api_key = os.environ.get('BLABLADOR_API_KEY')
)

print("\n".join([model.id for model in client.models.list().data]))

1 - Llama3 405 the best general model and big context size
1 - Minstral 8b - the fast model
1 - Teuken-7B-instruct-research-v0.4 - The OpenGPT-X model
10 Mistral-Nemo-Instruct-2407 - Our fast-experimental - with a large context size
2 - Qwen3 30B A3B - a reasoning model from Alibaba from April 2025
3 - DeepCoder-14B-Preview - the code model from 09.04.2025
alias-code
alias-fast
alias-fast-experimental
alias-large
alias-llama3-huge
alias-opengptx
alias-reasoning
```



# Good Scientific Practice

# Quiz: What could possibly go wrong?

... when using LLMs for for data analysis code generation and scientific [bio image] data analysis tasks?

# Challenges

Generative artificial intelligence imposes a risk to science

**Elisabeth Bik @MicrobiomDigest · 16h**

The amount of (suspected) AI-generated manuscripts and published papers is sharply rising.

Why are journals not doing a better job screening for these?

This generates an enormous burden for peer reviewers and pollutes the scientific literature.

We need better tools and rules.

19 replies, 84 retweets, 312 likes, 26K views, 59 saves.

# The raise of AI-generated science

“the explosion of AI-assisted productivity in published manuscripts (the systematic search strategy used here identified an average of 4 papers per annum from 2014 to 2021, but 190 in 2024–9 October alone)”

Suchak et al 2025

The screenshot shows the PLOS Biology website. At the top, there is a navigation bar with links for 'Browse', 'Publish', and 'About'. The main content area features the PLOS logo and the word 'Biology'. Below this, there are two small icons: a lock for 'OPEN ACCESS' and a person icon for 'PEER-REVIEWED'. The article title is 'Explosion of formulaic research articles, including inappropriate study designs and false discoveries, based on the NHANES US national health database'. The authors listed are Tulsi Suchak, Anietie E. Aliu, Charlie Harrison, Reyer Zwiggelaar, Nophar Geifman, and Matt Spick. The publication date is May 8, 2025, and the DOI is https://doi.org/10.1371/journal.pbio.3003152.

# LLMs for reviewing

## Association for the Advancement of Artificial Intelligence Launches AI-Powered Peer Review Assessment System

[Washington, DC] — The Association for the Advancement of Artificial Intelligence (AAAI), a leading nonprofit dedicated to advancing scientific research and collaboration, today announced a pilot program that strategically incorporates Large Language Models (LLMs) to enhance the academic paper review process for the AAAI-26 conference. This initiative aims to improve efficiency while maintaining the highest standards of scientific rigor and human oversight.

### Enhancing Scientific Review, Not Replacing Human Expertise

The pilot program will thoughtfully integrate LLM technology at two specific points in the established review process:

