

Bio-image Analysis Code Generation

Robert Haase

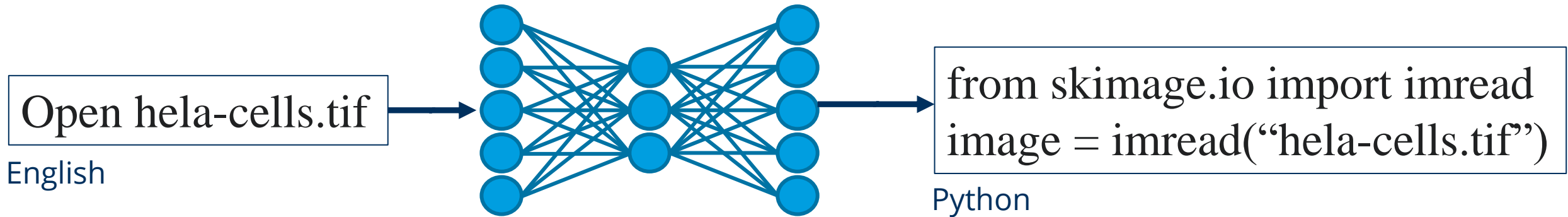


These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license.

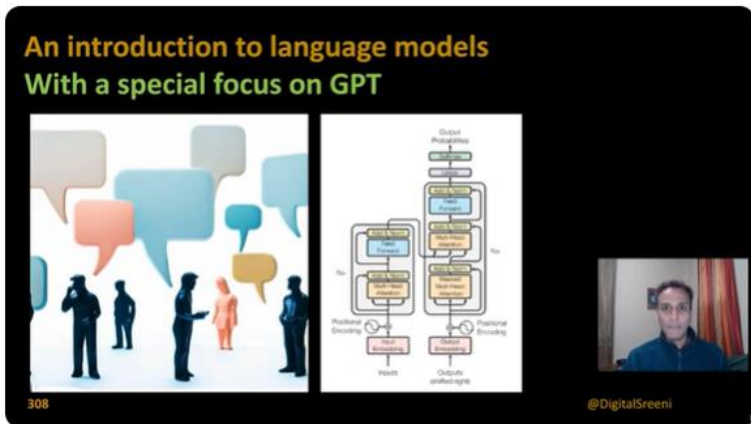
<https://doi.org/10.5281/zenodo.14001044>

Large Language Models (LLMs)

Text-to-text, translation, code generation



Large Language Models (LLMs)

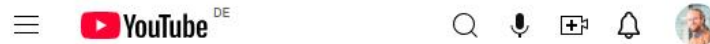


308 - An introduction to language models with focus on GPT



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=9Y7f4j396hl>

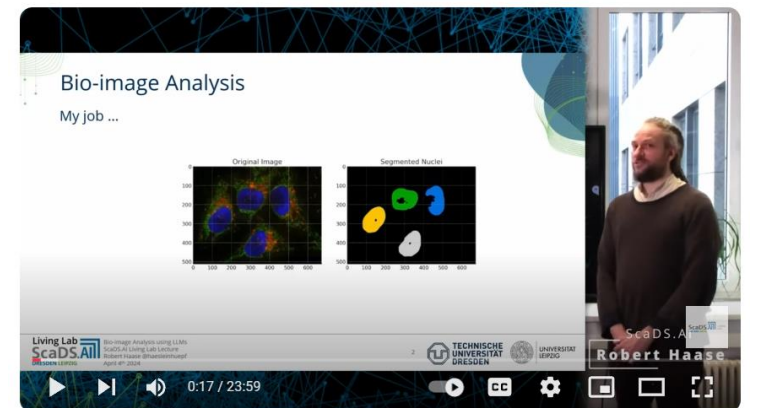


How Large Language Models Work



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>



Lecture: #24 Large Language Models for Bio-image Analysis - ScaDS.AI Dresden/Leipzig



389 views 6 months ago
In this lecture, ScaDS.AI Dresden/Leipzig scientific researcher Dr. Robert Haase talks about Large Language Models for Bio-image Analysis.
...more

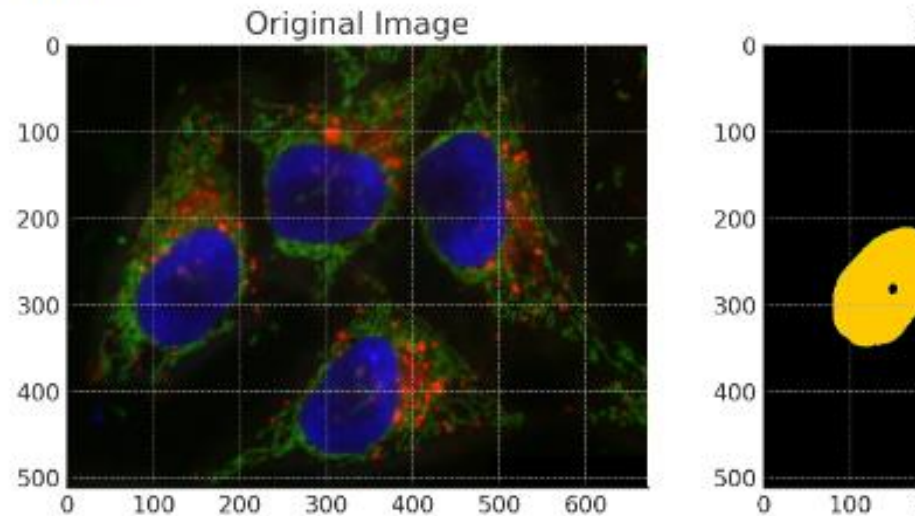
<https://www.youtube.com/watch?v=9dtVIVwk2eg>

Bio-image Analysis using Large-L

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

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 corres
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
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()
```

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

sklearn/ reqs.txt
examples/ setup.cfg
README.rst setup.py



Language Model



Generated PR

+20 -12

sklearn
gradient_boosting.py
helper.py
utils



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^{*1,2} John Yang^{*1,2} Alexander Wettig^{1,2}
Shunyu Yao^{1,2} Kexin Pei³ Ofir Press^{1,2} Karthik Narasimhan^{1,2}

¹Princeton University ²Princeton Language and Intelligence ³University of Chicago



bia-bob and git-bob



Both AI-assistants *can* generate code for bio-image analysis in your browser

- installed on your computer
- on Github.com

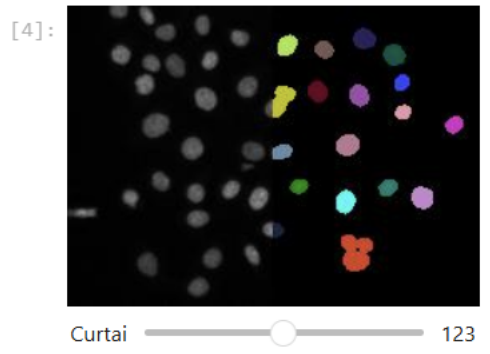
```
[3]: %%bob
I would like to segment nuclei in `image` using Voronoi-Otsu-Labeling.
```

This notebook may contain text, code and images generated by artificial intelligence. Used model: claude-3-5-sonnet-20241022, vision model: claude-3-5-sonnet-20241022, endpoint: None, bia-bob version: 0.23.3. 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>

```
[4]: import napari_segment_blobs_and_things_with_membranes as nsbatwm

# Segment nuclei using Voronoi-Otsu-Labeling
labels = nsbatwm.voronoi_otsu_labeling(image, spot_sigma=4, outline_sigma=1)

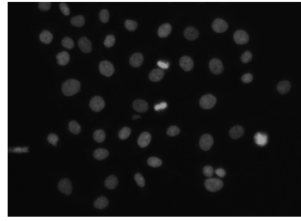
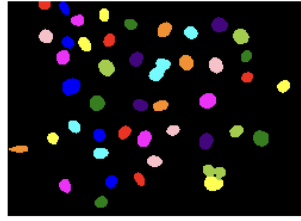
# Show original and segmentation result
stackview.curtain(image, labels)
```



github-actions bot commented now

This message was generated by [git-bob](#) (version: 0.11.2dev, 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 solution for nuclei segmentation using Voronoi-Otsu-Labeling by implementing a Jupyter notebook that processes the input image `human_mitosis_small.png`. The notebook performs the segmentation using `napari-segment-blobs-and-things-with-membranes` library and visualizes the results using `stackview`. The segmentation result was saved as `nuclei_segmentation_result.png`, showing the detected nuclei boundaries. No changes were needed to `requirements.txt` as all necessary libraries were already included.

-  Input image showing nuclei for segmentation analysis.
-  Output image showing the segmented nuclei using Voronoi-Otsu-Labeling.
- [nuclei_segmentation.ipynb](#) New Jupyter notebook implementing the nuclei segmentation workflow.
- [requirements.txt](#) Unchanged file as required libraries were already present.

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 >

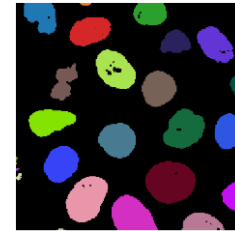
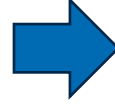
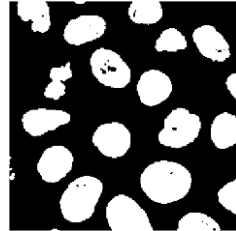
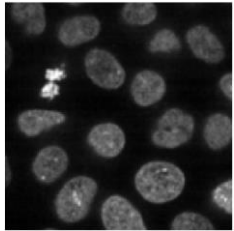
Certainly! Here's a simple Python code snippet that prints the letter "a":

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

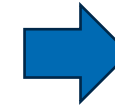
No matter how often you execute this code, the output is always guaranteed to be „a“.

How good are LLMs for Bio-image Analysis?

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



	area
0	955.0
1	31.0
2	815.0
3	1166.0
4	1135.0
	⋮



858.04

How good are 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)
```

```
[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],  
        [1,1,0,0,0],  
        [0,0,0,0,0],  
    ])) == 8
```

Prompt

Reference solution

Unit test (excerpt)

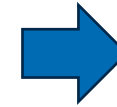
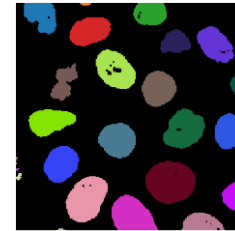
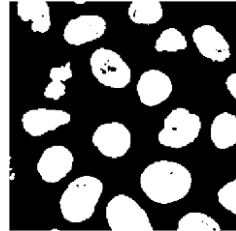
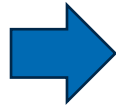
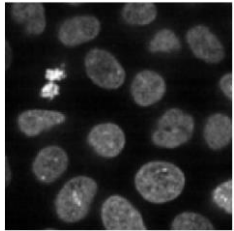
Open for your (!) contributions

18 more currently on their way...

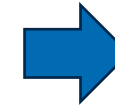
We formulated 57 of such test-cases (yet)

How good are LLMs for Bio-image Analysis?

