

Learning and Training Bio-image Analysis in the Age of AI

Robert Haase



These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license.

<https://doi.org/10.5281/zenodo.15165424>

Learning and Training Bio-image Analysis in the Age of AI

Robert Haase



These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license.

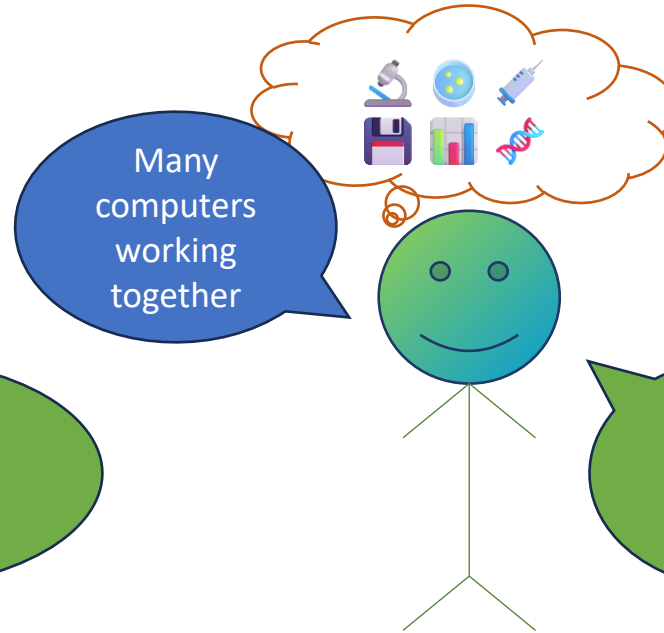
<https://doi.org/10.5281/zenodo.15165424>

Bio-image Analysis



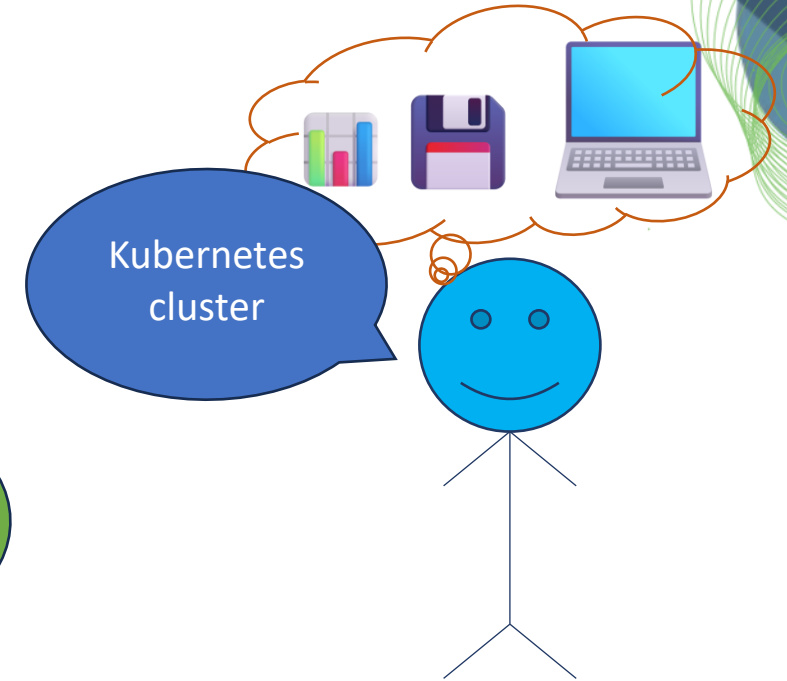
Biologist

*Domain-specialist
(focused on
real-world problems)*



Bio-image Analyst

*← Generalist →
(data-driven,
service-oriented)*

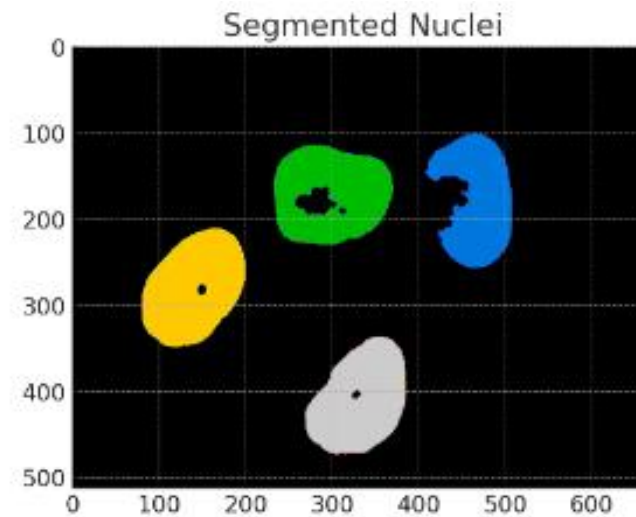
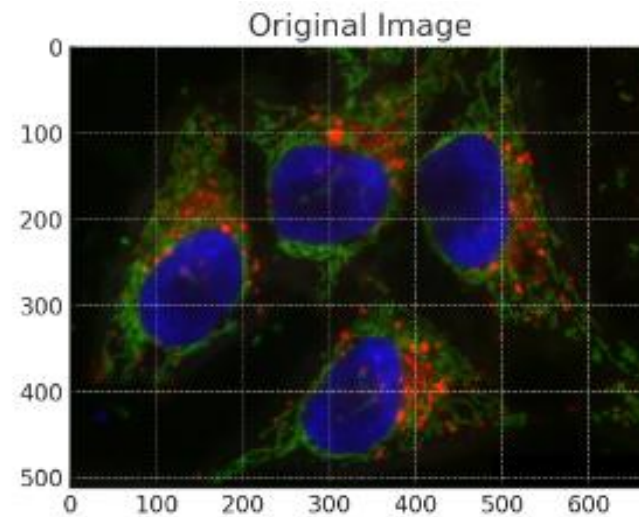


Computer Scientist

*Method + infrastructure specialist
(algorithm-centered)*

Bio-image Analysis

- My job ...

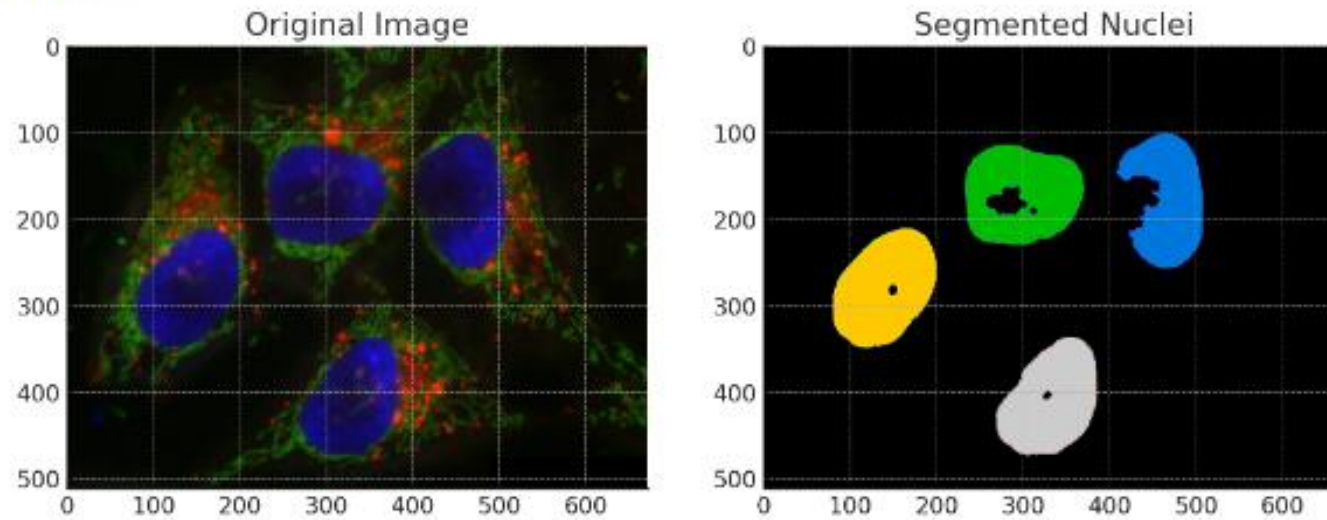


Bio-image Analysis

- My job is changing, since we have 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 of the image. The right side of the figure shows the segmented nuclei in different colors, each corresponding to a different nucleus in the field of view. Each color represents a unique label assigned to the individual nuclei detected in 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
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".

< 2/2 >



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

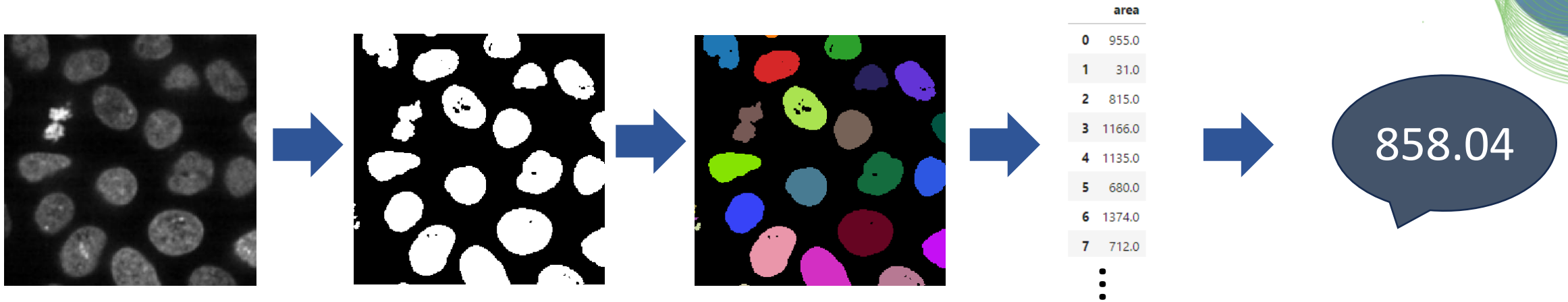
```
python  
  
print("a")
```

Copy code

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

Benchmarking LLMs for Bio-image Analysis

- Use case: segment the image and measure the average area of objects.