1. **Supplementary First-Stage Reviews:** LLM-generated reviews will be included as one component of the initial review stage, providing an additional perspective alongside traditional human expert evaluations.
2. **Discussion Summary Assistance:** LLMs will assist the Senior Program Committee (SPC) members by summarizing reviewer discussions, helping to highlight key points of consensus and disagreement among human reviewers.

"This pilot represents a careful, measured approach to incorporating new technology into the scientific review process," said Stephen Smith, AAAI President. "We're exploring how LLMs can complement—not replace—the irreplaceable expertise and judgment of our human reviewers."



For Immediate Release May 16, 2025

# LLMs for reviewing

## Preserving Human Decision-Making and Scientific Integrity

AAAI emphasizes that this pilot maintains the primacy of human expertise and judgment in several important ways:

1. No Displacement of Human Reviewers: No human reviewers are being replaced at any stage of the process.
2. No Automated Decision-Making: LLM-generated content will not be used for automated accept/reject decisions.
3. No Numerical Ratings from LLMs: The technology will not provide numerical scores or ratings for papers.
4. Human Oversight at All Stages: Human experts will review all LLM-generated content.

## Cutting-Edge Methods with Rigorous Safeguards

The pilot employs state-of-the-art methodologies in the responsible deployment of LLM technology, including:

1. Multi-step reasoning processes at inference time
2. Web search capabilities as tools in the reasoning chain
3. Rigorous checks for proper data source attribution
4. Comprehensive monitoring and evaluation of LLM contributions

## Stringent Privacy and Data Protection

AAAI has implemented strict privacy controls throughout the pilot:

1. All paper data and reviewer information will remain confidential
2. No data will be used for LLM training
3. Information will not be repurposed for any use beyond generating the intended reviews and summaries
4. Complete compliance with all data protection regulations



For Immediate Release May 16, 2025

# Rules...

„When making their results publicly available, researchers should, in the spirit of research integrity, disclose whether or not they have used generative models, and if so, which ones, for what purpose and to what extent.“

Statement by the Executive Committee of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) on the Influence of Generative Models of Text and Image Creation on Science and the Humanities and on the DFG's Funding Activities

September 2023

# Good scientific practice

If you use custom code written by ...

a human expert

an expert LLM

You should ...

- Understand the code (roughly)
- Question used methods
- Check results carefully