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



	area
0	955.0
1	31.0
2	815.0
3	1166.0
4	1135.0
⋮	



858.04

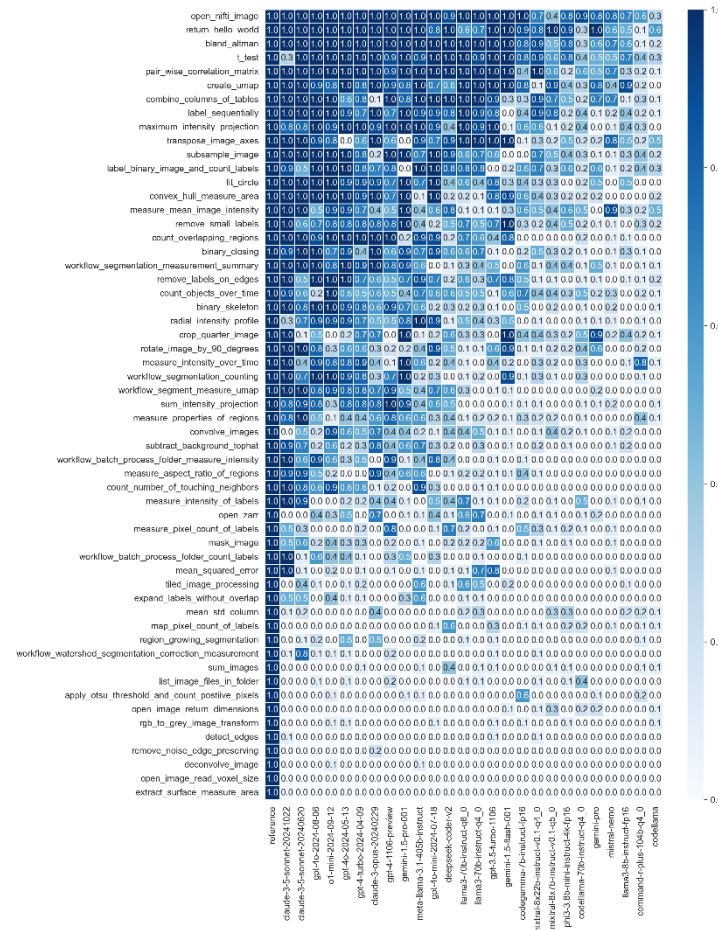
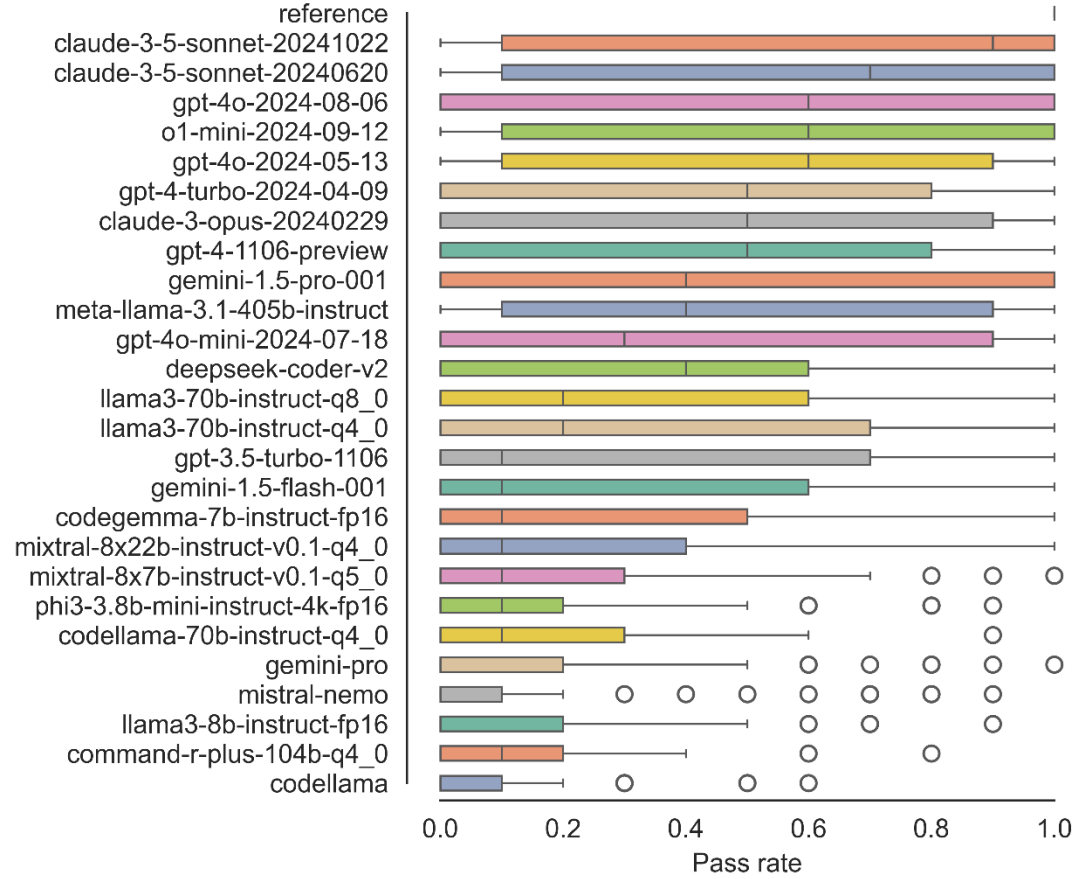
workflow_segmentation_measurement_summary	1.0	1.0	1.0	0.9	1.0	0.8	0.9	0.0	0.3	0.4	0.5	0.0	0.6	0.1	0.4	0.4	0.1	0.5	0.0	0.1	0.1
reference																					
claude-3-5-sonnet-20240620																					
gpt-4o-2024-05-13																					
gpt-4-turbo-2024-04-09																					
claude-3-opus-20240229																					
gpt-4-1106-preview																					
gemini-1.5-pro-001																					
gpt-4o-mini-2024-07-18																					
llama3-70b-instruct-q8_0																					
llama3-70b-instruct-q4_0																					
gpt-3.5-turbo-1106																					
gemini-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																					
gemini-pro																					
llama3-8b-instruct-fp16																					
command-r-plus-104b-q4_0																					
codellama																					

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

Large language models

Benchmarking LLMs for Bio-image Analysis

Summary: 57 use-cases (yet), 24 LLMs (yet), n=10



Prompt Engineering

To further improve
LLMs capabilities

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.



Here is the python code to achieve the task:

```
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

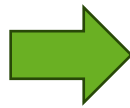
Modify a prompt until it works.

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

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.

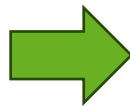
Assume the code is executed in a Jupyter notebook.

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')

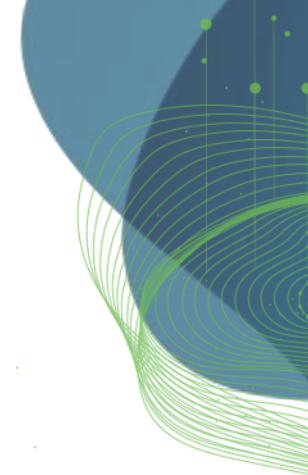
# Threshold using Otsu's method
thresh = filters.threshold_otsu(image)
binary_image = image > thresh

# 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

Scientific code writing: **bia-bob**



- First steps: Keep it simple

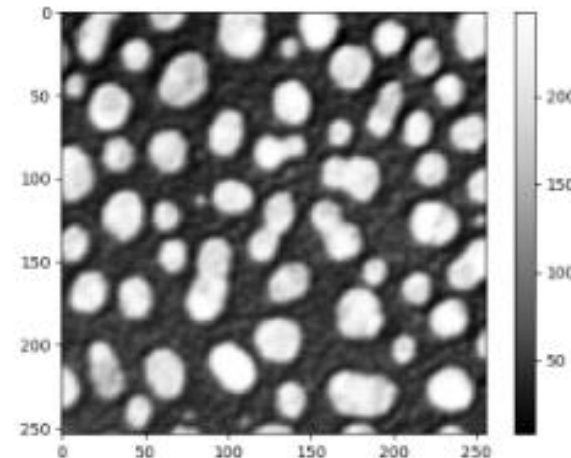
```
%bob Load the blobs.tif image file and show it.
```

I will load the `blobs.tif` image file and then display it using the appropriate visualization function.

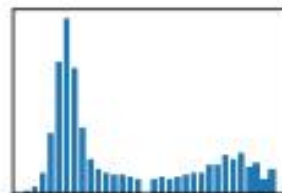
```
from skimage.io import imread
import stackview

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

# Display the image
stackview.insight(image)
```



shape	(254, 256)
dtype	uint8
size	63.5 kB
min	8
max	248



Scientific code writing: **bia-bob**

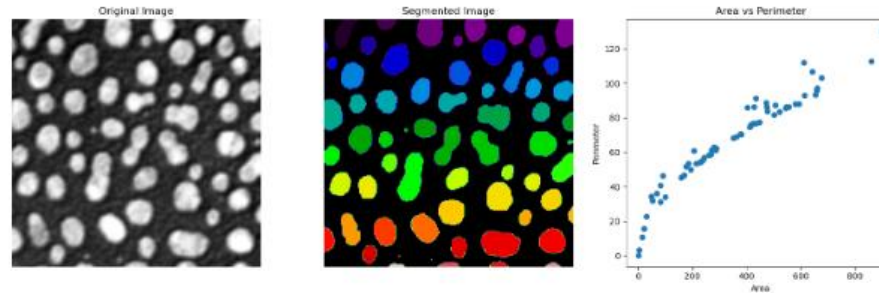


- After learning *how to prompt* for code: try more complex workflows

```
[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 pycyclesperanto_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: prompt engineering

Context-dependent / configurable 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:  
```\nfrom skimage.io import imread  
image = imread(filename)\n```\n
```

...


```
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: prompt engineering



## Context-dependent / configurable system prompt

- System message in git-bob config 
- Code hints in issue templates
- Chat history