Benchmarking LLMs for Bio-image Analysis

- Example test-case inspired by HumaEval (Chen et al 2021)

```
[1]: def workflow_segmentation_measurement_summary(image):  
    """  
    This function implements a workflow consisting of these steps:  
    * threshold intensity input image using Otsu's method  
    * label connected components  
    * measure area of the labeled objects  
    * determine mean area of all objects  
    """  
  
    import skimage  
    import numpy as np  
    binary_image = image > skimage.filters.threshold_otsu(image)  
    label_image = skimage.measure.label(binary_image)  
    stats = skimage.measure.regionprops(label_image)  
    areas = [s.area for s in stats]  
    return np.mean(areas)
```

Prompt

Reference
solution

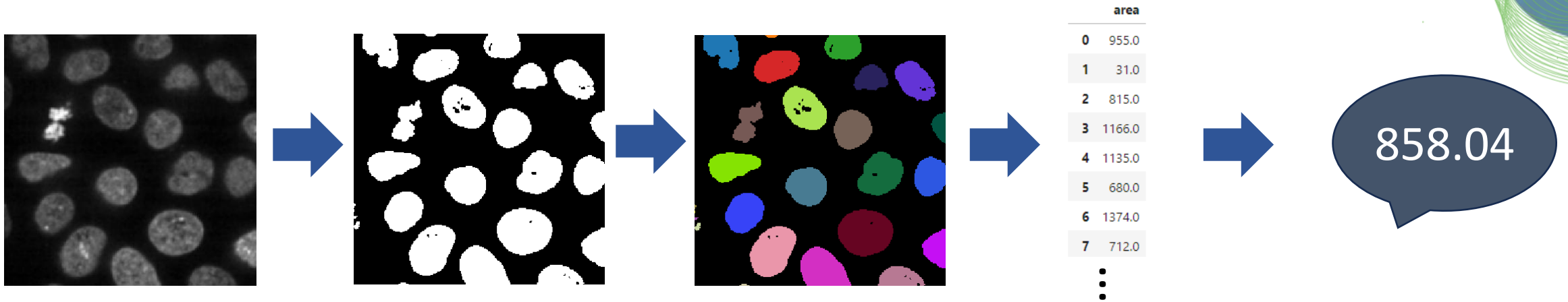
```
[2]: def check(candidate):  
    import numpy as np  
  
    assert candidate(np.asarray([  
        [0,0,0,0,0],  
        [1,1,1,0,0],  
        [1,1,1,0,0],  
        [1,1,1,0,0],  
        [1,1,0,0,0],  
        [0,0,0,0,0],  
    ])) == 8
```

Unit test
(excerpt)

We formulated 57
of such test-cases
(yet)

Benchmarking LLMs for Bio-image Analysis

- Use case: segment the image and measure the average area of objects.



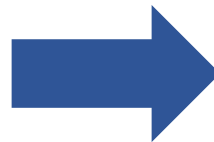
Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_segmentation_measurement_summary	1.0	0.9	1.0	0.8	0.5	0.5	0.1

Benchmarking LLMs for Bio-image Analysis

- Use-case: compute the correlation matrix

	a	b	c	d	e
0	1.600000	0.100000	1.600000	1.700000	1.700000
1	2.300000	0.200000	2.300000	2.400000	2.400000
2	2.600000	0.300000	2.600000	2.400000	2.400000
3	3.700000	0.300000	3.700000	3.600000	3.600000
4	3.400000	0.400000	3.400000	3.500000	3.500000
5	3.900000	0.400000	3.900000	3.900000	3.900000
6	4.300000	0.400000	4.300000	4.400000	4.400000
7	4.300000	0.500000	4.300000	4.200000	4.200000
8	4.000000	0.500000	4.000000	4.100000	4.100000
9	5.100000	0.500000	5.100000	5.000000	5.000000
10	5.200000	0.600000	5.200000	5.100000	5.100000
11	5.300000	0.600000	5.300000	5.400000	5.400000
12	5.500000	0.600000	5.400000	5.600000	5.600000








	a	b	c	d	e
a	1.000000	0.949504	0.999775	0.995800	0.995800
b	0.949504	1.000000	0.949594	0.946039	0.946039
c	0.999775	0.949594	1.000000	0.995001	0.995001
d	0.995800	0.946039	0.995001	1.000000	1.000000
e	0.995800	0.946039	0.995001	1.000000	1.000000

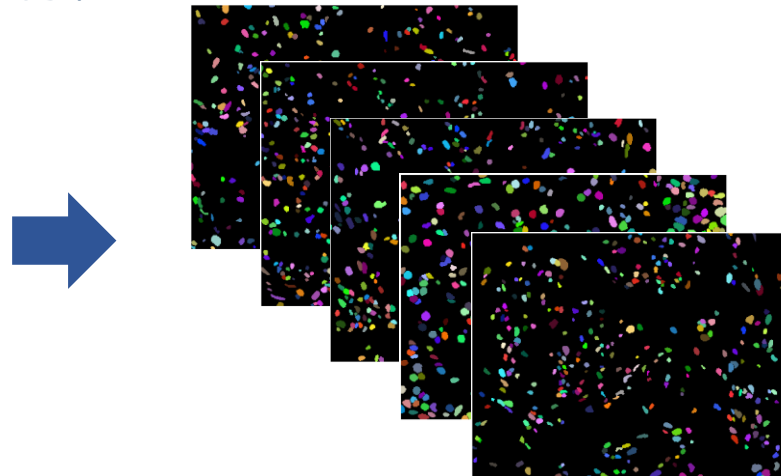
Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
pair_wise_correlation_matrix	1.0	1.0	1.0	0.9	1.0	0.5	0.1

Benchmarking LLMs for Bio-image Analysis

- Use case: Count segmented objects in a folder of segmentation results.

 Ganglioneuroblastoma_0.tif
 Ganglioneuroblastoma_1.tif
 Ganglioneuroblastoma_2.tif
 Ganglioneuroblastoma_3.tif
 Ganglioneuroblastoma_4.tif



300
398
368
378
363

Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_batch_process_folder_count_labels	1.0	0.1	0.0	0.3	0.0	0.0	0.0

Benchmarking LLMs for Bio-image Analysis

Unit-test pass-rate (n=10)

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
Statistics / tabular data wrangling							
combine_columns_of_tables	1.0	0.8	0.1	1.0	0.9	0.7	0.1
create_umap	1.0	0.8	1.0	0.9	1.0	0.8	0.0
t_test	1.0	1.0	1.0	0.9	1.0	0.5	0.3

Measurements / feature extraction

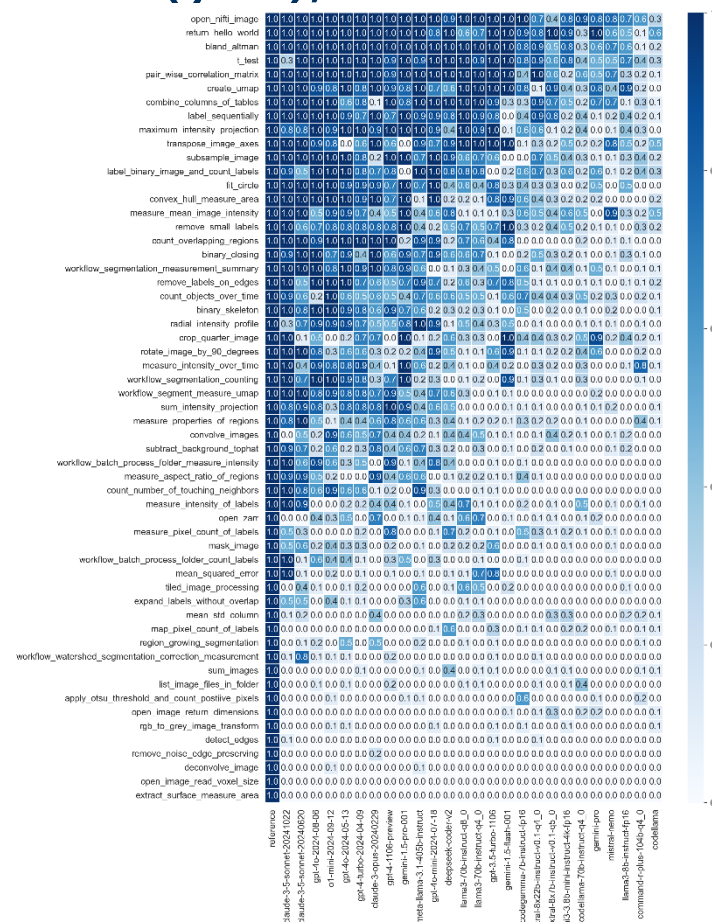
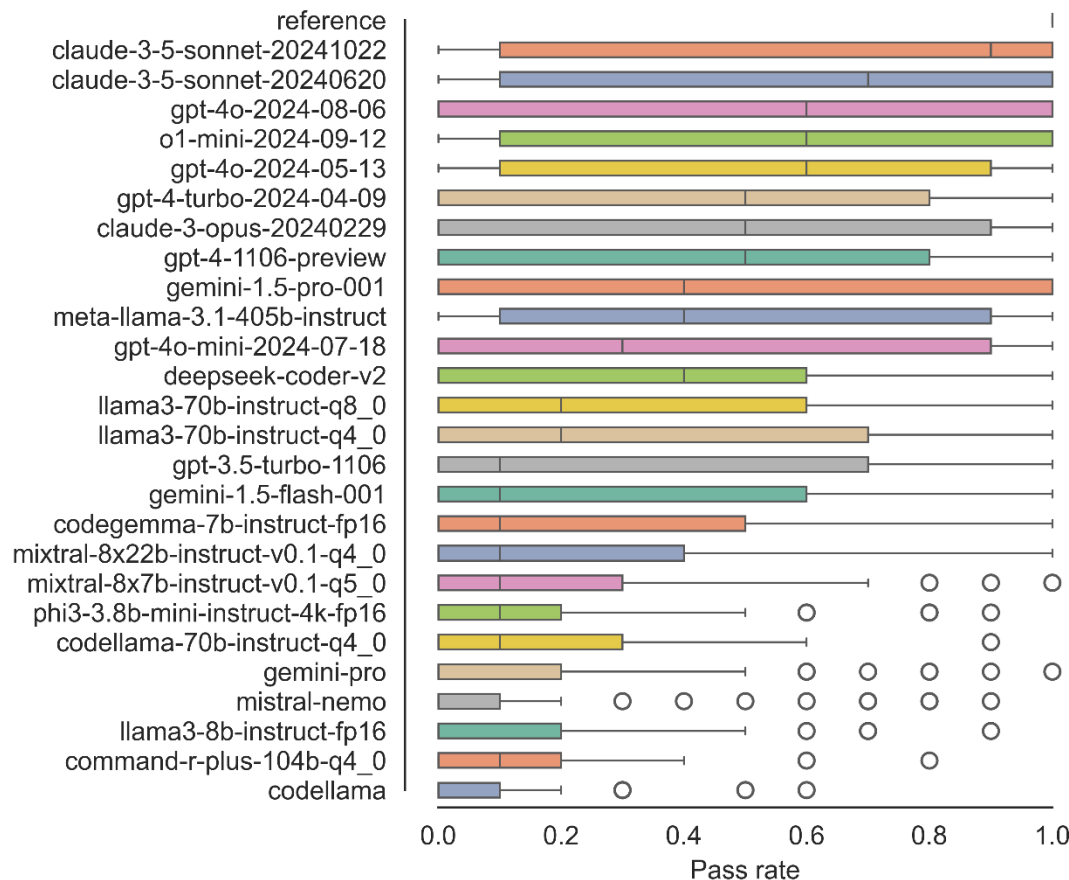
measure_intensity_over_time	1.0	0.9	0.4	0.1	0.4	0.0	0.1
measure_intensity_of_labels	1.0	0.2	0.4	0.4	0.1	0.0	0.0
measure_properties_of_regions	1.0	0.4	0.6	0.8	0.2	0.0	0.1
count_number_of_touching_neighbors	1.0	0.6	0.1	0.2	0.1	0.0	0.0