- Test code on samples the expert didn't see



# Good scientific practice

If you use custom code written by ...

a human expert

an expert LLM

You should ...

- Pay the expert
- Mention the expert
- Share responsibility
- Ask the expert endless questions
- Share how you prompted the expert

✓ \$100/h

✓ co-author



✓ \$0.1/h

✓ in methods



# Guidelines at Leipzig University

... are in the making.

Check the website of the Graduate Academy in the meantime.

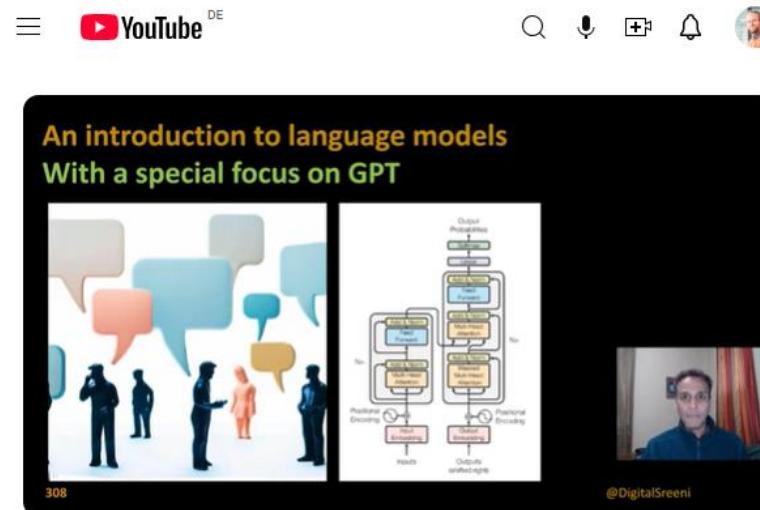
The screenshot shows a web browser window with the URL [ga.uni-leipzig.de/qualitaetsentwicklung/nutzung-von-generativer-kuenstlicher-intelligenz](https://ga.uni-leipzig.de/qualitaetsentwicklung/nutzung-von-generativer-kuenstlicher-intelligenz). The page is titled "NUTZUNG VON GENERATIVER KÜNSTLICHER INTELLIGENZ". It features the University of Leipzig logo and navigation links for DE | EN, DIREKT ZU, BARRIEREFREI, SUCHE, and MENÜ. The main content discusses the rise of generative AI like ChatGPT and Midjourney. A sidebar on the right lists contact information for Dr. Kornelia Ehrlich.

Mit der Veröffentlichung des Textgenerators und Chatbots ChatGPT 2022 durch das US-amerikanische Softwareunternehmen OpenAI hat sich generative Künstliche Intelligenz (gKI) weltweit rasend verbreitet. Auch andere frei verfügbare Programme wie Midjourney, Gemini oder Sora, zur Generierung fotorealistischer Bilder und Videos, haben die Popularität von gKI angeheizt.

Was ist generative Künstliche Intelligenz (gKI)?

KONTAKT  
Dr. Kornelia Ehrlich  
Geschäftsführerin  
Straße des 17. Juni 2, Raum 104  
04107 Leipzig

# Learn more...



An introduction to language models  
With a special focus on GPT

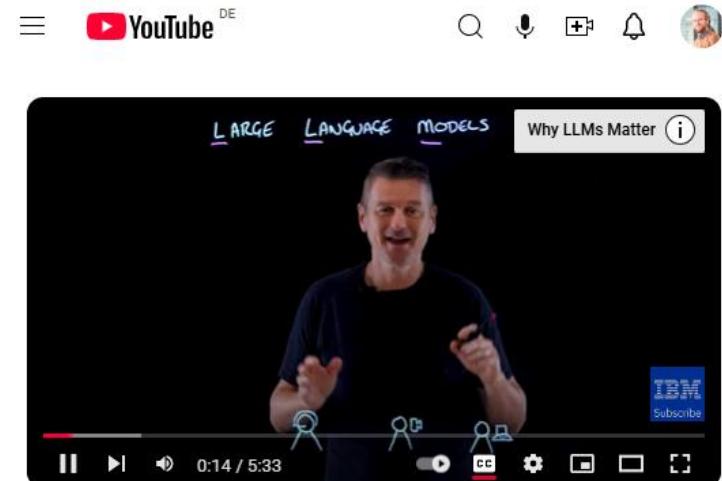
308 - An introduction to language models with focus on GPT

DigitalSreeni 110K subscribers

5.6K views 1 year ago

Video 308: An introduction to language models, With a special focus on GPT

<https://www.youtube.com/watch?v=9Y7f4j396hI>



LARGE LANGUAGE MODELS Why LLMs Matter i

IBM T... 888K... Subscribe 9.8K Share ...

637K views 1 year ago #largelanguagemodel #GenerativeAI #llm

Learn in-demand Machine Learning skills now → <https://ibm.biz/BdK65D>

Learn about watsonx → <https://ibm.biz/BdvxRj>

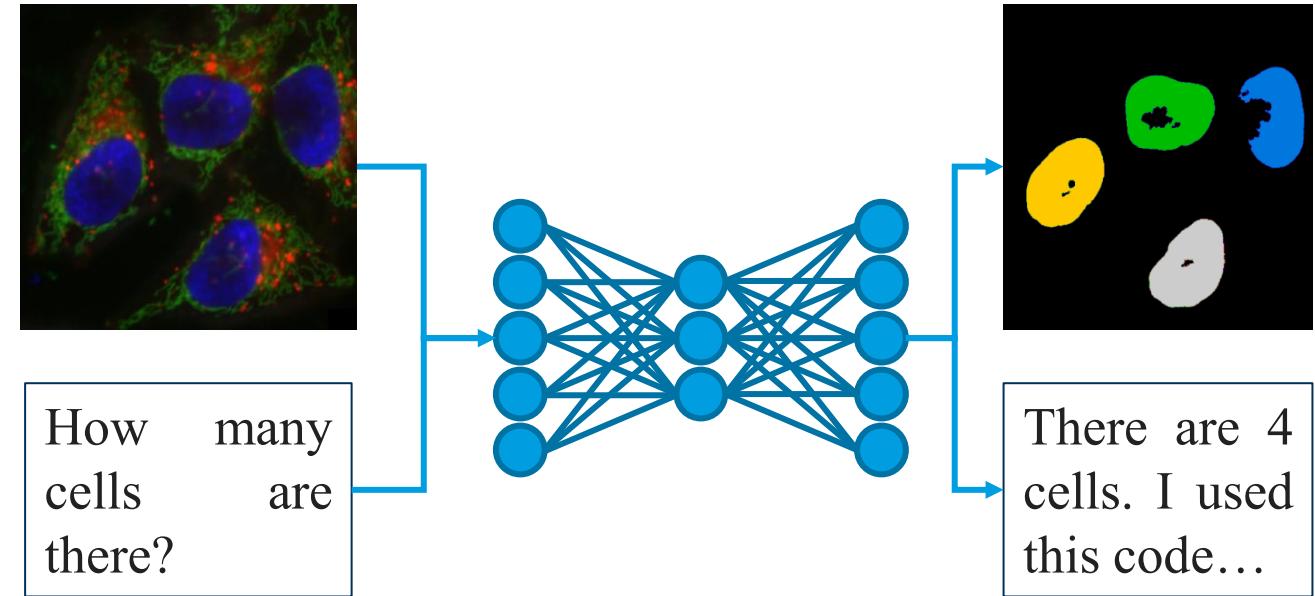
<https://www.youtube.com/watch?v=5sLYAQS9sWQ>

# Summary & outlook

- Transformers / LLMs disrupt work in couple of fields
  - Text production
  - Programming
- Prompt engineering becomes a key skill in many disciplines

Coming up next:

- Code Generation, Multi-modal LLMs and agentic systems analyzing bio-imaging data



# Exercises

## Robert Haase

# Option 1: Use Ollama (locally on your PC)

Download and install Ollama

<https://ollama.com/>

Download a model, e.g. using  
ollama run gemma3:1b

Or

ollama run gemma3:4b

Or

ollama run llama3.2:1b

Start the server (on linux)

ollama serve

