ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

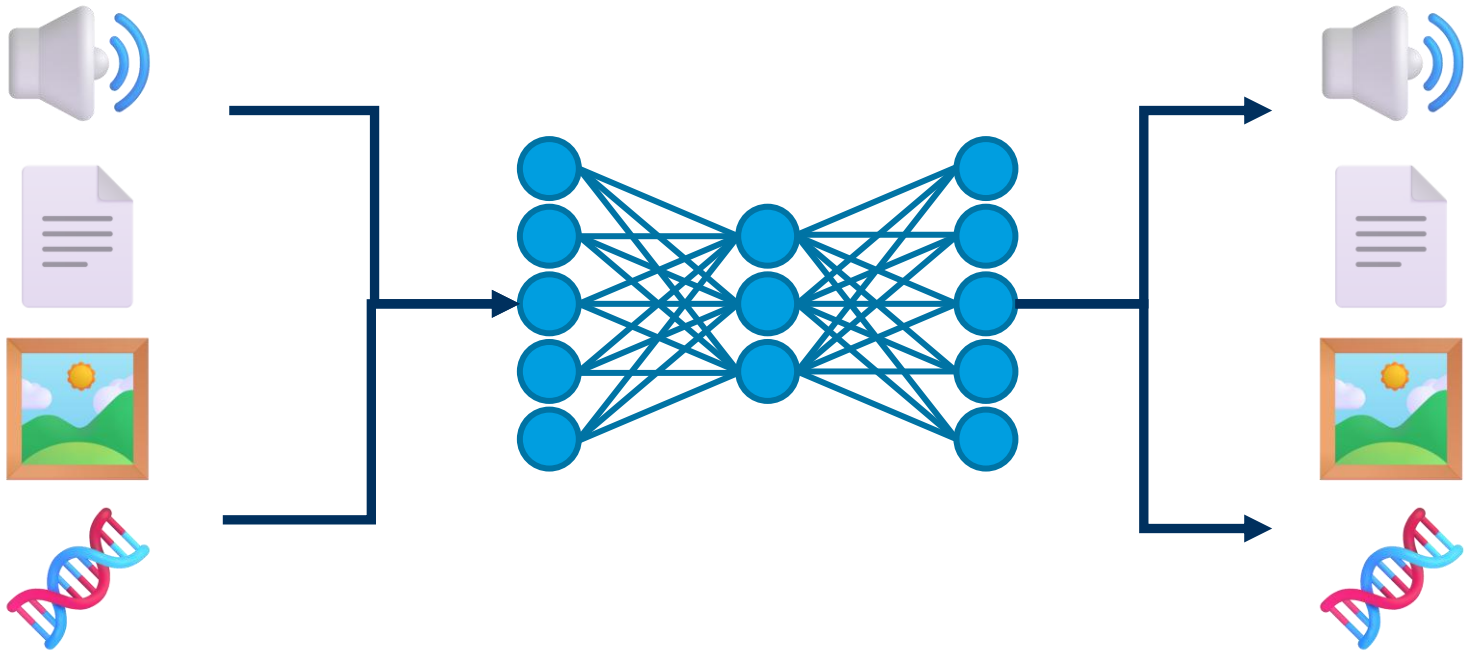
Multi-modal Language Models

[images + text]

Robert Haase

Multi-modal LLMs

Combining image, text and [...] data



Multi-modal LLMs

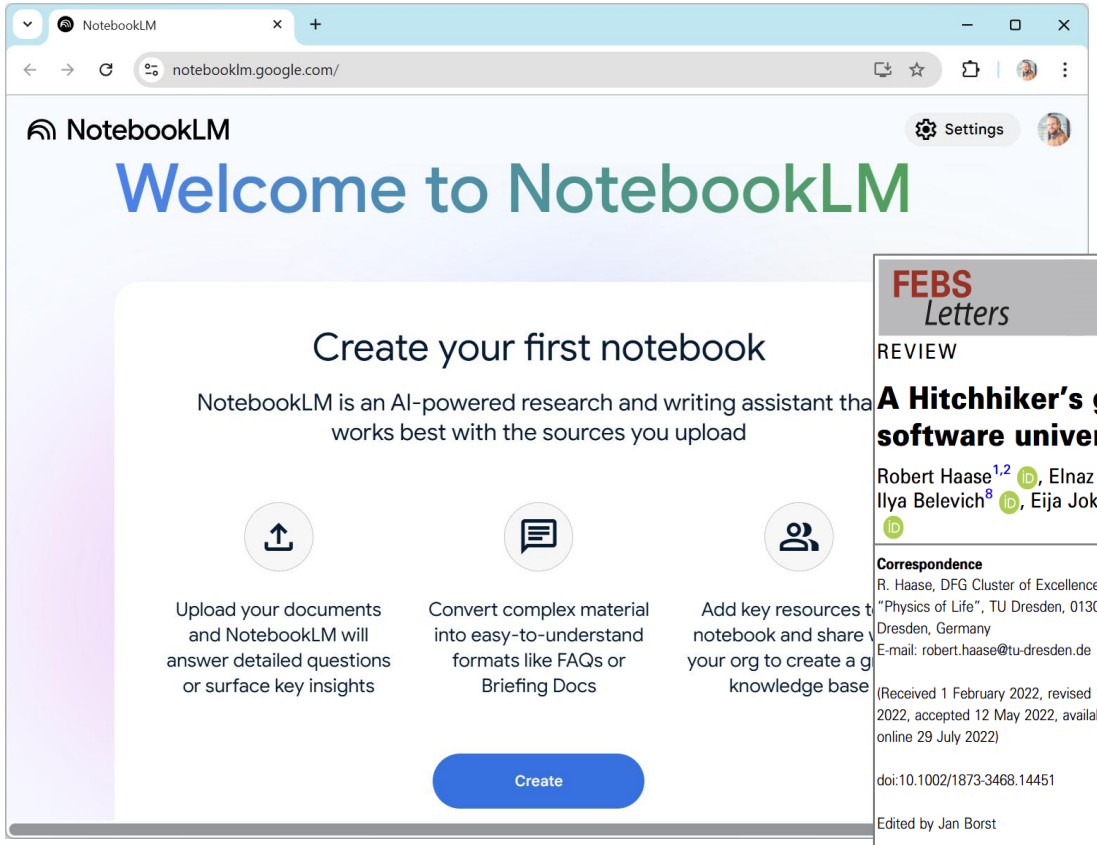
Combining image, text and [...] data, to gain new [biological] insights.