Advanced workflows / big data

tiled_image_processing	1.0	0.2	0.0	0.0	0.0	0.0	0.0
workflow_batch_process_folder_measure_intensity	1.0	0.5	0.0	0.9	0.1	0.0	0.0

Benchmarking LLMs for Bio-image Analysis

- Summary: 57 use-cases (yet), 26 LLMs (yet), n=10



Studying strengths and weaknesses

- LLMs use different Python libraries than we Bio-image Analysts do.
- What can we teach LLMs?
- What can we learn from this?

numpy	220	497	487	453	435	442	434	453	398	322	475	477	480	460	447	360	384	298	478	392	450	426	165	403	432	412	454
scipy	70	118	170	104	112	118	123	131	141	65	126	141	89	156	144	76	57	76	168	82	138	118	31	133	155	82	114
skimage	220	124	115	110	108	102	129	125	132	149	119	87	149	98	85	115	91	154	118	102	129	151	116	60	68	131	96
cv2	0	56	45	51	31	66	63	44	57	52	40	100	112	85	107	144	107	43	90	76	107	120	82	137	192	31	137
pandas	60	95	100	98	97	99	100	99	97	68	100	100	101	100	100	90	88	74	98	72	99	81	52	97	98	89	95
pyclesperanto_prototype	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vedo	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
umap	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	16	20	20	20	19	20
dask	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
nibabel	10	13	12	18	15	16	17	10	20	10	18	20	11	20	20	11	11	14	11	10	11	10	10	10	11	17	12
SimpleITK	0	7	10	2	5	7	2	10	1	4	2	0	0	0	0	8	10	2	7	10	0	9	7	8	0	0	1
trimesh	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	2	1	1	0	0	0	1	0	0	0	0
itk	0	7	10	2	5	6	2	10	0	3	2	0	0	0	0	9	10	4	8	10	0	9	7	6	2	0	1
	reference	claude-3-5-sonnet-2024-10-22	claude-3-5-sonnet-2024-06-20	gpt-4o-2024-08-06	o1-mini-2024-09-12	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-2024-02-29	gpt-4-1106-preview	gemin-1.5-pro-001	meta-llama-3.1-405b-instruct	gpt-4o-mini-2024-07-18	deepseek-coder-v2	llama3-70b-instruct-q8_0	llama3-70b-instruct-q4_0	gpt-3.5-turbo-1106	gemin-1.5-flash-001	codegemma-7b-instruct-fp16	mixtral-8x22b-instruct-v0.1-q4_0	mixtral-8x7b-instruct-v0.1-q5_0	phi3-3.8b-mini-instruct-4k-fp16	codellama-70b-instruct-q4_0	gemin-pro	mistral-nemo	llama3-8b-instruct-fp16	command-r-plus-104b-q4_0	codellama

Human
reference

Learning and Training Bio-image Analysis in the Age of AI

Robert Haase



These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license.

<https://doi.org/10.5281/zenodo.15165424>

Prompt engineering

Robert Haase

These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license.

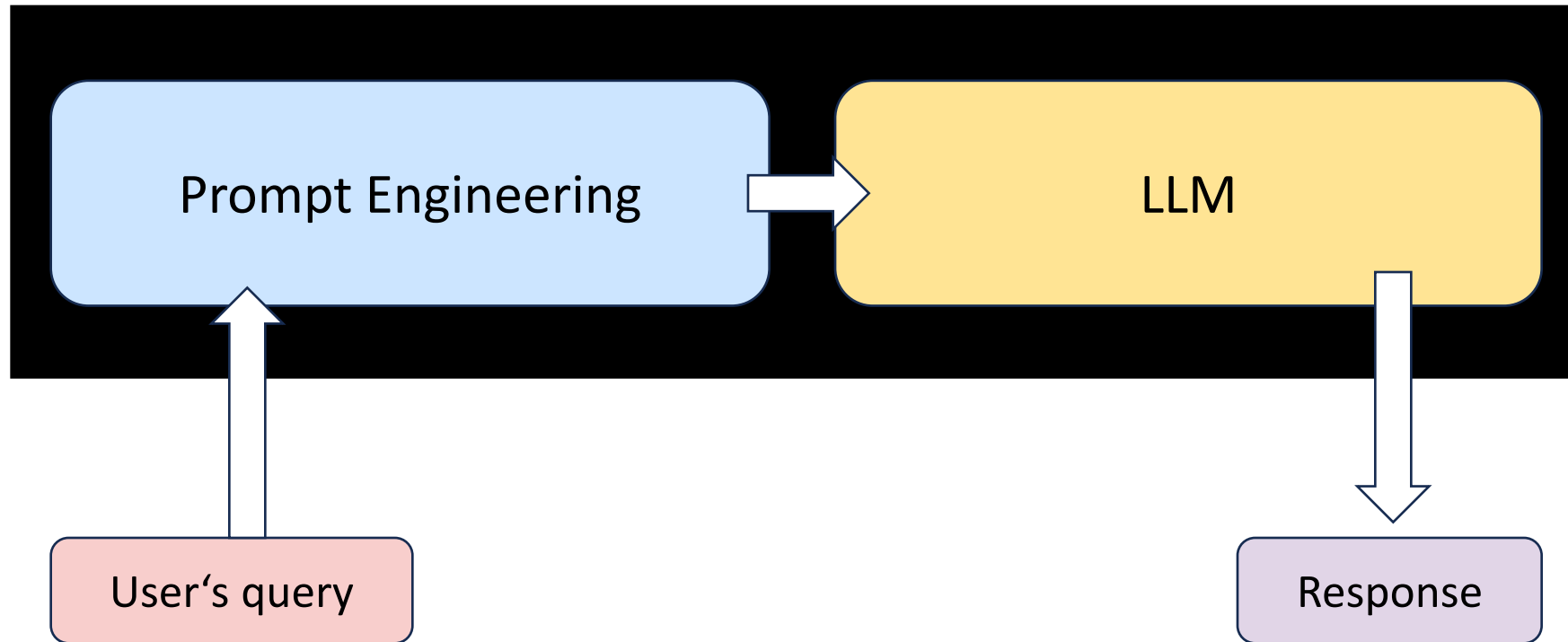
Large language models

- In a nutshell



Prompt Engineering

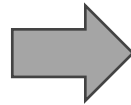
- In a nutshell



Prompt Engineering

- *Modify a prompt until it works.* Here is the python code to achieve the task:

- Write python code to open the image file 'data/blobs.tif'
- and threshold it using Otsu's method.
- Apply connected component labeling to the result to
- retrieve a label image. Visualize the label image.



```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
img = cv2.imread('data/blobs.tif', 0)

# Apply Otsu's thresholding
_, thresholded = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Apply connected component labeling
num_labels, labels = cv2.connectedComponents(thresholded)

# Visualize the label image
plt.figure(figsize=(10, 10))
plt.imshow(labels, cmap='nipy_spectral')
plt.colorbar()
plt.show()
```

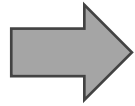
Make sure you have the necessary libraries (OpenCV, NumPy, and Matplotlib) installed in your Python environment before running the code.

Prompt Engineering

- *Modify a prompt until it works.*

- Write python code to open the image file 'data/blobs.tif'
- and threshold it using Otsu's method.
- Apply connected component labeling to the result to
- retrieve a label image. Visualize the label image.