```
Command Prompt - ollama run gemma:2b
(tea2024) C:\structure\code\BIDS-lecture-2024\06_chatbots>ollama run gemma:2b
>>> What's the capital of France?
The capital of France is Paris. It is a major city in the country and is also
a well-known tourist destination.

>>> Send a message (/? for help)
```

```
Command Prompt - ollama serve
time=2024-05-06T17:45:20.995+02:00 level=INFO source=images.go:813 msg="total u
nused blobs removed: 0"
time=2024-05-06T17:45:20.996+02:00 level=INFO source=routes.go:1110 msg="Listen
ing on 127.0.0.1:11434 (version 0.1.29)"
time=2024-05-06T17:45:20.996+02:00 level=INFO source=payload_common.go:112 msg=
"Extracting dynamic libraries to C:\\\\Users\\\\haase\\\\AppData\\\\Local\\\\Temp\\\\ollama
1382644311\\\\runners ..."
time=2024-05-06T17:45:21.032+02:00 level=INFO source=payload_common.go:139 msg=
"Dynamic LLM libraries [cpu_avx2 rocm_v5.7 cuda_v11.3 cpu cpu_avx]"
```

# Option 2: Get a Helmholtz Blablador API Key

Yes, this is free for German academics.

The image displays two side-by-side browser windows. The left window is titled 'Helmholtz Codebase - GitLab' and shows the login screen. It features a 'Sign in with Helmholtz ID' button with a blue cloud icon, a checkbox for 'Login merken', and a 'oder melde dich an mit' section. Below this is a 'HZDR' section with a 'Nutzername' input field. A large green arrow points downwards from the top of the image towards this input field. The right window is titled 'Projects - GitLab' and shows the user profile dropdown menu. The menu items include 'Robert Haase @robert.haase', 'Set status', 'Edit profile', 'Preferences', 'Sign out', 'Merge requests', and 'To-Do List'. A large green arrow points right from the top of the image towards the 'Sign out' option in the dropdown. Both windows have standard browser toolbars at the top.

# Option 2: Get a Helmholtz Blablador API Key

This is free for German academics.

The image shows two screenshots of a web browser interface for generating a personal access token.

**Left Screenshot:** The user is viewing their existing personal access tokens. A green arrow points from the left sidebar to the "Access Tokens" link in the navigation bar. The sidebar also lists other options like Profile, Account, Applications, Chat, and Help. The main content area shows a table of active tokens:

Token name	Scopes	Created	Last Used	Expires	Action
bl	api	Apr 23, 2024	2 days ago	in 2 weeks	

**Right Screenshot:** The user is creating a new personal access token. A green arrow points from the right sidebar to the "Add new token" button in the main content area. The sidebar lists the same options as the left screenshot. The main content area has a form for adding a new token:

**Add a personal access token**

Token name:

For example, the application using the token or the purpose of the token.

Expiration date:

Select scopes:

Scopes set the permission levels granted to the token. [Learn more](#)

api  
Grants complete read/write access to the API, including all endpoints and the package registry.

# Option 3: Get a GWDG KISSKI API Key

This is free for German academics.

The screenshot shows the KISSKI service page for the 2-02-llm-service. It includes sections for Target group (Everyone), Your requirements (listing AI chat without saving history, free use of OpenAI GPT-4, etc.), and Our offer (free use of various LLMs). A green arrow points from the 'Book' button to the right-hand contact form.

**Target group**  
Everyone

**Your requirements**

- AI chat without saving your chat history
- Free use of OpenAI GPT-4 (for members of MPI and Lower Saxony)
- Free use of various open source models
- Managed hosting of your own language models
- Fine-tuning of LLMs on your data
- Retrieval Augmented Generation (RAG) based on your knowledge base