```
40 - name: Install dependencies
41 run: |
42 python -m pip install --upgrade pip
43 pip install git-bob
44 pip install -e .
45
46
47 - name: Run git-bob
48 env:
49 ANTHROPIC_API_KEY: "${{ secrets.ANTHROPIC_API_KEY }}"
50
51
52
53
54
55
56
57
58 SYSTEM_MESSAGE: |
59 You are an extremely skilled python developer. Your name is git-bob. You are sometimes called github-actions bot.
60 You can solve programming tasks and review code.
61 You are an expert when it comes to using the stackview Python library.
62 When you use stackview, you always start by importing the library: `import stackview`.
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109 When asked to solve a specific problem, you keep your code changes minimal and only solve the problem at hand.
110 You cannot execute code.
111 You cannot retrieve information from other sources but from github.com.
112 Do not claim anything that you don't know.
113 If you do not know the answer to a question, just say that you don't know and tag @haesleinhuepf so that he can a
114 In case you are asked to review code, you focus on the quality of the code.
115 VISION_SYSTEM_MESSAGE: |
116 You are an AI-based vision model with excellent skills when it comes to describing image. When describing an imag
117 * What is shown in the image.
118 * If the image shows clearly distinct objects in its channels, these structures are listed for each channel indiv
119 * You speculate how the image was acquired.
```

# Under the hood: prompt engineering



Context-dependent /  
**configurable** system  
prompt

- System message in git-bob config
- Code hints in issue templates
- Chat history

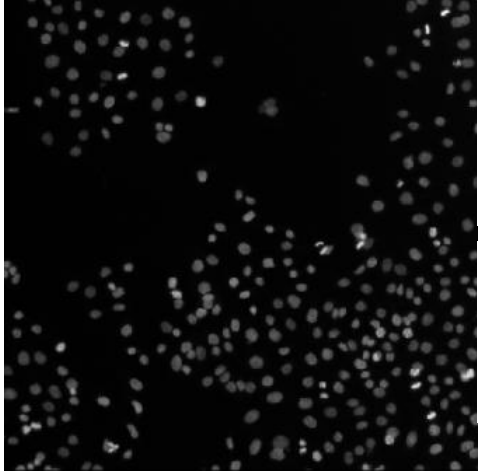
The screenshot shows a GitHub issue template for 'Biolmage Analysis'. The title is 'Issue: Biolmage Analysis'. Below the title, there is a description: 'Submit a microscopy image for AI-assisted analysis. If this doesn't look right, choose a different type.' There are input fields for 'Add a title' (containing '[Biolmage Analysis]:') and 'Add a description'. The description field has a 'Write' tab selected. Below the description field, there are sections for 'Analysis Goal (What should be done / analysed?)', 'Image Upload' (with a drag-and-drop instruction), 'Python Tools' (with a list of libraries: numpy, scikit-image, stackview, scipy), and a 'Note' about data privacy. At the bottom, there is a note about the review process and a link to 'git-bob, an AI-assistant for image analysis suggestions'.

## Viewing images using stackview

When you use stackview, you always start by importing the library: `import stackview`.

- Showing an image stored in variable `image` and a segmented image stored in variable `labels` on top with animated blending. Also works with two images or two label images.  
`stackview.animate_curtain(image, labels)`
- Showing an animation / timelapse image stored in variable `image`.  
`stackview.animate(image)`
- Save an animation / timelapse stored in variable `image` with specified frame delay to a file.  
`stackview.animate(image, filename="output.gif", frame_delay_ms=100)`
- Display an image stored in a variable `image` (this also works with label images). Prefer `stackview.insight` over `matplotlib.pyplot.imshow!`  
`stackview.insight(image)`
- Display an image as a label image explicitly.  
`stackview.imshow(image, labels=True)`

# Including vision language models (VLMs)



```
[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?
```

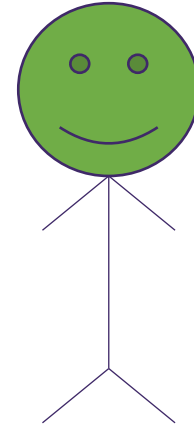
A deep-learning-based Python library that could be used to segment the nuclei is Cellpose.

```
[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!

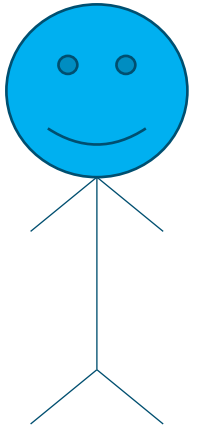


That's right!



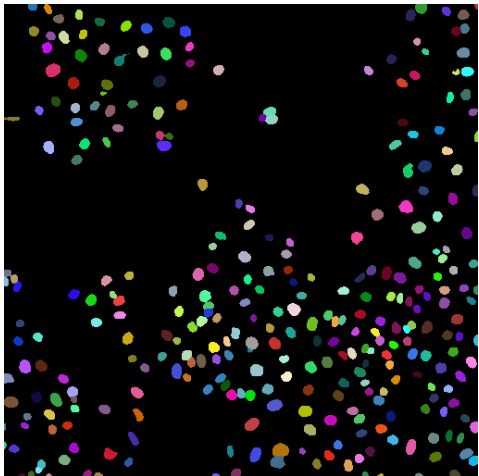
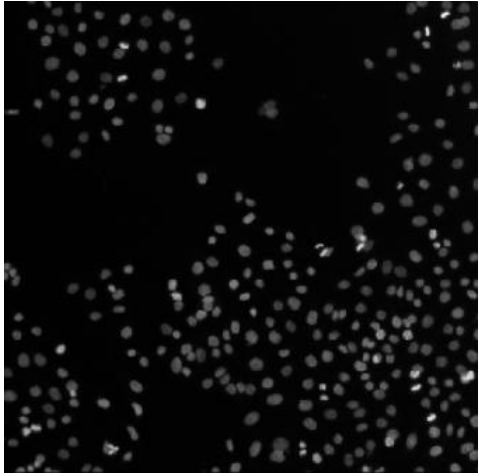
Biologist

That makes sense!



Computer scientist

# Including vision language models (VLMs)



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

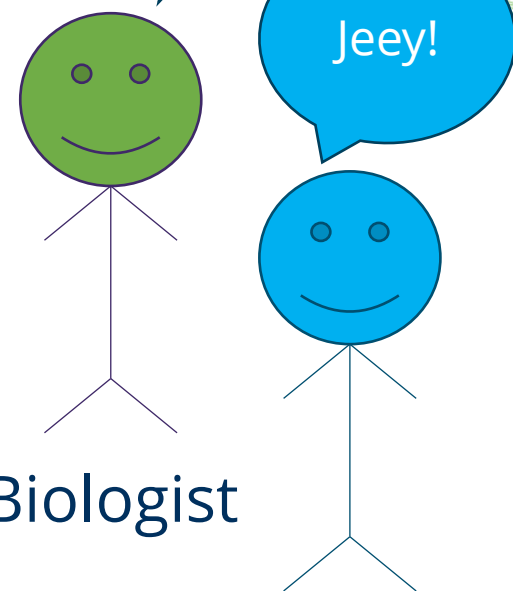
Load the pre-trained model
model = models.Cellpose(model_type='nuclei')

[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.



Biologist

Computer scientist



# Including vision language models (VLMs)

## [BioImage Analysis]: Cell segmentation - algorithm selection #78

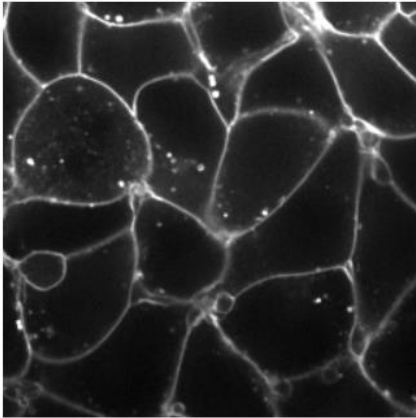
Open haesleinhuepf opened this issue 6 minutes ago · 1 comment



haesleinhuepf commented 6 minutes ago

Owner

What algorithms / Python packages would be suited for segmenting cells in this kind of microscopy image?



git-bob answer



github-actions bot commented 5 minutes ago

Contributor

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.

Hi! 🙌

For segmenting cells in membrane-stained microscopy images like this, I recommend trying these approaches:

1. Using `scikit-image`'s watershed segmentation with markers from cell centers would work well here. Here's a quick example notebook:

```
import numpy as np
from skimage import io, filters, segmentation, feature
```



2. Alternatively, you could try Cellpose which is specialized for cell segmentation tasks: <https://github.com/mouseland/cellpose>

Let me know if you'd like me to elaborate on any of these approaches!

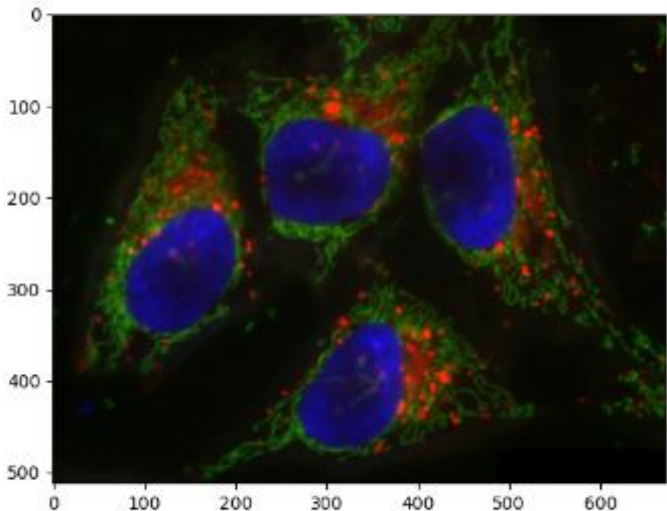
Best regards,  
git-bob



# Generating notebooks using vision models

Ask %%bob to generate a Jupyter notebook

```
hela_cells = imread("hela-cells-8bit.tif")
stackview.insight(hela_cells)
```



```
shape (512, 672, 3)
dtype uint8
size 1008.0 kB
min 0
max 255
```



Present Bob an image like this

```
%%bob hela_cells
Please write a new Jupyter notebook for processing this image.
Its filename is `hela-cells-8bit.tif`.
At the beginning of the notebook describe the image it is made for.
I would like to segment the objects in the blue channel.
Write Python-code for doing this and please add explanatory notebook
cells in between explaining what you're doing in detail as I'm a
Python-beginner.

Thanks :-)
```

A notebook has been saved as `segmenting_hela_cells_blue_channel.ipynb`.



# Generating notebooks using vision models



The screenshot displays a Jupyter Notebook interface with two panes. The left pane, titled 'generate\_notebooks.ipynb', shows the process of generating a notebook. It includes a code cell with the following Python code:

```
[3]: hela_cells = imread("hela-cells-8bit.tif")
stackview.insight(hela_cells)
```

The output of this cell shows a 3D visualization of HeLa cells with a histogram on the right. The histogram shows the distribution of pixel values across the image. Below the visualization, a code cell shows the interaction with the 'bob' model:

```
[4]: %%bob hela_cells
```

The model's response is: "Please write a new Jupyter notebook for processing this image. Its filename is 'hela-cells-8bit.tif'. At the beginning of the notebook describe the image it is made for. I would like to segment the objects in the blue channel. Write Python-code for doing this and please add explanatory notebook cells in between explaining what you're doing in detail as I'm a Python-beginner. Thanks :-)"

The right pane, titled 'segmenting\_hela\_cells\_blue.ipynb', shows the generated notebook content. It includes a title 'HeLa Cells Image Processing', an introduction to the notebook's purpose, and a code cell for importing necessary libraries:

```
[1]: import numpy as np
import matplotlib.pyplot as plt
from skimage import io, filters, measure, morphology, color
```

The code cell also includes comments and a call to display the original image:

```
Set a figure size
plt.rcParams['figure.figsize'] = [10, 8]

Read the image
image = io.imread('hela-cells-8bit.tif')

Display the original image
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()
```

The output of this code cell shows the original image of HeLa cells.



# Initializing bia-bob

When you run it for the first time, and want to use ...

- OpenAI GPT 4omni (optional, as it's default)

```
from bia_bob import bob
bob.initialize("gpt-4o-2024-08-06", vision_model="gpt-4o-2024-08-06")
```

- Anthropic claude

```
from bia_bob import bob
bob.initialize(model="claude-3-5-sonnet-20240620", vision_model="claude-3-5-sonnet-20240620")
```

- Github Models

```
bob.initialize(
 endpoint='github_models',
 model='Phi-3.5-mini-instruct')
```

- KISSKI Models

```
from bia_bob import bob
bob.initialize(endpoint="https://chat-ai.academiccloud.de/v1",
 model="meta-llama-3.1-70b-instruct")
```

# Scientific code writing

## Reproducibility

- Who has written code when and why?
- What were results of the code in its first version?
- If code was modified, what did the developers discuss in that context?
- How do we integrate AI in our coding workflows?