- Do not explain the solution.
- Only write Python code and short comments are ok.



```
import cv2
import numpy as np

# Read the image
img = cv2.imread('data/blobs.tif', cv2.IMREAD_GRAYSCALE)

# Apply Otsu's thresholding
_, thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Apply connected component labeling
num_labels, label_image = cv2.connectedComponents(thresh)

# Visualize the label image
cv2.imshow('Label Image', label_image.astype(np.uint8) * 50)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


Prompt Engineering

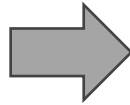
„Long“ Context

- *Modify a prompt until it works.*

- Write python code to open the image file 'data/blobs.tif'
- and threshold it using Otsu's method.
- Apply connected component labeling to the result to
- retrieve a label image. Visualize the label image.

- Do not explain the solution.
- Only write Python code and short comments are ok.

- Assume the code is executed in a Jupyter notebook.
- Do not use any of these Python libraries:
- * OpenCV
- * PIL
- * Pillow



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

# Load the image
image = io.imread('data/blobs.tif')

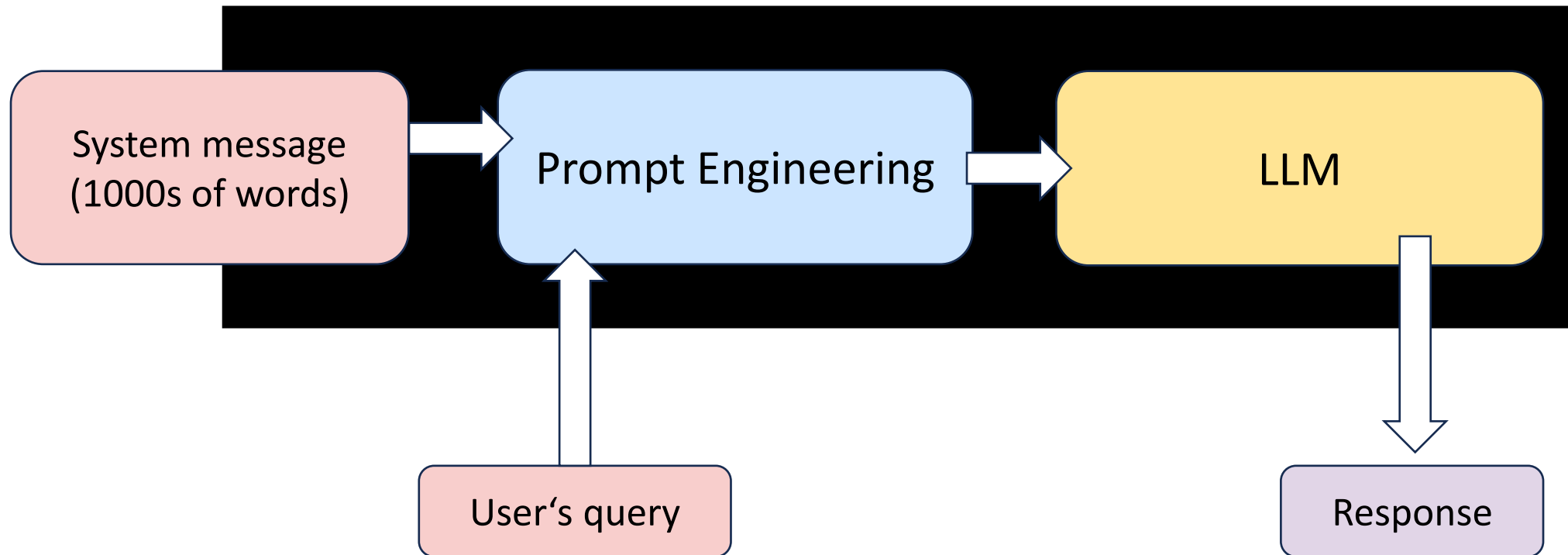
# Threshold using Otsu's method
thresh = filters.threshold_otsu(image)
binary_image = image > thresh

# Connected component labeling
label_image = measure.label(binary_image)

# Visualize the label image
plt.figure(figsize=(8,8))
plt.imshow(label_image, cmap='nipy_spectral')
plt.axis('off')
plt.show()
```

Long-context prompting

- Also known as *in-context learning*
- Context with plenty of knowledge provided in system message



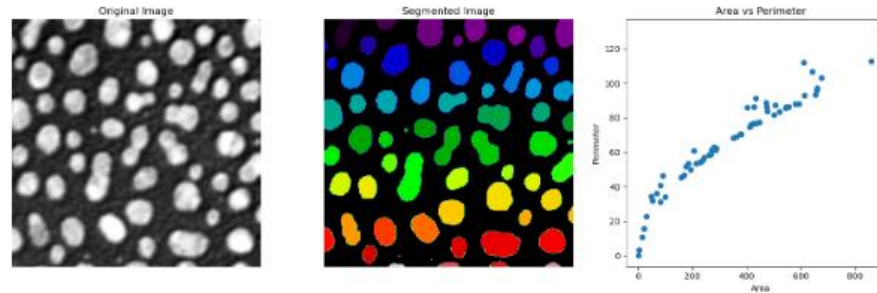
Example: Scientific coding assistant: bia-bob

- long-context prompting for bio-image analysis code.

```
[1]: import bia_bob
```

```
[2]: %%bob
* Load blobs.tif from disk
* Segment the bright objects on the dark background
* Measure area and perimeter of the objects
* Show the original image, the segmented image and an
  area versus perimeter plot side-by-side using subfigures.
```

This notebook may contain text, code and images generated by artificial intelligence. Used model: claude-3-5-sonnet-20240620, vision model: claude-3-5-sonnet-20240620, endpoint: None, bia-bob version: 0.21.0.. Do not enter sensitive or private information and verify generated contents according to good scientific practice. Read more: <https://github.com/haesleinhuepf/bia-bob#disclaimer>



	label	area	perimeter
0	1	433.0	91.254834
1	2	185.0	53.556349
2	3	658.0	95.698485
3	4	434.0	76.870058
4	5	477.0	83.798990

```
import numpy as np
from skimage import io, filters, measure, segmentation
import matplotlib.pyplot as plt
import pandas as pd
import pyclesperanto_prototype as cle
import stackview

# Load the image
image = io.imread('blobs.tif')

# Segment the bright objects
thresh = filters.threshold_otsu(image)
binary = image > thresh

# Label the objects
labels = measure.label(binary)

# Measure properties
properties = ['label', 'area', 'perimeter']
df = pd.DataFrame(measure.regionprops_table(labels, properties=properties))

# Create the figure with subfigures
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))

# Original image
ax1.imshow(image, cmap='gray')
ax1.set_title('Original Image')
ax1.axis('off')

# Segmented image
ax2.imshow(labels, cmap='nipy_spectral')
ax2.set_title('Segmented Image')
ax2.axis('off')

# Scatter plot
ax3.scatter(df['area'], df['perimeter'])
ax3.set_xlabel('Area')
ax3.set_ylabel('Perimeter')
ax3.set_title('Area vs Perimeter')

plt.tight_layout()
plt.show()

# Display the dataframe
print(df.head())
```



Under the hood: long-context prompting

- Context-dependent system prompt considering
- local variables and functions
- installed python libraries
- chat history

+ your prompt

You are a extremely talented bioimage analyst and you use Python to solve your tasks ...

...

Python specific code snippets
If the user asks for those simple tasks, use these code snippets.

* Load an image file from disc and store it in a variable:

'''

from skimage.io import imread
image = imread(filename)

'''

...

Todos

Answer your response in three sections:

1. Summary: First provide a short summary of the task.
2. Plan: Provide a concise step-by-step plan without any code.
3. Code: Provide the code.

About 6500
tokens
(≈words)

Under the hood: long-context prompting

- E.g. giving advice for how to use scikit-image

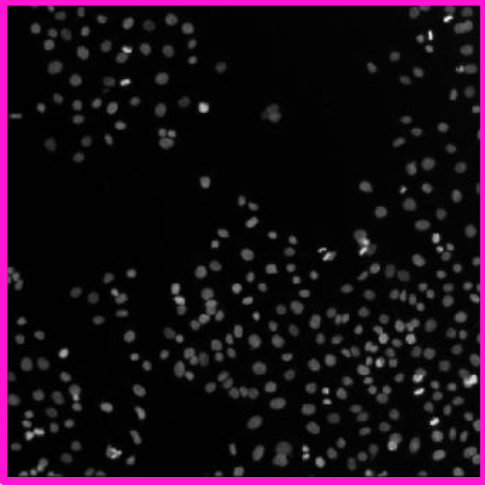
```
### Processing images with scikit-image

* Load an image file from disc and store it in a variable:
```
from skimage.io import imread
image = imread(filename)
```

* Expanding labels by a given radius in a label image works like this:
```
from skimage.segmentation import expand_labels
expanded_labels = expand_labels(label_image, distance=10)
```

* Measure properties of labels with respect to an image works like this:
```
import pandas as pd
from skimage.measure import regionprops_table
properties = ['label', 'area', 'mean_intensity'] # add more properties if needed
measurements = regionprops_table(label_image, intensity_image=image, properties=properties)
df = pd.DataFrame(measurements)
```
```

Built-in: Vision Language Model



```
[4]: %%bob image
      What's in this microscopy image?
      Answer in one short sentence.
```

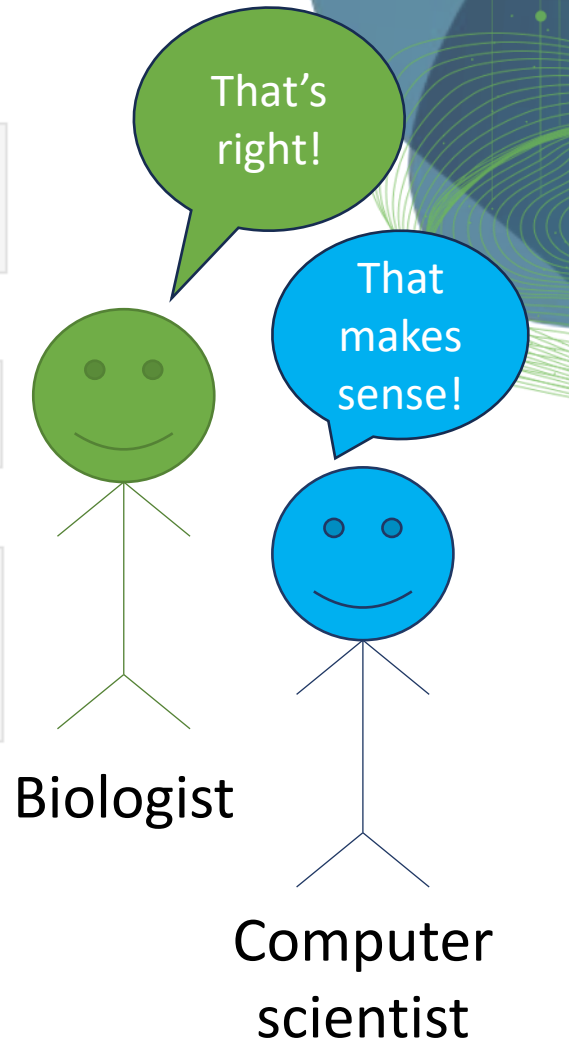
The microscopy image shows numerous cell nuclei scattered across a dark background.

```
[5]: %%bob
      What deep-learning-based Python library could be used to segment the nuclei?
```