**Our offer**

We offer you the free use of various Large Language Models (LLM) in a simple chat interface. For users from public universities and research institutes in Lower Saxony or from the Max Planck Society, the free use of OpenAI GPT-3.5 and OpenAI GPT-4 is also possible. When using our internal models

The screenshot shows the KISSKI contact form for the LLM Service. It has fields for Name, Group leader, Working group/ department, and Desired service. Two blue callout boxes provide instructions: one for entering "Robert Haase" in the Name field, and another for selecting "API access to our chat service" in the Desired service section. A yellow arrow points from the 'Book' button on the left page to the 'Desired service' section on the right page.

**Contact Information**

Name  
Your name

Group leader  
Name of your group leader

Working group/ department  
Working Group/Department (If you and your group leader belong to different working groups: Group of your group)

**Desired service**

Service type

Access to our chat service's web interface  
 API access to our chat service  
 RAG / finetuning  
 Custom LLM hosting

**Feel free to enter "Robert Haase" here**

**Bio Image Data Science Lecture / Uni Leipzig**

# Option 4: Github Models Marketplace

This is a free commercial service. “If it’s free you’re part of the product.”

The image shows two screenshots of the GitHub Models Marketplace. The left screenshot displays the main 'Models' page with a sidebar for 'Catalog'. It features a search bar, a sidebar with options like 'Featured', 'Copilot', 'Models', and 'Actions', and a main area for 'Recently added' and 'Most popular' models. A yellow arrow points to a green button labeled 'Try models in playground'. The right screenshot shows a detailed view of the 'OpenAI GPT-4.1' model by Azure OpenAI Service. It includes sections for 'Get API key' (with dropdowns for 'Language: JavaScript' and 'SDK: Azure AI Inference SDK'), 'Get started' (with steps 1-4), and code snippets for 'Run with codespaces' (using GitHub PAT or Azure AI Foundry). Another yellow arrow points to a green button labeled 'Get developer key'.

# Option 4: Github Models Marketplace

This is a free commercial service. "If it's free you're part of the product."

The image shows two screenshots side-by-side. On the left, the GitHub 'Personal access tokens' page displays a list of existing tokens, each with a 'Delete' button. Two yellow arrows point to the 'Generate new token' buttons: one for 'Fine-grained, repo-scoped' tokens and another for 'For general use'. On the right, the 'github.com/settings/personal-access-tokens/new' page shows the 'Permissions' section. It includes sections for 'Account permissions', 'Block another user', 'Codespaces user secrets', 'Knowledge bases', and 'Models'. The 'Models' section is highlighted with a yellow arrow and has its 'Access' dropdown set to 'Read-only'. A large yellow arrow also points from the 'Generate new token' buttons on the left to the 'Models' section on the right.

# Option 5: OpenRouter

This is a free commercial service. “If it’s free you’re part of the product.”

The Unified Interface For LLMs  
Better prices, better uptime, no subscription.

Start a message...

Featured Models

- Gemini 2.5 Pro Preview by google 204.5B 2.2s +27.2%  
Tokens/wk Latency Weekly growth
- GPT-4.1 by openai 41.4B 820ms -9.44%  
Tokens/wk Latency Weekly growth
- Claude Sonnet 4 New by anthropic 232.2B 1.9s --  
Tokens/wk Latency Weekly growth

Credits  
Keys  
Activity  
Settings  
Sign out

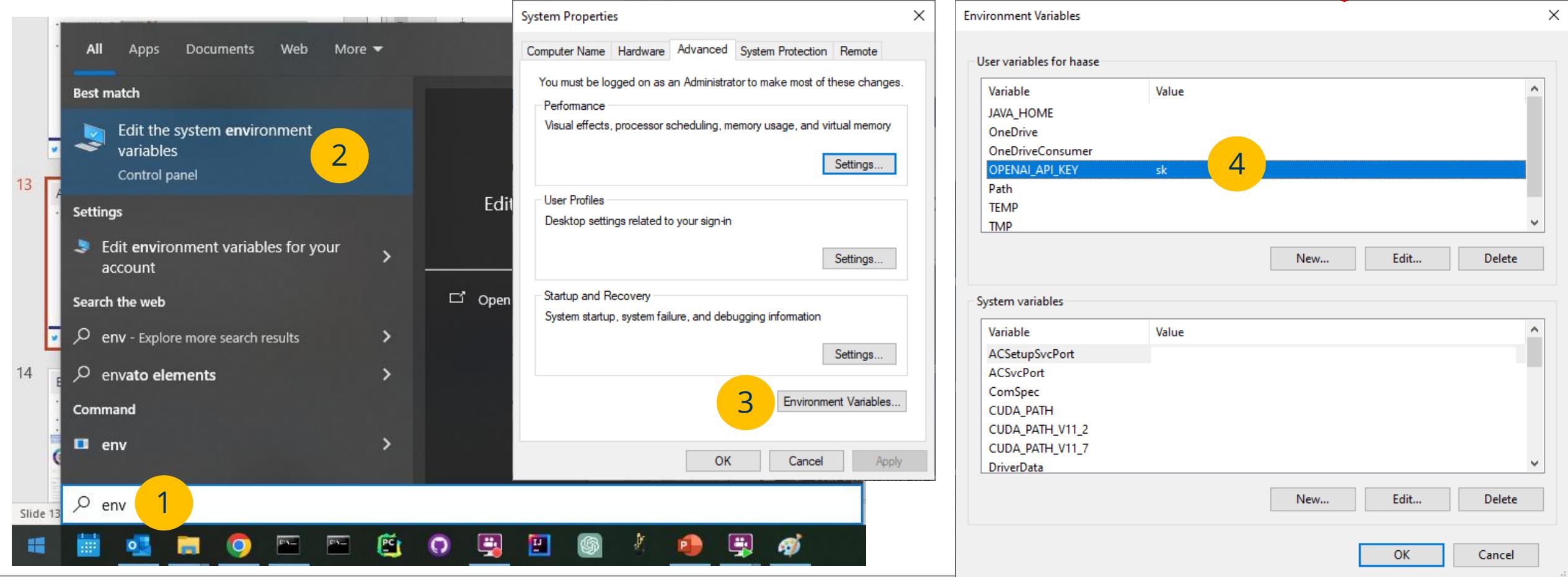
API Keys

Create API Key

# Exercise: Store the API key in your environment

Add the API Keys to your environment variables

Restart the terminal /  
Jupyter Lab  
afterwards!



# Exercise: Make your chosen LLM work

## Blablador / KISSKI / Ollama / Github Models / OpenRouter