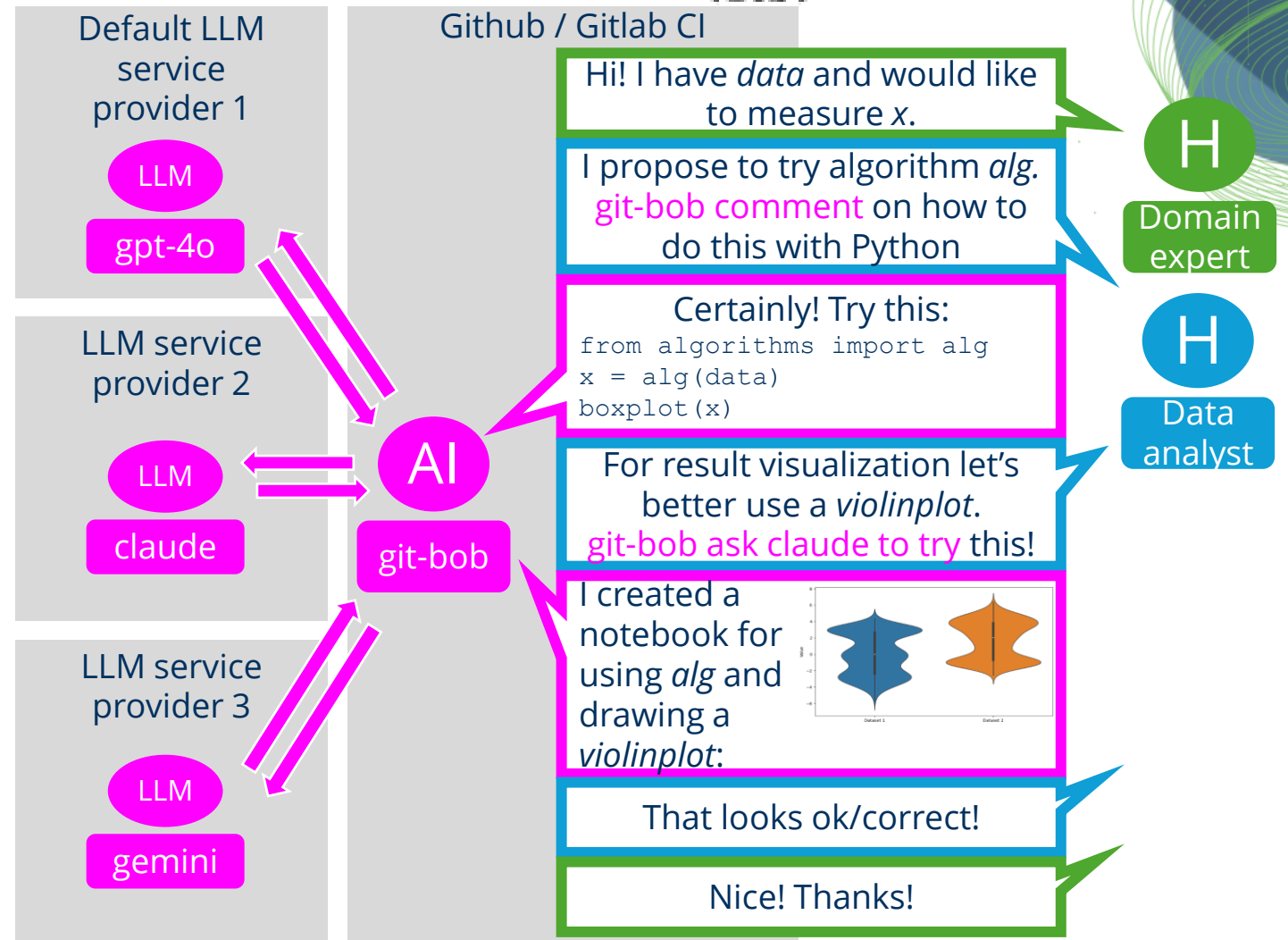
git can help with this

# Scientific code writing: git-bob



## Reproducibility

- Who has written code when and why?
- What were results of the code in its first version?
- If code was modified, what did the developers discuss in that context?
- How do we integrate AI in our coding workflows?



# Scientific code writing: git-bob



## Reproducibility

- Who has written code when and why?
- What were results of the code in its first version?
- If code was modified, what did the developers discuss in that context?
- How do we integrate AI in our coding workflows?

Iterative programming instead of recursion.

Open

haesleinhuepf opened this issue now · 0 comments



haesleinhuepf commented now

Owner ...

In playground/test.py there is a function that computes the faculty of a number. It uses recursion. It would be great if the function worked iteratively instead.



haesleinhuepf commented now

Owner Author ...

git-bob think about this.



github-actions[bot] reacted with thumbs up emoji

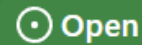
# Scientific code writing: git-bob



## Reproducibility

- Who has written code when and why?
- What were results of the code in its first version?
- If code was modified, what did the developers discuss in that context?
- How do we integrate AI in our coding workflows?

## Iterative programming instead of recursion.



haesleinhuepf opened this issue 2 minutes ago · 2 comments



github-actions bot commented 2 minutes ago

Contributor

This message was generated by [git-bob](#) (version: 0.2.6, model: gpt-4o-2024-08-06), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

```
def faculty(number):
 result = 1
 for i in range(1, number + 1):
 result *= i
 return result

for i in range(10):
 print("The faculty of", i, "is", faculty(i))
```


# Scientific code writing: git-bob




## Reproducibility

- Who has written code when and why?
- What were results of the code in its first version?
- If code was modified, what did the developers discuss in that context?
- How do we integrate AI in our coding workflows?


## Iterative programming instead of recursion.



 haesleinhuepf opened this issue 6 minutes ago · 3 comments · May be fixed by #232

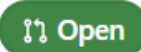
• • •

 haesleinhuepf commented now Owner Author ...

git-bob implement this!



  github-actions (bot) mentioned this issue now

Replace recursive faculty computation with an iterative method in playground/test.py. #232 

# Scientific code writing: git-bob



## Reproducibility

- Who has written code when and why?
- What were results of the code in its first version?
- If code was modified, what did the developers discuss in that context?
- How do we integrate AI in our coding workflows?

Replace recursive faculty computation with an iterative method in playground/test.py. #232

Open github-actions wants to merge 1 commit into main from git-bob-mod-sesEhgK1ld

Conversation 0 Commits 1 Checks 0 Files changed 1

github-actions bot commented 1 min

This message was generated by [git-bob](#) (version: 0.2) assistant. It can make mistakes and has [limitations](#). C

Replaced the recursive implementation of the `faculty` function in `playground/test.py` with an iterative approach by avoiding the limits and overhead associated for large inputs, while maintaining the function number.

closes [#231](#)

```
@@ -1,8 +1,8 @@
1 1 def faculty(number):
2 - if number == 0:
3 - return 1
4 - else:
5 - return number * faculty(number - 1)
2 + result = 1
3 + for i in range(1, number + 1):
4 + result *= i
5 + return result
```

Replaced the recursive function with function to compute...

Human peer reviewing AI work





# Trigger words

**git-bob comment** (aliases: **answer, think about, respond, review**)

- Only responds in text-form

**git-bob try**

- Generates + runs code, shows resulting images

**git-bob solve** (aliases: **implement, apply**)

- Generates + runs code + sends pull-request

**git-bob ask ... to ...**


- Answer using a specified LLMs.

Example: **git-bob ask claude-3-5-20241022 to solve this!**

# Reviewing pull-requests: git-bob






Conversation 1    Commits 1    Checks 1    Files changed 3


 **haesleinhuepf** commented on Aug 1 Owner

With this I'm adding support for solving github issues

git-bob comment



  enable solving github issues 7e7e0a

 **github-actions** (bot) commented on Aug 1 Contributor

This comment was generated by [git-bob](#), an AI-based assistant.

Thank you for the pull request!

The newly added functionalities and improvements significantly extend our ability to interact with GitHub issues programmatically. Here are some observations and comments:

- Dependencies:**
  - Added `blablado` to `install_requires` in `setup.cfg`. Ensure this addition aligns with the overall project requirements and is properly documented.
- Code Changes in `_github_utilities.py`:**
  - The new utility functions are well-documented and provide essential capabilities to list issues, get detailed issue information, list repository files, retrieve file contents, update files in a new branch, send pull requests, and solve GitHub issues.
  - Ensure that environment variables like `GITHUB_API_KEY` are managed securely and that users are instructed on how to set them up.
- Enhancement in `_terminal.py`:**
  - The change to identify "git-bob solve" as a command for initiating issue resolution is a valuable addition. This allows for more dynamic interaction with GitHub issues through the terminal.
  - The `solve_github_issue` function within the CLI context adds great value for automated issue resolution.
- Suggestions:**
  - Consider adding unit tests for the new functionalities to ensure they work as expected and to maintain robustness.

...adds valuable functionality. Thanks again for your

AI peer reviewing human work

#### 4. Suggestions:

- Consider adding unit tests for the new functionalities to ensure they work as expected and to maintain robustness.



# Scientific code writing: **git-bob**

“I will document the code later.”, finished PhD student just before leaving the lab.



# Scientific code writing: **git-bob**

~~“I will document the code later.”~~

“**A**I will document the code later.”



haesleinhuepf commented now

It would be great to have numpy-style docstrings for all functions in the file `src/git_bob/_github_utilities.py`

`git-bob solve`



Open github-actions wants to merge 1 commit into `main` from `mod-cnqvE9n552`

Conversation 0 Commits 1 Checks 0 Files changed 1



github-actions bot commented now

This comment was generated by `git-bob`, an AI-based assistant.

This pull request adds numpy-style docstrings to all functions in the file `src/git_bob/_github_utilities.py` to address issue [#22](#).

closes [#22](#)



Update file content

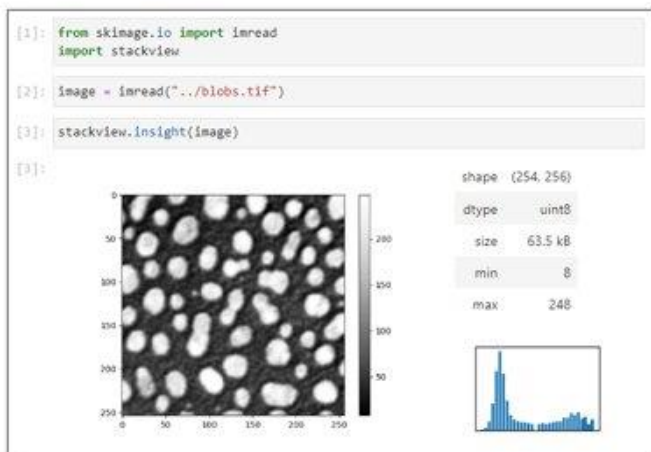
Verified

8304d14



# Optimizing documentation

E.g. to make code more readable



```
from bia_bob import bob
```

```
%%bob modify the notebook `mod_notebook.ipynb` by
adding explanatory markdown cells between the code cells.
Also add a markdown cell at the very beginning that
introduces the entire notebook shortly.
```

The modified notebook has been saved as `mod_notebook_1.ipynb`.

## Introduction

In this notebook, we will demonstrate how to load a bioimage, visualize it using stackview, and explore the image interactively. We will be using an image of blobs for this example. The steps include loading the image and visualizing it using different stackview functionalities.

