

Deep Learning + Large Language Models for Bio-image Analysis

Robert Haase

Reusing Material from

Alex Krull (MPI CBG), Martin Weigert (EPFL Lausanne), Uwe Schmidt (MPI CBG), Ignacio Arganda-Carreras (Universidad del País Vasco), Johannes Soltwedel, Till Korten (PoL TU Dresden), Loic A. Royer (CZ Biohub), Alexandr Dibrov (CSBD/MPI-CBG Dresden), Aditya Ramesh et al (OpenAI) and Alexandr Khrapichev (University of Oxford), Carsen Stringer et al (HHMI Janelia)

Deep Learning and Large language models

- How it started



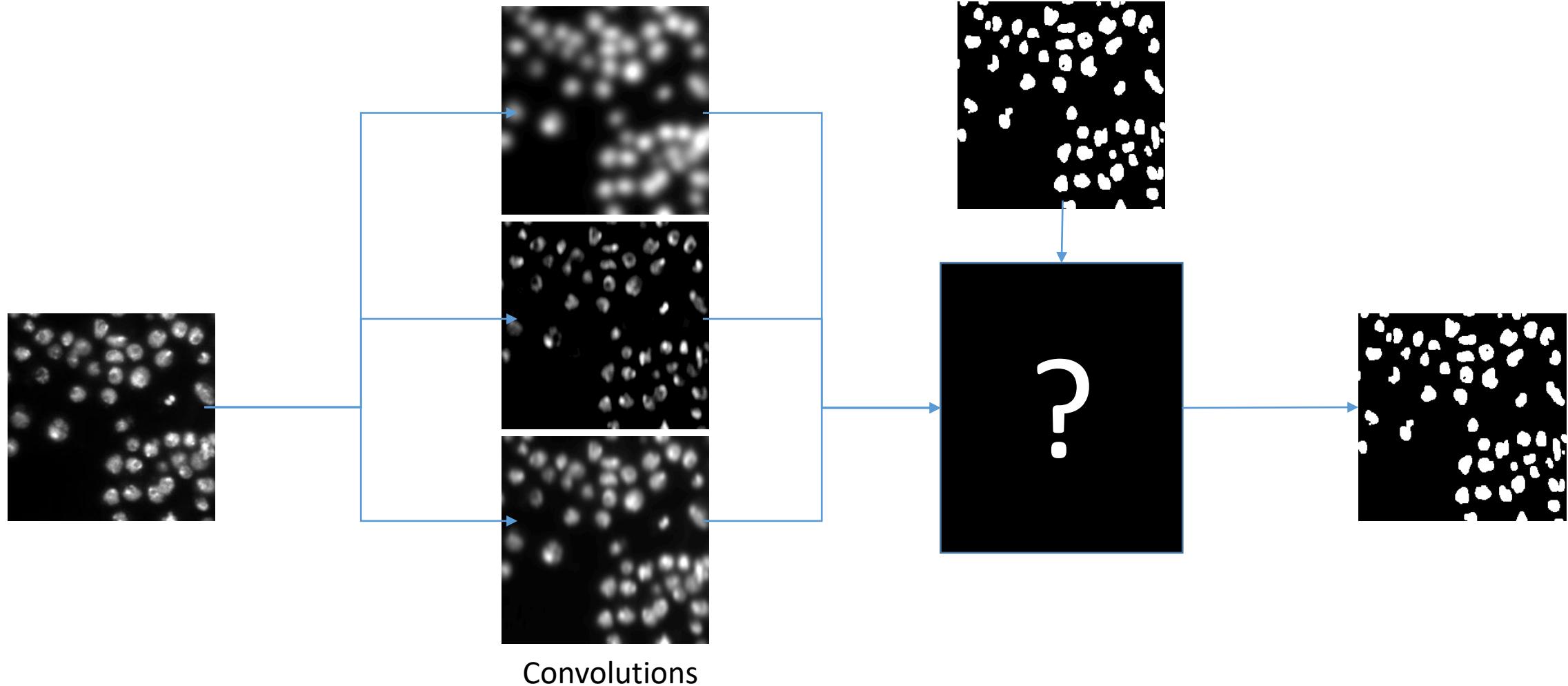
- How it's going

(GitHub copilot)

```
• Robert Haase +4 *
@catch_NaN
def agglomerative_clustering(
    reg_props: pd.DataFrame, cluster_number: int, n_neighbors: int
) -> Tuple[str, np.ndarray]:
    """
    """
```

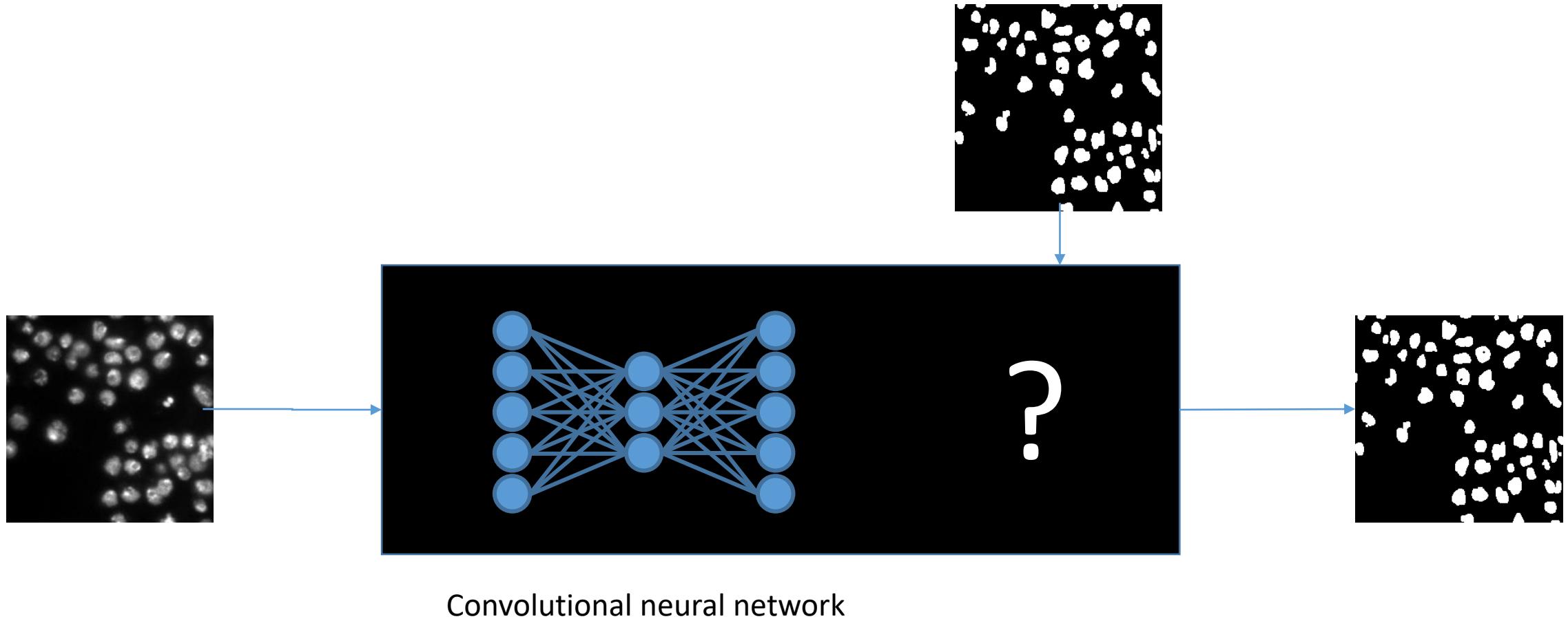
Machine learning for image analysis

- Recap: In classical supervised machine learning, we typically select features for training our classifier



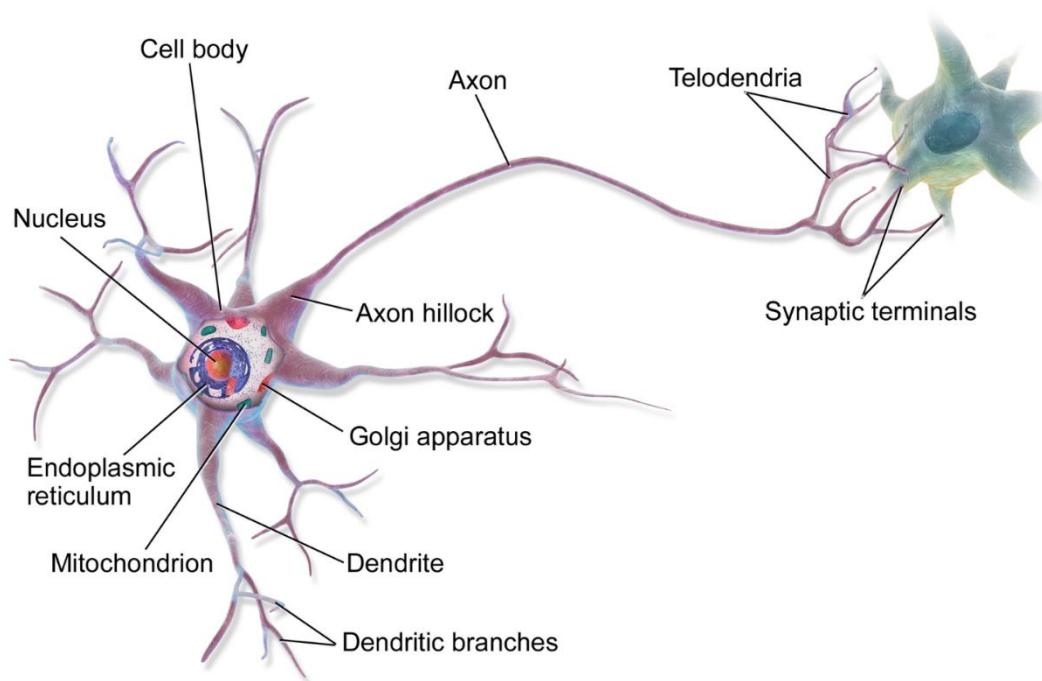
Deep learning for image analysis

- In deep learning, this selection becomes part of the black box

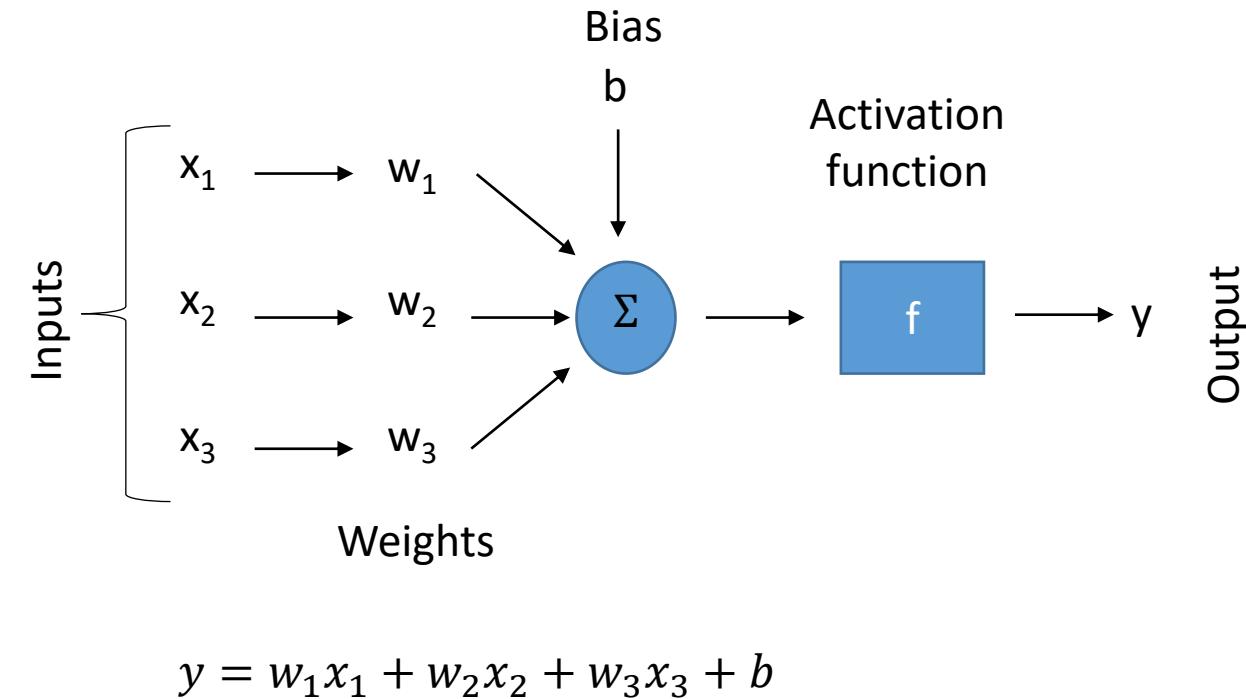


Neural networks

- How biologists see neurons



- How computer scientists see neurons
“perceptron”



Neuron image source:

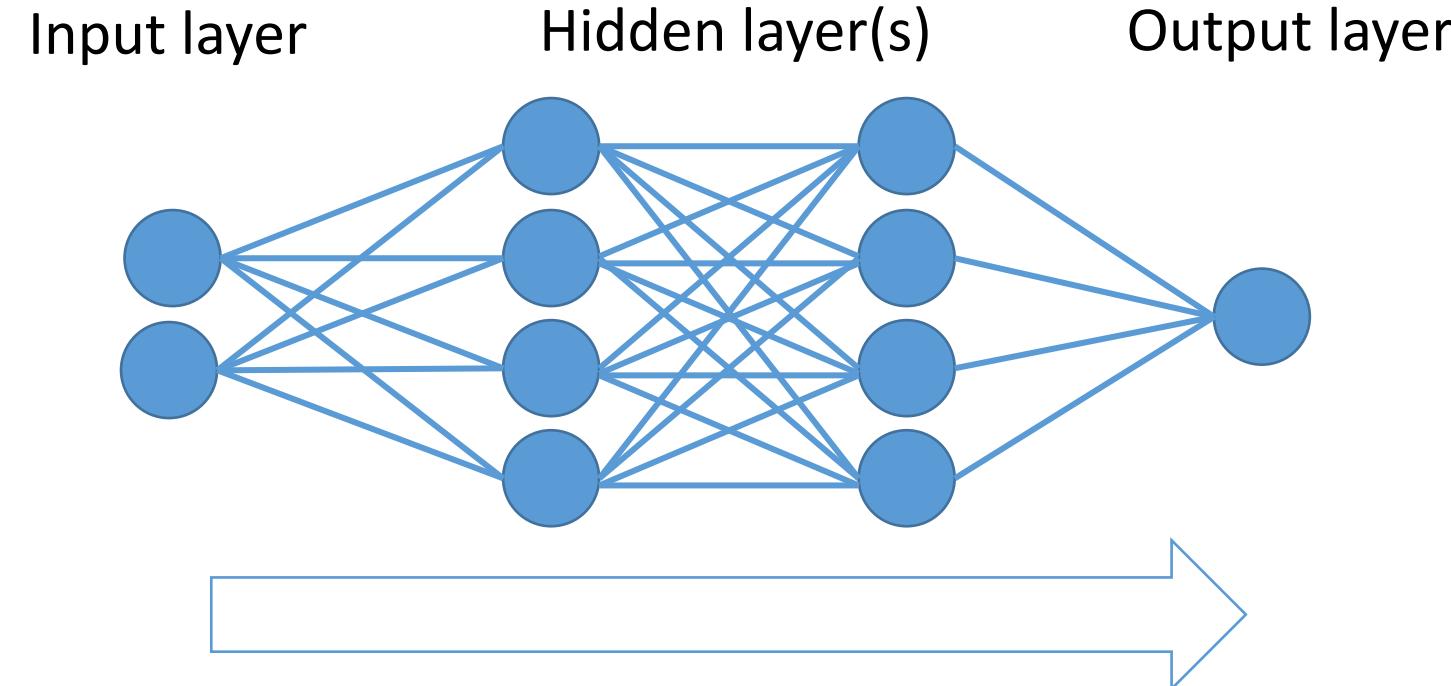
https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png

Licensed CC-BY 3.0 by [BruceBlaus](#)

Neural Networks

- Early form: “Multilayer Perceptron”
- fully connected class of feedforward artificial neural network

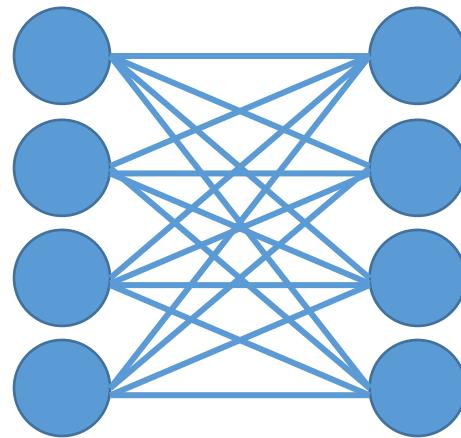
If there are *many* hidden layers, we speak of a *deep* neural network



Convolutional neural networks

- Layer types

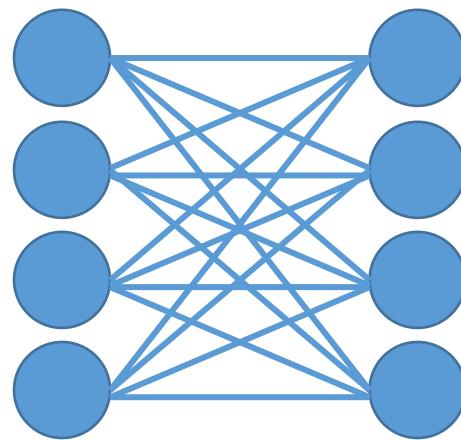
Fully connected layer



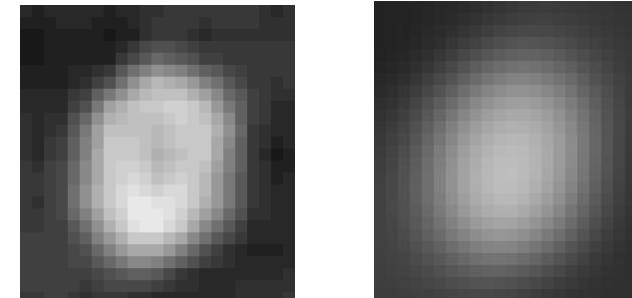
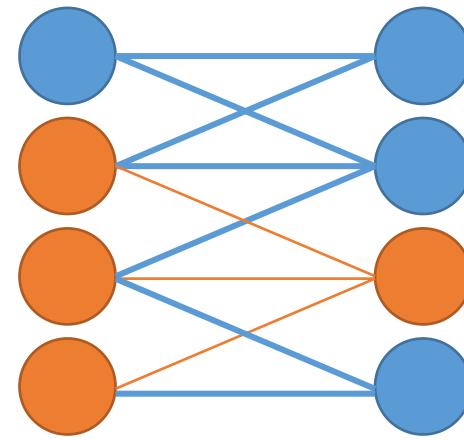
Convolutional neural networks

- Layer types

Fully connected layer



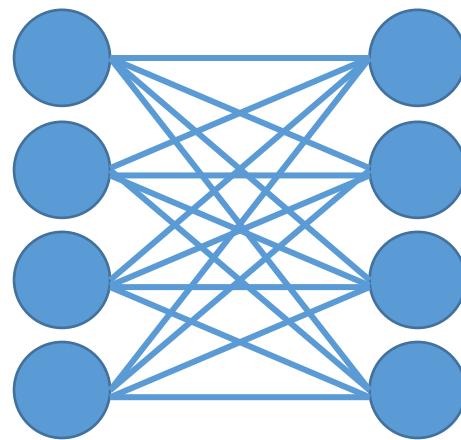
Convolutional layer



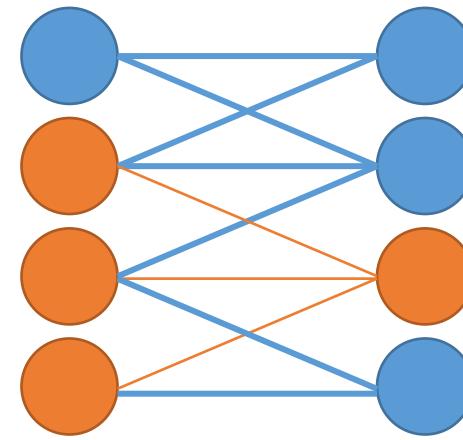
Convolutional neural networks

- Layer types

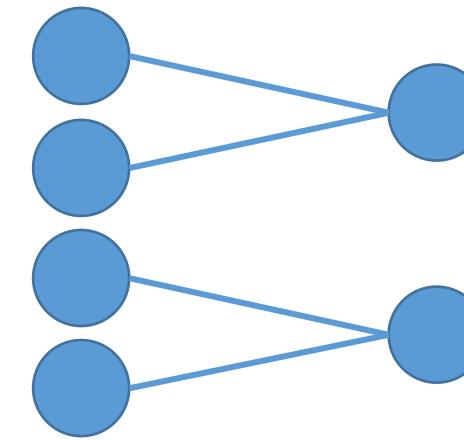
Fully connected layer



Convolutional layer



Pooling layer
("Max pool", "Average pool")