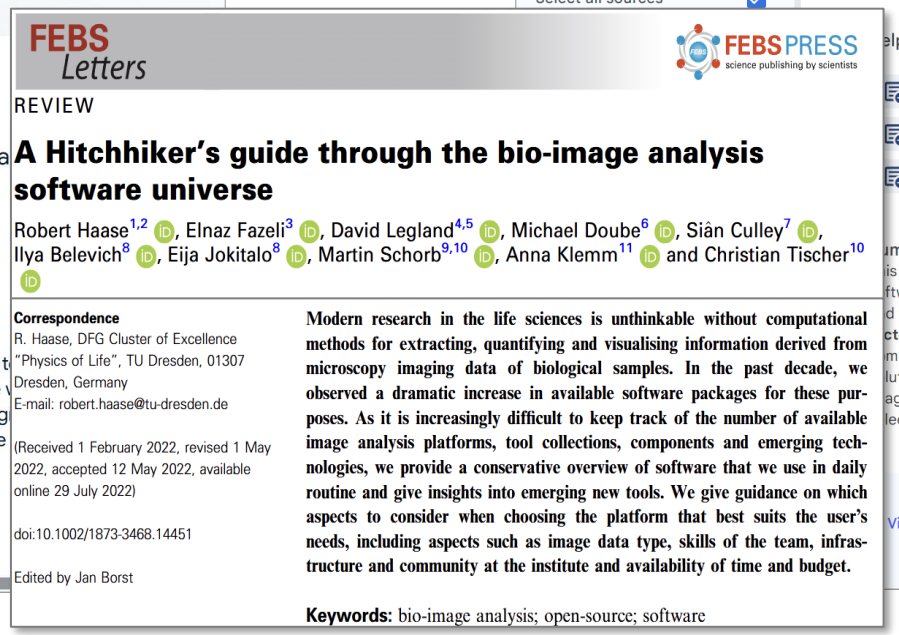
- Chat / translation / text-generation: 📄 -> 📄
- Voice-chat / -translation: 🔊 -> 🔊
- Text-to-speech (TTS): 📄 -> 🔊
- Speech to text (STT): 🔊 -> 📄
- Vision / image classification / image description / image-to-text: 🖼️ -> 📄
- Image generation, text-to-image: 📄 -> 🖼️
- Image variation, inpainting, image segmentation: 🖼️ -> 🖼️
- Genotype-phenotype translation: 🧬 -> 📄
- Sequence-prediction / protein design: 📄 -> 🧬

Focus of today's lecture

Use case: Paper to Podcast

Text to audio conversion: Google NotebookLM ->




FEBS Letters

FEBS PRESS science publishing by scientists

REVIEW

A Hitchhiker's guide through the bio-image analysis software universe

Robert Haase[1,2], Elnaz Fazeli[3], David Legland[4,5], Michael Doube[6], Siân Culley[7], Ilya Belevich[8], Eija Jokitalo[8], Martin Schorb[9,10], Anna Klemm[11] and Christian Tischer[10]

Correspondence
R. Haase, DFG Cluster of Excellence "Physics of Life", TU Dresden, 01307 Dresden, Germany
E-mail: robert.haase@tu-dresden.de

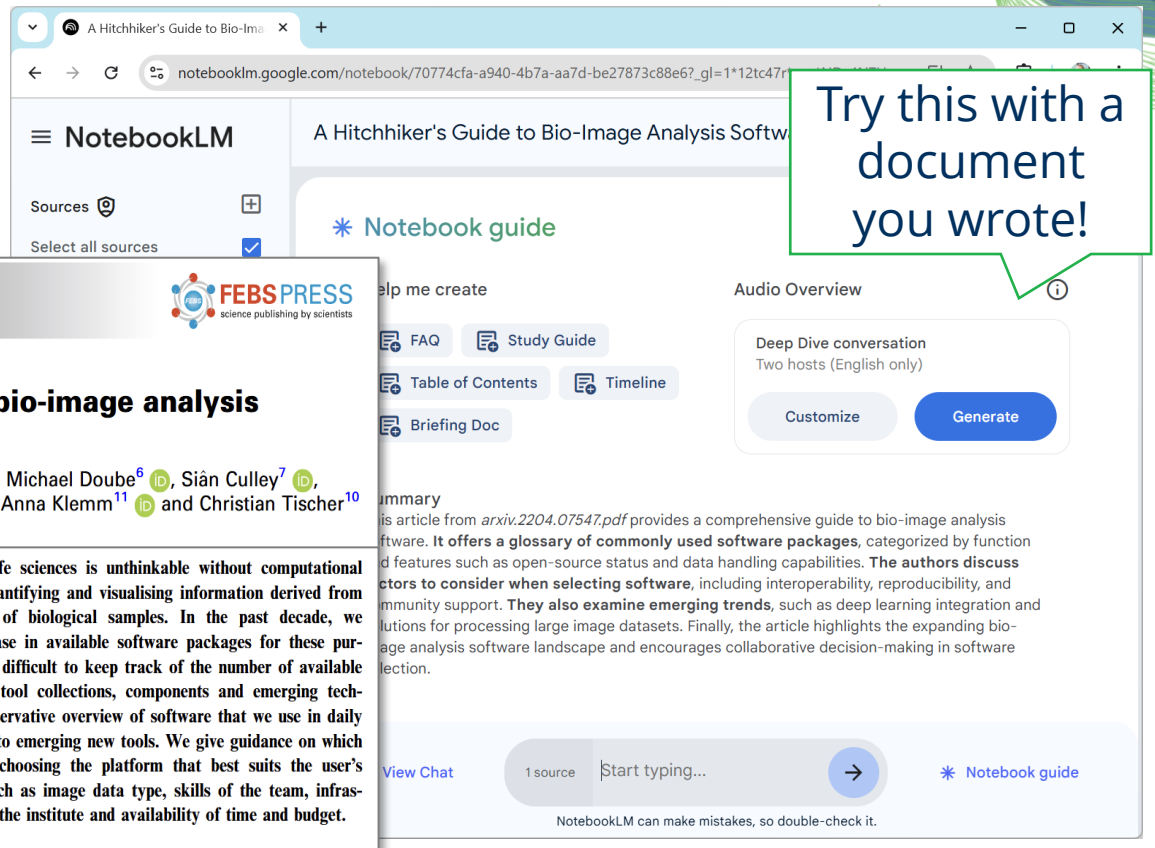
(Received 1 February 2022, revised 1 May 2022, accepted 12 May 2022, available online 29 July 2022)