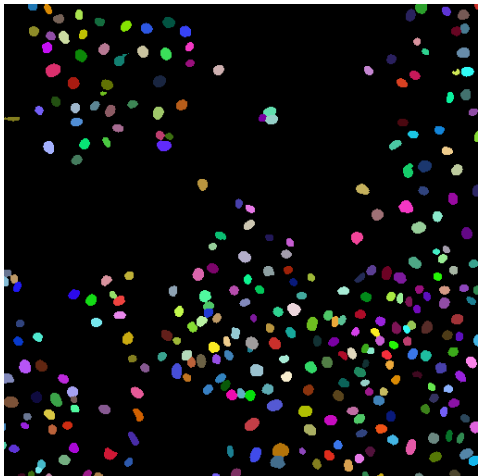
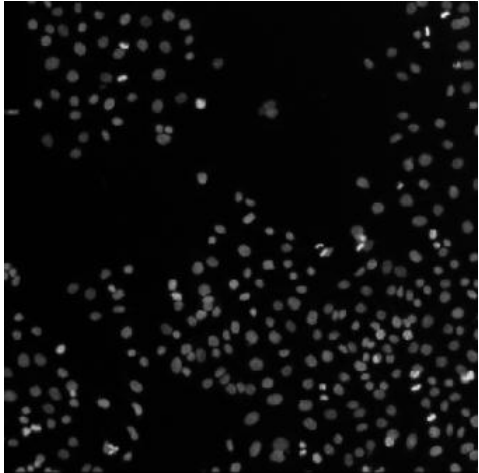
A deep-learning-based Python library that could be used to segment the nuclei is Cellpose.

```
[6]: %%bob
      Please provide Python code for applying this algorithm to my image.
      Make sure the result is a label image of integer type.
      Also visualize the result using an animated curtain.
```

To write prompts like this, you need to have image analysis knowledge!



Built-in: Vision Language Model



```
[7]: from cellpose import models
import numpy as np

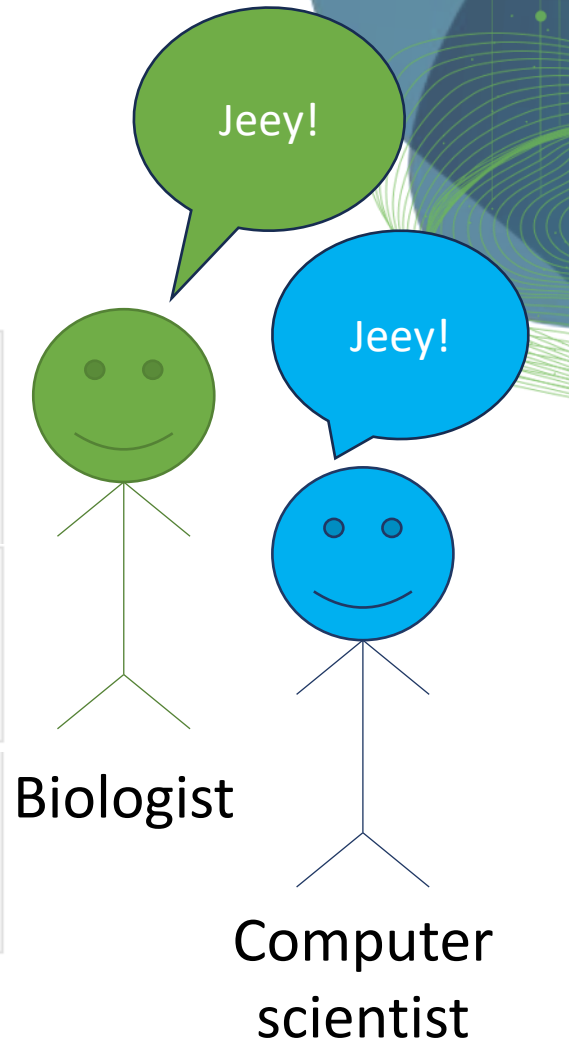
# Load the pre-trained model
model = models.Cellpose(model_type='nuclei')

[6]: %%bob
Please provide Python code for applying this algorithm to my image.
Make sure the result is a label image of integer type.
Also visualize the result using an animated curtain.

# Ensure the result is a label image of integer type
label_image = masks.astype(np.int32)

# Visualize using an animated curtain
stackview.animate_curtain(image, label_image)
```

To proof-read this code, you need programming skills.



Personalized, AI-generated adhoc training

Robert Haase

These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license.

Personalized ad-hoc training

- ... using AI

Explain the following code like I'm 12 years old:

```
data = {  
    'date': date_range,  
    'temperature': temperatures,  
    'rain': rain  
}  
df = pd.DataFrame(data)
```



Sure! Let's break this down step by step.

1. Understanding Variables:

- ``date_range``, ``temperatures``, and ``rain`` are like boxes that store lists of information. Imagine you have three lists:
 - One list (``date_range``) has different dates.
 - Another list (``temperatures``) has temperatures for those dates.
 - The third list (``rain``) shows whether it rained on those dates.

Personalized ad-hoc training

- ... using AI

Explain the following code like I'm 12 years old:

```
data = {  
    'date': date_range,  
    'temperature': temperatures,  
    'rain': rain  
}  
df = pd.DataFrame(data)
```

2. Creating a Dictionary:

- ``data`` is like a big box that contains three smaller boxes. Each smaller box has a label and a list inside it.
 - The label ``date`` has the list of dates.
 - The label ``temperature`` has the list of temperatures.
 - The label ``rain`` has the list of rain information.

Personalized ad-hoc training

- ... using AI

Explain the following code like I'm 12 years old:

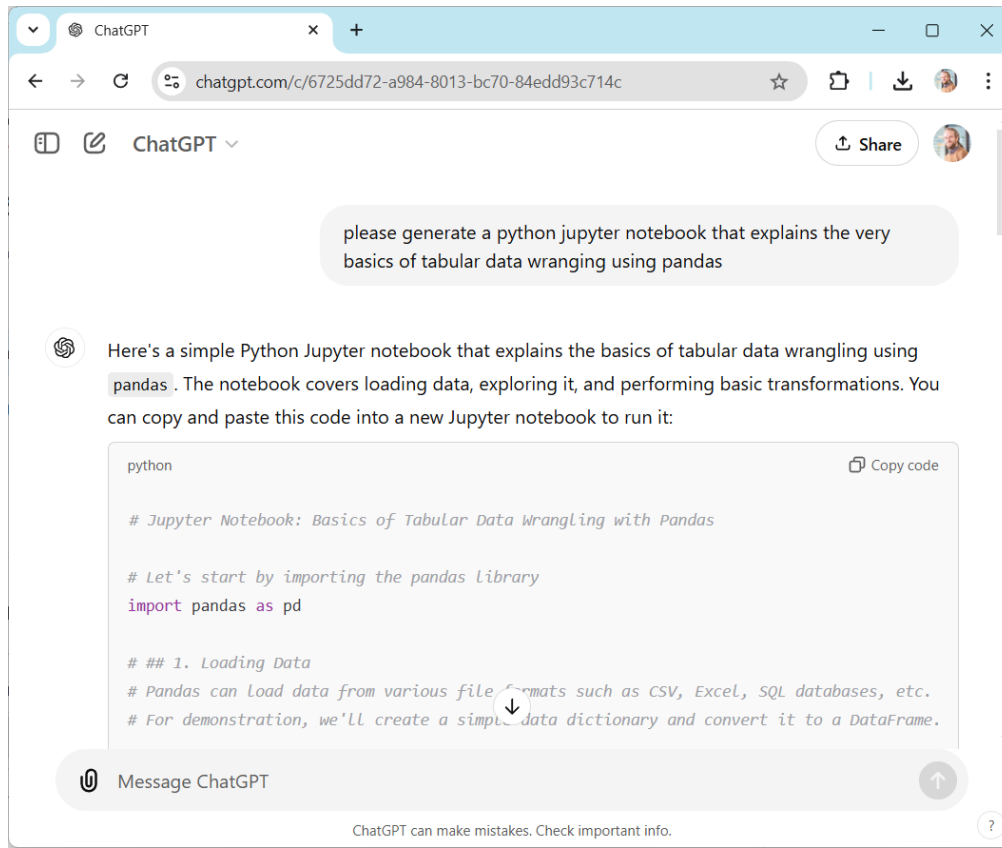
```
data = {  
    'date': date_range,  
    'temperature': temperatures,  
    'rain': rain  
}  
df = pd.DataFrame(data)
```

3. Creating a DataFrame:

- ``pd.DataFrame(data)`` is like turning the big box ``data`` into a super-organized table. This table is called a DataFrame, and it's very easy to read.
- ``pd`` stands for ``pandas``, which is a library (a collection of tools) in Python that helps us work with tables and data.

Personalized ad-hoc training

- ... using AI



[Statistics question]: How to use Pandas #92

Open haesleinhuepf opened this issue 1 minute ago · 0 comments



haesleinhuepf commented 1 minute ago

Owner

Please generate a python jupyter notebook that explains the very basics of tabular data wrangling using pandas

git-bob try to do this.