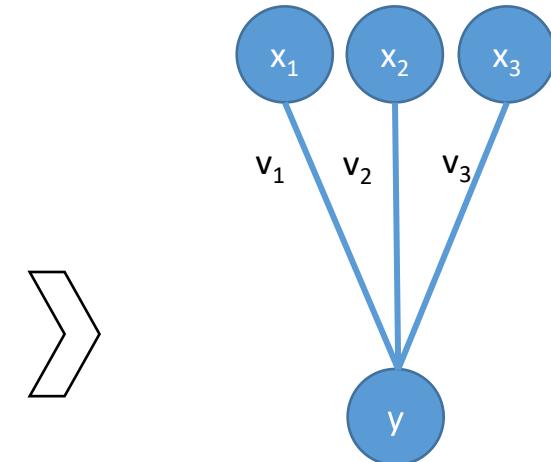
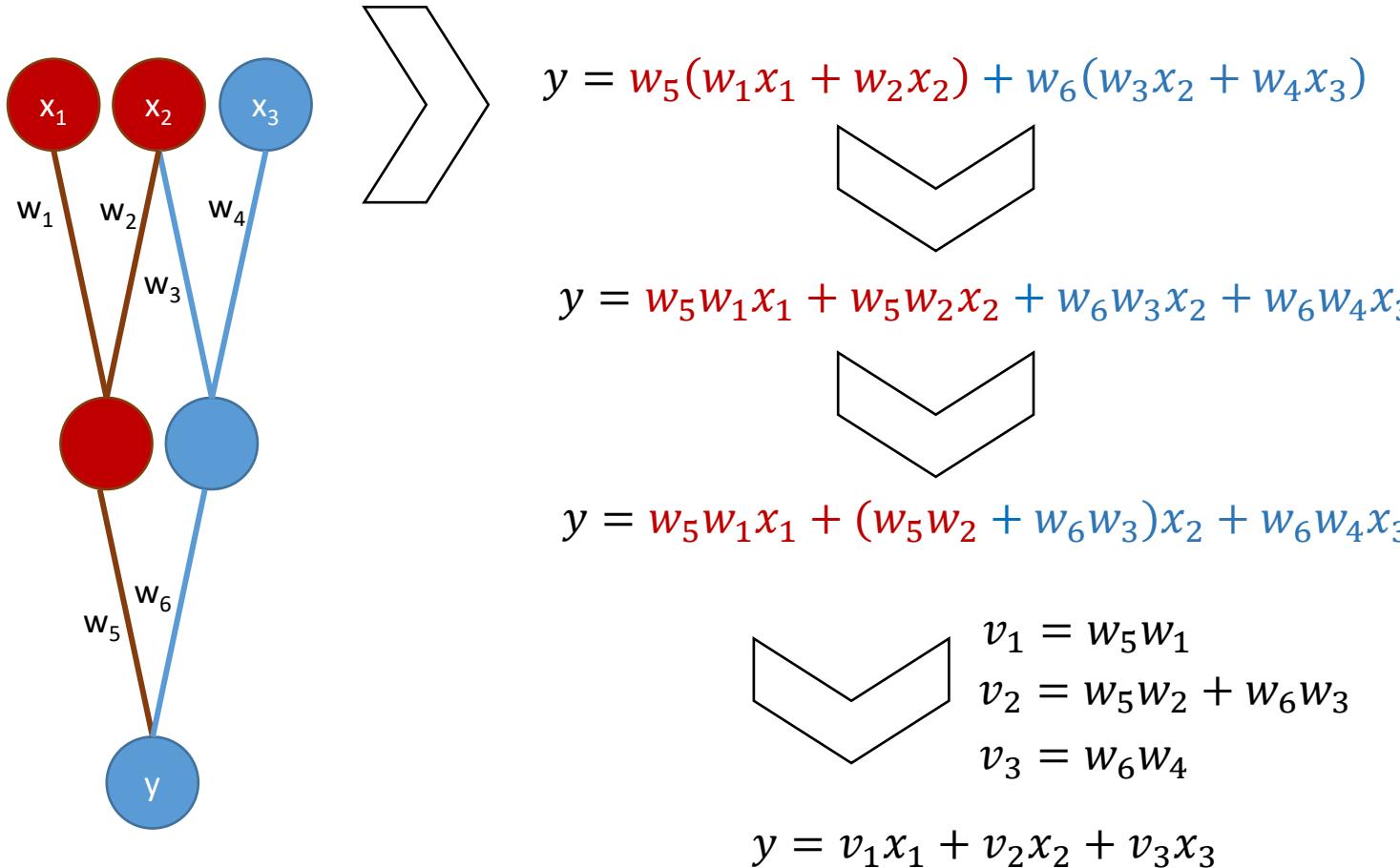
3	15	1	13
9	7	0	10
11	5	5	3
1	8	9	6

Max pooling

15	13
11	9

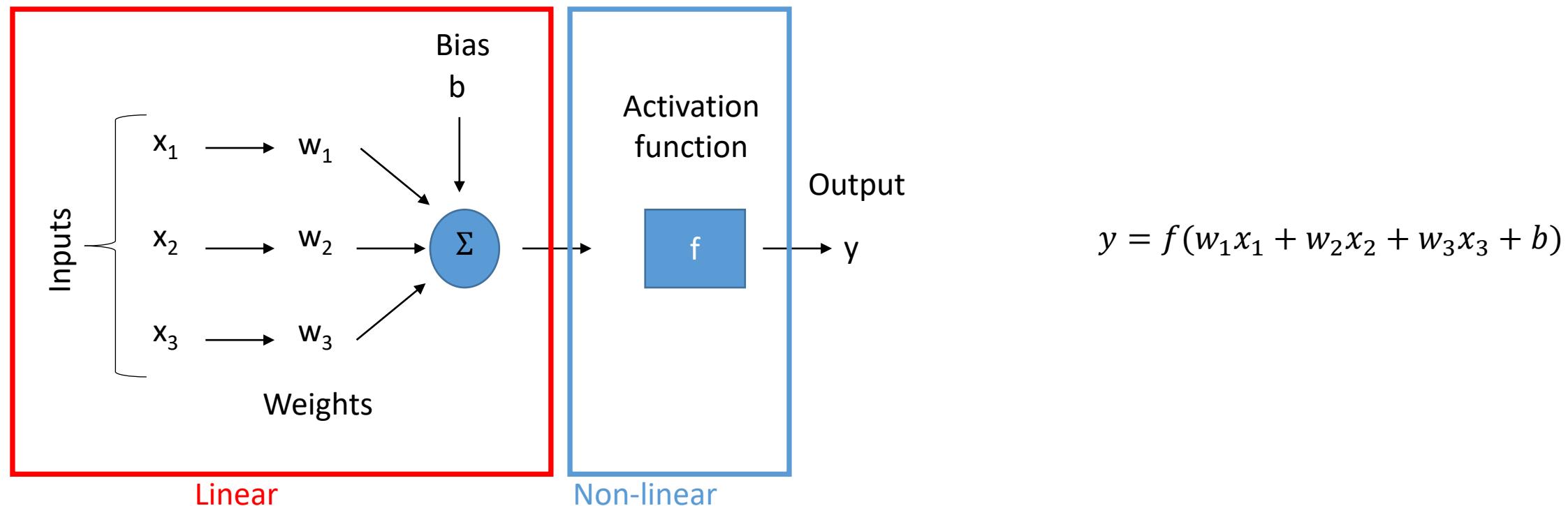
Convolutional neural networks

- Assuming we had no activation functions in the network layers can be reduced by eliminating brackets!



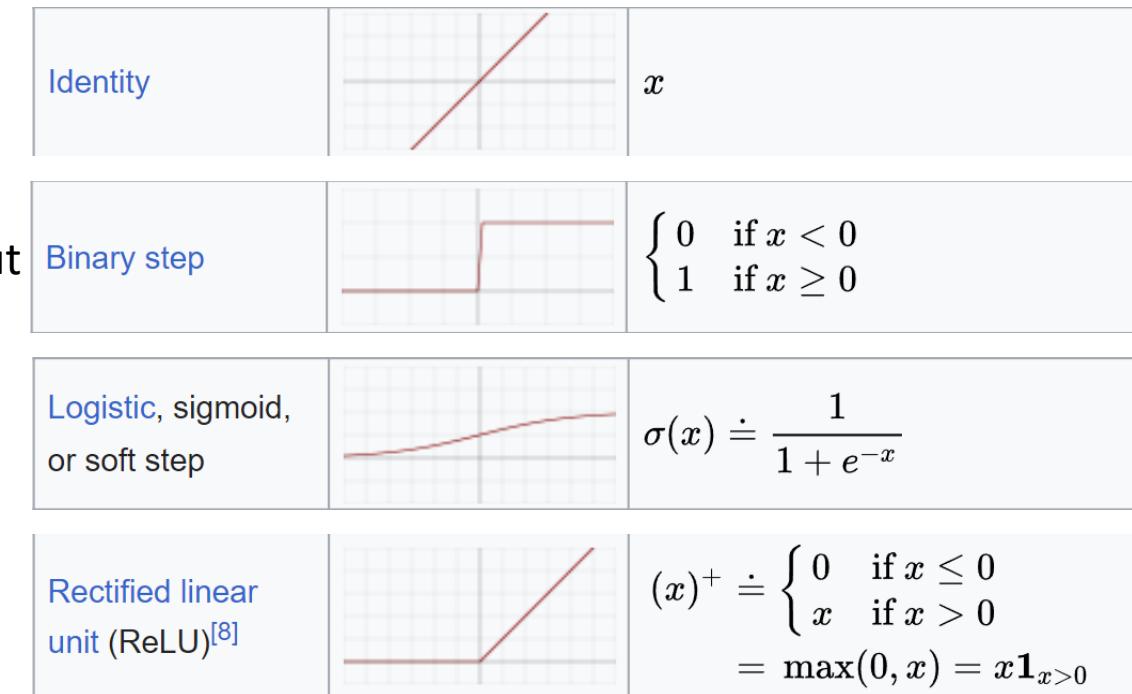
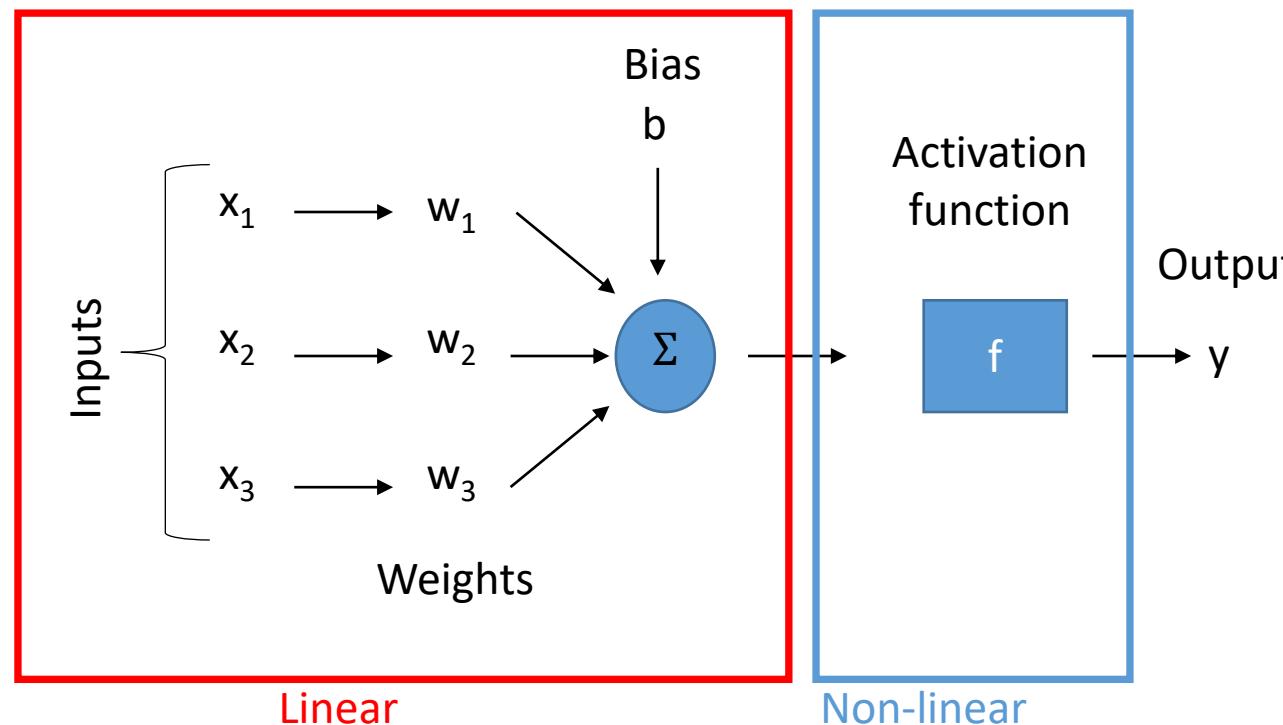
Activation functions

- Introduction of *non-linearity* and *activation functions* enabled what we call *deep-learning* today.



Activation functions

- Introduction of *non-linearity* and *activation functions* enabled what we call *deep-learning* today.

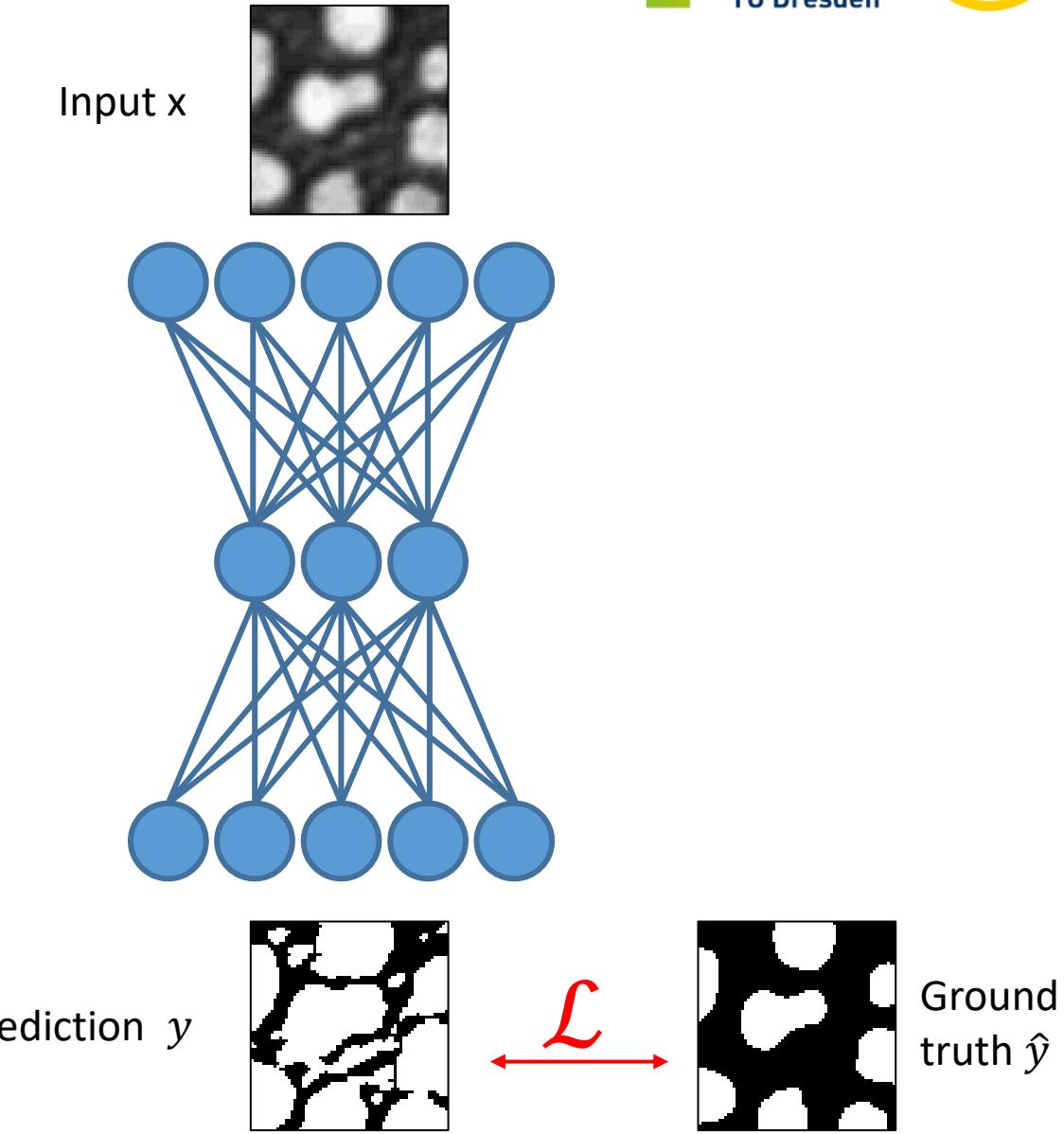


Learning: Back propagation

- Learning is an optimization problem
- Step 0: Initialize the network randomly
 - Weights
 - Bias
- Step 1: Forward pass the input through the network, get an initial prediction
- Step 2: Compare the output with the ground truth, compute the error (loss function)
 - The loss function can be freely defined.
 - Example: mean squared error

$$\mathcal{L}(y, \hat{y}) = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - y_i)^2$$

- Step 3: Update weights



Back-Propagation Algorithm

The loss function can be expanded from

$$\mathcal{L}(y, \hat{y}) = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - y_i)^2$$

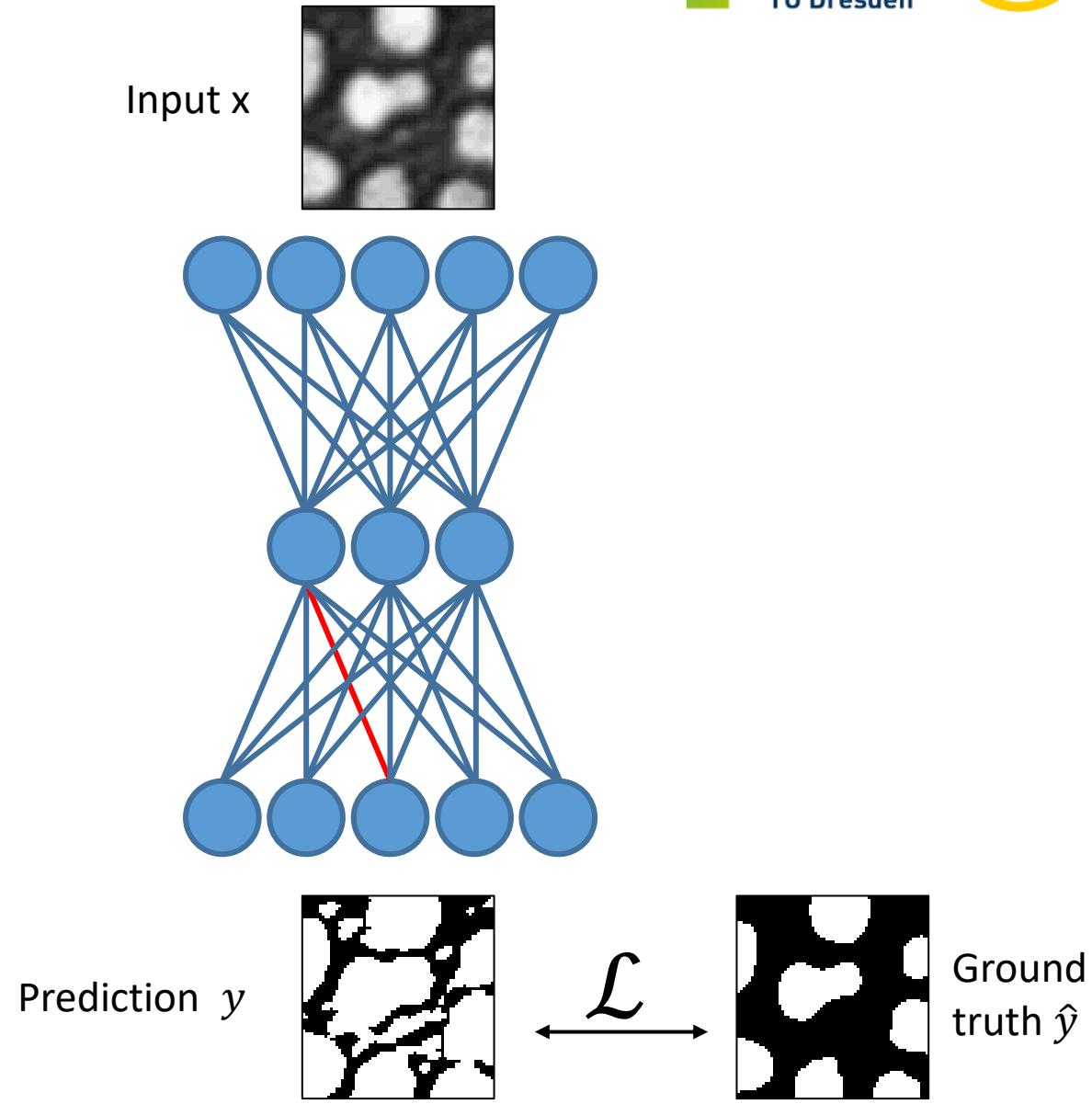
as the prediction depends on inputs x weights w and bias b

$$\mathcal{L}(\hat{y}, x, w) = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - (w^T x_i + b))^2$$

Derivatives with respect to w and b :

$$\frac{\partial \mathcal{L}(\hat{y}, x, w)}{\partial w} = -\frac{1}{N} \sum_{i=1}^N 2x_i(\hat{y}_i - (w^T x_i + b))$$

$$\frac{\partial \mathcal{L}(\hat{y}, x, w)}{\partial b} = -\frac{1}{N} \sum_{i=1}^N 2(\hat{y}_i - (w^T x_i + b))$$



Back-Propagation Algorithm

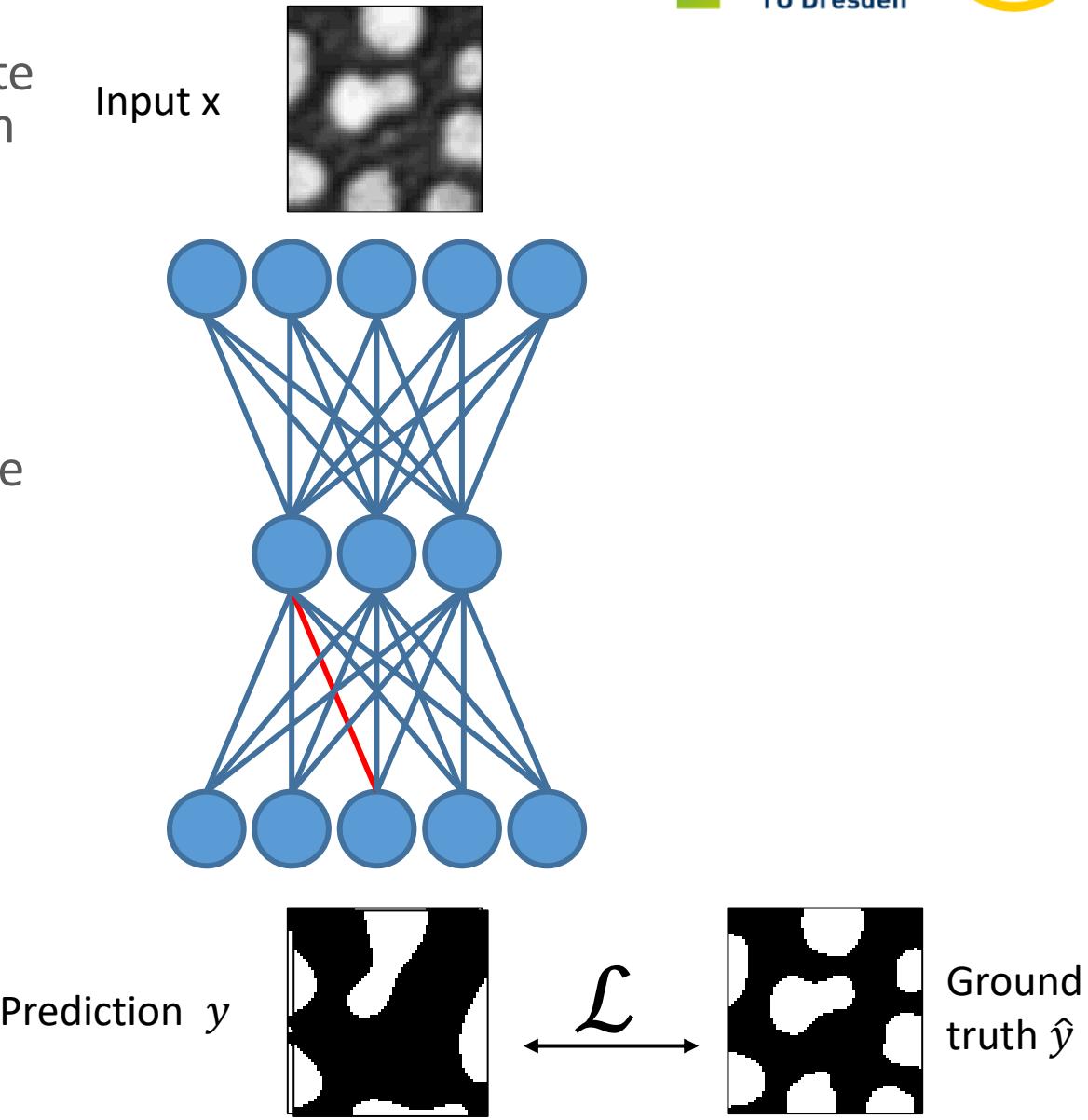
- The principle of the BackProp algorithm is to calculate the gradient of the loss function with respect to each trainable parameters of the ANN, i.e.