## Step 1: Import necessary libraries

We will start by importing the necessary libraries. In this case, `skimage.io` for loading the image and `stackview` for visualization.

```
from skimage.io import imread
import stackview
```

## Step 2: Load the image

Next, we will load the image file into a variable. The image used in this notebook is `blobs.tif`.

```
image = imread("../blobs.tif")
```

## Step 3: Visualize the image using stackview insight

We will use `stackview.insight` to visualize the loaded image. This function allows for inspecting the image

```
stackview.insight(image)
```

shape (254, 256)  
dtype uint8  
size 63.5 kB  
min 8  
max 248

# Translating notebooks



"Please translate notebook <xyz.ipynb> to <language>."

**10\_nd\_image\_data.ipynb**

### Multidimensional image stacks

Multidimensional image data can be handled in a similar way to multi-channel image data.

### Three-dimensional image stacks

There are also images with three spatial dimensions: X, Y, and Z. Typical examples in microscopy and in medical imaging include a look at an Magnetic Resonance Imaging (MRI) data set.

```
image.io import imread
skimage import imshow
matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tf13d1.tif')
```

### Image slicing

Since three dimensions are spatial dimensions, we can also make slices orthogonal to the image plane and corresponding to the anatomical planes. To orient the images correctly, we can transpose their axes by adding `[:, :, T]` by the end.

```
sagittal = image_stack[:, :, 128].T
coronal = image_stack[:, 128, :].T
transverse = image_stack[96]
```

```
fig, axs = plt.subplots(1, 3, figsize=(15,15))

show orthogonal planes
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagittal')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transverse');
```



### Videos

If an image dataset has a temporal dimension, we call it a video. Processing videos works similar to multi-channel images and image stacks. Let's open a microscopy dataset showing yeast

**10\_nd\_image\_data.de.ipynb**

### Multidimensional Bildstapel

Multidimensionale Bilddaten können ähnlich wie mehrkanalige Bilddaten behandelt werden.

### Dreidimensionale Bildstapel

Es gibt auch Bilder mit drei räumlichen Dimensionen: X, Y und Z. Typische Beispiele finden sich in der Mikroskopie und in der medizinischen Bildgebung. Schauen wir uns ein Magnetresonanztomographie (MRT) Datensatz an:

```
[1]: from skimage.io import imread
from stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tf13d1.tif')
```

### Bildschnitte

Da alle drei Dimensionen räumliche Dimensionen sind, können wir auch Schnitte orthogonal zur Bildebene machen, die den Anatomischen Ebenen entsprechen. Um die Bilder korrekt zu orientieren, können wir ihre Achsen transponieren, indem wir `[:, :, T]` hinzufügen.

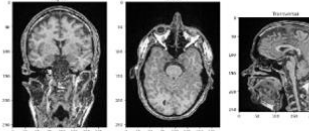
```
[2]: sagittal = image_stack[:, :, 128].T
coronal = image_stack[:, 128, :].T
transverse = image_stack[96]

fig, axs = plt.subplots(1, 3, figsize=(15,15))

orthogonale Ebenen anzeigen
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagittal')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transversal');
```



### Videos

Wenn ein Bilddatensatz eine zeitliche Dimension hat, nennen wir ihn ein Video. Die Verarbeitung von Videos funktioniert ähnlich wie die mehrkanaliger Bilder und Bildstapel. Öffnen wir einen Mikroskopie-

**10\_nd\_image\_data.es.ipynb**

### Pilas de imágenes multidimensionales

Los datos de imágenes multidimensionales se pueden manejar de manera similar a los datos de imágenes multicanal.

### Pilas de imágenes tridimensionales

También hay imágenes con tres dimensiones espaciales: X, Y y Z. Puedes encontrar ejemplos típicos en microscopía y en imágenes médicas. Echemos un vistazo a un conjunto de datos de imágenes por Resonancia Magnética (IRM):

```
[1]: from skimage.io import imread
from stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tf13d1.tif')
```

### Corte de imágenes

Como las tres dimensiones son dimensiones espaciales, también podemos hacer cortes ortogonales al plano de la imagen y correspondientes a Planos Anatómicos. Para orientar correctamente las imágenes, podemos transponer sus ejes añadiendo `[:, :, T]` al final.

```
[2]: sagittal = image_stack[:, :, 128].T
coronal = image_stack[:, 128, :].T
transverse = image_stack[96]

fig, axs = plt.subplots(1, 3, figsize=(15,15))

mostrar planos ortogonales
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagittal')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transversal');
```



### Videos

Si un conjunto de datos de imágenes tiene una dimensión temporal, lo llamamos video. Procesar videos funciona de manera similar a las imágenes multicanal u a las pilas de imágenes. Vamos a abrir un

**10\_nd\_image\_data.fr.ipynb**

### Piles d'images multidimensionnelles

Les données d'images multidimensionnelles peuvent être traitées de manière similaire aux données d'images multi-canaux.

### Piles d'images tridimensionnelles

Il existe aussi des images avec trois dimensions spatiales: X, Y et Z. Vous trouvez des exemples typiques en microscopie et en imagerie médicale. Regardons un ensemble de données d'imagerie par résonance magnétique (IRM):

```
[1]: from skimage.io import imread
from stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tf13d1.tif')
```

### Coupe d'image

Comme les trois dimensions sont des dimensions spatiales, nous pouvons également faire des coupes orthogonales au plan de l'image et correspondant aux plans anatomiques. Pour orienter correctement les images, nous pouvons transposer leurs axes en ajoutant `[:, :, T]` à la fin.

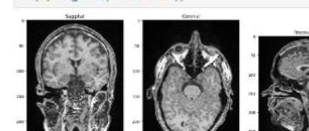
```
[2]: sagittal = image_stack[:, :, 128].T
coronal = image_stack[:, 128, :].T
transverse = image_stack[96]

fig, axs = plt.subplots(1, 3, figsize=(15,15))

montrer des plans orthogonaux
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagittal')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transverse');
```



### Vidéos

Si un ensemble de données d'image a une dimension temporelle, nous l'appelons une vidéo. Le traitement des vidéos fonctionne de



# bia-bob versus git-bob



- [image] data analysis tasks
- Close interaction with data
- HPC clusters, notebooks in the cloud

- Sustainable software development
- Python library maintenance
- No local hardware / cluster needed

```
[16]: %%bob
Write a function that does PCA with all numerical columns in
big_dataframe and add the result as new columns the dataframe
```

This notebook may contain text, code and images generated by artificial intelligence. Used model: gpt-4o-2024-08-06, vision model: gpt-4o-2024-08-06, endpoint: None, bia-bob version: 0.22.0.. 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>

```
[17]: from sklearn.decomposition import PCA

def add_pca_components_to_df(df, n_components=2):
 # Select numeric columns
 numeric_df = df.select_dtypes(include='number')

 # Perform PCA
 pca = PCA(n_components=n_components)
 pca_result = pca.fit_transform(numeric_df)

 # Add PCA results as new columns to the original dataframe
 for i in range(n_components):
 df[f'PCA_{i+1}'] = pca_result[:, i]

 return df
```



haesleinhuepf commented 2 minutes ago

There are no comments in playground/test.py

git-bob solve





github-actions bot mentioned this issue 1 minute ago

Add comments to playground/test.py

#21

Open

# Comparison with ChatGPT and friends

	ChatGPT*	 bia-bob	 git-bob
• Can generate code for bio-image analysis	✓	✓	✓
• Can search the internet	✓	✗	✗
• Knows what variables, functions and libraries are used in your code.	✗	✓	✓
• Can involve multiple LLMs / agents in one session	✗	✓	✓
• Can involve multiple humans in none session	✗	✗	✓
• Costs	0-20\$ / month	< 0.05\$ / session	< 0.05\$ / session



# Generating stories



## Bio-image analysis unrelated



haesleinhuepf commented 24 minutes ago

Owner ...

Write a short story about the adventures of the rabbit in the forest. The rabbit finds the golden carrot and shares it with its friends. Also paint a picture illustrating the story.

story-teller try to do this!



git-bob`s use-cases range much wider than bia-bob`s.



github-actions commented 22 minutes ago

Contributor ...

This message was generated by story-teller / git-bob (version: 0.1.1) using model: claude-3.5-sonnet-20241022. [Log] as experimental AI-based assistant. It can make mistakes and has limitations. Check its messages carefully.

I created a delightful children's story about a rabbit who discovers a magical golden carrot in the forest and learns the importance of sharing with friends. To bring the story to life, I painted a cheerful illustration showing a white rabbit holding a golden carrot, surrounded by woodland friends in a colorful forest setting, perfect for engaging young readers aged around 5 years old.

- [rabbit\\_story.md](#) Created a new children's story about a rabbit finding and sharing a golden carrot with forest friends.



Draw a colorful illustration of a white rabbit holding a golden carrot in the forest with animal friends.

git-bob-playground / rabbit\_story.md

github-actions[bot] Created a heartwarming story about sh... be9ef4f · 24 minutes ago

15 lines (8 loc) · 1013 Bytes

Preview Code Blame



## The Magical Golden Carrot

Once upon a time, in a lush green forest, there lived a tiny white rabbit named Hoppy. One sunny morning, while hopping through the forest, Hoppy spotted something glowing behind a bush. To his surprise, it was a sparkling golden carrot!

"Oh my!" exclaimed Hoppy, his pink nose twitching with excitement. "I've never seen anything so beautiful!"

But instead of eating it all by himself, Hoppy had a wonderful idea. He called all his forest friends - deer, squirrel, and little mouse - to share his magical discovery.

Together, they had a wonderful picnic, sharing pieces of the golden carrot. As they ate, the carrot made them all glow with happiness and friendship. From that day on, they remained the best of friends, always sharing their special finds in the forest.

[A cheerful illustration showing a white rabbit surrounded by forest animals, all sharing a glowing golden carrot under a big tree. The scene is filled with warm colors and friendly smiles.]

The End

# 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



# Important note

Whatever you enter in #biabob / #gitbob / ChatGPT / Claude / Gemini / any other LLM-based system will be sent over the internet to people you don't know.