doi:10.1002/1873-3468.14451

Edited by Jan Borst

Modern research in the life sciences is unthinkable without computational methods for extracting, quantifying and visualising information derived from microscopy imaging data of biological samples. In the past decade, we observed a dramatic increase in available software packages for these purposes. As it is increasingly difficult to keep track of the number of available image analysis platforms, tool collections, components and emerging technologies, we provide a conservative overview of software that we use in daily routine and give insights into emerging new tools. We give guidance on which aspects to consider when choosing the platform that best suits the user's needs, including aspects such as image data type, skills of the team, infrastructure and community at the institute and availability of time and budget.

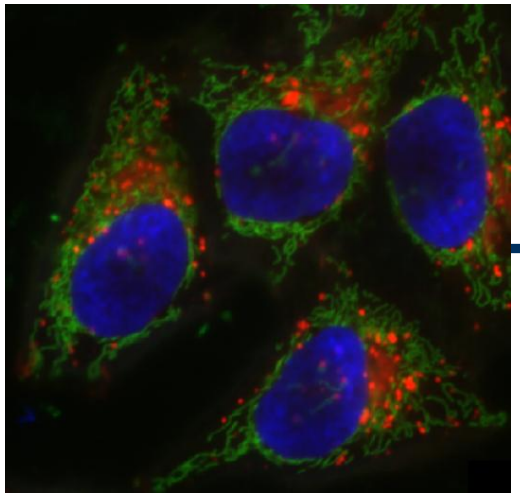
Keywords: bio-image analysis; open-source; software



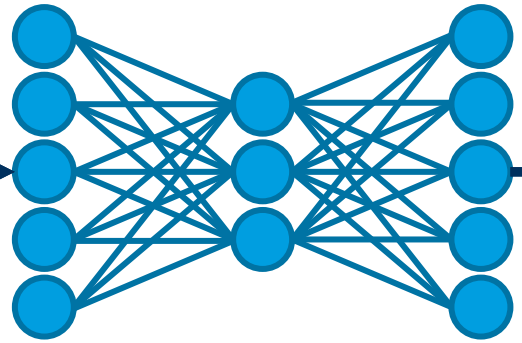
Try this with a document you wrote!

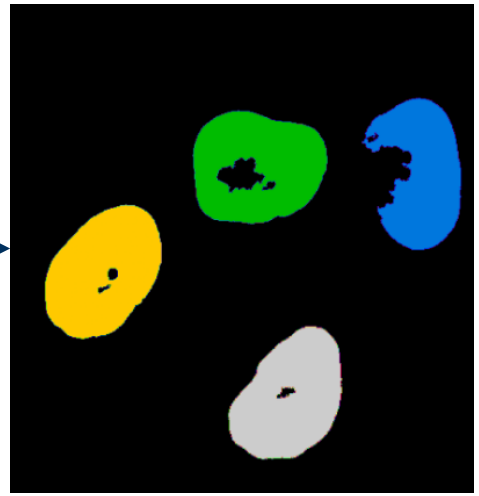
Multi-modal LLMs

Combining image, text and [...] data, to gain new [biological] insights.



How many cells are there?

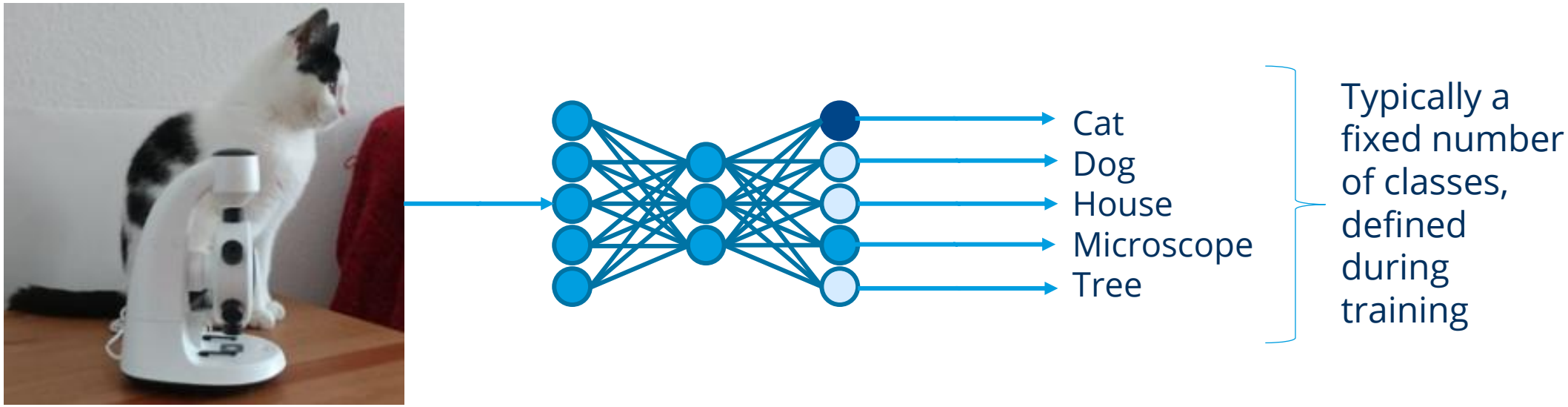




There are 4 cells. I just marked their nuclei.