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^k}$$

- where w_{ij}^k is the i:th weight of node j in layer k, which will allow the optimization algorithm to update the weights step by step using stochastic gradient descent

$$w_{ij}^k = w_{ij}^k + \eta \frac{\partial \mathcal{L}}{\partial w_{ij}^k}$$

- where η is the step length, in this context known as the **learning rate**.
 η can be varied during training.



Train dataset (e.g. 80% of the data)

- Used for training directly

Validation dataset (10% of the data)

- After every iteration see if the model overfits

Test dataset (10% of the data)

- Final evaluation after training is finished (once)

Underfitting

- A trained model that is not even able to properly process the data it was trained on

Overfitting

- A model that is able to process data it was trained on well
- It processes other data poorly

<https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c>

The U-net

- The U-net is a very common DNN architecture in biological image processing.
 - Contraction: Increase the “What”, decrease the “Where”
 - Expansion: Increase the “Where”, decrease the “What”

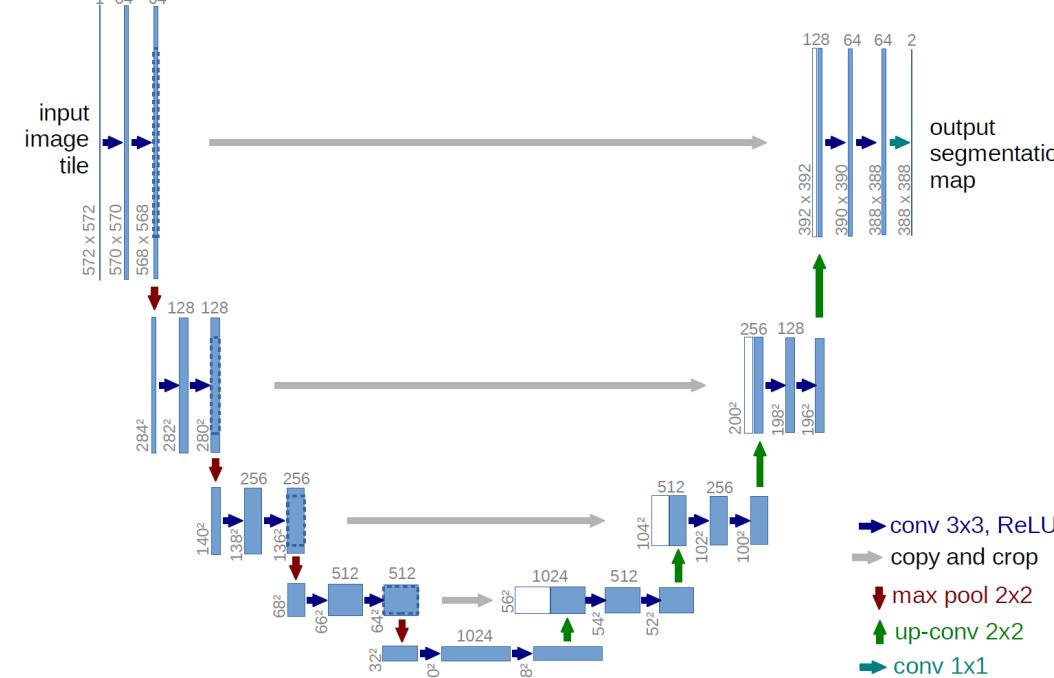
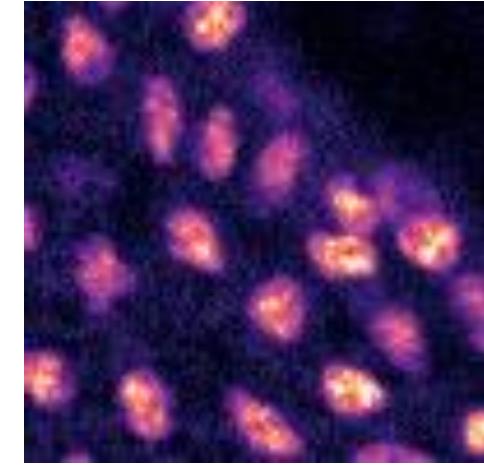
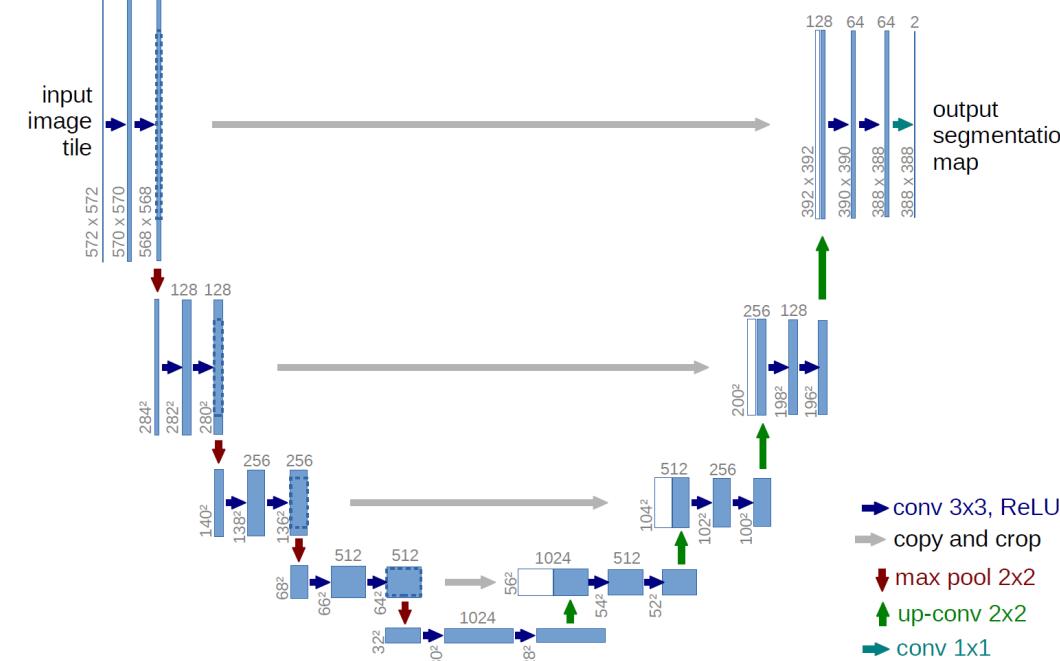
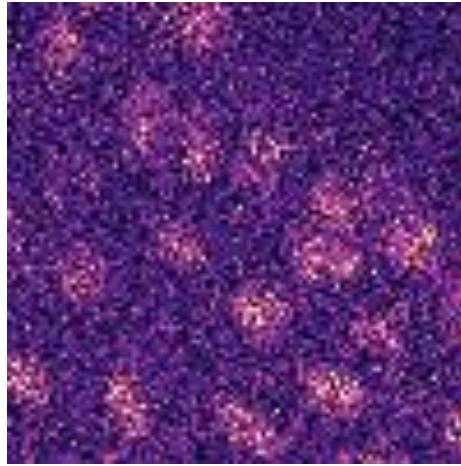


Image denoising: CARE

- Content aware image restoration (CARE)
- Image acquisition of pairs of images: A high-quality and a low-quality image.
- Problem: shot noise, biology moves!



Source: Ronneberger (2015) <https://arxiv.org/pdf/1505.04597.pdf>
Slide adapted from: Martin Weigert, EPFL Lausanne

Image denoising: CARE

- Important to use on the same conditions/structures/staining that the networks were trained on!

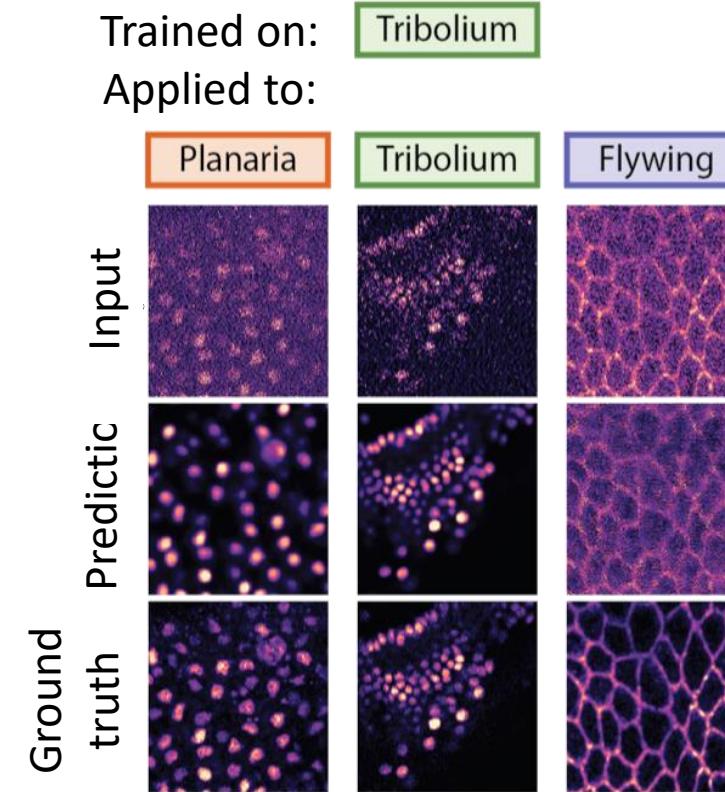
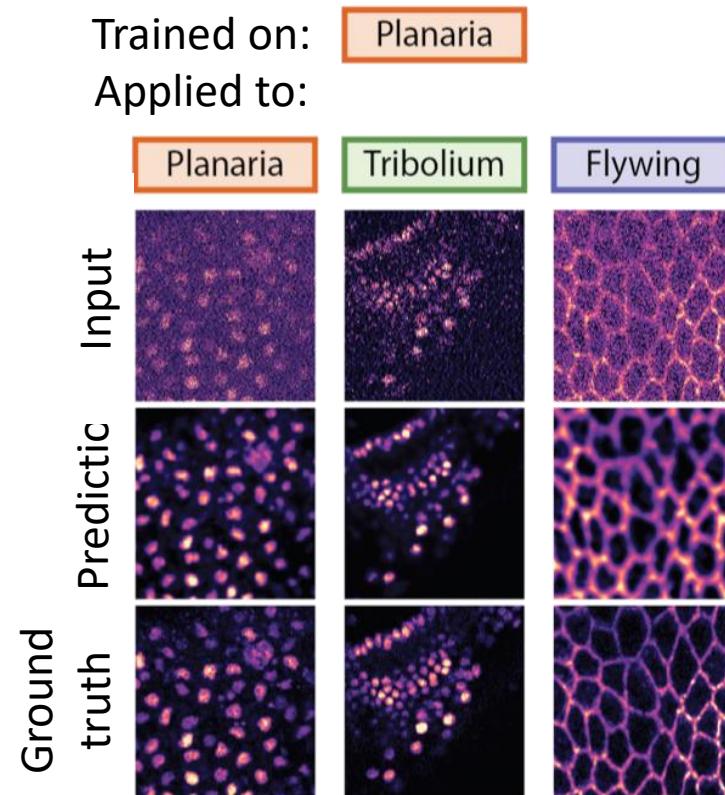


Image denoising

- Noise2Void

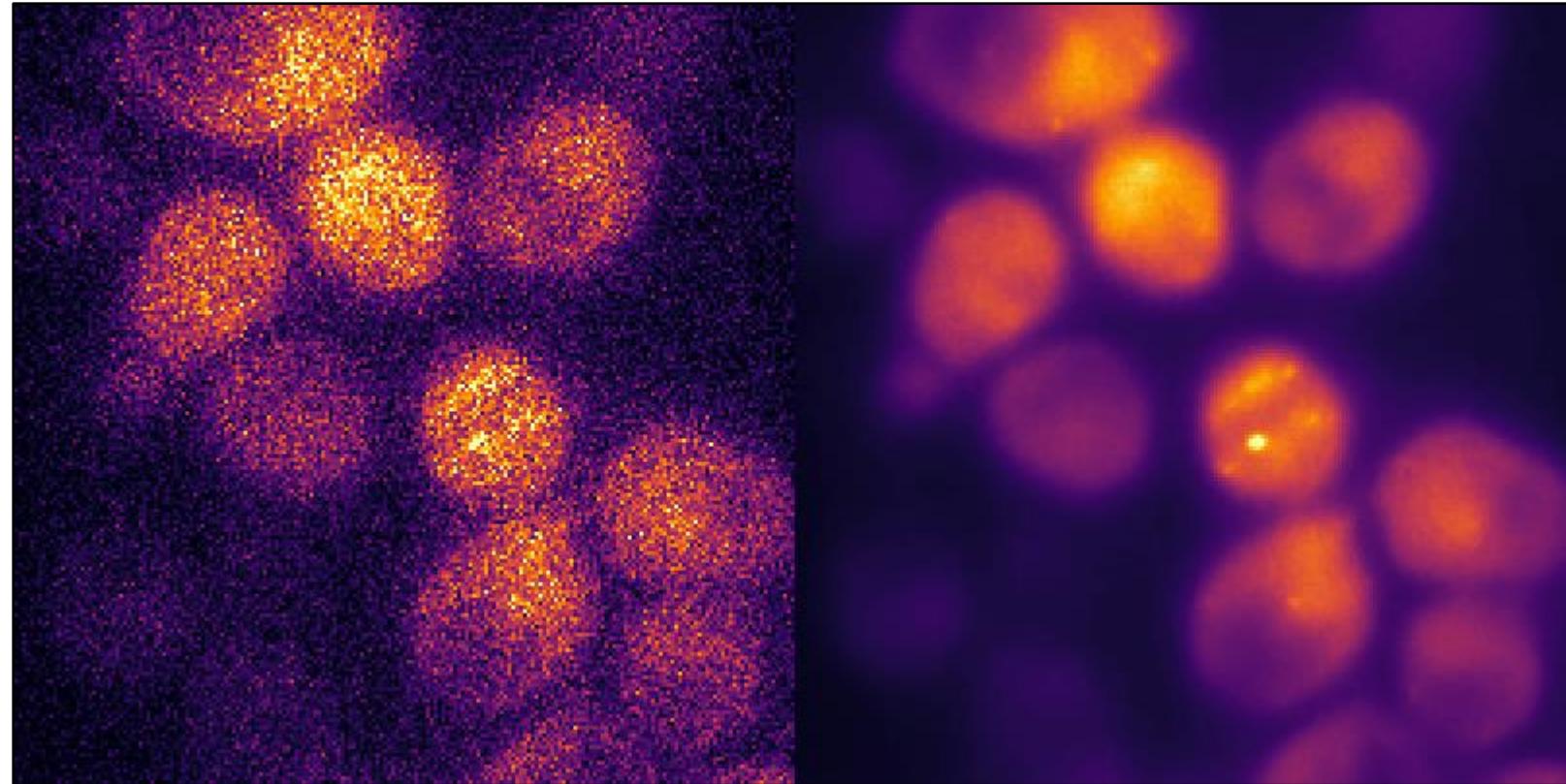
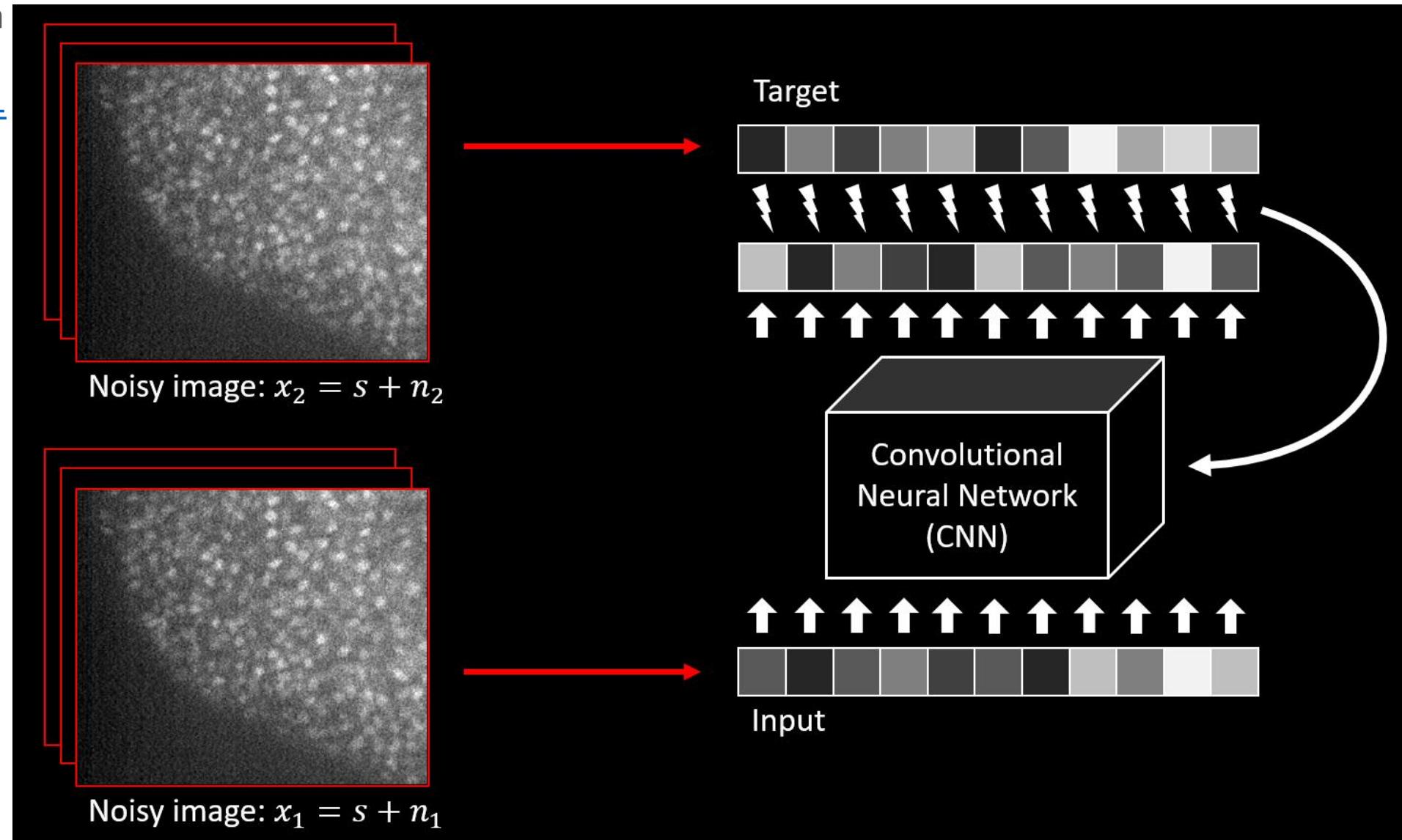


Image denoising

- Noise2Noise by NVidia
(Lehtinen 2018)
<https://arxiv.org/pdf/1803.04189.pdf>



- Noise2void: Krull et al (2019)
<https://arxiv.org/abs/1811.10980>
- Noise2self: Batson and Royer (2019)
<https://arxiv.org/abs/1901.11365>