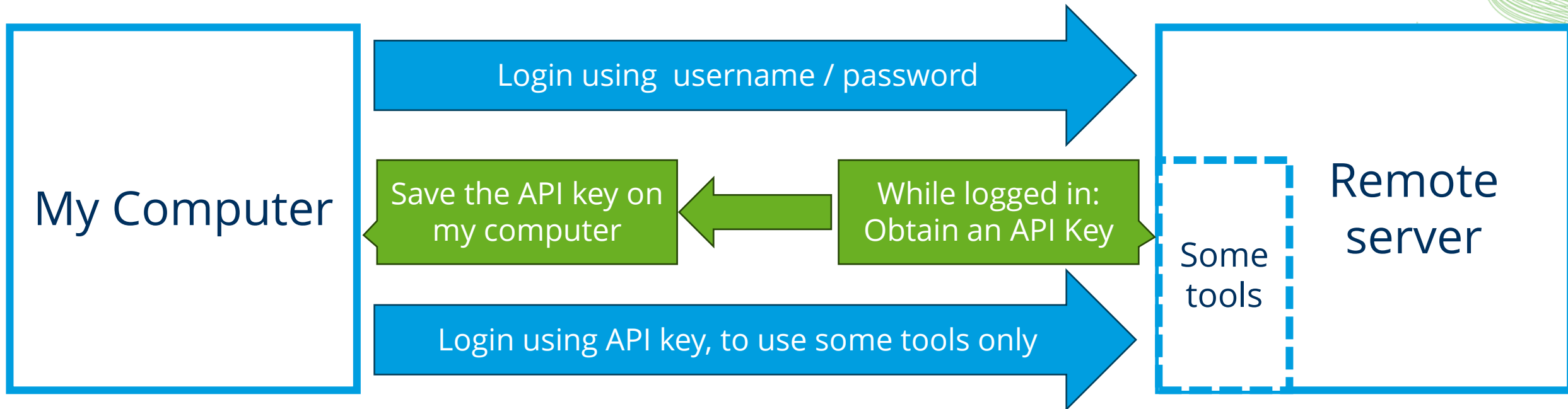
**DO NOT enter private / sensitive / personal information.**

# Installation: bia-bob



# 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  
(unless you trust them)

API key can be revoked /  
invalidated any time,

# Anthropic / Claude API Key

To use claude, you need an API Key from Anthropic (usage may cost money)

The image shows three sequential browser screenshots illustrating the steps to create an Anthropic API key:

- Step 1:** The browser is at `anthropic.com/api`. The page features the heading "Build with Claude" and a "Start building" button highlighted with a yellow arrow labeled "1".
- Step 2:** The browser is at `console.anthropic.com/dashboard`. The dashboard shows a "Get API keys" button highlighted with a yellow arrow labeled "2".
- Step 3:** The browser is at `console.anthropic.com/settings/keys`. The "API keys" section is visible, with a "+ Create Key" button highlighted by a yellow arrow labeled "3".

# OpenAI API Key

Create an OpenAI API Key (usage may cost money)

The image displays two browser screenshots illustrating the steps to create an OpenAI API key. The left screenshot shows the OpenAI API landing page with a yellow arrow labeled '1' pointing to the 'Sign up' button and a yellow arrow labeled '3' pointing to the 'API keys' link in the sidebar. The right screenshot shows the 'API keys' management page with a yellow arrow labeled '2' pointing to the 'API keys' link in the sidebar and a yellow arrow labeled '4' pointing to the '+ Create new secret key' button.



# GitHub Models API Key

Create a GitHub Models API Key (usage may cost money)

The image shows a sequence of three browser screenshots illustrating the steps to create a GitHub Models API key:

- Step 1:** The first screenshot shows the GitHub Marketplace homepage. A yellow arrow labeled '1' points to the 'OpenAI GPT-4o' model card in the 'Models' section.
- Step 2:** The second screenshot shows the 'OpenAI GPT-4o' model page. A yellow arrow labeled '2' points to the 'Get started' button.
- Step 3:** The third screenshot shows the 'Get started' sidebar on the model page. A yellow arrow labeled '3' points to the 'Go to personal access tokens' button.

# GitHub Models API Key

## Create a GitHub Models API Key

Free service by Github /  
Microsoft Azure

No need to  
configure any  
rights. Just create a  
key with default  
settings.

Marketplace Personal Access Tokens (Classic)

github.com/settings/tokens

Settings / Developer Settings

Personal access tokens (classic) Generate new token

Tokens you have generated that can be used to access the [GitHub API](#).

Token Name	Expiration Date	Last Used	Action
llm4 — repo	Expires on Tue, Dec 31 2024.		
llm3 — repo	Expired on Sat, Aug 24 2024.	Last used within the last 2 months	Delete
llm2 — repo	Expired on Fri, Aug 23 2024.	Last used within the last 2 months	Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

# KISSKI / GWDG API Keys

## Create an KISSKI API Key



Free service by  
German Academics

**KISSKI**  
KI-Servicezentrum für sensible und kritische Infrastrukturen

About us Target Groups Services News DE EN

### Chat AI

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 in Lower Saxony or members of Max Planck Institutes, the free use of OpenAI GPT-3.5 and OpenAI GPT-4 is also possible. When using our internal models, your conversations are never stored on the server side. With the OpenAI models, Microsoft may store your conversations for up to 30 days to prevent abuse. However, your data is never used for training or similar purposes. Hosting your own models is also possible on request.

We also offer fine-tuning of your models and Retrieval Augmented Generation (RAG), an effective and cost-efficient technique to provide an LLM with up-to-date access to the knowledge in your documents.

Requirements

Service type

Software

Contact person

Ali Doost Hosseini  
Jonathan Decker

Planned start date

from now on

Chat AI

**KISSKI**  
KI-Servicezentrum für sensible und kritische Infrastrukturen

Home > Services

### LLM Service

Contact Information

Name

Your name

Group leader

Name of the group leader

Working group/ department

Name of the working group/ department you belong to

**KISSKI**  
KI-Servicezentrum für sensible und kritische Infrastrukturen

### Desired service

Service type

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

Requirements

Use LLMs for code-generation and bio-image analysis.

Please briefly describe what you need from us.

# Exercise: Store the API key in your environment

## Add the OPEN AI API Key to your environment variables

Enter  
OPENAI\_API\_KEY  
Or  
ANTHROPIC\_API\_KEY  
Or  
GH\_MODELS\_API\_KEY

Key here

The screenshot shows the Windows System Properties dialog box with the Environment Variables tab selected. The 'User variables for haase' section contains a list of variables, with 'OPENAI\_API\_KEY' highlighted. A 'New System Variable' dialog box is open, showing 'Variable name:' and 'Variable value:' fields. Annotations include: 1. Search for 'env' in the Start menu. 2. Click 'Edit the system environment variables'. 3. Click 'Environment Variables...'. 4. Click 'New...'. Callouts indicate where to enter the API key name and the key value.

Variable	Value
JAVA_HOME	
OneDrive	
OneDriveConsumer	
OPENAI_API_KEY	
Path	
TEMP	
TMP	

Variable	Value
ACSetupSvcPort	
ACSvcPort	
ComSpec	
CUDA_PATH	
CUDA_PATH_V11_2	
CUDA_PATH_V11_7	
DriverData	

# Setup an environment on your computer

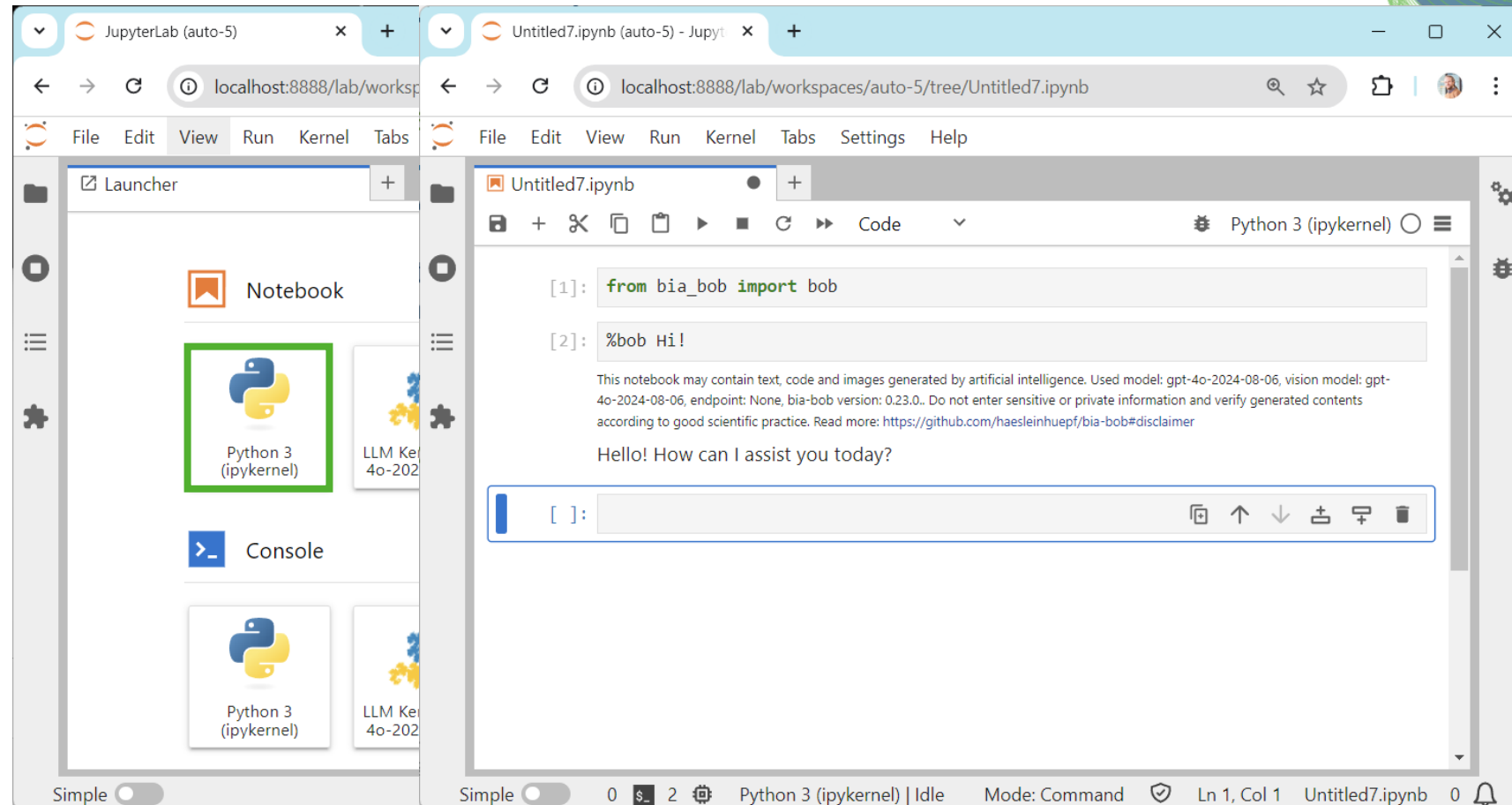
`conda env create -f https://github.com/haesleinhuepf/bia-bob/raw/main/environment.yml`

Setup a new environment using a predefined set of Python image-analysis related libraries

`conda activate bob-env`

Log in to this environment

jupyter lab



The screenshot displays the JupyterLab web interface. On the left is the 'Launcher' view with a 'Python 3 (ipykernel)' environment selected. The main area shows a notebook with two code cells: `from bia_bob import bob` and `%bob Hi!`. The output of the second cell is 'Hello! How can I assist you today?'. A disclaimer is visible below the code cells.

# Bia-bob installation on the web

haesleinhuepf/bia-bob: BIA Bot

github.com/haesleinhuepf/bia-bob?tab=readme-ov-file

README BSD-3-Clause license

## Installation

You can install `bia-bob` using `conda/pip`. It is recommended to use a `conda/mamba` environment. If you have never used `conda`, see [this link](#) **first**.