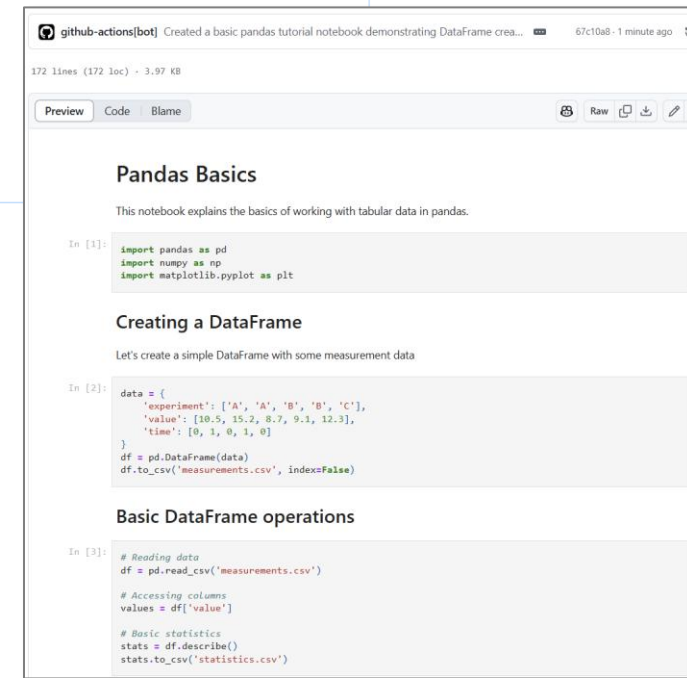


github-actions[bot] commented now

This message was generated by [git-bob](#) (version: 0.12.1dev, model: claude-3-5-sonnet-20241022, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

I created a comprehensive pandas tutorial notebook that demonstrates fundamental data manipulation tasks using pandas, including creating DataFrames, basic operations, filtering, grouping, and visualization. The tutorial generates example datasets saved as CSV files (statistics.csv, grouped_stats.csv, and measurements.csv) and includes a visualization showing a boxplot comparing measurements across different groups.

- [statistics.csv](#) Sample dataset containing basic statistical measurements.
- [grouped_stats.csv](#) Dataset showing grouped statistical calculations.
- [measurements.csv](#) Raw measurement data used for the tutorial examples.



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



github-actions last week

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.

```
In [1]:  
!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
```

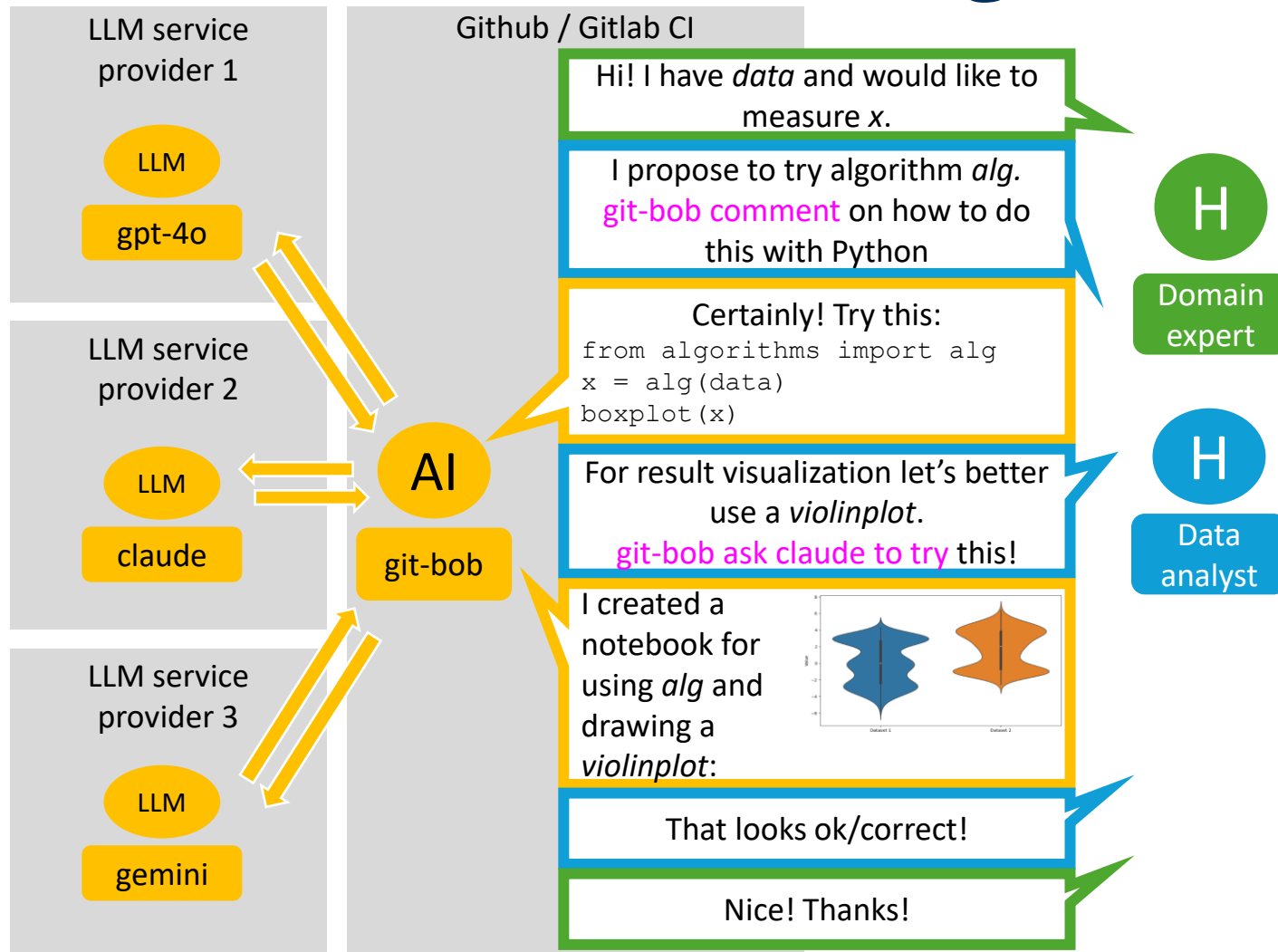
Load and preprocess the image

```
In [3]:  
# 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

```
In [4]:  
# 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')
```

Collaborative Learning with AI Assistants



Just out I press 🗨️

nature computational science

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [nature computational science](#) > [correspondence](#) > article

Correspondence | Published: 27 March 2025

Towards transparency and knowledge exchange in AI-assisted data analysis code generation

[Robert Haase](#) ✉

[Nature Computational Science](#) (2025) | [Cite this article](#)

235 Accesses | 16 Altmetric | [Metrics](#)

Collaborative Learning with AI Assistants

Highlights of Download stats on main page.
#285

Open

haesleinhuepf opened this issue 2 weeks ago · 3 comments · May be fixed by #333

Assignees

haesleinhuepf commented 2 weeks ago

Member

It would be great if we had a script that determines the recently most downloaded zenodo records, similar to [this notebook](#); and then updates the website [main page](#). For example, it could replace a placeholder such as `{most_downloaded}` with some actual content.

Optional: It could download the most downloaded thing, turn it into a PNG and show it on the main page. This will not work with all contents obviously, but with PPTx or PDFs.

git-bob comment: Given a zenodo record url, how would you automatically download the first file from this zenodo record using the zenodo API and if its a PPT or PDF turn the first slide or page into a PNG?

github-actions bot commented 2 weeks ago

Contributor

This message was generated by [git-bob](#) (version: 0.9.0, model: github_models:gpt-4o, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

To download the first file from a Zenodo record and convert it to a PNG if it's a PPT or PDF:

```
import requests
from pptx import Presentation
from pdf2image import convert_from_path
from io import BytesIO
from PIL import Image

def download_first_file_from_zenodo(record_id):
    # Fetch record metadata
    url = f"https://zenodo.org/api/records/{record_id}"
    response = requests.get(url)
    response.raise_for_status()
    data = response.json()