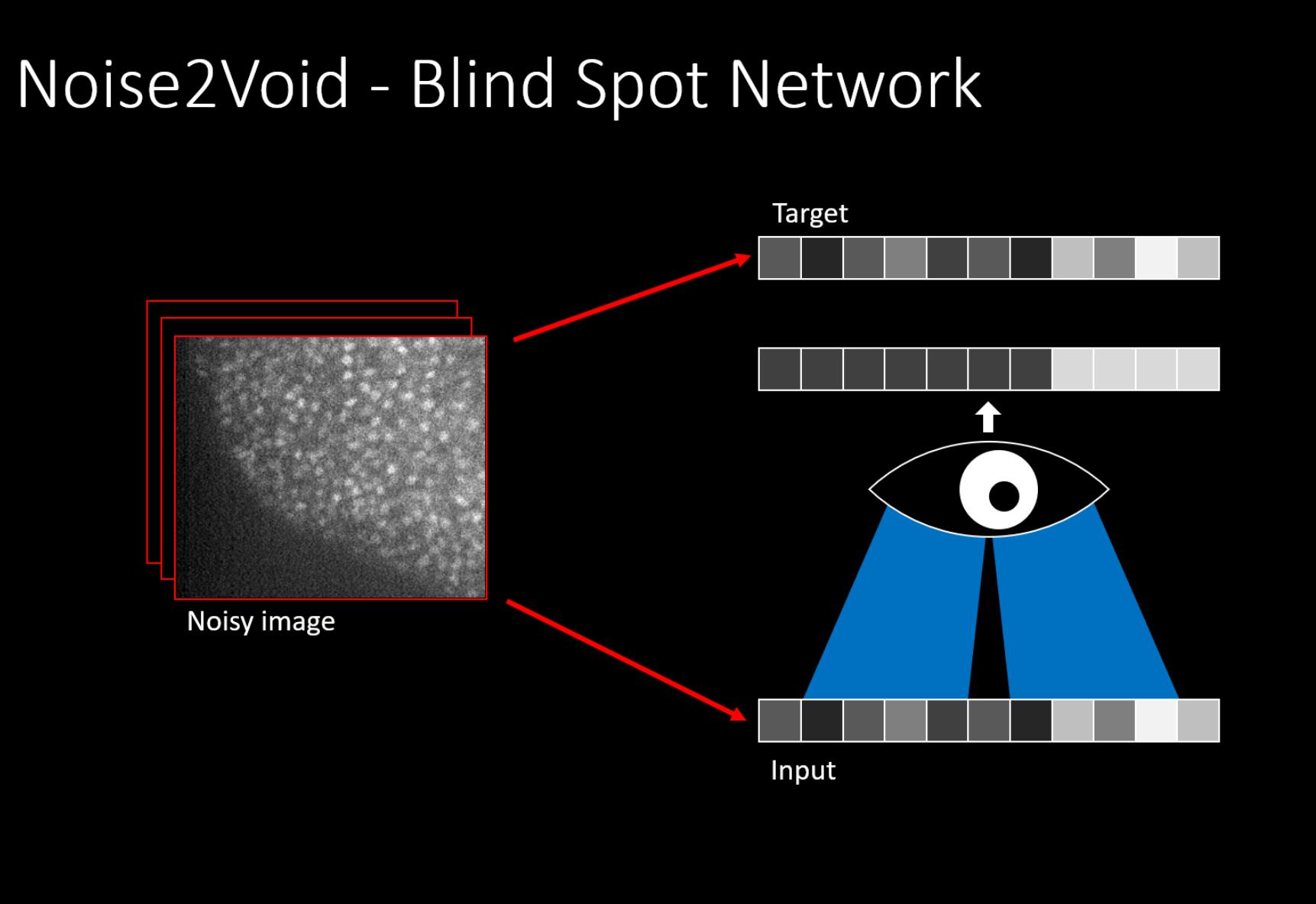
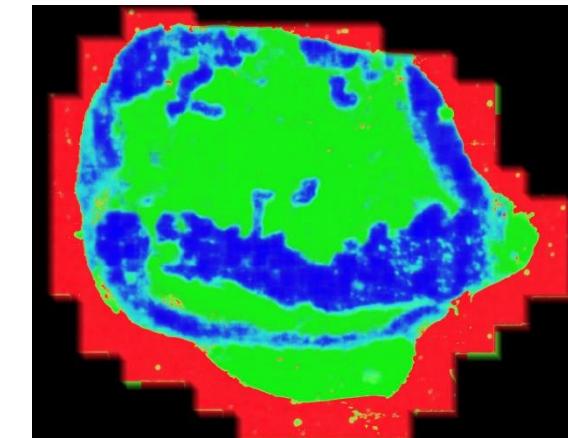
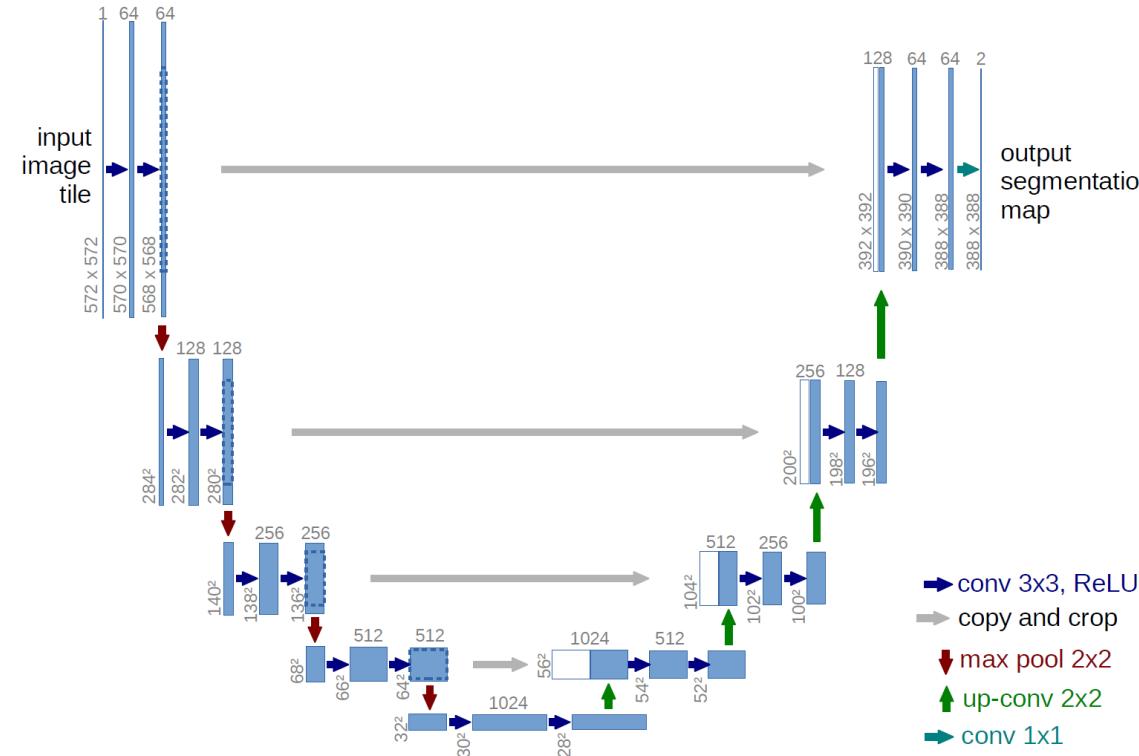
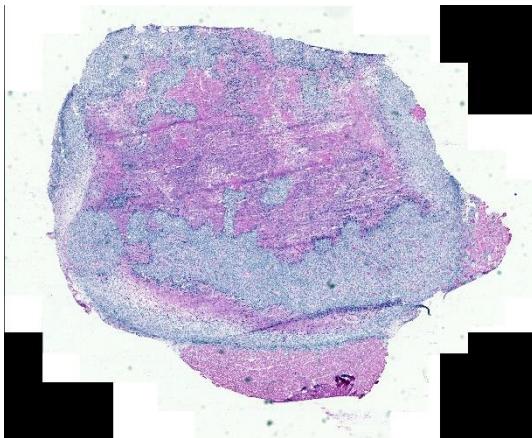


Image segmentation

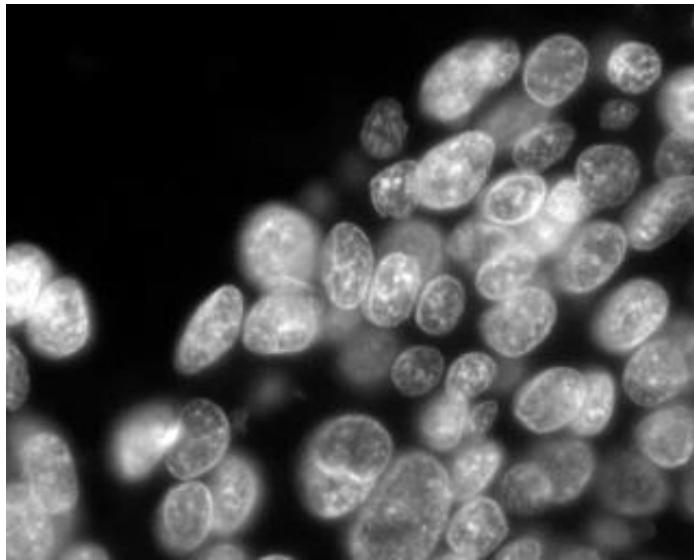
- U-net for pixel classification
 - > Semantic segmentation
 - > Probability maps



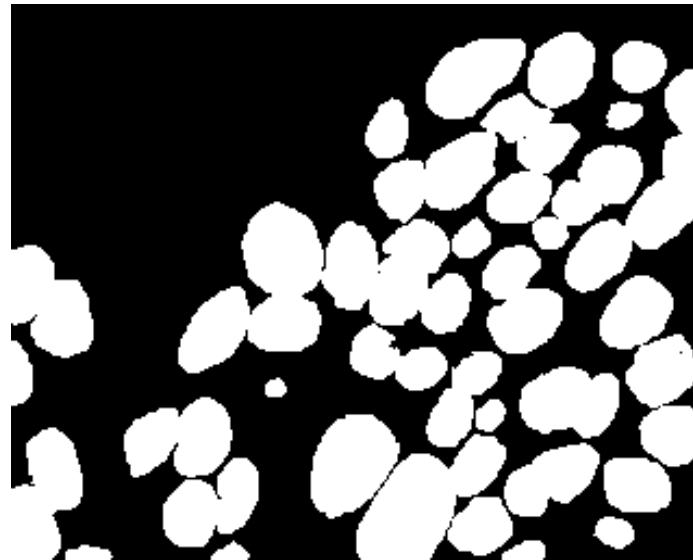
Sources: Ronneberger (2015) <https://arxiv.org/pdf/1505.04597.pdf>
Johannes Soltwedel, OncoRay, TU Dresden

Nuclei/cell segmentation

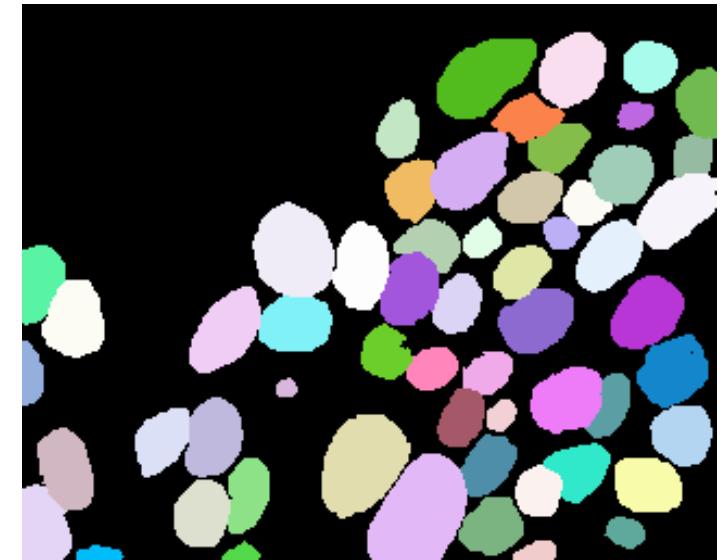
- Recap: Terminology



Input Image



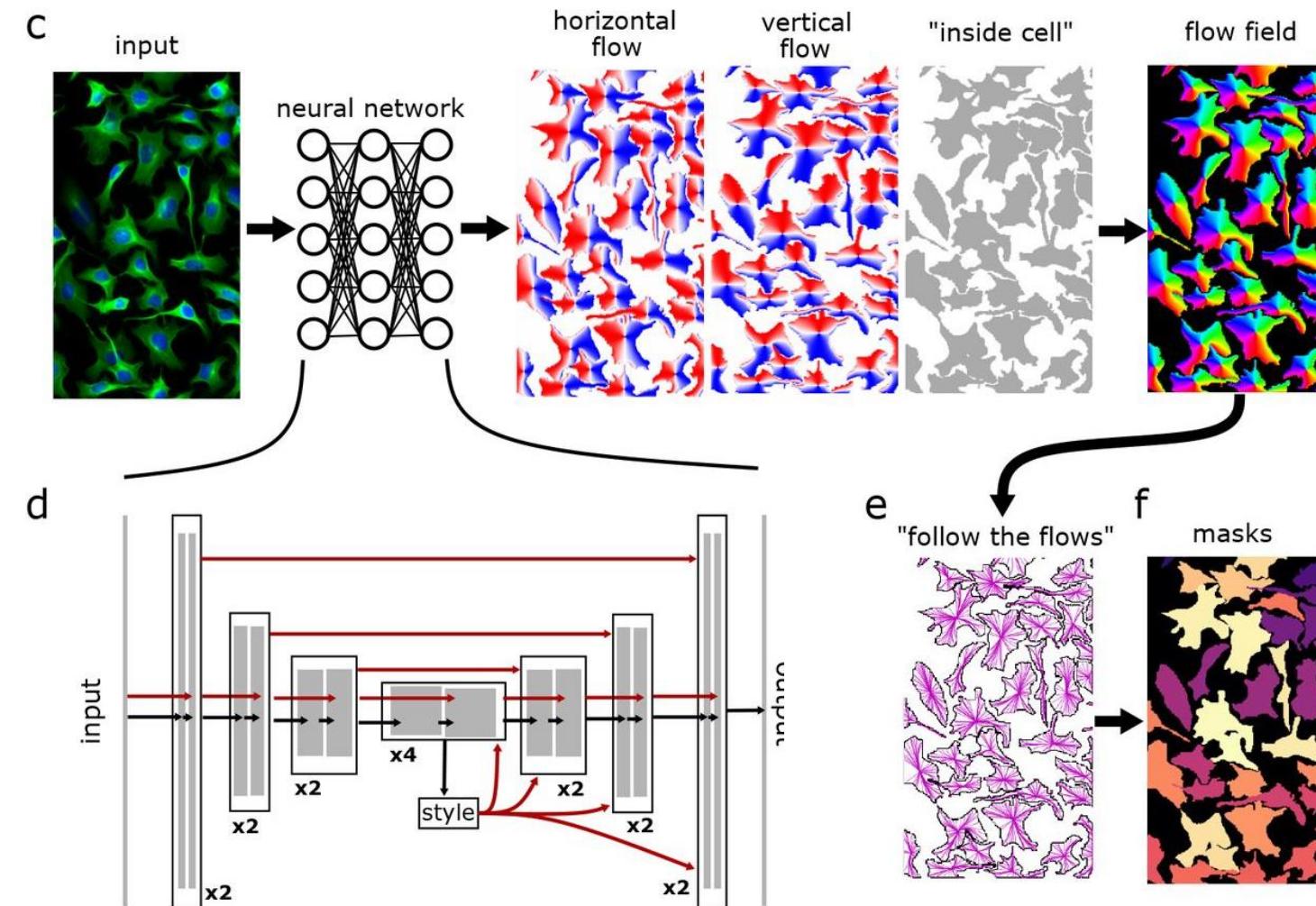
Semantic Segmentation
(foreground/background)



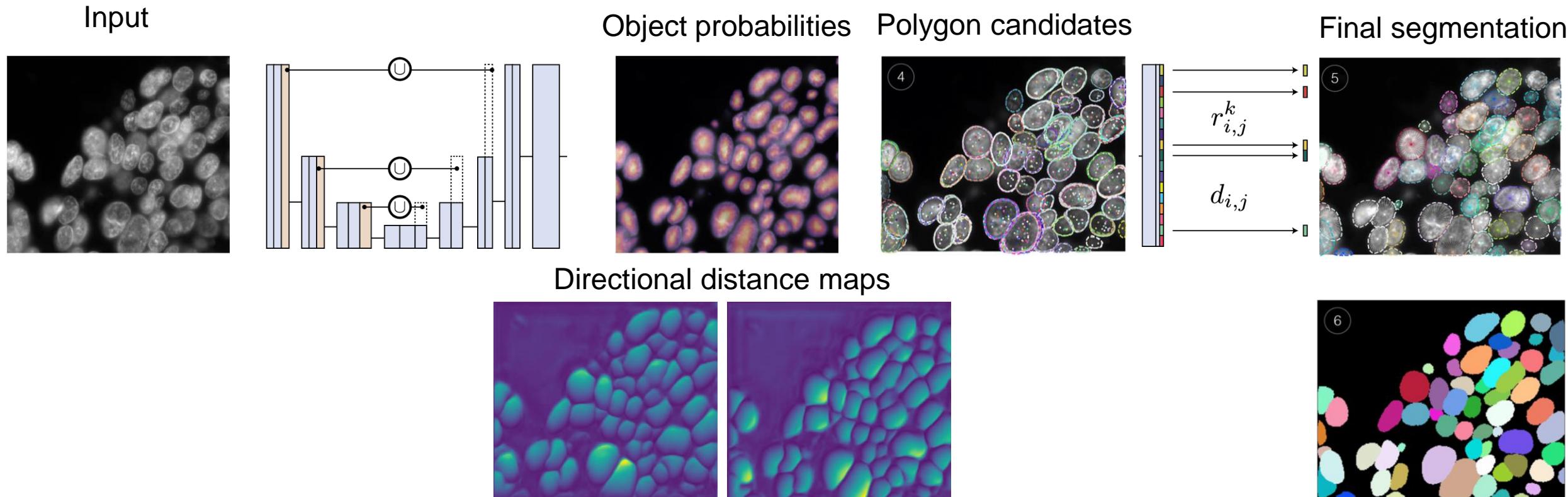
Instance Segmentation
(individual cells)

CellPose

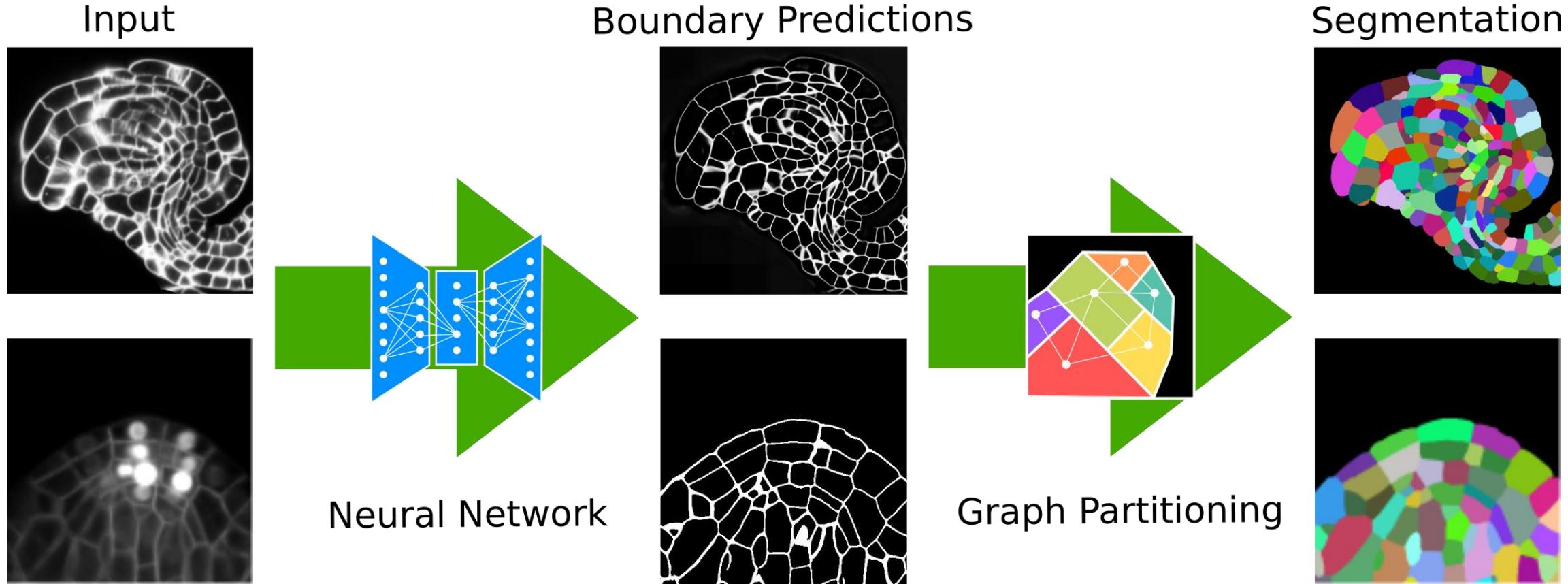
- Cell/Nuclei – segmentation based on flow-fields
- Technically similar to Watershed, but with a deep-learning based altitude-image



- Prediction of probable object centers + polygon outlines
- Non-maximum-suppression of less likely polygons

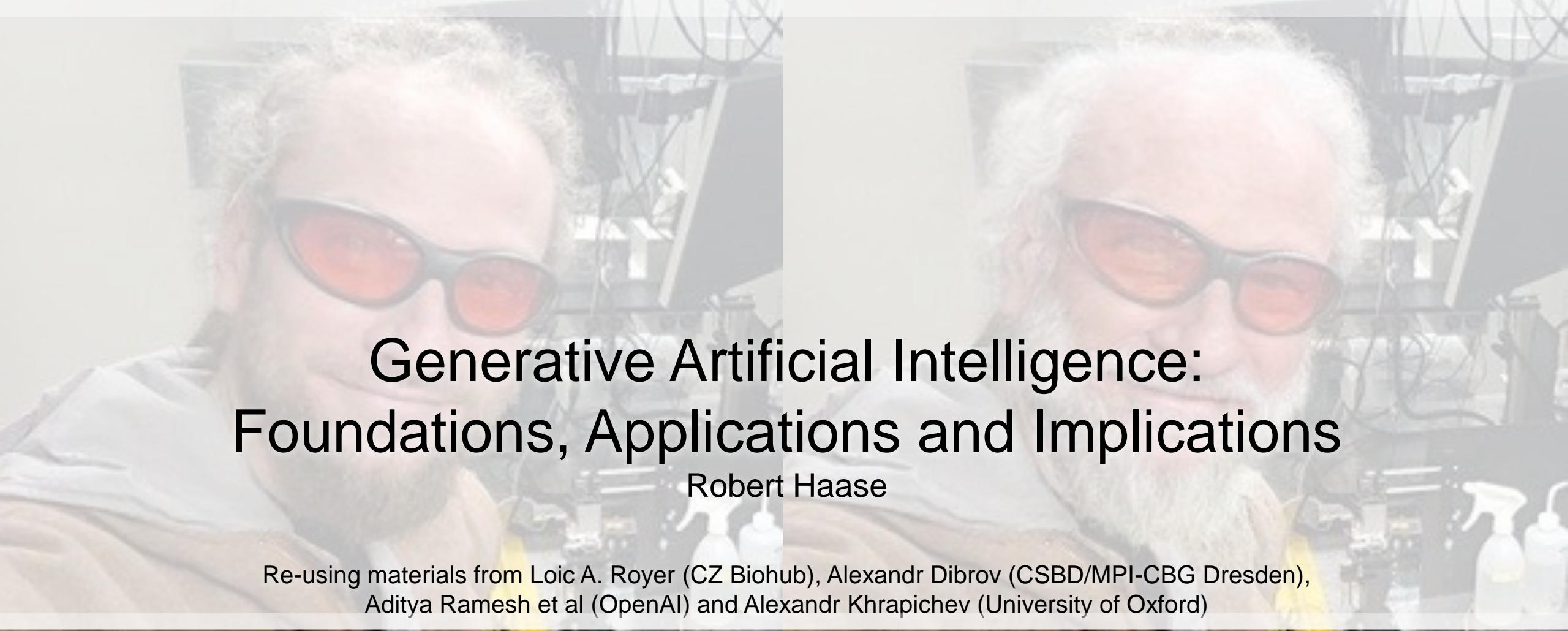


- Combination of neural networks + graph partitioning



Summary: Deep Learning for Bio-image Analysis

- [Convolutional] Neural Networks is a decade old technology that enabled breakthroughs in various fields during the last decade. Examples in Microscopy:
 - Image Denoising
 - Image Segmentation
- Common scheme: Smart algorithms for processing input/output of neural networks
 - ~~Image in, segmentation out~~
- Deep-Learning based often represent state-of-the-art techniques with respect to result quality
- Training these models is computationally expensive and requires a certain level of expertise
- If Otsu-Thresholding + Connected Component Labeling does the job, don't dive into deep learning!