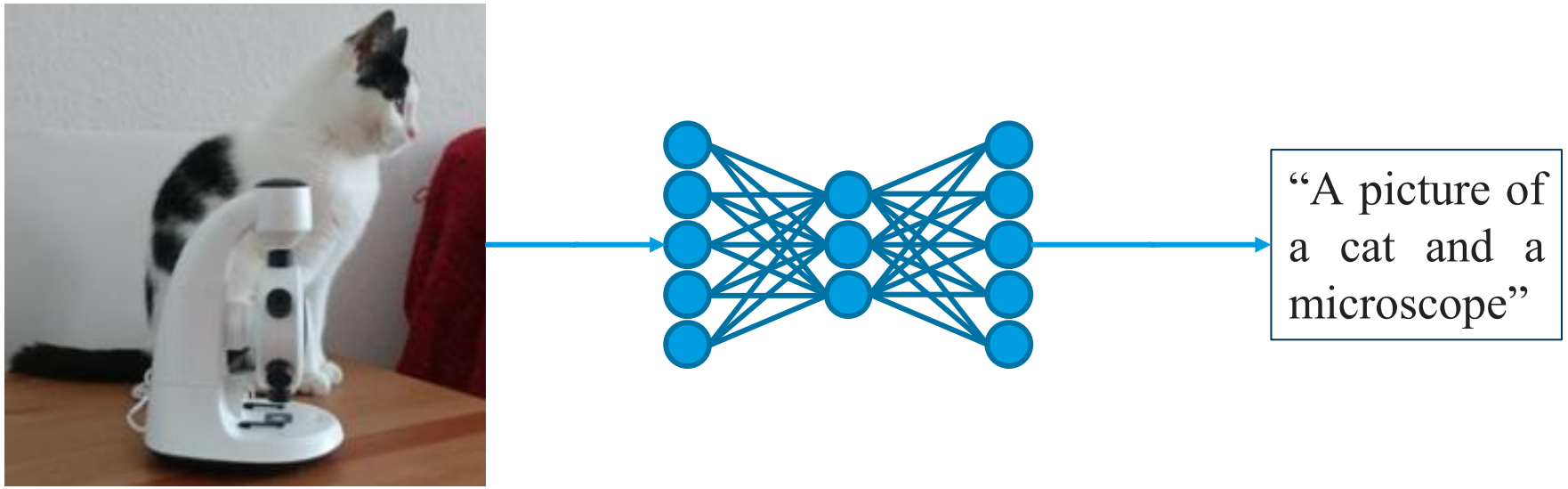
Vision Models

- Classifying images (decades old research field 🥱)