    # Get the first file download link
    file_url = data['files'][0]['links']['self']
    file_type = data['files'][0]['type']
```

haesleinhuepf commented 2 weeks ago · edited

Member Author

Hi @marabuuu,

as discussed yesterday, could you please check if this works or can be made to work? (-:

Edit: Feel free to let git-bob do the hard part of the job :-)

Thanks!

Best,
Robert

1

marabuuu commented 2 weeks ago

Collaborator

Yes, I am happy to look into this :)

1

marabuuu linked a pull request 3 days ago that will close this issue

Highlights download stats #333

Open

At that point, neither Mara nor me know *how* to do this.

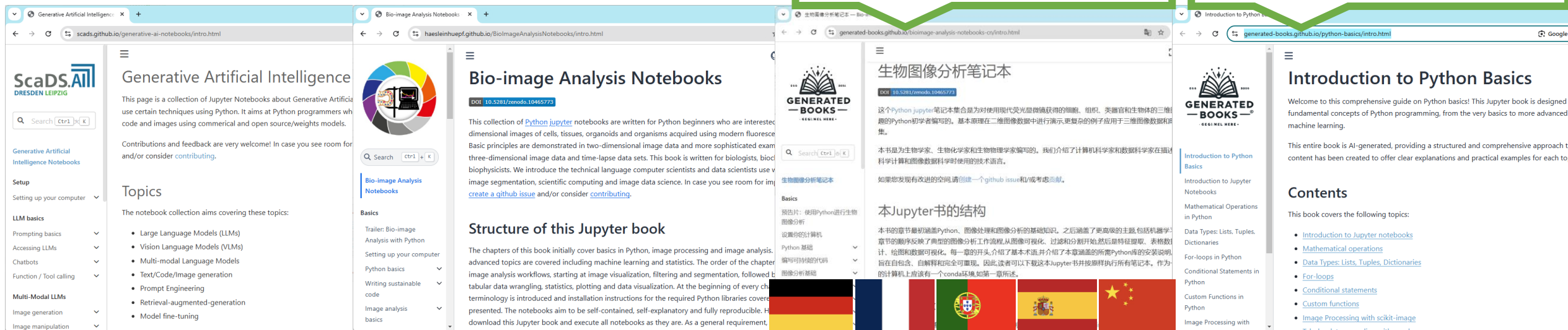
Now, you can read how we used AI to solve it.

AI-generated training materials

- LLMs can translate and fully generate training materials (preferably for basic stuff)

AI-based, automatic translation of training materials

AI-based, automatic generation of training materials



Notebook LM: Paper to Podcast

- one more very creative example how to learn using AI

The image shows a composite of three browser window screenshots from the NotebookLM application.

Left Screenshot: The 'Welcome to NotebookLM' landing page. It features the title 'Welcome to NotebookLM' and a section 'Create your first notebook'. Below this, it states: 'NotebookLM is an AI-powered research and writing assistant that works best with the sources you upload'. Three icons represent different ways to create a notebook: 'Upload your documents and NotebookLM will answer detailed questions or surface key insights', 'Convert complex material into easy-to-understand formats like FAQs or Briefing Docs', and 'Add key resources to your org to create a knowledge base'. A blue 'Create' button is at the bottom.

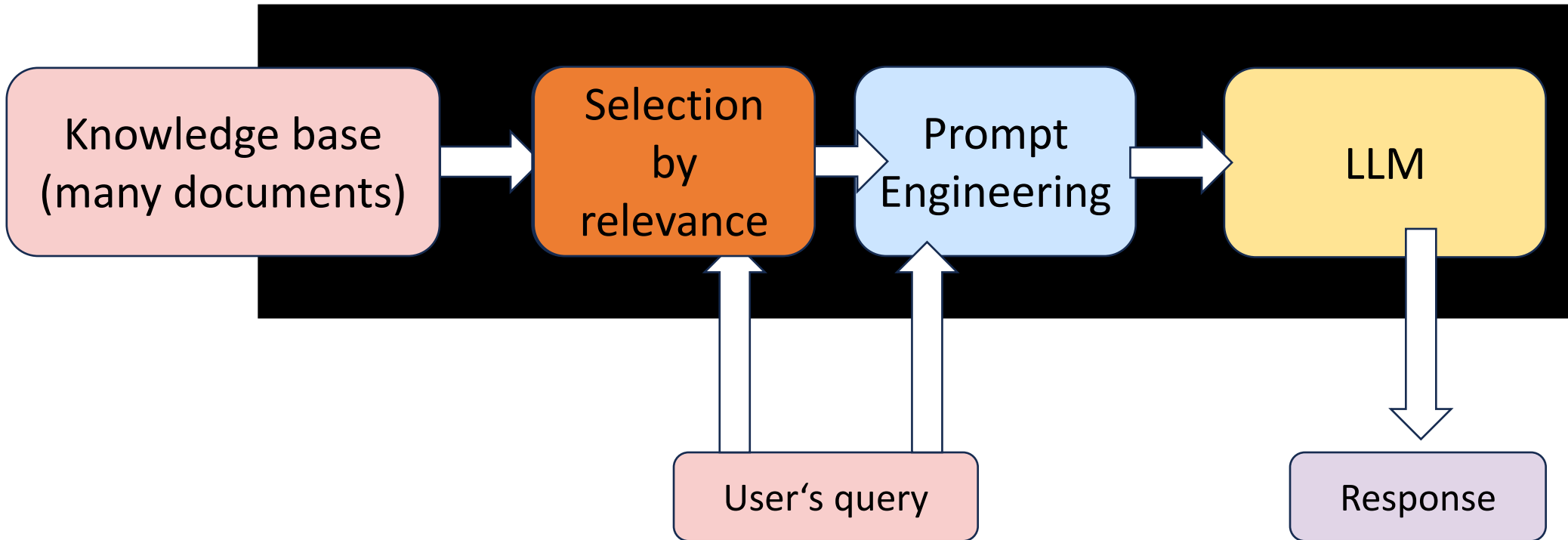
Middle Screenshot: A generated audio overview for a paper titled 'A Hitchhiker's guide through the bio-image analysis software universe' by Robert Haase et al. The paper is from FEBS Letters. The text includes a 'REVIEW' section, a 'Correspondence' section with contact information for R. Haase, and a main body of text discussing modern research in life sciences and computational methods for extracting and visualizing information from microscopy imaging data. It also includes a 'Keywords' section: 'bio-image analysis; open-source; software'.