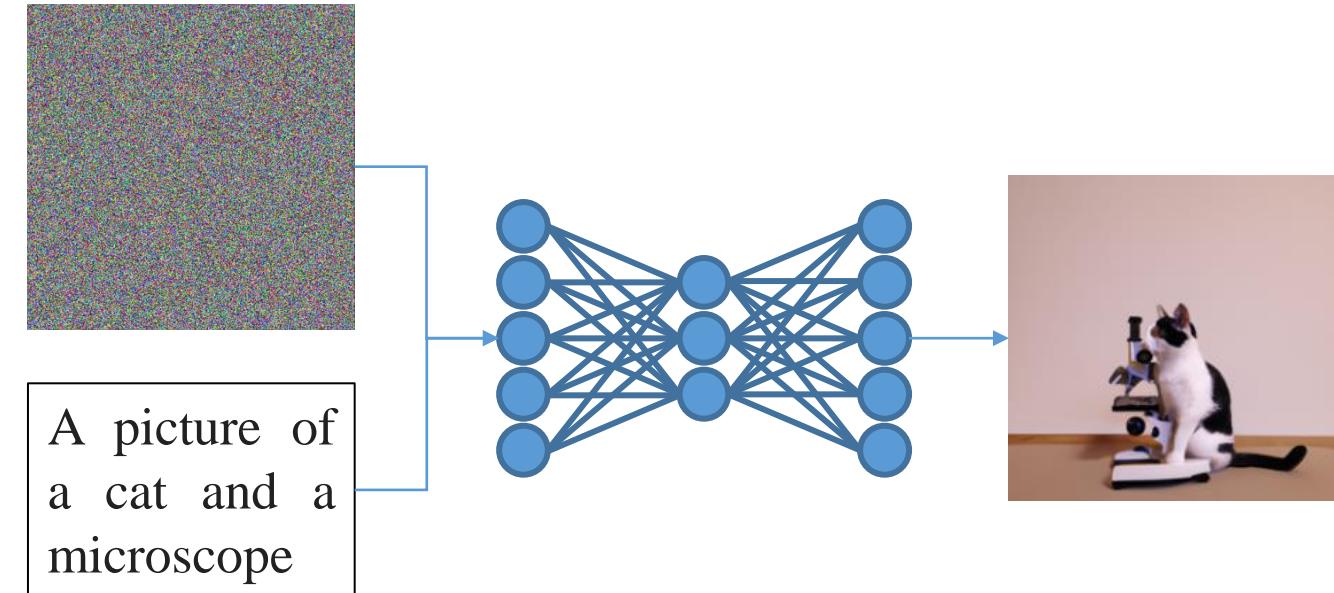
Generative Artificial Intelligence: Foundations, Applications and Implications

Robert Haase

Re-using materials from Loic A. Royer (CZ Biohub), Alexandr Dibrov (CSBD/MPI-CBG Dresden),
Aditya Ramesh et al (OpenAI) and Alexandr Khrapichev (University of Oxford)

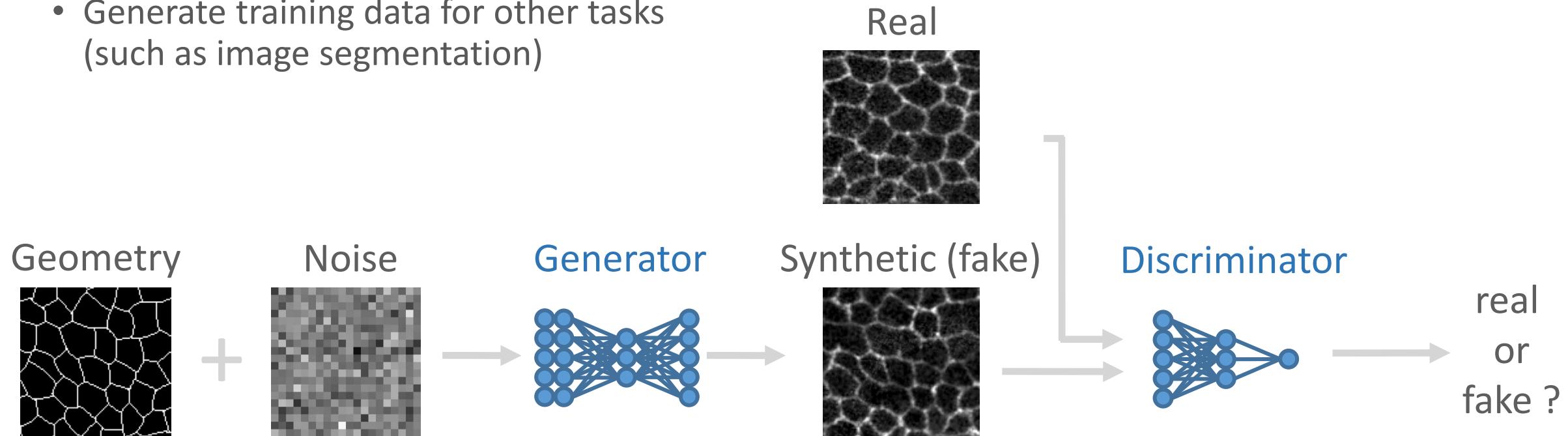
Generative Artificial Intelligence

- Definition: “Generative artificial intelligence [...] is a type of artificial intelligence (AI) system capable of generating text, images, or other media in response to prompts.”¹
- Commonly based on Neural Networks
- Bridges fields:
 - Natural Language Processing (NLP)
 - Computer Vision (CV)
- Use-cases
 - Translating text
 - Writing emails, text, grant proposals
 - Summarizing articles
 - Writing code
 - General question answering
 - Image generation



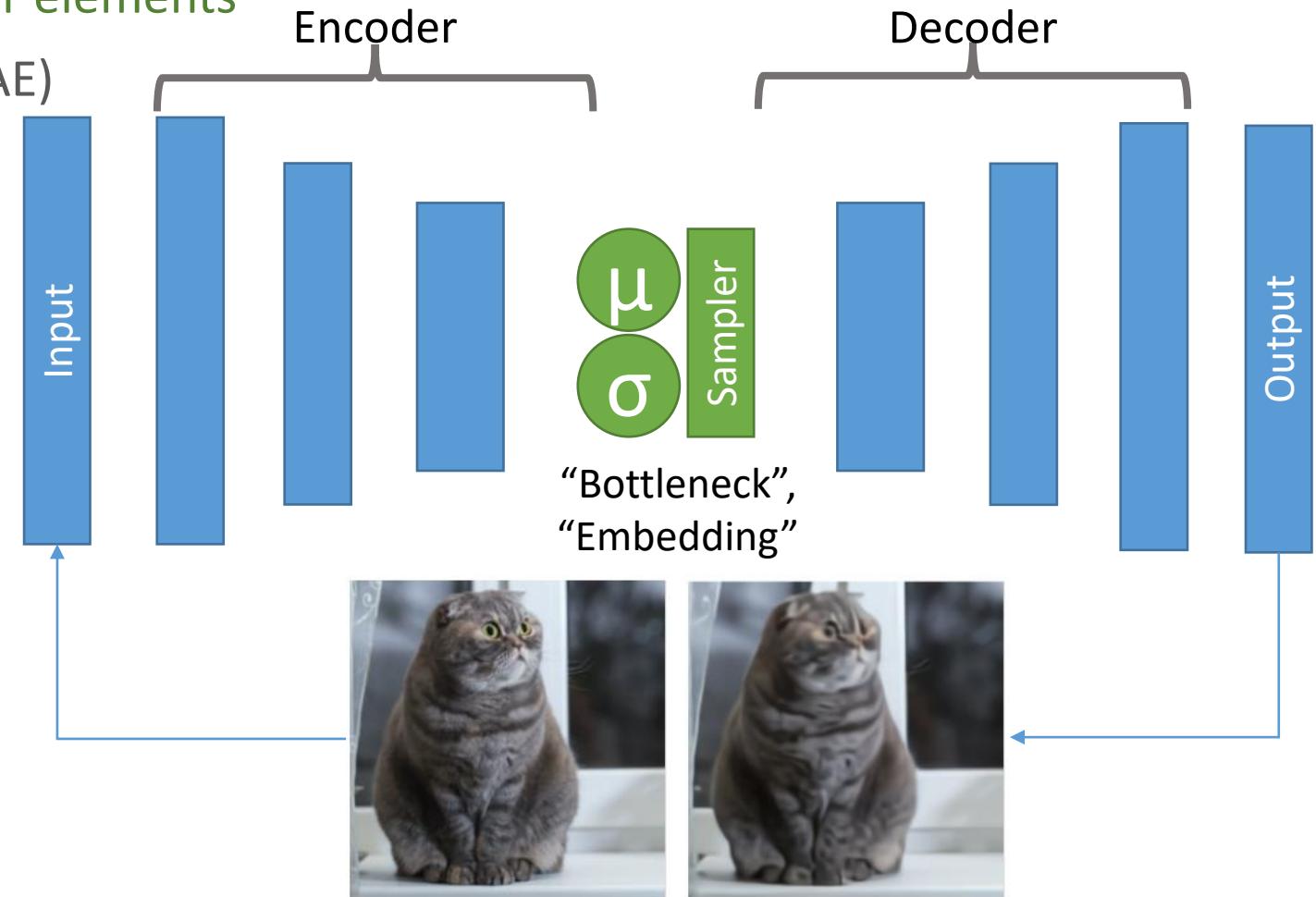
How does it work?

- Combination of **neural networks**
- Example: Generative Adversarial Networks (GANs)
- Use-cases:
 - Generate training data for other tasks (such as image segmentation)



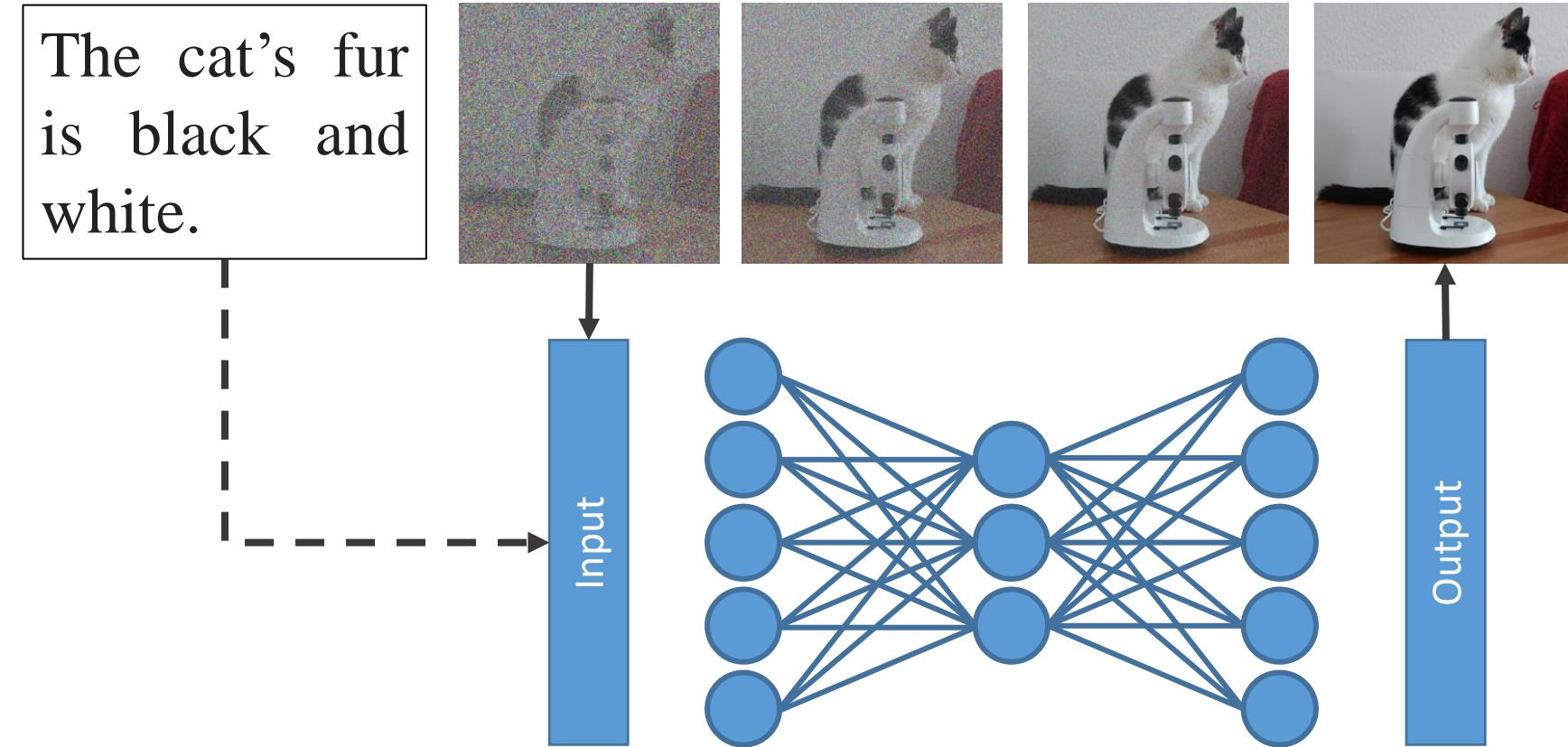
How does it work?

- Combination of **neural networks** + **other elements**
- Example: Variational Auto-Encoders (VAE)
- Use-cases:
 - Combine images (cat, microscope)



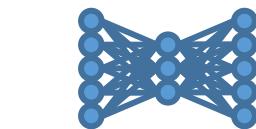
How does it work?

- Combination of neural networks + other elements + various data sources
- Examples: GPT / DALL-E, Stable Diffusion
- Use-case:
 - Generate image from noise + text



How does it work?

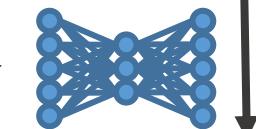
- Combination of neural networks + other elements + various data sources
- Examples: GPT / DALL-E, Stable Diffusion
- Use-case:
 - Generate image from noise + text



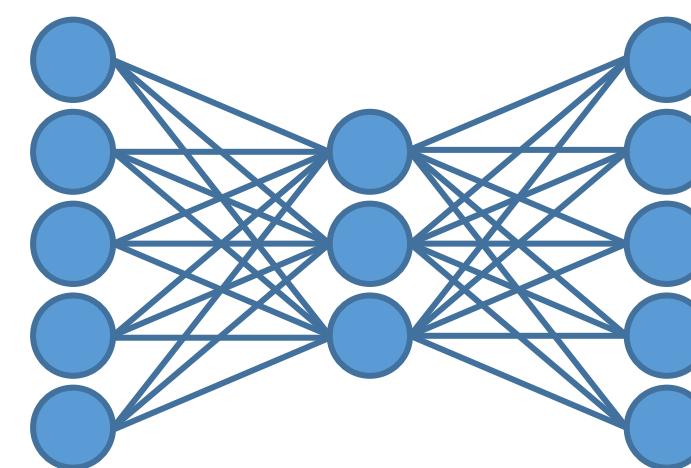
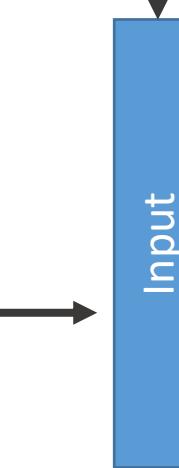
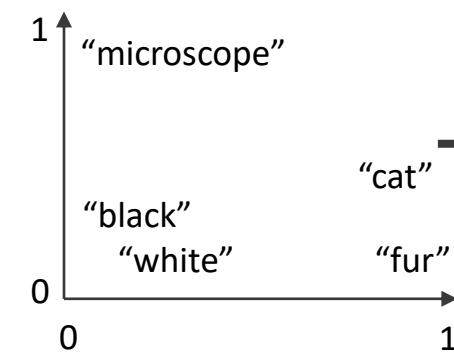
Self-attention

The cat's fur is black and white.

The cat's fur
is black and
white.



Word Embedding

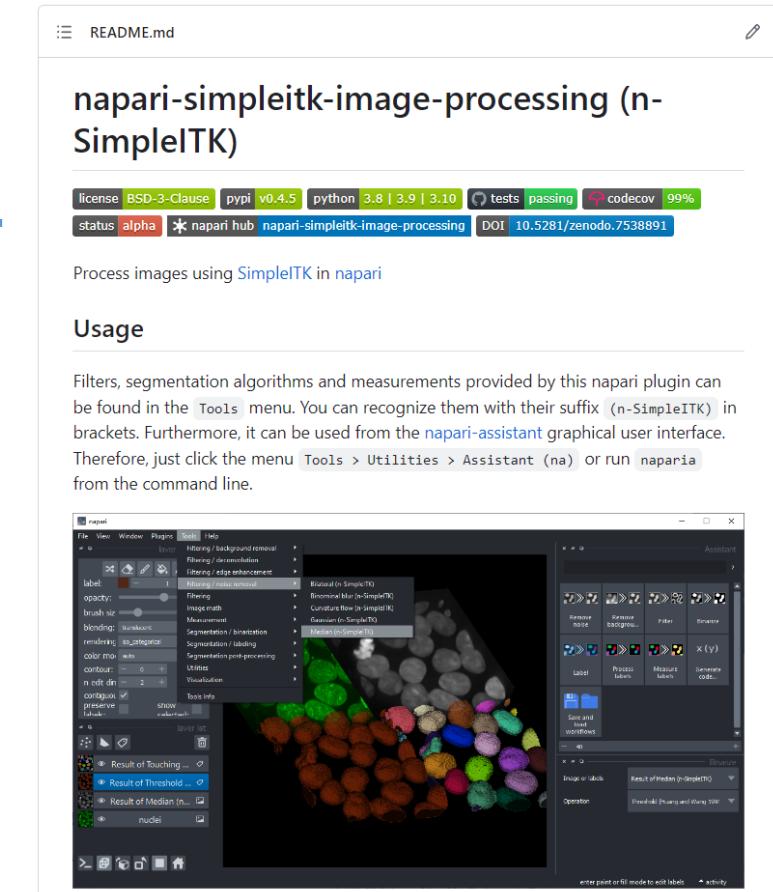
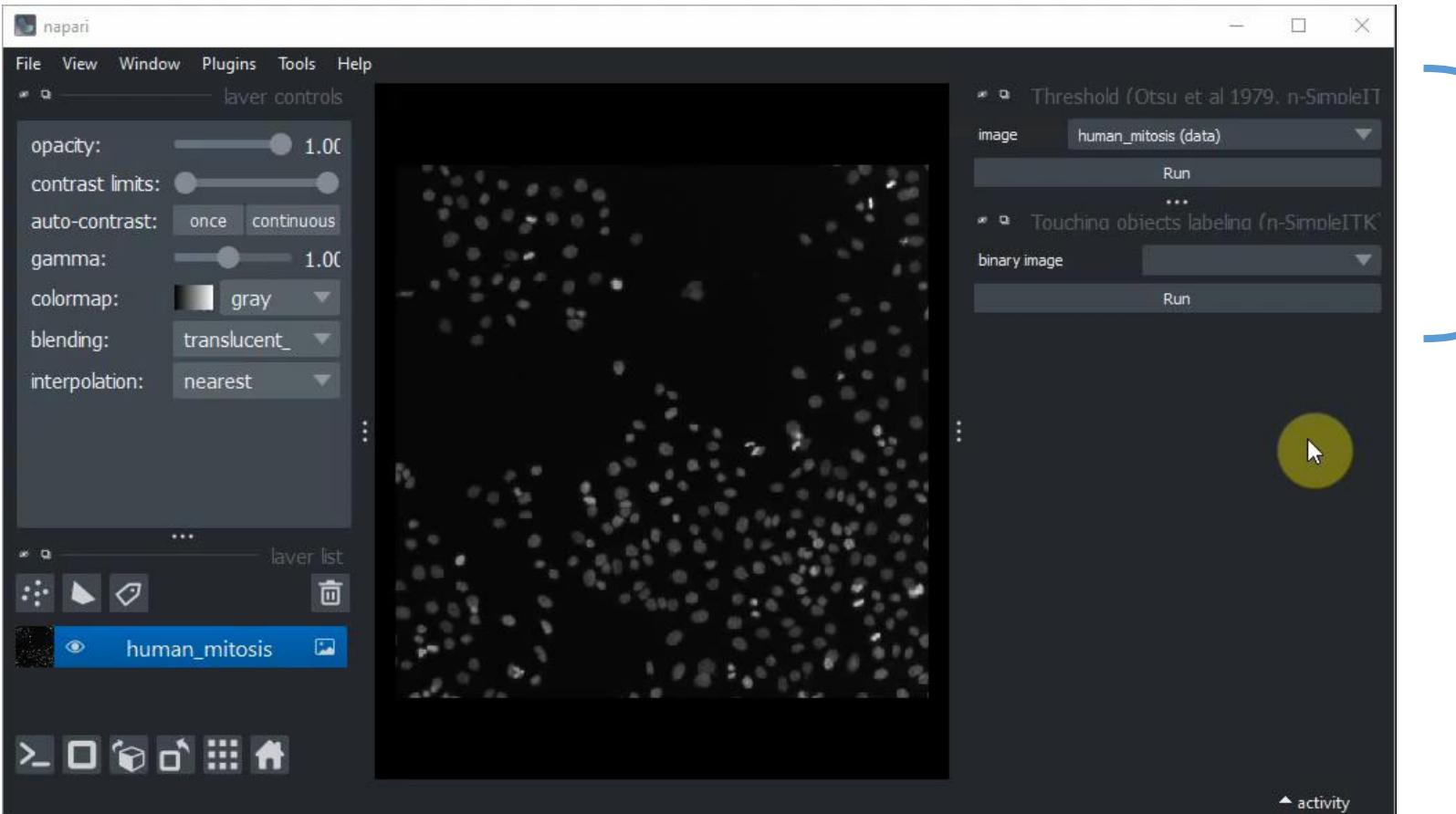


Deconstruction

- Deconstruction is a method in software engineering to
 - understand how existing software works
 - prevent reinventing the wheel
 - identify limitations
 - identify bottlenecks
- Related methods
 - Reverse engineering
 - Code review
 - Pair programming
- In the age of computers writing code, reading code is a key skill.

Deconstruction of napari-chatGPT

- Napari is a Python-based image viewer, extensible via plugins / “widgets”.



The top part shows the GitHub README page for the 'napari-simpleitk-image-processing' plugin. It includes the plugin's name, license (BSD-3-Clause), PyPI version (v0.4.5), supported Python versions (3.8 | 3.9 | 3.10), test status (tests passing), code coverage (codecov 99%), and GitHub status (alpha, napari hub). Below is a brief description: 'Process images using SimpleITK in napari'. The bottom part shows a screenshot of the graphical user interface for the plugin, which is a tabbed interface with tabs like 'Label', 'Filtration / denoising', 'Segmentation / binarization', and 'Segmentation post-processing'. It shows a 3D volume with various colored regions and a list of operations on the right.