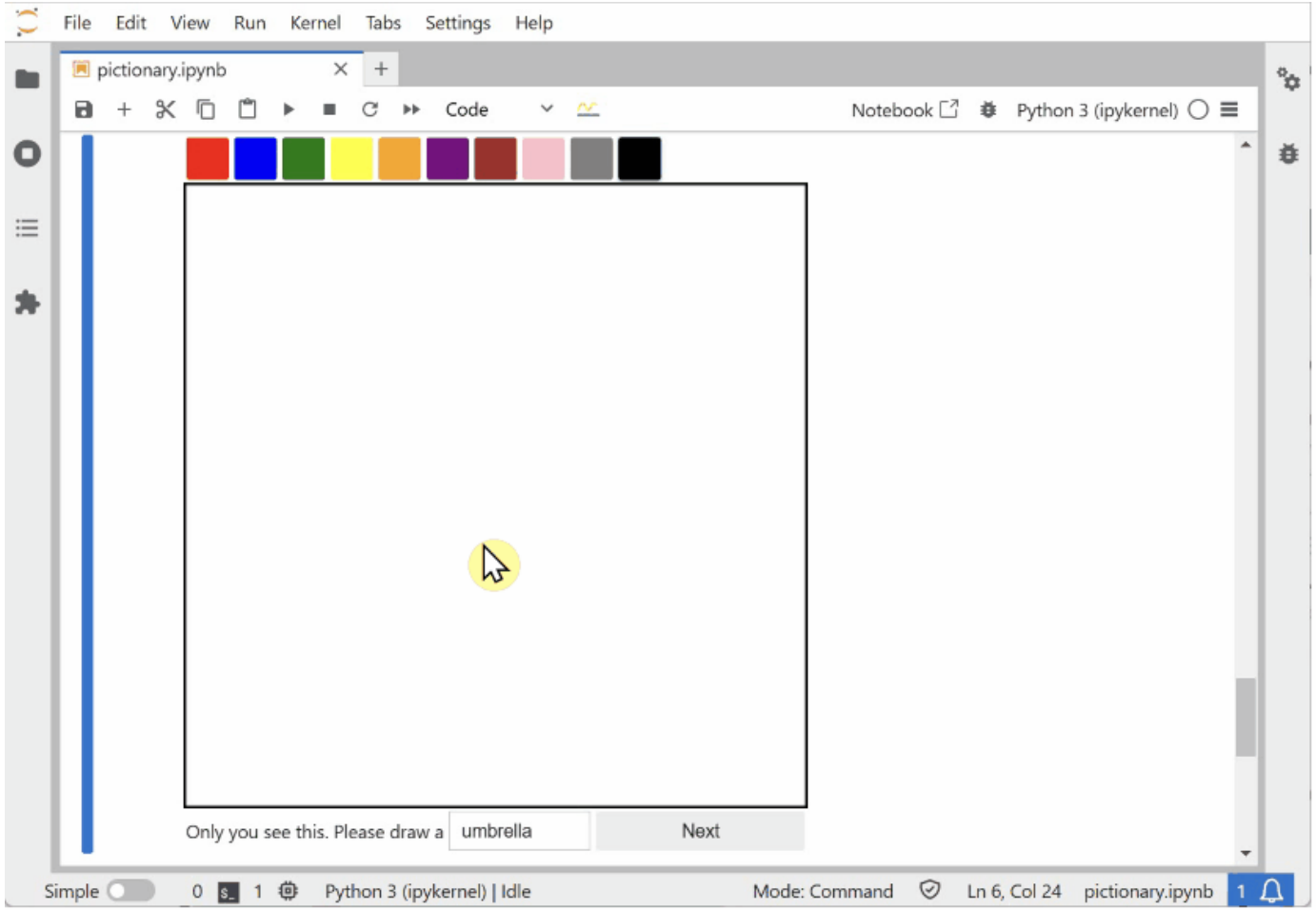


Vision Language Models

- Classifying images
- Describing images



Gamification: VLM-Pictionary

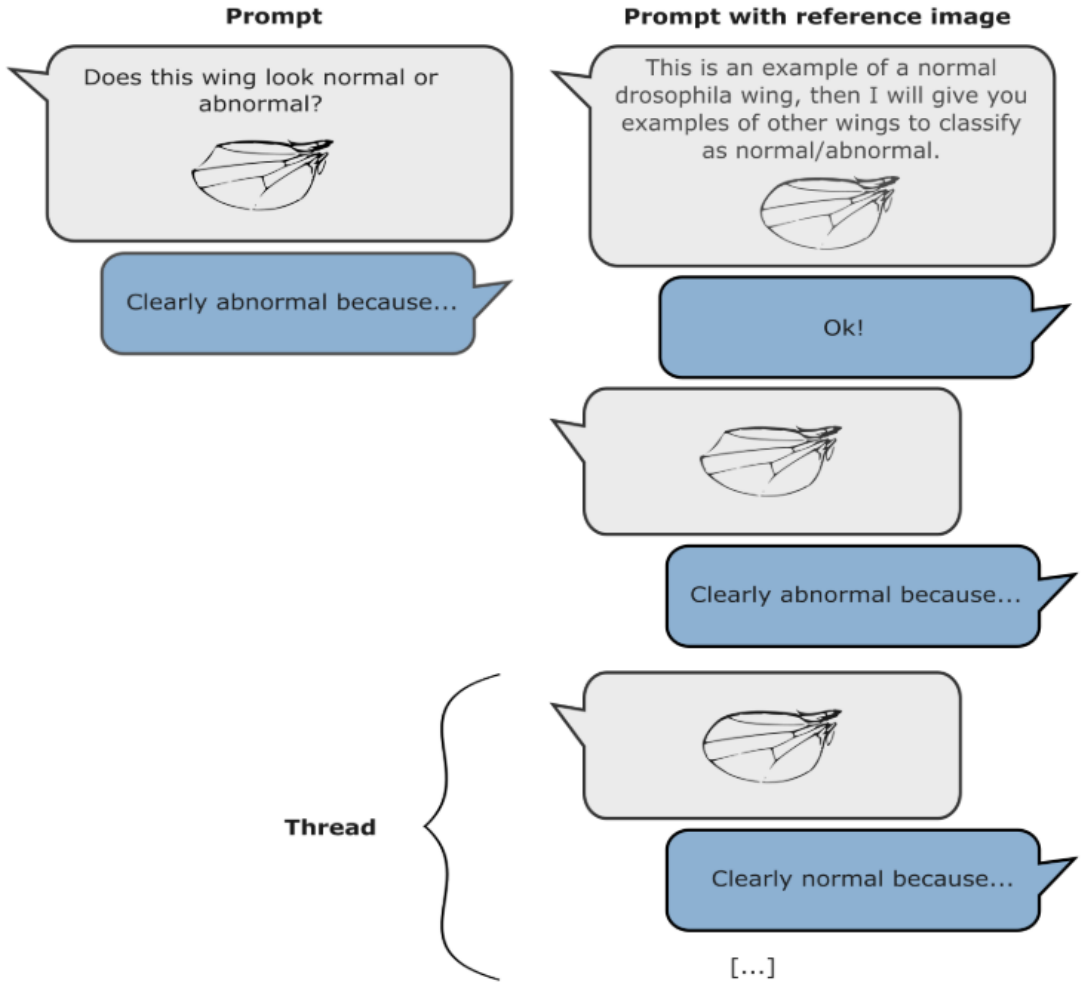


Use case: Descriptive biology

Table 1. Dataset: phenotypes analyzed and example images.

Phenotype	# Imgs	Example
Loss of wing veins (V-)	10	
Ectopic wing veins (V+)	20	
Integrity of wing margin (WM)	23	
Wing surface adhesion (WA)	27	

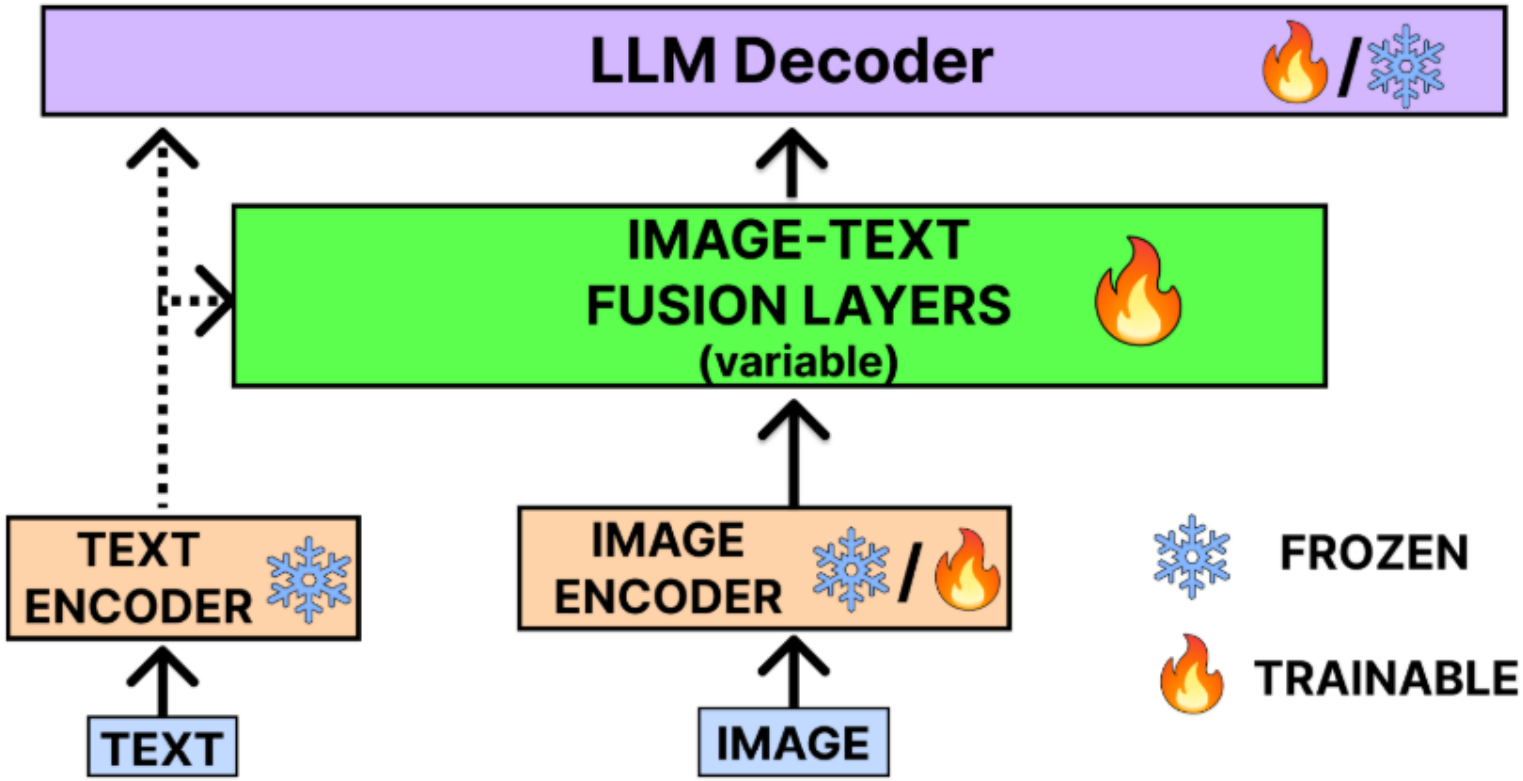
"… while visual language models are in their infancy, they already show potential for multiple applications in automated phenotyping studies. We encourage the community to carefully test them… "