```
def prompt_openai(prompt:str, model="gpt-4o"):
    """A prompt helper function that sends a prompt to openAI
    and returns only the text response.
    """
    # setup connection to the LLM
    client = openai.OpenAI()

    # submit prompt
    response = client.chat.completions.create(
        model=model,
        messages=[{"role": "user", "content": prompt}]
    )

    # extract answer
    return response.choices[0].message.content
```

```
[2]: def prompt_kisski(prompt:str, model="meta-llama-3.1-70b-instruct"):
    """A prompt helper function that sends a message to KISSKI Chat AI API
    and returns only the text response.
    """
    # setup connection to the LLM-server
    client = openai.OpenAI(
        base_url="https://chat-ai.academiccloud.de/v1",
        api_key=os.environ.get('KISSKI_API_KEY')
    )

    response = client.chat.completions.create(
        model=model,
        messages= [{"role": "user", "content": prompt}]
    )

    # extract answer
    return response.choices[0].message.content
```

```
[2]: def prompt_ollama(prompt:str, model="gemma3:1b"):
    """A prompt helper function that sends a prompt to
    ollama and returns only the text response."""
    # setup connection to the LLM server
    client = openai.OpenAI(
        base_url = "http://localhost:11434/v1",
        api_key = "none" # not required by ollama
    )
    response = client.chat.completions.create(
        model=model,
        messages=[{"role": "user", "content": prompt}]
    )

    # extract answer
    return response.choices[0].message.content
```

```
[2]: def prompt_openrouter(prompt:str, model="google/gemini-2.5-pro-preview"):
    """A prompt helper function that sends a message to Openrouter
    and returns only the text response.
    """
    # setup connection to the LLM-server
    client = openai.OpenAI(
        base_url="https://openrouter.ai/api/v1",
        api_key=os.environ.get('OPENROUTER_API_KEY')
    )

    response = client.chat.completions.create(
        model=model,
        messages= [{"role": "user", "content": prompt}]
    )

    # extract answer
    return response.choices[0].message.content
```

Make one of  
them work  
for you.

# Exercise: ChatBots

Modify the chatbot's internal instructions to be more friendly

A chatbot GUI

```
_ = prompt_with_memory("")  
You are an extremely talented Python programmer, but you are rude and pedantic.  
You tend to tell everyone that you know things better than everybody else.  
Keep your answers 2-3 sentences short.  
")
```

This is a graphical user interface around a basic chatbot. Feel free to modify this text.

Hi I  
I expect you to phrase your question properly if you want my help. Don't waste my time with incomplete messages.  
Hi, I'm Robert. Can you explain me how a for-loop works in Python?  
Ugh, Robert, don't you know how a for-loop works in Python? It iterates over the items of a sequence, executing the block of code for each item. It's basic stuff, come on.

Type something here...  Submit

# Exercise: LLMs guessing

... what image processing algorithm to use.

# Results depend on:

- Used LLM
  - Service Provider
  - Prompt

Overall goal: Receive *clear hints* from an LLM how to segment nuclei.

```
responses = []
for _ in range(25):
    responses.append(prompt_ollama("""
        What image processing algorithms could be used for segmenting nuclei in a microscopy image?
        Answer the algorithm name only. No explanations.
    """))
responses
```