Deconstruction of napari-chatGPT

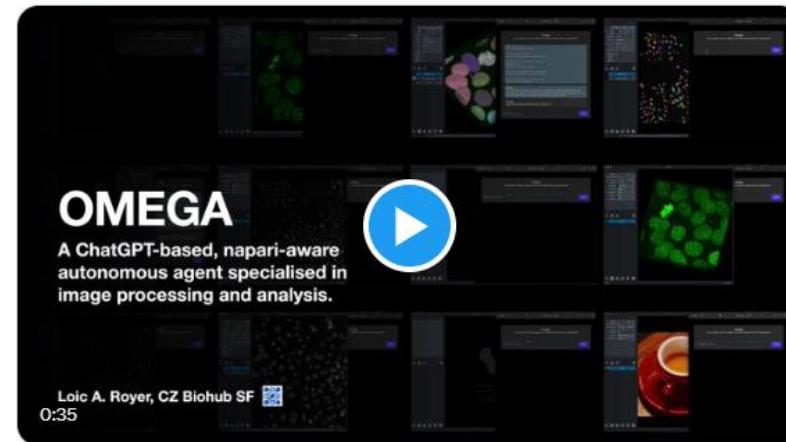
- Napari-chatGPT can automate programming plugins / “widgets”.



Loïc A. Royer    
@loicaroyer

#ChatGPT + @napari_imaging

Releasing my latest ~~weekend~~ project: Omega, an autonomous LLM agent that writes image processing and analysis code, fixes its mistakes, accesses the napari viewer, makes widgets, & more!
[github.com/royerlab/napari...](https://github.com/royerlab/napari-chatgpt)
@LangChainAI @OpenAI
#OmegaAgent

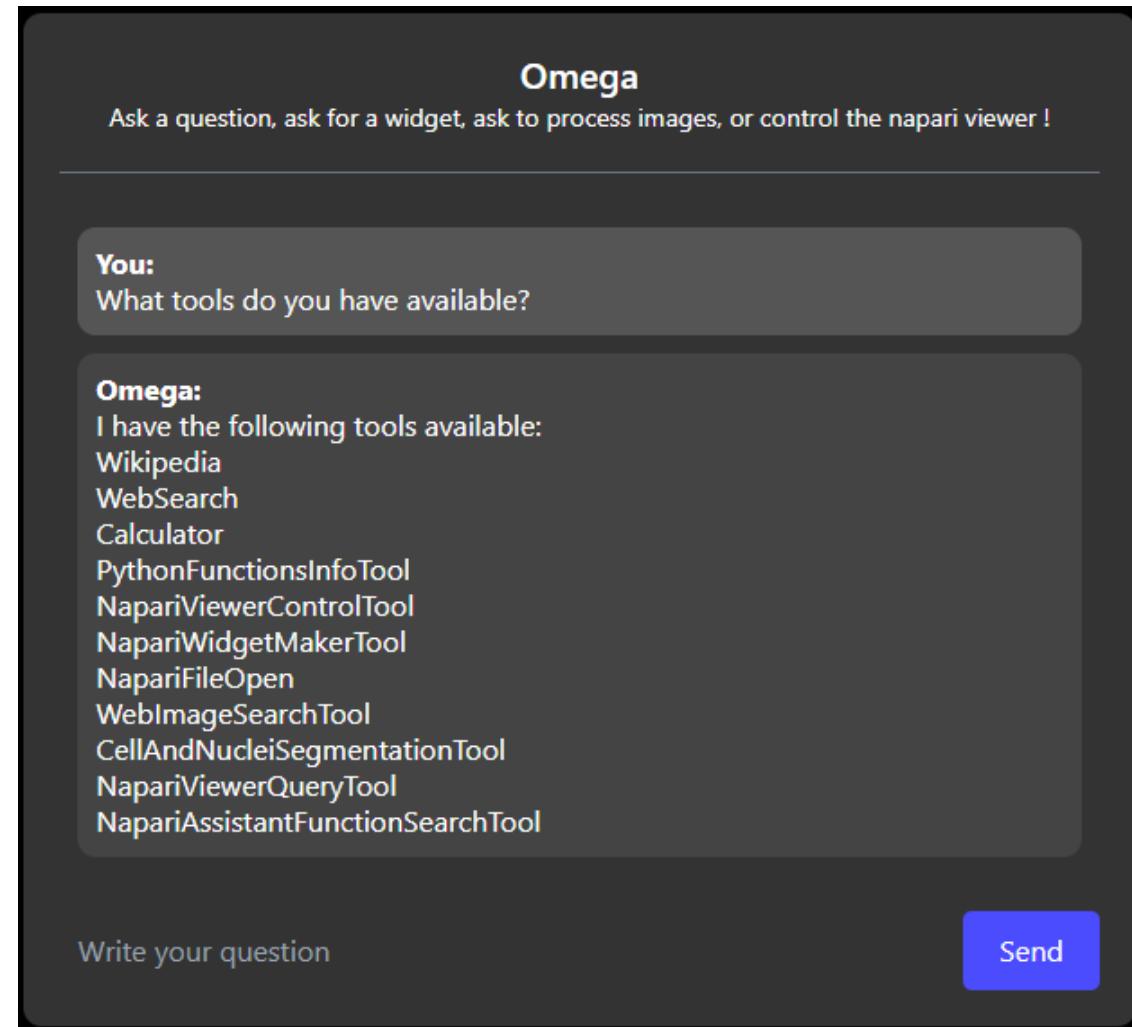
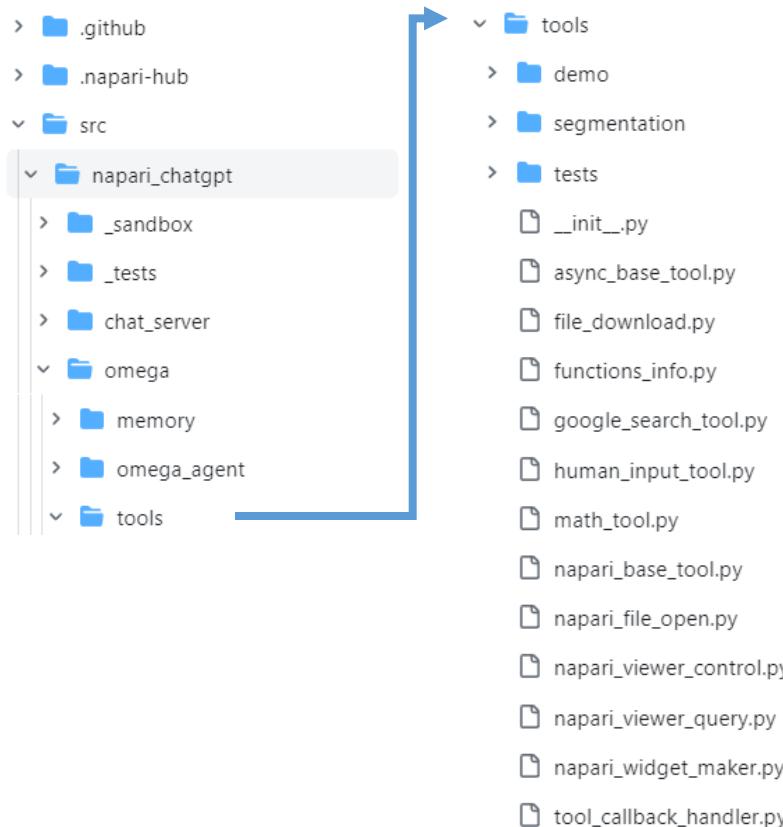


5:19 AM · May 3, 2023 · 183.6K Views

208 Retweets 44 Quotes 770 Likes 424 Bookmarks

Deconstruction of napari-chatGPT

- Napari-chatGPT defines a list of “tools”



The screenshot shows a dark-themed AI interface titled "Omega". At the top, it says "Ask a question, ask for a widget, ask to process images, or control the napari viewer !". Below this, there's a "You:" input field containing "What tools do you have available?". The AI response, under "Omega:", lists various tools: Wikipedia, WebSearch, Calculator, PythonFunctionsInfoTool, NapariViewerControlTool, NapariWidgetMakerTool, NapariFileOpen, WebImageSearchTool, CellAndNucleiSegmentationTool, NapariViewerQueryTool, and NapariAssistantFunctionSearchTool. At the bottom, there's a "Write your question" input field and a blue "Send" button.

Deconstruction of napari-chatGPT

- Napari-chatGPT defines a list of “tools”

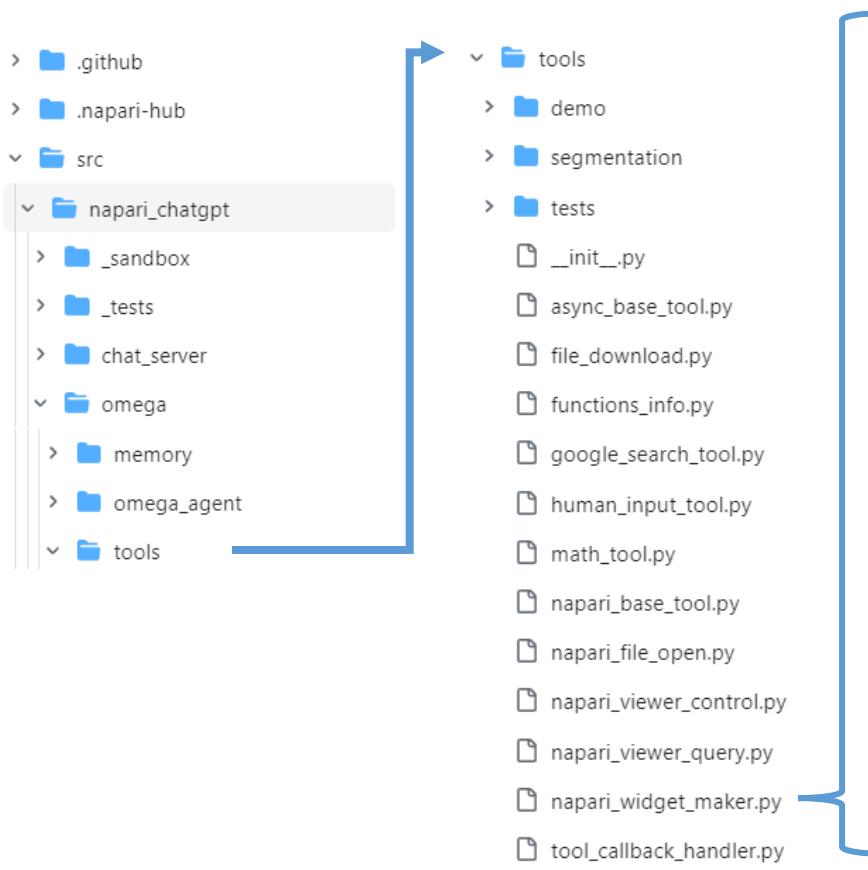
The image shows a file explorer on the left and a code editor on the right. The file explorer highlights the 'napari_chatgpt' folder, which contains subfolders like '.github', '.napari-hub', 'src', and 'napari_chatgpt'. The 'napari_chatgpt' folder is expanded to show files such as '_init_.py', 'async_base_tool.py', 'file_download.py', 'functions_info.py', 'google_search_tool.py', 'human_input_tool.py', 'math_tool.py', 'napari_base_tool.py', 'napari_file_open.py', 'napari_viewer_control.py', 'napari_viewer_query.py', 'napari_widget_maker.py', and 'tool_callback_handler.py'. A blue bracket on the right side groups the 'tools' folder and its contents. The code editor displays two Python functions:

```
1  from napari_chatgpt.omega.tools.async_base_tool import AsyncBaseTool
2  from napari_chatgpt.utils.google import search_overview
3
4
5  class GoogleSearchTool(AsyncBaseTool):
6      name = "GoogleSearch"
7      description = "Useful for when you need to answer questions by querying the web."
8
9      def _run(self, query: str) -> str:
10         """Use the tool."""
11         result = search_overview(query=query)
12         return result
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58  def search_overview(query: str,
59                      num_results: int = 3,
60                      lang: str = "en",
61                      max_text_snippets: int = 5,
62                      max_query_freq_hz: float = 1.0) -> str:
63      url = f"https://www.google.com/search?q={query}&num={num_results}&hl={lang}"
64      text = text_from_url(url,
65                           max_text_snippets=max_text_snippets,
66                           max_query_freq_hz=max_query_freq_hz)
67      return text
```

https://github.com/royerlab/napari-chatgpt/blob/aa00b48d0523da588ece5dc8cdb006c895c4b473/src/napari_chatgpt/omega/tools/google_search_tool.py
https://github.com/royerlab/napari-chatgpt/blob/aa00b48d0523da588ece5dc8cdb006c895c4b473/src/napari_chatgpt/utils/google.py#L58
Code origin: <https://github.com/Nv7-GitHub/googlesearch>

Deconstruction of napari-chatGPT

- Napari-chatGPT defines a list of “tools”



```
10    _napari_widget_maker_prompt = """
11
12 Task:
13 You competently write image processing and image analysis functions in python given a plain text request.
14 The function should be pure, self-contained, effective, well-written, syntactically correct.
15 The function should work on 2D and 3D images, and images of any number of dimensions (nD),
16 unless the request is explicit about the number of dimensions.
17 The widget should do all and everything that is asked, but nothing else or superfluous.
18
19 Instructions for Function Signature:
20 - Integers, floats, boolean, or any other type accepted by the magicgui library.
21 - Decorate the function with the magicgui decorator: '@magicgui(call_button='Run')' where <function_name>
22 - DO NOT CREATE A NEW INSTANCE OF A NAPARI VIEWER, use the one provided in the variable: 'viewer'.
23 - DO NOT write code to add the widget to the napari window by calling viewer.window.add_dock_widget().
24
25 The function signature should have a type hint for the return, e.g. -> ImageData or -> Image
26
27 {generic_codegen_instructions}
28
29 Request:
30 {input}
```

Our prompt goes here

https://github.com/royerlab/napari-chatgpt/blob/aa00b48d0523da588ece5dc8cdb006c895c4b473/src/napari_chatgpt/omega/tools/napari_widget_maker.py#L10

Prompt engineering

- chatGPT can solve simple image analysis tasks

Prompt

```
simple_question = """  
Write Python code only and no additional explanatory text.  
  
Write a python program, that  
* loads the file `../../data/blobs.tif`,  
* labels objects in this image, and  
* visualize results.  
  
Assume this program would be executed in a Jupyter notebook.  
It is not necessary to save the results. Show the results in Jupyter.  
"""
```

Response

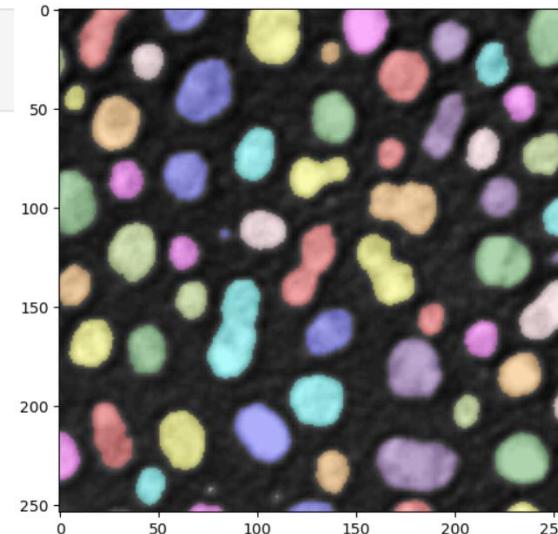
```
code = prompt(simple_question)  
print(code)  
  
```python  
from skimage.io import imread
from skimage.filters import threshold_otsu
from skimage.measure import label
from skimage.color import label2rgb
import matplotlib.pyplot as plt

image = imread(' ../../data/blobs.tif')
thresh = threshold_otsu(image)
binary = image > thresh
labels = label(binary)
image_label_overlay = label2rgb(labels, image=image)

fig, ax = plt.subplots(figsize=(10, 6))
ax.imshow(image_label_overlay)

plt.show()
```
```

Result



Prompt engineering

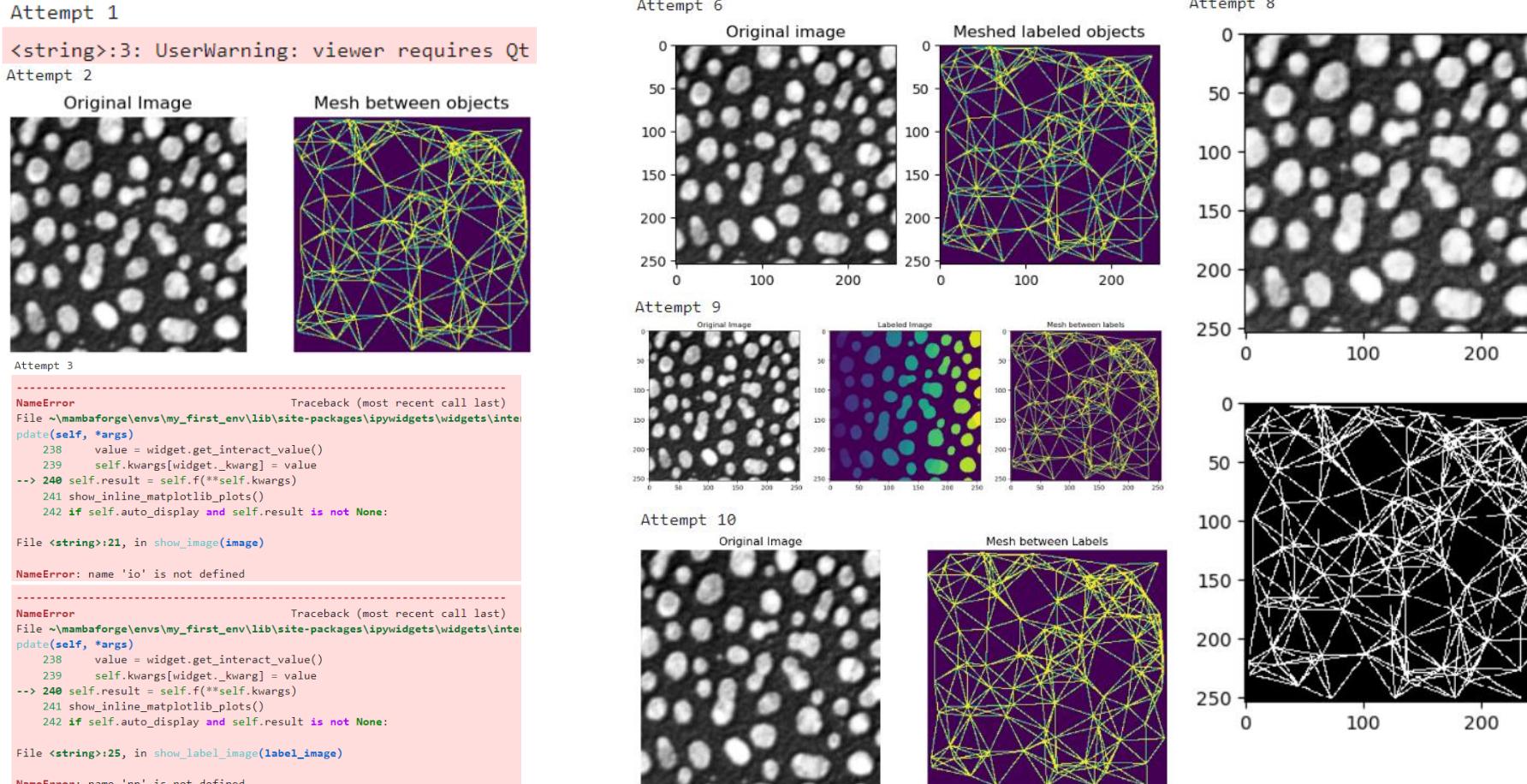
- With more advanced tasks, it might need additional help.

```
simple_question = """  
Write Python code only and no additional explanatory text.  
  
Write a python program, that  
* loads the file `../../data/blobs.tif`,  
* labels objects in this image,  
* and draws a mesh between labels with a maximum distance of 50 pixels.  
  
Assume this program would be executed in a Jupyter notebook.  
It is not necessary to save the results. Show the results in Jupyter.  
"""
```

```
more_sophisticated_question = """  
Please program some python code like a professional would.  
Write Python code only and no additional explanatory text.  
  
Write a python program, that  
* loads the file `../../data/blobs.tif`,  
* labels objects using voronoi-otsu-labeling,  
* and draws a mesh between labels with a maximum distance of 50 pixels.  
  
I have this code snippet for segmenting an image:  
import pyclesperanto_prototype as cle  
label_image = cle.voronoi_otsu_labeling(image)  
  
And this is the code snippet for drawing a mesh between objects in a label image:  
mesh = cle.draw_mesh_between_proximal_labels(labels, maximum_distance:int)  
  
Assume this program would be executed in a Jupyter notebook.  
It is not necessary to save the results. Show the results in Jupyter.  
"""
```

Prompt engineering

- With more involved tasks, it might need additional help.



The *more sophisticated* prompt produced useful results in 5 out of 10 runs.