It is recommended to install `bia-bob` in a `conda-environment` for bio-image analysis.

```
conda env create -f https://github.com/haesleinhuepf/bia-bob/blob/main/conda-environment.yml
```

You can then activate this environment...

```
conda activate bob_env
```

OR install bob into an existing environment:

```
pip install bia-bob
```

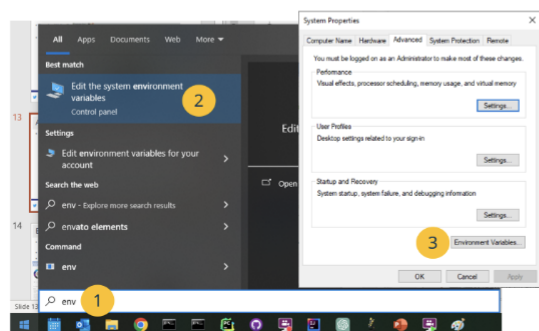
haesleinhuepf/bia-bob: BIA Bot

github.com/haesleinhuepf/bia-bob?tab=readme-ov-file#installation

README BSD-3-Clause license

## Setting API keys

For using LLMs from remote service providers, you need to set an [API key](#) (short cryptic texts such as "proj\_sk\_asdasdasd" which allow you to log into services without entering your username and password. Many online services require a key for billing; others enable you to use their free services only after obtaining a key. In the following sections, you find links to a couple of LLM services providers compatible with `bia-bob`. After obtaining the key, you need to add it to the environment variables of your computer. On Windows, you can do this by 1) searching for "Environment Variables" in the Windows search bar, 2) clicking on "Edit the system environment variables", 3) clicking on "Environment Variables", 4) clicking on "New" in the "System variables" section to create a new variable with the name specified below (e.g. `OPENAI_API_KEY`) and the value of your API key.



haesleinhuepf/bia-bob: BIA Bot

github.com/haesleinhuepf/bia-bob?tab=readme-ov-file#installation

README BSD-3-Clause license

## Using OpenAI / ChatGPT as backend

Create an [OpenAI API Key](#) and add it to your environment variables named `OPENAI_API_KEY` as explained on [this link](#).

You can then initialize Bob like this (optional, as that's default):

```
from bia_bob import bob
bob.initialize("gpt-4o-2024-08-06", vision_model="gpt-4o-2024-08-06")
```

## Using Anthropic / Claude

Create an [Anthropic API Key](#) and add it to your environment variables named `ANTHROPIC_API_KEY`.

You can then initialize Bob like this:

```
from bia_bob import bob
bob.initialize(model="claude-3-5-sonnet-20240620", vision_model="claude-3-5-sonnet-20240620")
```

## Using Github Models Marketplace

If you are using the models from [Github Models Marketplace](#), please [create a GITHUB API key \(with default settings\)](#) and store it for accessing the models in an environment variable named `GITHUB_API_KEY`.

# Installation: git-bob



# Installation: git-bob

Upload `git-bob.yml` file to `.github/workflows/` in your repository.

```
40 - name: Install dependencies
41 run: |
42 python -m pip install --upgrade pip
43 pip install git-bob
44 pip install -e .
45
46
47 - name: Run git-bob
48 env:
49 ANTHROPIC_API_KEY: "${{ secrets.ANTHROPIC_API_KEY }}"
50
51
52
53
54
55
56
57
58 SYSTEM_MESSAGE: |
59 You are an extremely skilled python developer. Your name is git-bob. You are sometimes called github-actions bot.
60 You can solve programming tasks and review code.
61 You are an expert when it comes to using the stackview Python library.
62 When you use stackview, you always start by importing the library: `import stackview`.
```

Consider pinning a specific version here to avoid updates being installed automatically.

Modify the system message to your use-case



# Installation: git-bob

## Configure API keys

The screenshot shows the GitHub repository settings for 'stackview' by user 'haesleinhuepf'. The 'Settings' tab is selected, and the 'Actions secrets / New secret' dialog is open. The dialog contains a 'Name' field with the text 'OPENAI\_API\_KEY' and an empty 'Secret' field. A green 'Add secret' button is at the bottom of the dialog. Below the dialog, a message states 'This repository has no secrets.' with a green 'New repository secret' button. In the background, the 'Actions' menu item is highlighted in the left sidebar, and the 'Secrets and variables' menu item is also visible. Blue arrows point to the 'Settings' tab, the 'Name' field, the 'Secret' field, the 'Actions' menu item, and the 'New repository secret' button.

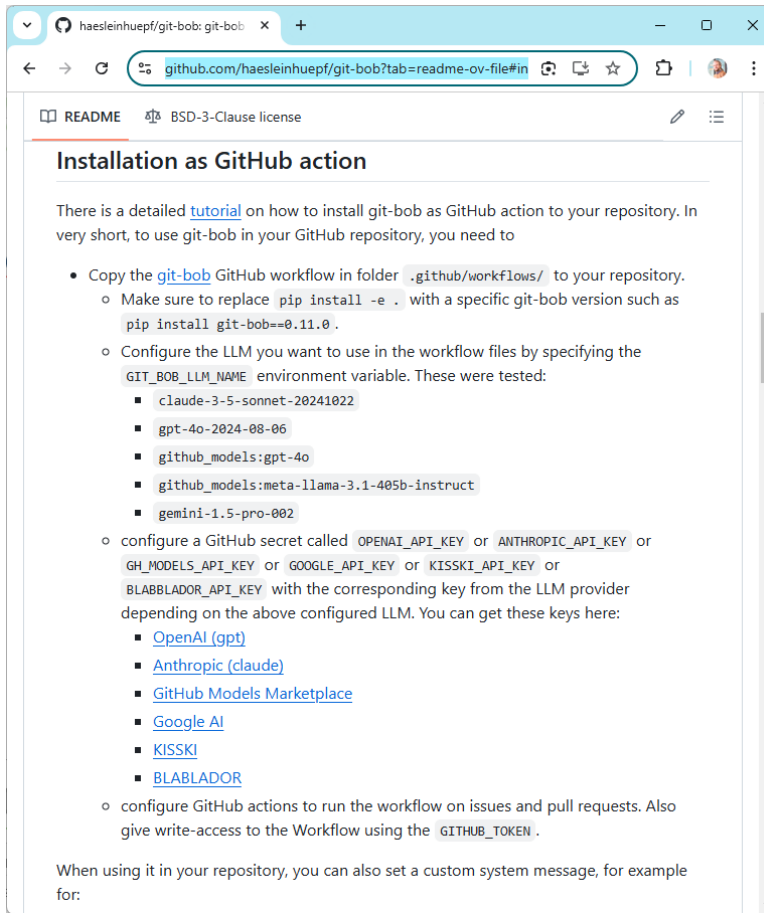
# Installation: git-bob

Set access rights in the Github Repository Settings under *Actions > General > Workflow permissions*

The screenshot displays the GitHub Repository Settings interface. On the left sidebar, the 'Actions' menu is expanded, and the 'General' sub-menu is selected, indicated by a blue arrow. The main content area is titled 'Workflow permissions' and contains the following text: 'Choose the default permissions granted to the GITHUB\_TOKEN when running workflows in this repository. You can specify more granular permissions in the workflow using YAML. [Learn more about managing permissions.](#)' Below this text, there are two radio button options: 'Read and write permissions' (selected, indicated by a blue arrow) and 'Read repository contents and packages permissions'. Under the selected option, it states: 'Workflows have read and write permissions in the repository for all scopes.' Below these options, there is a section for 'Pull request permissions' with the text: 'Choose whether GitHub Actions can create pull requests or submit approving pull request reviews.' The option 'Allow GitHub Actions to create and approve pull requests' is checked, indicated by a blue arrow. A 'Save' button is located at the bottom of the settings area.

# Installation: git-bob on the web

## Github



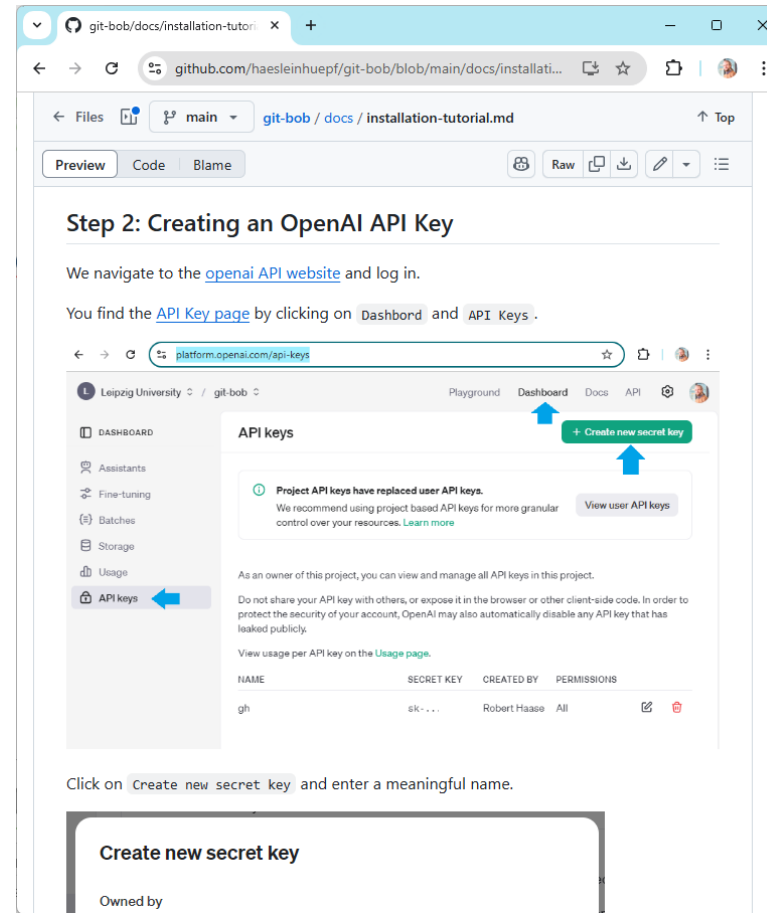
Installation as GitHub action

There is a detailed [tutorial](#) on how to install git-bob as GitHub action to your repository. In very short, to use git-bob in your GitHub repository, you need to

- Copy the [git-bob](#) GitHub workflow in folder `.github/workflows/` to your repository.
  - Make sure to replace `pip install -e .` with a specific git-bob version such as `pip install git-bob==0.11.0`.
  - Configure the LLM you want to use in the workflow files by specifying the `GIT_BOB_LLM_NAME` environment variable. These were tested:
    - `claude-3-5-sonnet-20241022`
    - `gpt-4o-2024-08-06`
    - `github_models:gpt-4o`
    - `github_models:meta-llama-3.1-405b-instruct`
    - `gemini-1.5-pro-002`
  - configure a GitHub secret called `OPENAI_API_KEY` or `ANTHROPIC_API_KEY` or `GH_MODELS_API_KEY` or `GOOGLE_API_KEY` or `KISSKI_API_KEY` or `BLABLADOR_API_KEY` with the corresponding key from the LLM provider depending on the above configured LLM. You can get these keys here:
    - [OpenAI \(gpt\)](#)
    - [Anthropic \(claude\)](#)
    - [GitHub Models Marketplace](#)
    - [Google AI](#)
    - [KISSKI](#)
    - [BLABLADOR](#)
  - configure GitHub actions to run the workflow on issues and pull requests. Also give write-access to the Workflow using the `GITHUB_TOKEN`.

When using it in your repository, you can also set a custom system message, for example for:

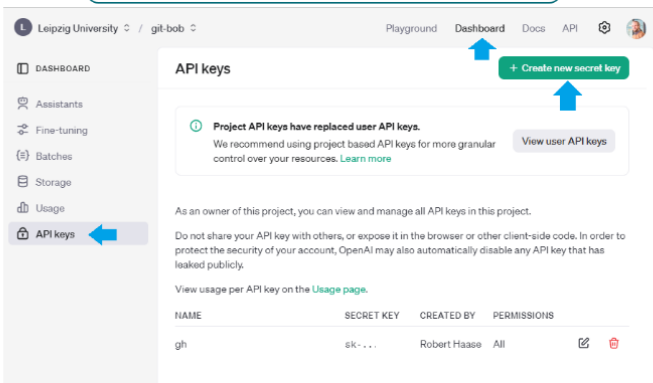
## Github



Step 2: Creating an OpenAI API Key

We navigate to the [openai API website](#) and log in.