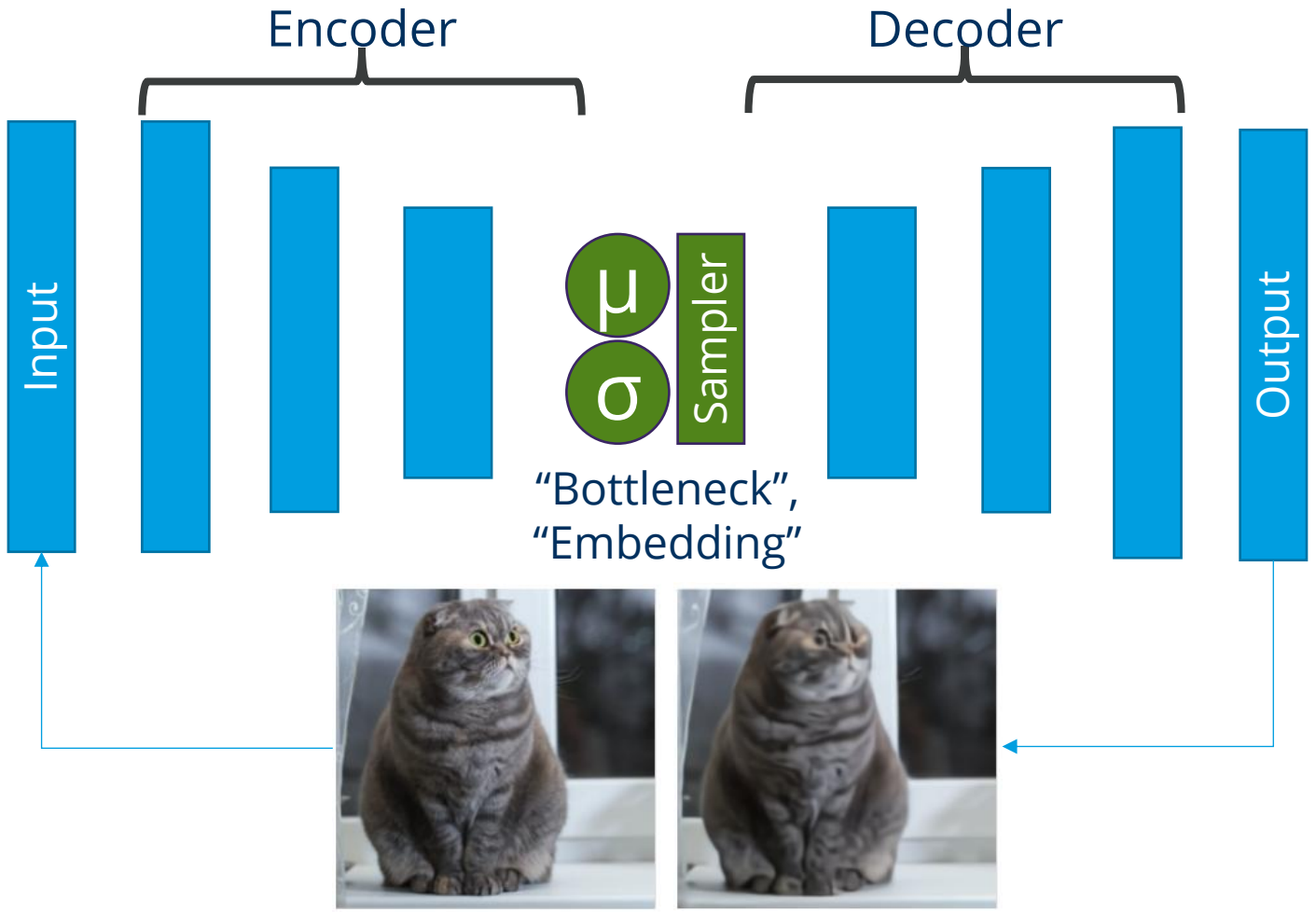
Cropped from Paci & Nanni (2024), licensed CC-BY 4.0 https://www.biorxiv.org/content/10.1101/2024.05.27.594652v1

Vision Language Models (VLMs)