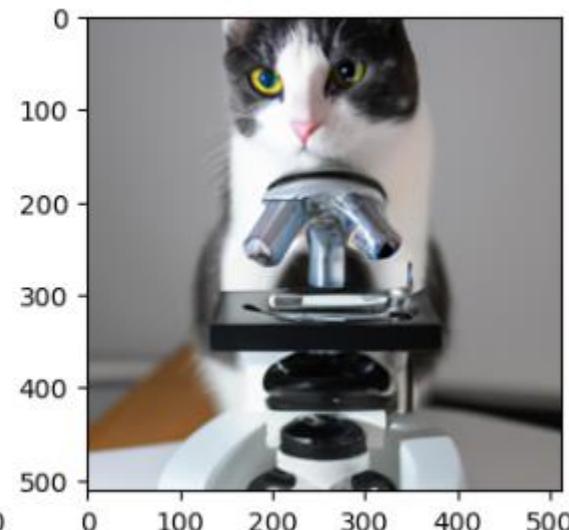
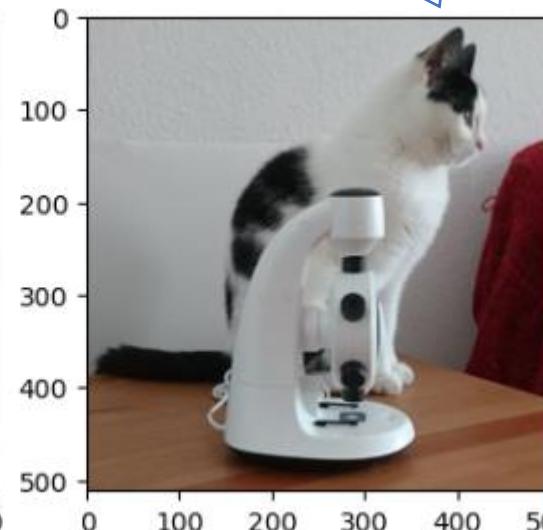
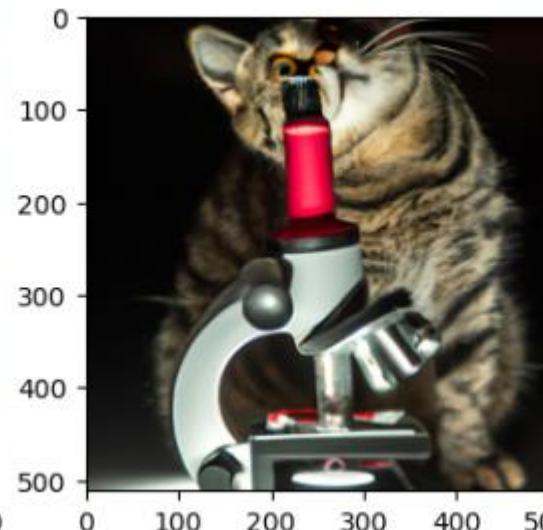
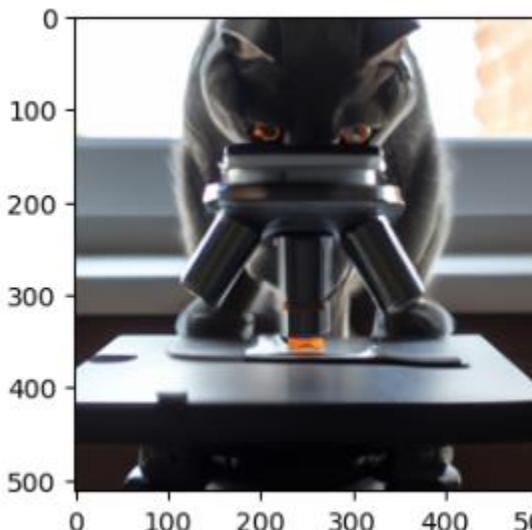
The *more sophisticated* prompt had errors in 4 out of 10 runs.

Prompt engineering

- Prompts can be used to write code, but also to generate images, e.g. with DALL-E.
One can generate quite realistic images given a detailed prompt.

```
cat_microscope_prompt = """  
Image of a cat sitting behind a microscope.  
"""
```

One cat
is real.



Adapted from:

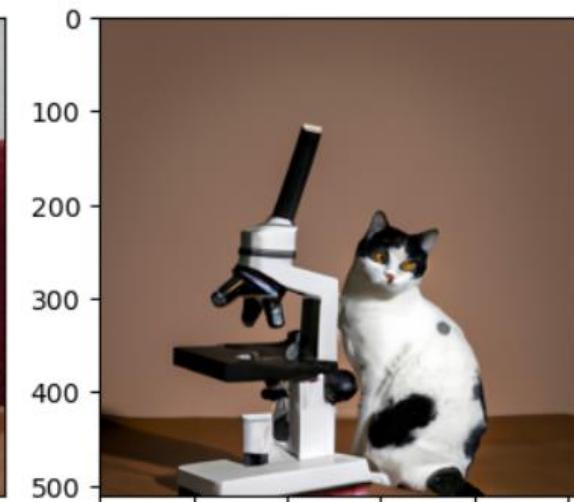
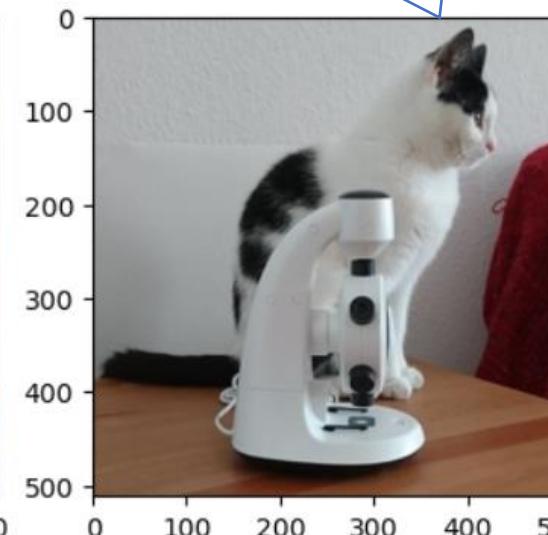
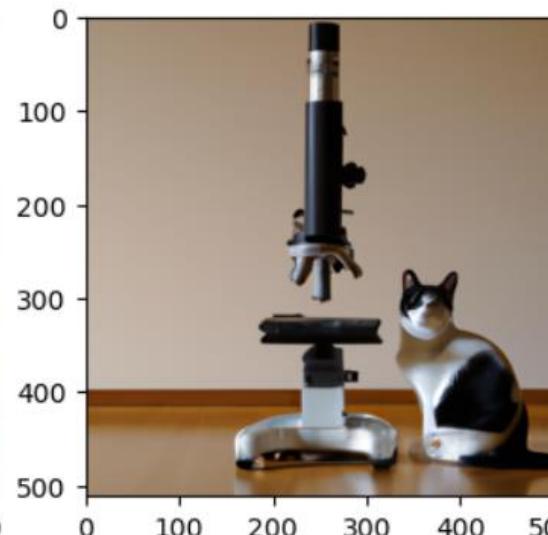
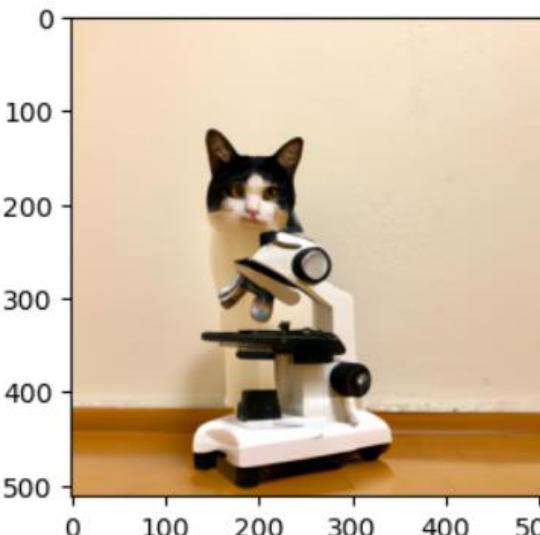
https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/07_prompt_engineering/02_generating_images.html

Prompt engineering

- Prompts can be used to write code, but also to generate images, e.g. with DALL-E.
One can generate quite realistic images given a detailed prompt.

```
[5]: cat_microscope_prompt = """  
Image of a cat sitting behind a microscope.  
Both are on a brown floor in front of a white wall.  
The cat is mostly white and has some black dots.  
The cat sits straight.  
The cat is a bit larger than the microscope.  
"""
```

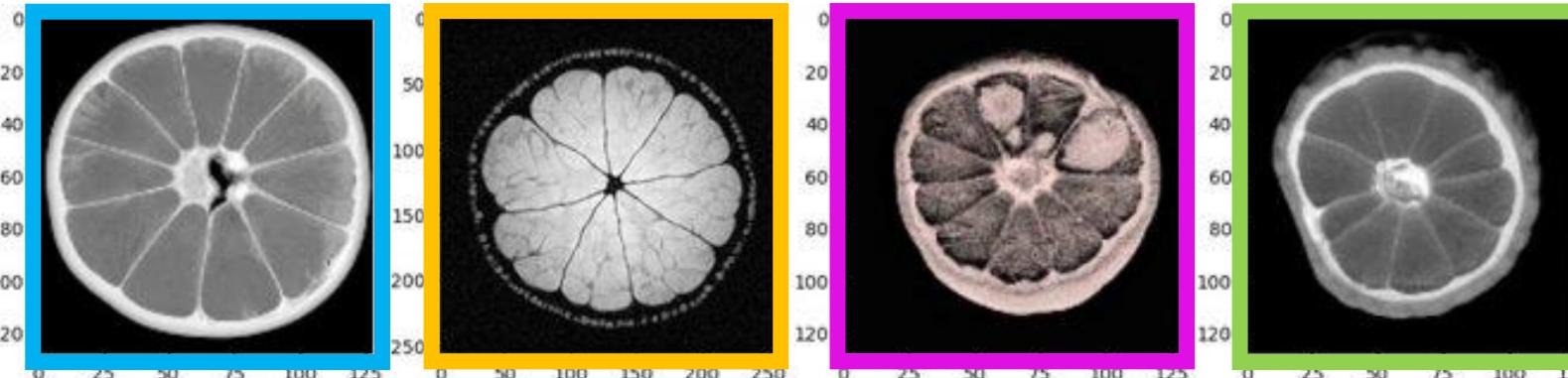
One cat
is real.



Prompt engineering

- Prompts can be used to write code, but also to generate images, e.g. with DALL-E.
One can generate quite realistic images given a detailed prompt.

```
mri_prompt = """  
A single, high resolution, black-white image of  
a realistically looking orange fruit slice  
imaged with T2-weighted magnetic resonance imaging (MRI).  
"""
```



Robert Haase @haesleinhuepf · 20h

Fun poll time! Which of these images shows a real MRI image of an orange? (Credits: licensed CC-BY 4.0 by Alexandr Khrapichev, University of Oxford; the other images were generated by @openai's DALL-E)

Please vote below, RT and if you can explain why, please comment! 😊

0 8 1 16.8%
1 6 2 67.6%
2 15 3 3.4%
3 4,880 4 12.3%

179 votes · Final results

- Quality assurance results depend not only on the language model, but also on the knowledge of the observers.

Deconstruction of napari-chatGPT

- LangChain is used to combine tools.
- It uses chatGPT under the hood.

```
def upper_case(text:str):
    return text.upper()

def reverse(text:str):
    return text[::-1]
```

```
tools = [
    Tool(
        name="Upper case",
        func=upper_case,
        description="Useful for making a text uppercase or capital letters."
    ),
    Tool(
        name="Reverse",
        func=reverse,
        description="Useful for making reversing order of a text."
    ),
]
```



⚡ Building applications with LLMs through composability ⚡

 lint passing  test passing  linkcheck passing  downloads/month 1M  License MIT

```
[4]: memory = ConversationBufferMemory(memory_key="c")
      llm=ChatOpenAI(temperature=0)
```

```
[5]: agent = initialize_agent(
    tools,
    llm,
    agent=AgentType.CHAT_CONVERSATIONAL_REACT_DESCRIPTION,
    memory=memory
)
```

Deconstruction of napari-chatGPT

- After combining tools, large language model and memory in an *agent*, you can interact with it.

```
[6]: agent.run(input="Hi, I am Robert")
```

```
[6]: 'Nice to meet you, Robert!'
```

```
[7]: agent.run(input="What's my name?")
```

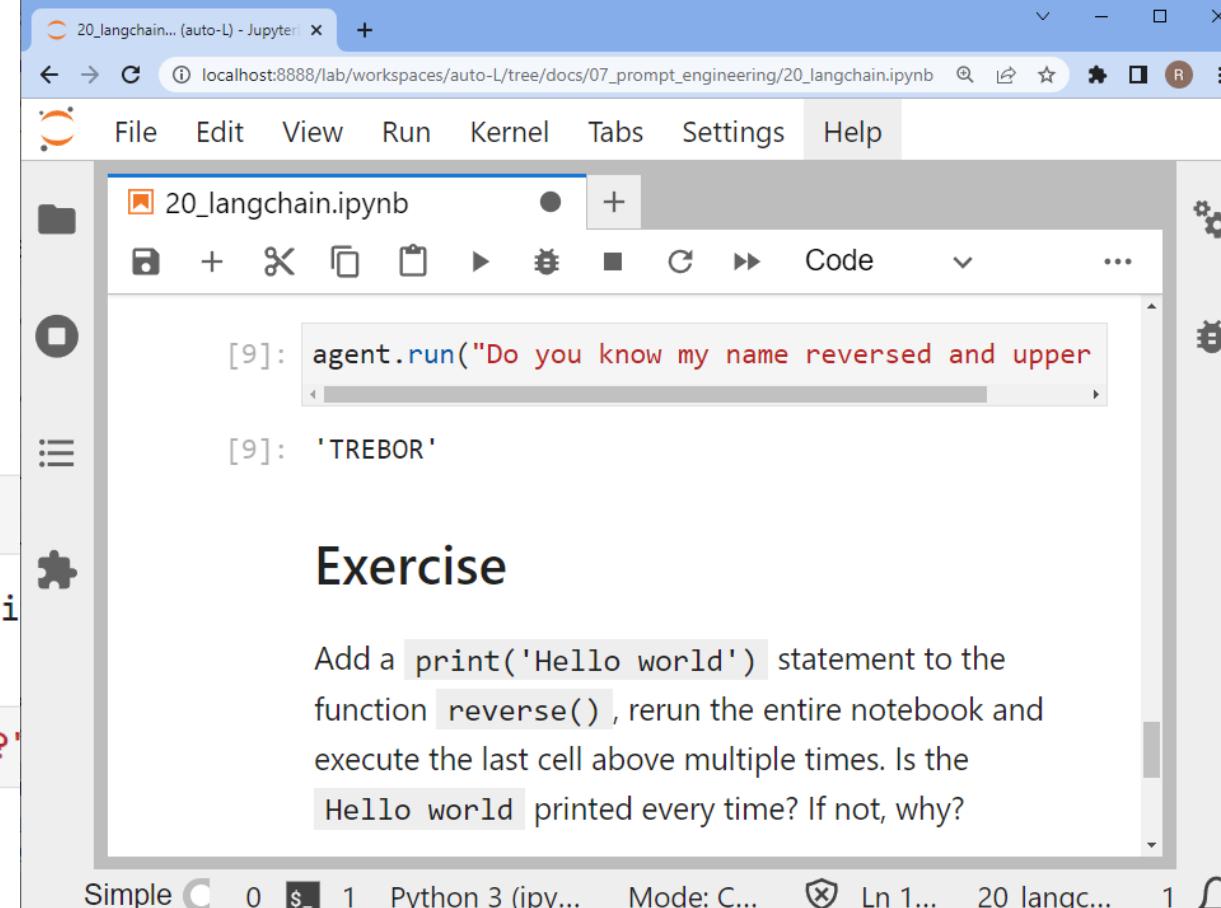
```
[7]: 'Your name is Robert'
```

```
[8]: agent.run("Can you reverse my name?")
```

```
[8]: "The response to your last comment was 'treboR', which is"
```

```
[9]: agent.run("Do you know my name reversed and upper case?")
```

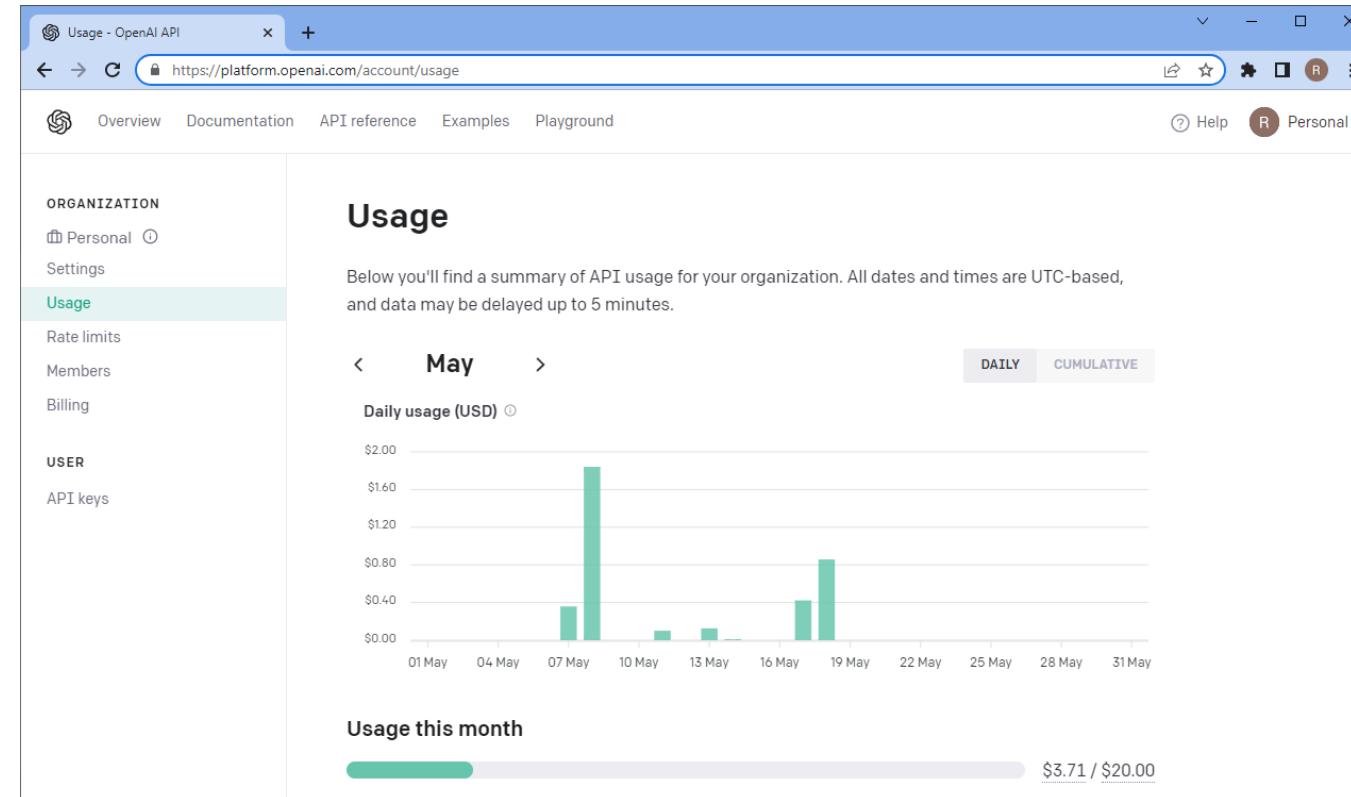
```
[9]: 'TREBOR'
```



The screenshot shows a Jupyter Notebook window titled '20_langchain.ipynb' running on 'localhost:8888'. The notebook contains several code cells and their outputs. Cell [6] outputs 'Nice to meet you, Robert!'. Cell [7] outputs 'Your name is Robert'. Cell [8] outputs the instruction 'Can you reverse my name?'. Cell [9] outputs 'Do you know my name reversed and upper case?' followed by 'TREBOR'. A large text block labeled 'Exercise' provides instructions: 'Add a `print('Hello world')` statement to the function `reverse()`, rerun the entire notebook and execute the last cell above multiple times. Is the `Hello world` printed every time? If not, why?' The bottom status bar shows the notebook is in Python 3 mode.

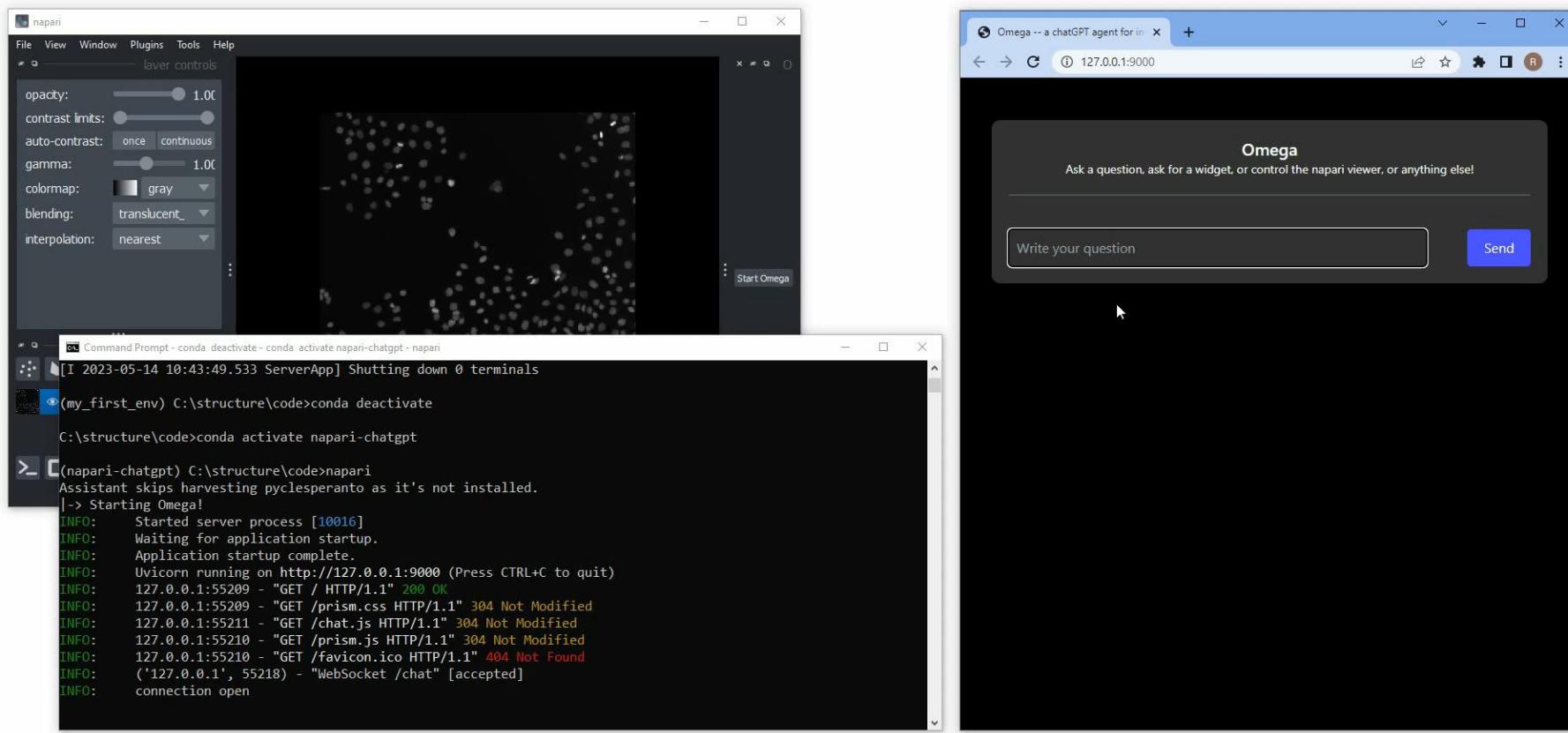
How much is the fish?

- Executing napari-ChatGPT and LangChain requires an OpenAI account.
- Using OpenAI infrastructure costs real money.