You find the [API Key page](#) by clicking on `Dashboard` and `API Keys`.

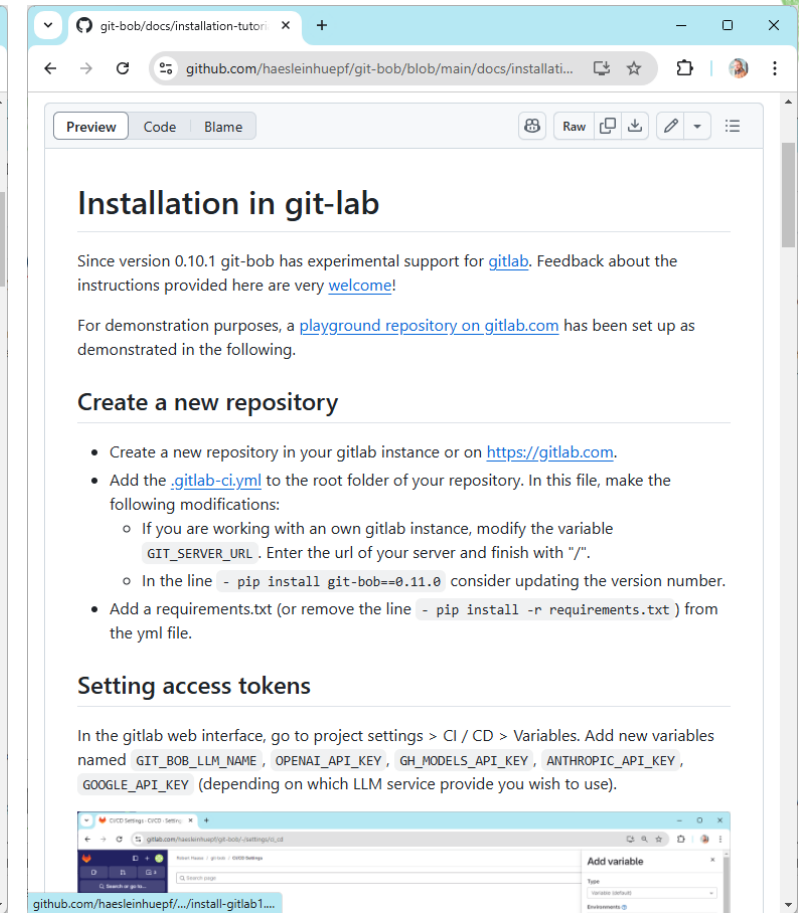


Click on `Create new secret key` and enter a meaningful name.

Create new secret key

Owned by

## Gitlab (experimental)



Installation in git-lab

Since version 0.10.1 git-bob has experimental support for [gitlab](#). Feedback about the instructions provided here are very [welcome!](#)


For demonstration purposes, a [playground repository on gitlab.com](#) has been set up as demonstrated in the following.

### Create a new repository

- Create a new repository in your gitlab instance or on <https://gitlab.com>.
- Add the [.gitlab-ci.yml](#) to the root folder of your repository. In this file, make the following modifications:
  - If you are working with an own gitlab instance, modify the variable `GIT_SERVER_URL`. Enter the url of your server and finish with `/`.
  - In the line `- pip install git-bob==0.11.0` consider updating the version number.
  - Add a `requirements.txt` (or remove the line `- pip install -r requirements.txt`) from the `yml` file.

### Setting access tokens

In the gitlab web interface, go to project settings > CI / CD > Variables. Add new variables named `GIT_BOB_LLM_NAME`, `OPENAI_API_KEY`, `GH_MODELS_API_KEY`, `ANTHROPIC_API_KEY`, `GOOGLE_API_KEY` (depending on which LLM service provide you wish to use).



# Exercises

# Exercise

Ask bia-bob to segment nuclei, before and after installing pyclesperanto!

```
[1]: import bia_bob

[2]: %%bob
Please segment the nuclei in the `skimage.data.human_mitosis()` dataset using Voronoi-

This notebook may contain text, code and images generated by artificial intelligence. Used model: gpt-4o-2024-08-06, vision model: gpt-4o-2024-08-06, endpoint: None, bia-bob version: 0.23.4. 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

[3]: from skimage import data
from skimage.filters import threshold_otsu
from skimage.segmentation import watershed
from scipy import ndimage as ndi
import numpy as np
import matplotlib.pyplot as plt

Load the human mitosis image
image = data.human_mitosis()

Calculate the Otsu threshold
thresh = threshold_otsu(image)

Create a binary mask
binary = image > thresh

Calculate the distance transform
distance = ndi.distance_transform_edt(binary)

Generate markers for the watershed
markers, _ = ndi.label(ndi.gaussian_filter(distance, sigma=3) < 0.1)

Perform the Voronoi-Otsu-Labeling (Watershed segmentation)
labels = watershed(-distance, markers, mask=binary)

Display the original and labeled images
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
ax = axes.ravel()

ax[0].imshow(image, cmap='gray')
ax[0].set_title('Original Image')

ax[1].imshow(labels, cmap='nipy_spectral')
```

```
[1]: import bia_bob

[2]: %%bob
Please segment the nuclei in the `skimage.data.human_mitosis()` dataset using Voronoi-

This notebook may contain text, code and images generated by artificial intelligence. Used model: gpt-4o-2024-08-06, vision model: gpt-4o-2024-08-06, endpoint: None, bia-bob version: 0.23.4. 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

[3]: from skimage.data import human_mitosis
import pyclesperanto as cle
import stackview

Load the human mitosis image
image = human_mitosis()

Apply Voronoi-Otsu-Labeling
labels = cle.voronoi_otsu_labeling(image)

Display the segmented image
stackview.insight(labels)

[3]:
```

shape	(512, 512)
dtype	uint32
size	1024.0 kB
min	0
max	323



After installing pyclesperanto!

Learn more!  
I2K virtual, Oct 30<sup>th</sup> 14:00 CEST

# Exercise

Login with your Github account and create a new issue. I'll chat with you and the AI to make it happen.

Bio-image analysis or kids' stories - your choice!



## [BioImage Analysis]: Nuclei segmentation #6

Open haesleinhuepf opened this issue 6 minutes ago · 1 comment

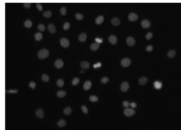
haesleinhuepf commented 6 minutes ago

### Analysis Goal (What should be done / analysed?)

Segment the nuclei in this image using Voronoi Otsu Labeling. While loading the image, make sure to use the first channel only.

### Image Upload

Drag & drop your microscopy image here (JPG, PNG, GIF, e.g. 512x512 pixels, 2D only).



### Python Tools

- List of Python libraries we should use for answering this:
  - numpy
  - napari-segment-blobs-and-things-with-membranes
  - stackview

Note: Your images and the text you enter here may be sent to OpenAI's online service where we use a large language model to answer your request. Do not upload any data you cannot share openly. Also do not enter any private or secret information. By submitting this Github issue, you confirm that you understand these conditions.

git bob try to do this!

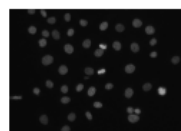
- Detailed instructions for bio-image analysis using Python (feel free to modify)

haesleinhuepf self assigned this 6 minutes ago


github-actions commented 4 minutes ago

The message was generated by gpt-4o (version: 0.12.0, model: gpt-4o-mini, log), an experimental AI-based assistant. It can make mistakes and has limitations. Check its messages carefully.

The task involved segmenting nuclei in a microscopy image using Voronoi Otsu Labeling. We downloaded the image "human\_mitosis\_small.png" and created a Jupyter notebook named "nuclei\_segmentation\_notebook.ipynb" to perform the segmentation. The segmented result was then saved as "segmented\_nuclei.png". These changes were minimal and focused on the task of nuclei segmentation using the specified image analysis techniques.



Downloaded and saved microscopy image for analysis.



Saved the output image after nuclei segmentation.

- nuclei\_segmentation\_notebook.ipynb Created a notebook to perform nuclei segmentation using specified methods.

## [Story Telling]: Adventure in the forest #5

Open haesleinhuepf opened this issue 8 minutes ago · 1 comment

haesleinhuepf commented 8 minutes ago

### What should the story be about?

I'd like to read a story about a duck and a cat meeting in the forest while searching for food. They have an adventure on their way to the ice cream stand. Also paint a picture of both happily snacking.

Note: Your images and the text you enter here may be sent to OpenAI's online service (Terms of use) or Anthropic's Claude online service (Terms of service) or Google AI (Terms of service) where we use a large language model to answer your request. Do not upload any data you cannot share openly. Also do not enter any private or secret information. By submitting this Github issue, you confirm that you understand these conditions.

story teller try to do this!

haesleinhuepf self assigned this 8 minutes ago

github-actions commented 6 minutes ago

The message was generated by story-teller / gpt-4o (version: 0.12.0, model: gpt-4o-mini, log), an experimental AI-based assistant. It can make mistakes and has limitations. Check its messages carefully.

In this update, a delightful story prompt about a duck and a cat embarking on an adventure in the forest was added. Additionally, an image depicting the duck and cat happily snacking was painted to bring the story to life.

- github/ISSUE\_TEMPLATE/story\_telling.mdx Added a story prompt about a duck and a cat having an adventure in the forest.



- Painted an image of the duck and cat snacking together.

During solving this task, the following errors occurred:

- Error processing github/ISSUE\_TEMPLATE/story\_telling.mdx: Modifying workflow files is not allowed.

# Summary & outlook

- LLMs can generate code to analyze biological microscopy images
- *We can* learn how to use them
- *We need* to understand bio-image-analysis, no matter if we write code ourselves or if we generate code using LLMs



<https://doi.org/10.5281/zenodo.14001044>

# Acknowledgements

## Collaborators & contributors

Christian Tischer (EMBL Heidelberg), Jean-Karim Hériché (EMBL Heidelberg), Nico Scherf (MPI CBS Leipzig), Loïc A. Royer (CZ Biohub), Elena Nicolay (UFZ Leipzig), Mara Lampert (ScaDS.AI / TU Dresden), Kevin Yamauchi (ETH Zurich / Basel), Seth Hinz, Teun A.P.M. Huijben, Jordão Bragantini (CZ Biohub San Francisco), Ilan Theodoro (CZ Biohub and the State University of Campinas, Brazil), Ian C. (University of British Columbia), Pradeep Rajasekhar (WEHI Australia), Stéphane Rigaud (Institut Pasteur Paris)

## Communities & platforms



## Funding



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.



<https://doi.org/10.5281/zenodo.14001044>



BIA Code Generation  
#I2K2024 Virtual  
Robert Haase  
@haesleinhuepf  
October 28<sup>th</sup> 2024

RH acknowledges the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research „Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig“, project identification number: ScaDS.AI



TECHNISCHE  
UNIVERSITÄT  
DRESDEN



UNIVERSITÄT  
LEIPZIG