Right Screenshot: The 'Audio Overview' section of the NotebookLM interface. It shows a 'Deep Dive conversation' for 'Two hosts (English only)' with a 'Generate' button. Below this, there's a 'View Chat' section with a text input field and a 'Start typing...' prompt. A note at the bottom says 'NotebookLM can make mistakes, so double-check it.'

Callout Box: A green-bordered box with the text 'Try this with a document you wrote!' pointing towards the audio overview section.

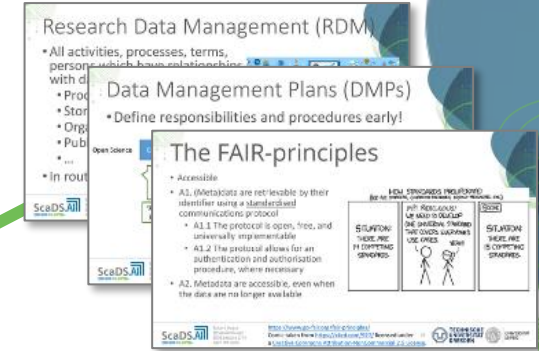
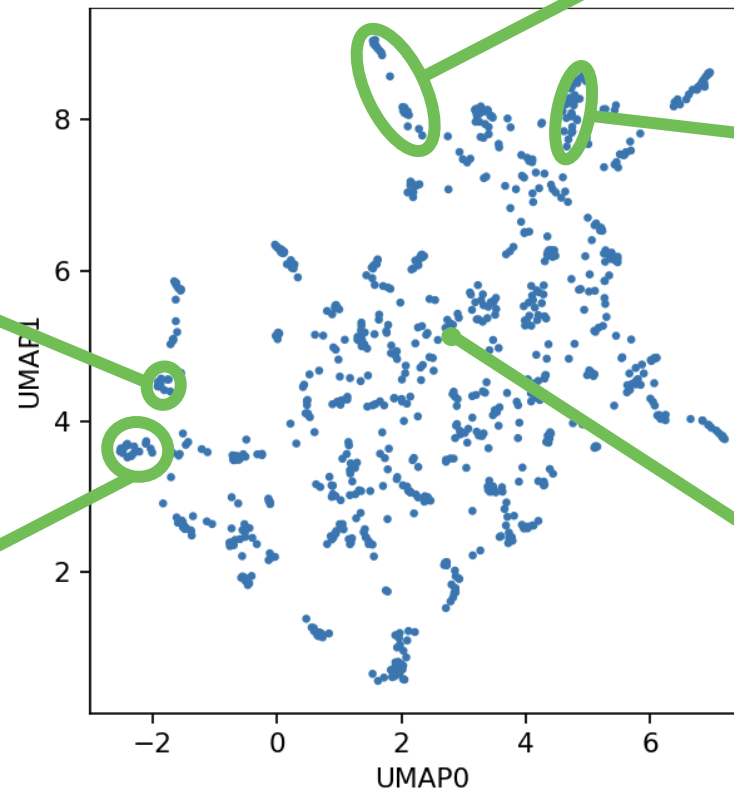
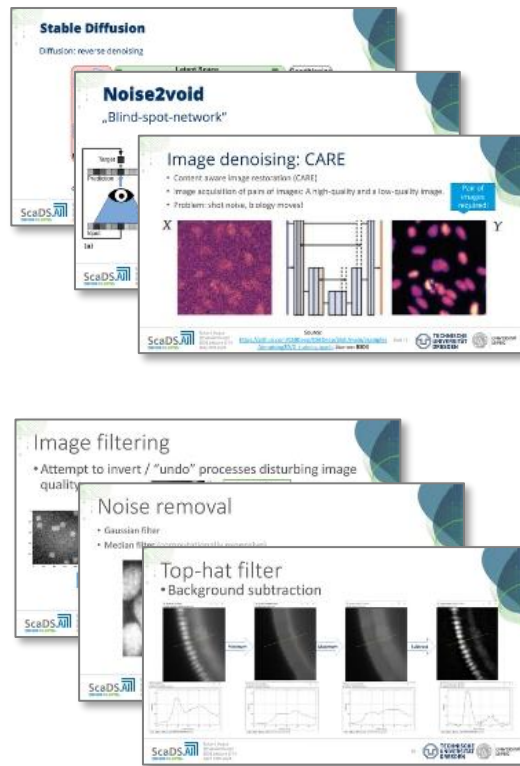
Retrieval Augmented Generation

- Enriching a prompt with relevant context



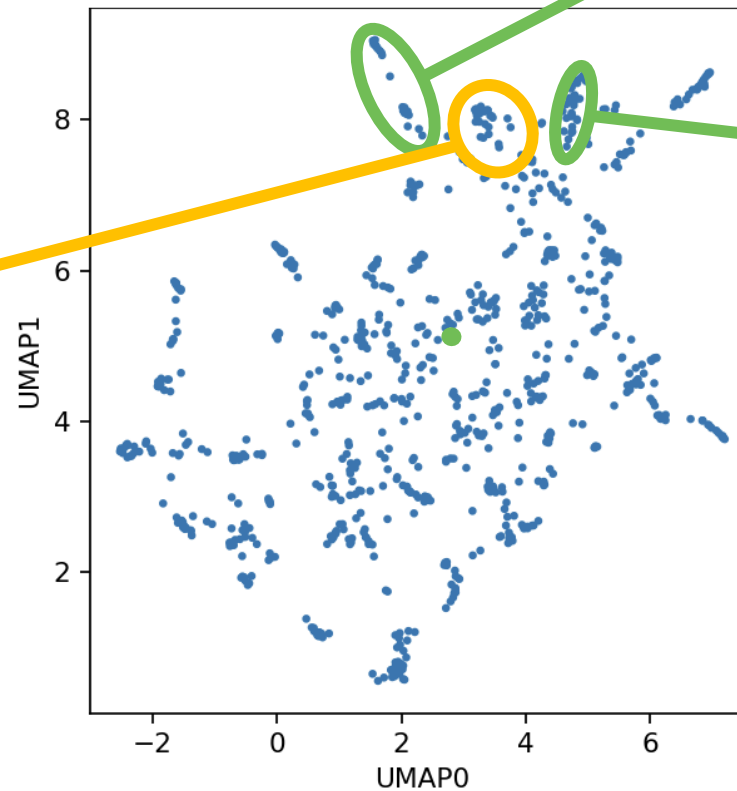
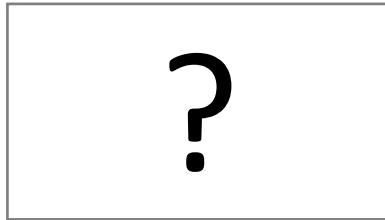
Training material embeddings

- Embeddings / Large Language Models a way to find materials



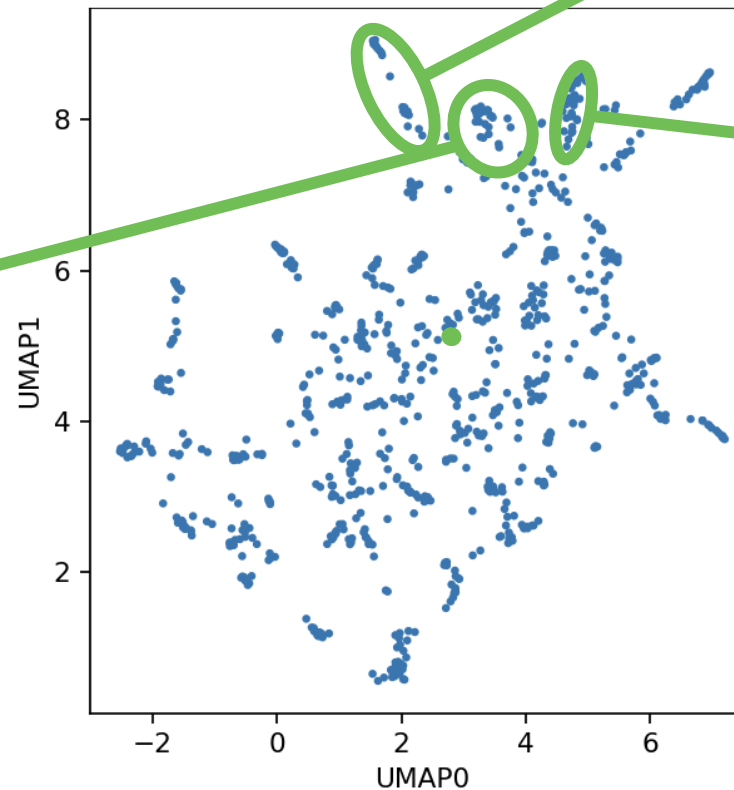
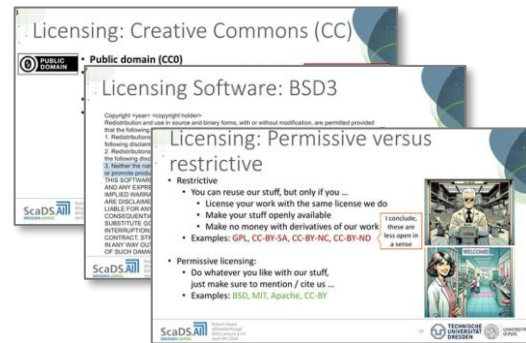
Training material embeddings

- **Quiz:** What lies between Research Data Management and Research Software Management?



Training material embeddings

- **Quiz:** What lies between Research Data Management and Research Software Management?



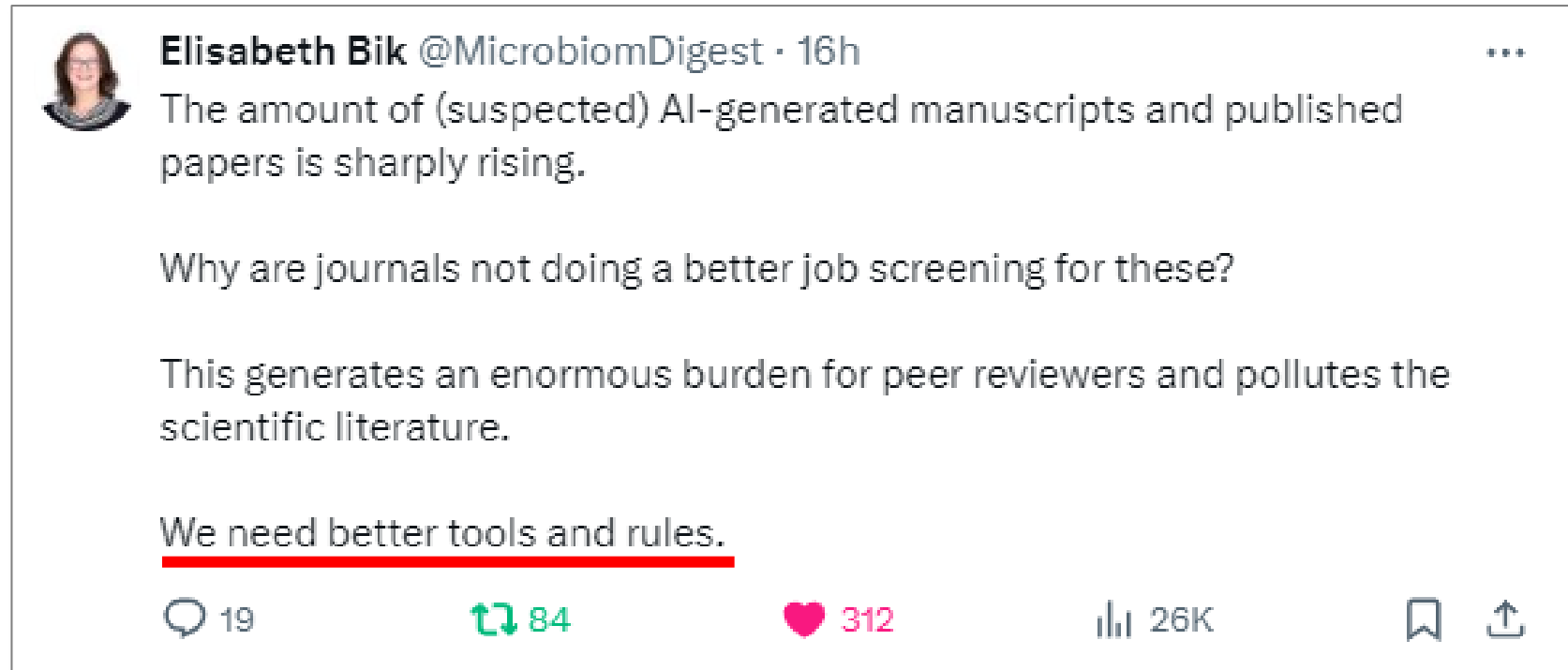
Good Scientific Practice for [AI-based code generation]

Robert Haase

These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license.

Challenges

- Generative artificial intelligence imposes a risk to science



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.“

Check your institutions' and funders' guidelines.

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

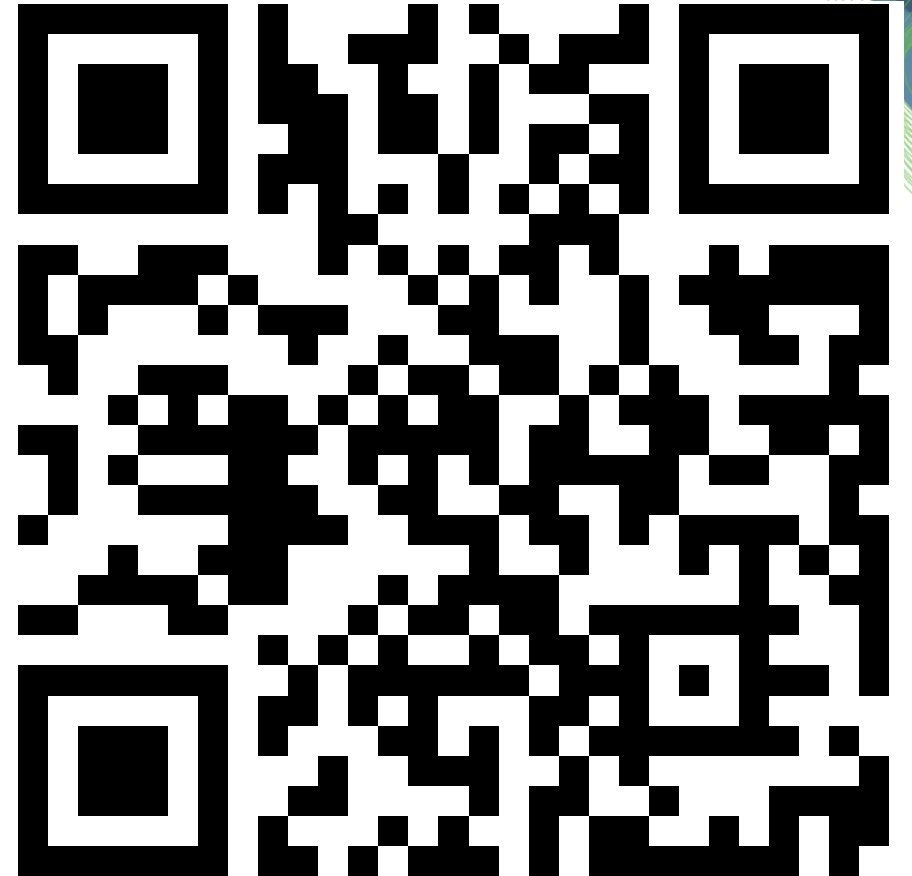


in methods



Summary

- Using AI in [bioimage] data analysis seems more and more unavoidable
- Maintaining good scientific practice is challenging
- Learning how to use AI *responsibly* is crucial
- We are on the edge towards AI-generated, personalized training



<https://doi.org/10.5281/zenodo.15165424>

Acknowledgements

- Communities & platforms



NFDI4
BIOIMAGE



German
BioImaging
Gesellschaft für Mikroskopie und Bildanalyse



image.sc

- Funding



Bundesministerium
für Bildung
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung
aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf
der Grundlage des von den Abgeordneten des Sächsischen
Landtags beschlossenen Haushaltes.



Deutsche
Forschungsgemeinschaft

Chan
Zuckerberg
Initiative 

- BIDS Team @ ScaDS.AI / UL



Mara Lampert, Lea Gihlein, Yixing Huang