A little warning by the end

- napari-chatGPT executes code and installs software on your machine.
- Use it with care! E.g. in a virtual machine / sandbox



Generative Artificial Intelligence

- Challenges
 - Data safety / security
 - Computational cost of training neural networks
 - CO2-footprint/climate change
 - Accessibility
 - Bias: “a nice photo of a human”
 - Hallucinations
 - Glitch tokens
 - False information / fake news
 - Debugging is hard



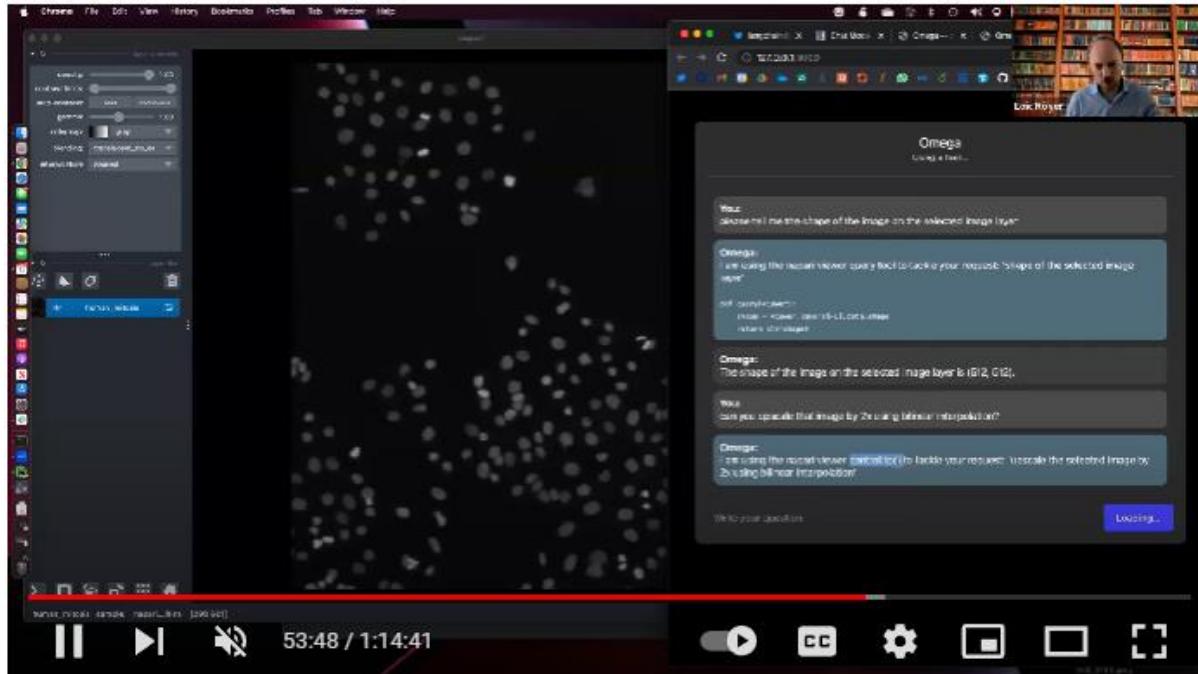
Model: Default (GPT-3.5)

R Who developed Gradoop?



Gradoop is an open-source framework for scalable graph analytics. It was developed by the [Hasso Plattner Institute \(HPI\) in Potsdam, Germany](#). The main contributors to Gradoop are researchers and developers from the HPI's Database Systems and Information Management (DIMA) group, led by [Prof. Dr. Felix Naumann](#). The project started in 2015 as part of the HPI's research on large-scale graph data management and analysis. Since then, it has gained popularity among researchers and industry professionals working with graph data.

Further reading / watching



Code review of Napari-ChatGPT by Loic Royer (CZI Biohub)



Scientific Computing ...
122 subscribers

Subscribe

Like 6

Dislike

Share

...

https://www.youtube.com/watch?v=JMo6Sn-L_j4



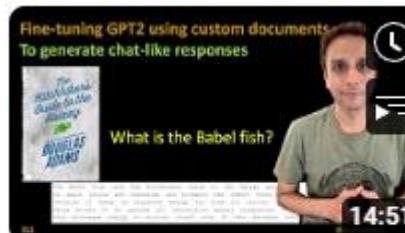
DigitalSreeni

@DigitalSreeni 71.3K subscribers 380 videos

This channel walks you through the entire process of learning to code in P ... >



308 - An introduction to language models with focu...



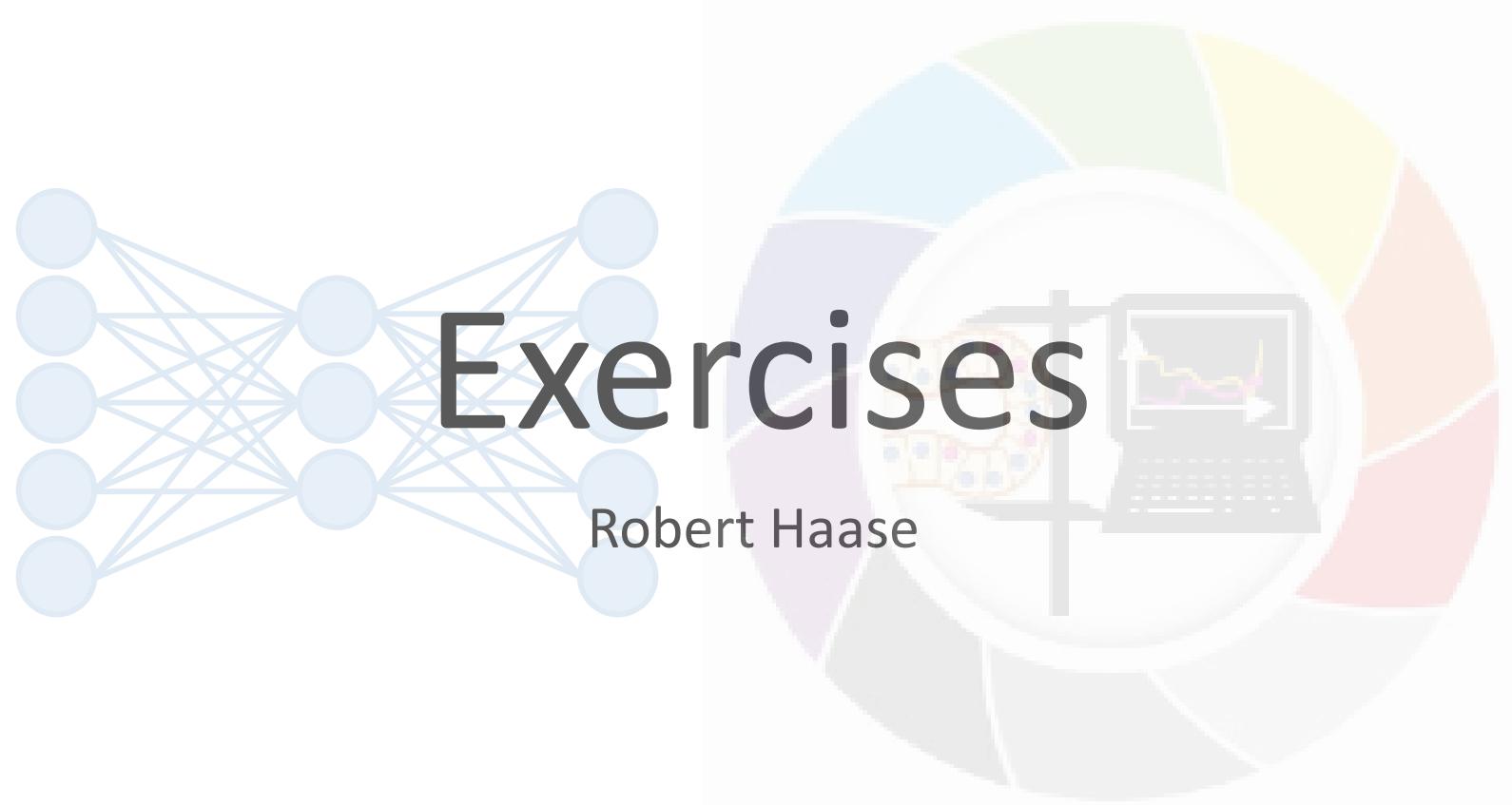
311 - Fine tuning GPT2 using custom documents



309 - Training your own Chatbot using GPT



<https://www.youtube.com/c/digitalsreeni>



Deep-learning based nuclei segmentation

Use CellPose and StarDist to segment the nuclei in the human-mitosis example dataset of scikit-image.

- [https://colab.research.google.com/github/BiAPoL/Bio-image Analysis with Python/blob/main/11 deep learning/stardist.ipynb](https://colab.research.google.com/github/BiAPoL/Bio-image_Analysis_with_Python/blob/main/11_deep_learning/stardist.ipynb)
- [https://colab.research.google.com/github/BiAPoL/Bio-image Analysis with Python/blob/main/11 deep learning/cellpose.ipynb](https://colab.research.google.com/github/BiAPoL/Bio-image Analysis with Python/blob/main/11_deep_learning/cellpose.ipynb)

The screenshot shows two stacked Google Colab notebooks. The top notebook, titled 'Image segmentation with StarDist', contains code for loading an image and using the StarDist model to predict instances. The bottom notebook shows the resulting segmented nuclei as a scatter plot with colored dots on a black background. The x-axis ranges from 0 to 400, and the y-axis ranges from 0 to 400.

```
[6]: labels, details = model.predict_instances(image)
stackview.insight(labels)

1/1 [=====] - 1s 1s/step
```

The screenshot shows two stacked Google Colab notebooks. The top notebook, titled 'Image Segmentation with CellPose', contains code for loading an image and using the CellPose model to evaluate it. The bottom notebook shows the resulting segmented nuclei as a scatter plot with colored dots on a black background. The x-axis ranges from 0 to 400, and the y-axis ranges from 0 to 400.

```
[5]: channels = [0,0] # This means we are processing single-channel greyscale images.

[6]: labels = masks.astype(np.uint32)
stackview.insight(labels)
```

The screenshot shows two stacked Google Colab notebooks. The top notebook, titled 'Image Segmentation with CellPose', contains code for installing necessary Python libraries and then using the CellPose model to evaluate the image. The bottom notebook shows the resulting segmented nuclei as a scatter plot with colored dots on a black background. The x-axis ranges from 0 to 400, and the y-axis ranges from 0 to 400.

```
[1]: !pip install cellpose stackview ipycanvas==0.11
```

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib

```
[1]: Requirement already satisfied: cycler>=0.10 in /usr/local/lib
```

Requirement already satisfied: fonttools>=4.22.0 in /usr/local

```
[1]: Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local
```

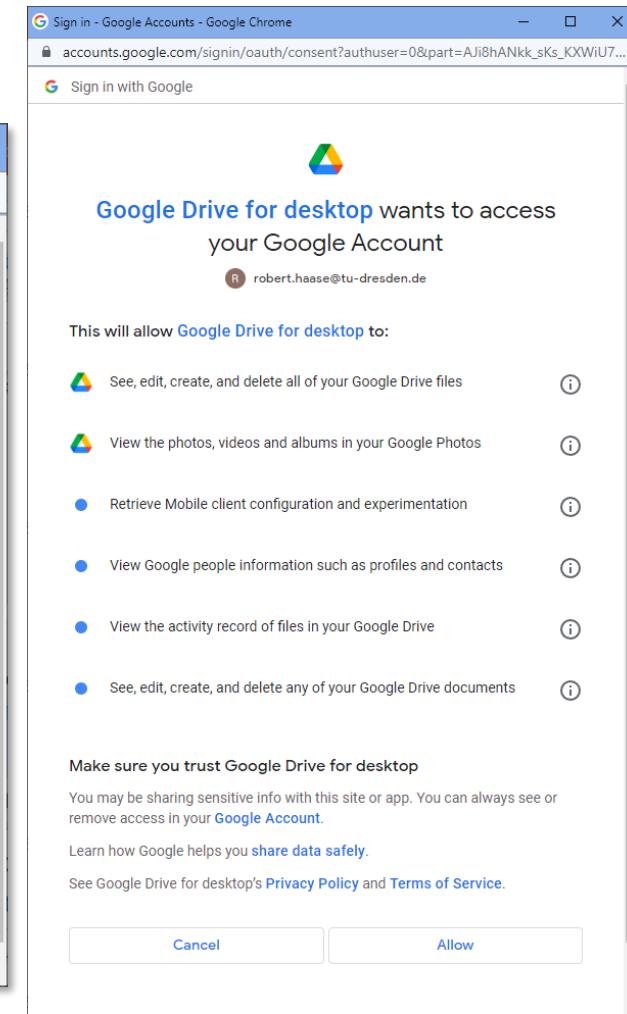
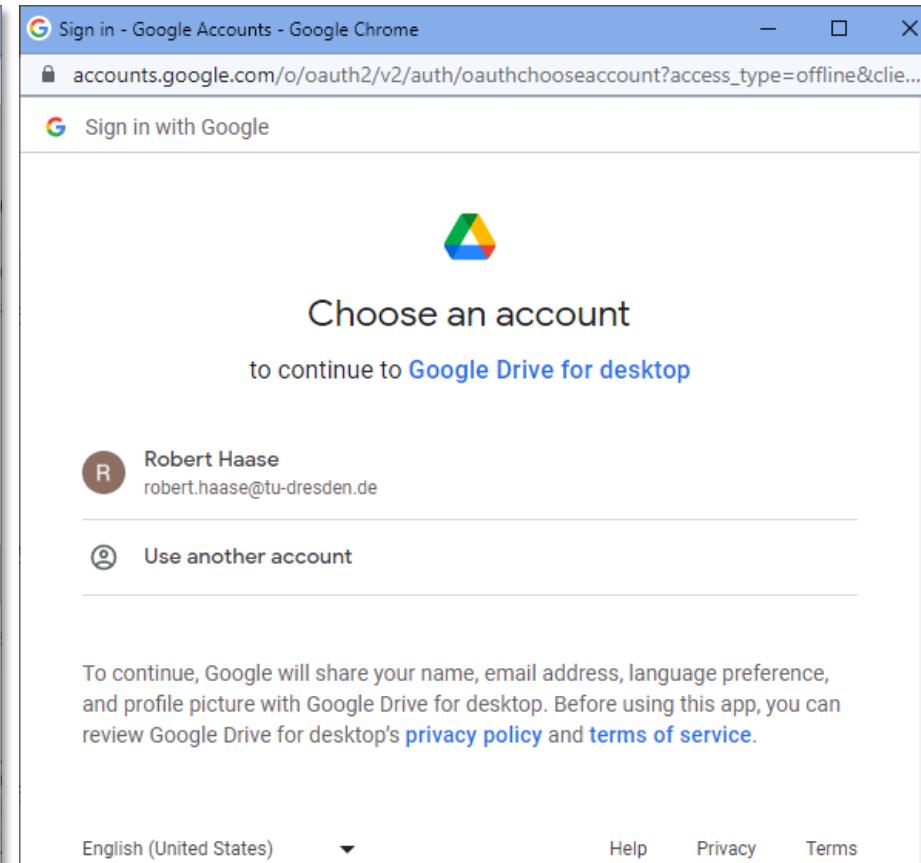
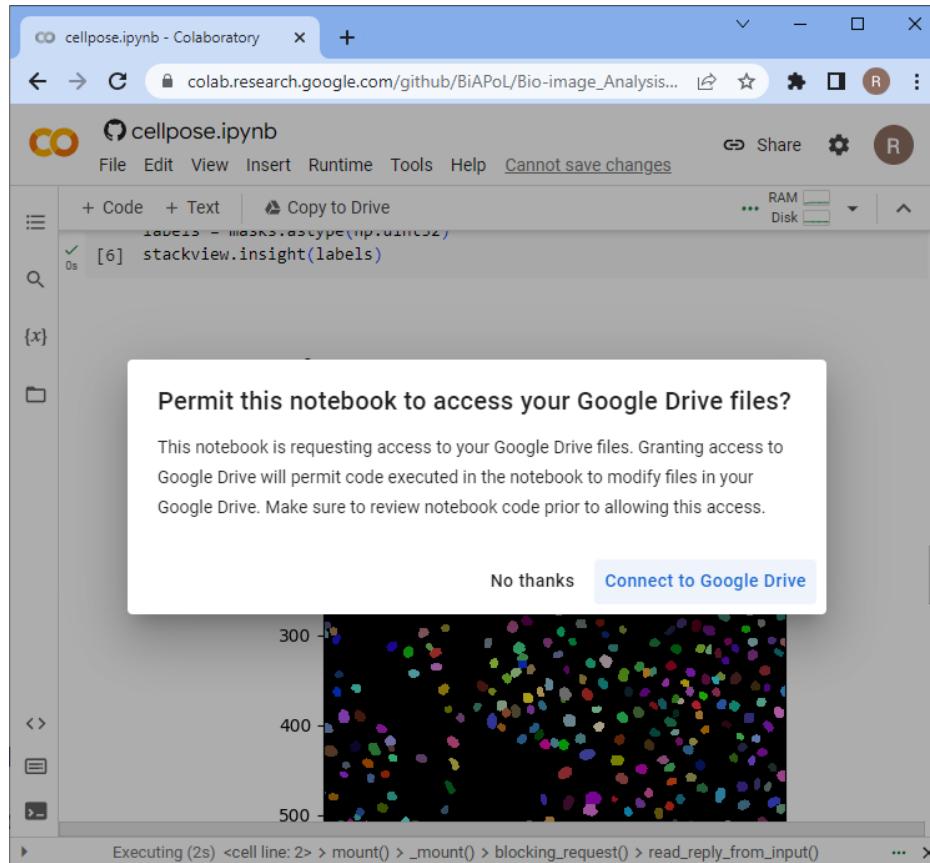
Requirement already satisfied: packaging>=20.0 in /usr/local

```
[1]: https://www.nature.com/articles/s41592-020-01018-x
```

```
[6]: labels = masks.astype(np.uint32)
stackview.insight(labels)
```

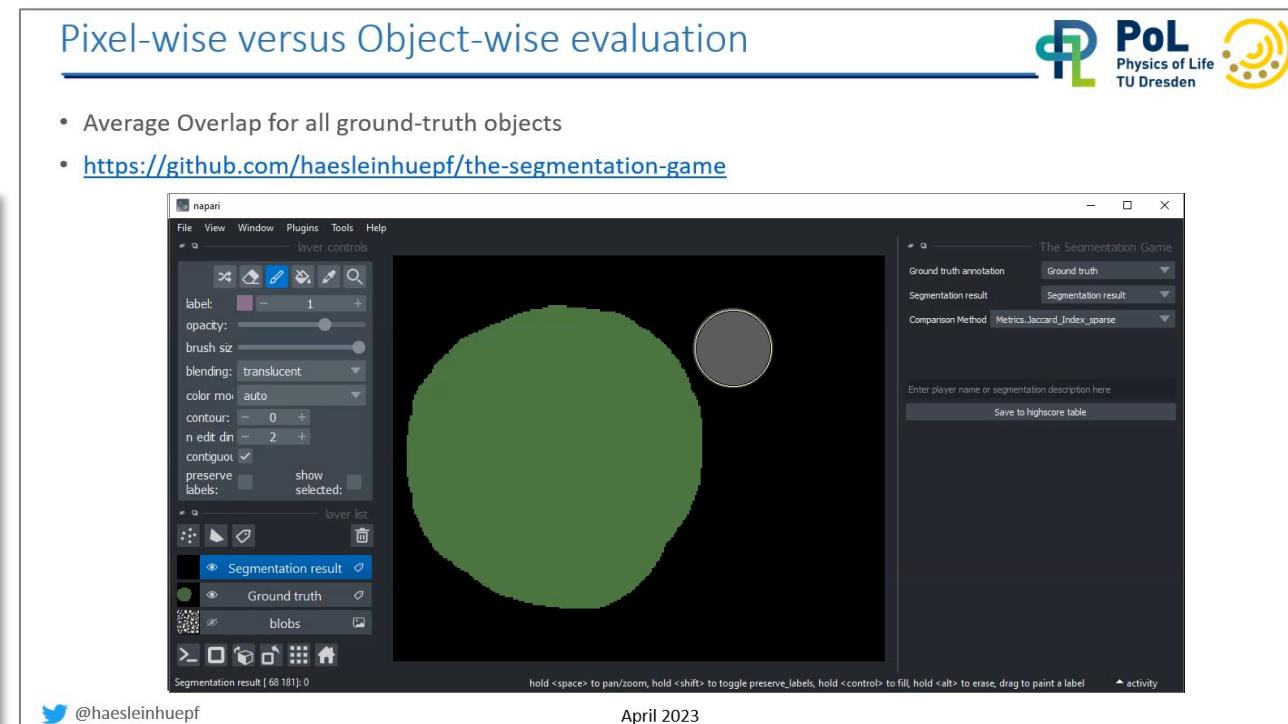
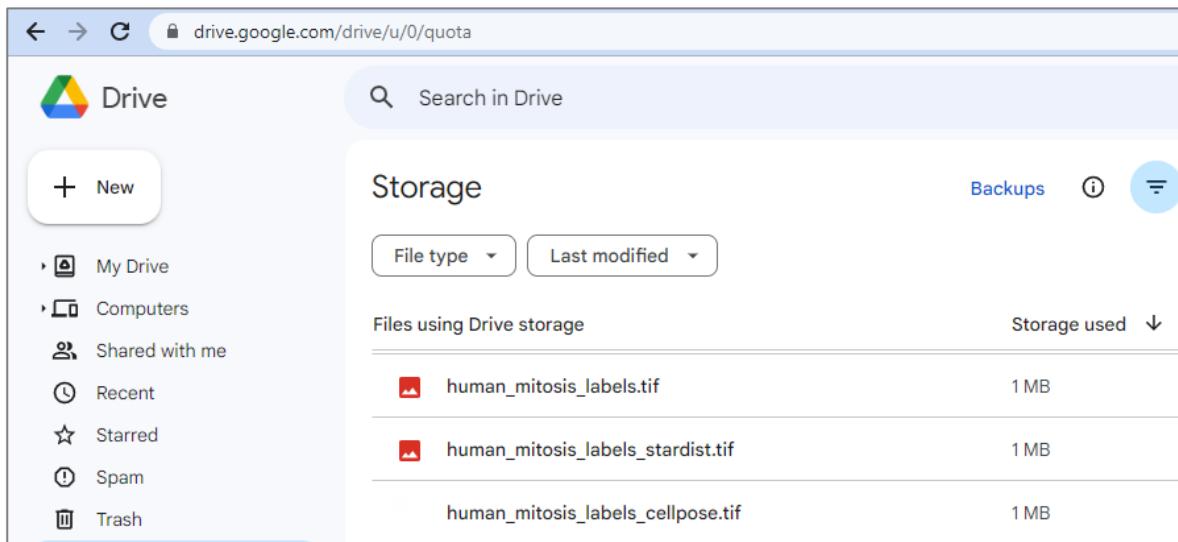
Connecting Google Drive

- Store the resulting label images to your Google Drive



Segmentation quality comparison

- Download the segmentation results of CellPose and StarDist from your Google Drive.
- Measure the quality of both compared to a sparse annotation.
- Which algorithm is better?



<https://drive.google.com/drive/u/0/quota>
<https://github.com/haesleinhuepf/the-segmentation-game>