Goal: Describe images

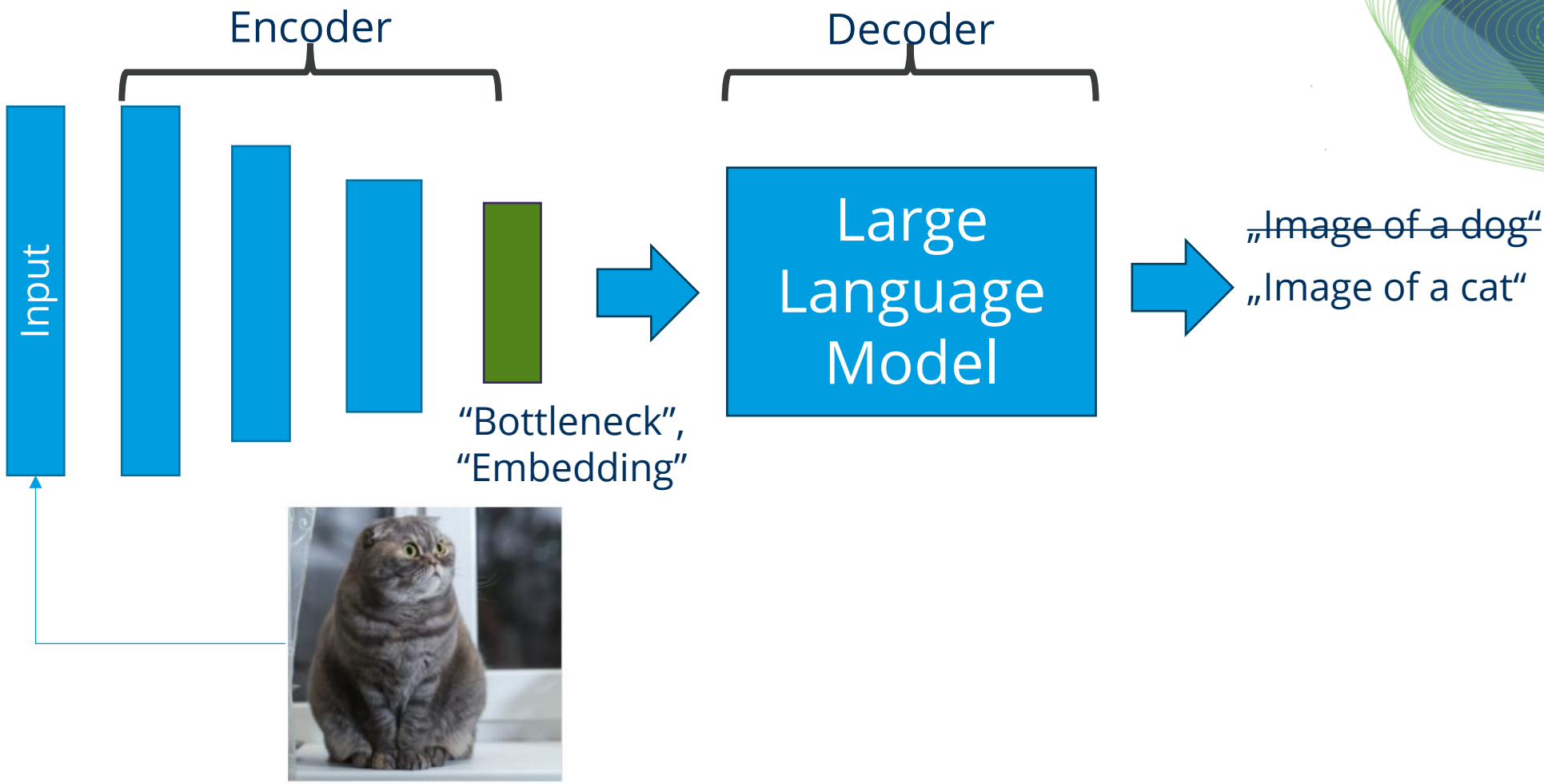


Variational Auto-Encoder



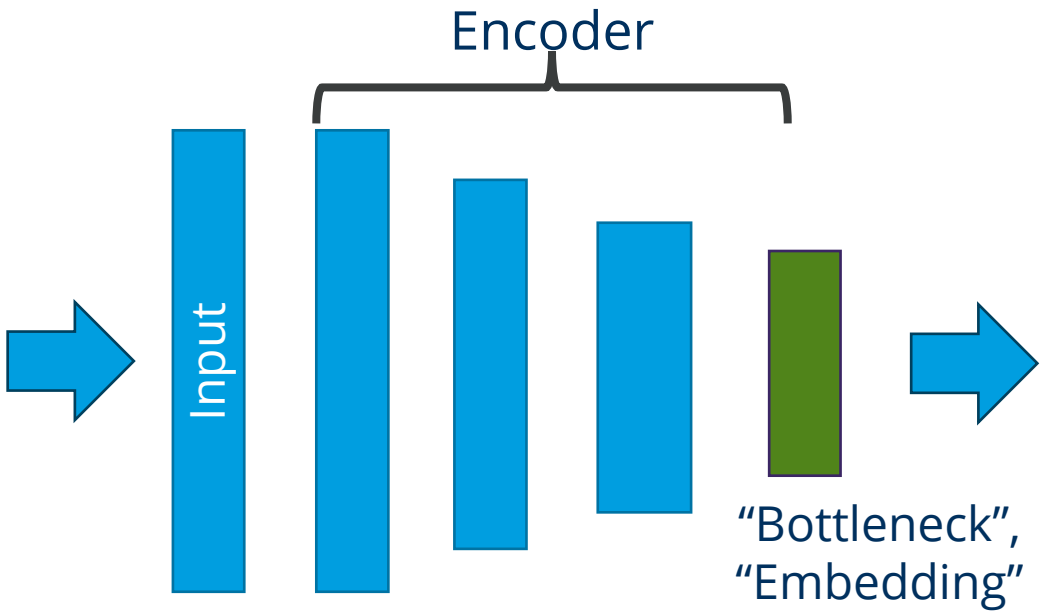
Cat picture source: Ramesh et al. 2021, licensed CC-BY 4.0 https://arxiv.org/pdf/2102.12092.pdf

Image classification -> image describing



Cat picture source: Ramesh et al. 2021, licensed CC-BY 4.0 https://arxiv.org/pdf/2102.12092.pdf

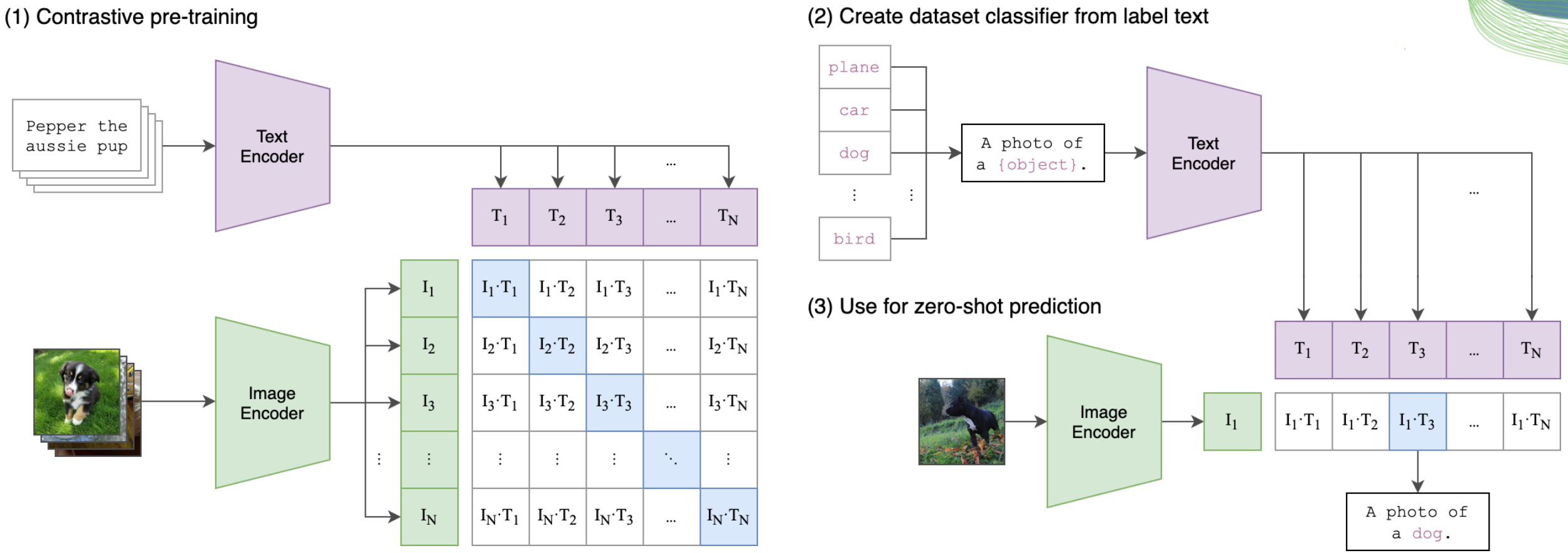
Image classification -> image describing



VLMs can only know terms which were in the training data.

Contrastive Language-Image Pre-Training

„CLIP“ Transformers



Source: Radford et al 2021, License MT
https://arxiv.org/abs/2103.00020
https://github.com/OpenAI/CLIP

CLIP transformers in Python

Using huggingface 🤗

Downloads 500 MB

```
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
```

```
options = ["a photo of a cat",
           "a photo of a dog"]
```

```
options = ["a photo of a cat",
           "a photo of a dog",
           "a photo of a microscope"]
```

```
inputs = processor(text=options, images=image, return_tensors="pt", padding=True)
outputs = model(**inputs)
```

...

```
label_probabilities
```

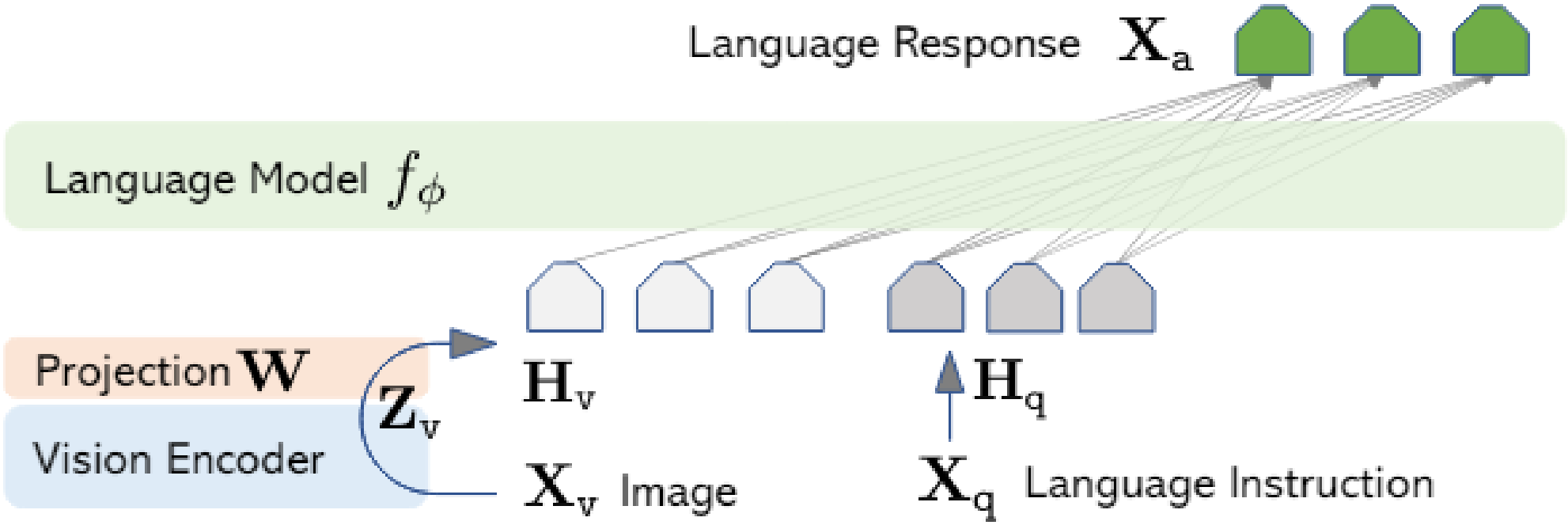
```
{'a photo of a cat': 0.9907298684120178,
 'a photo of a dog': 0.009270114824175835}
```

```
label_probabilities
```

```
{'a photo of a cat': 0.1352911740541458,
 'a photo of a dog': 0.0012659047497436404,
 'a photo of a microscope': 0.8634429574012756}
```

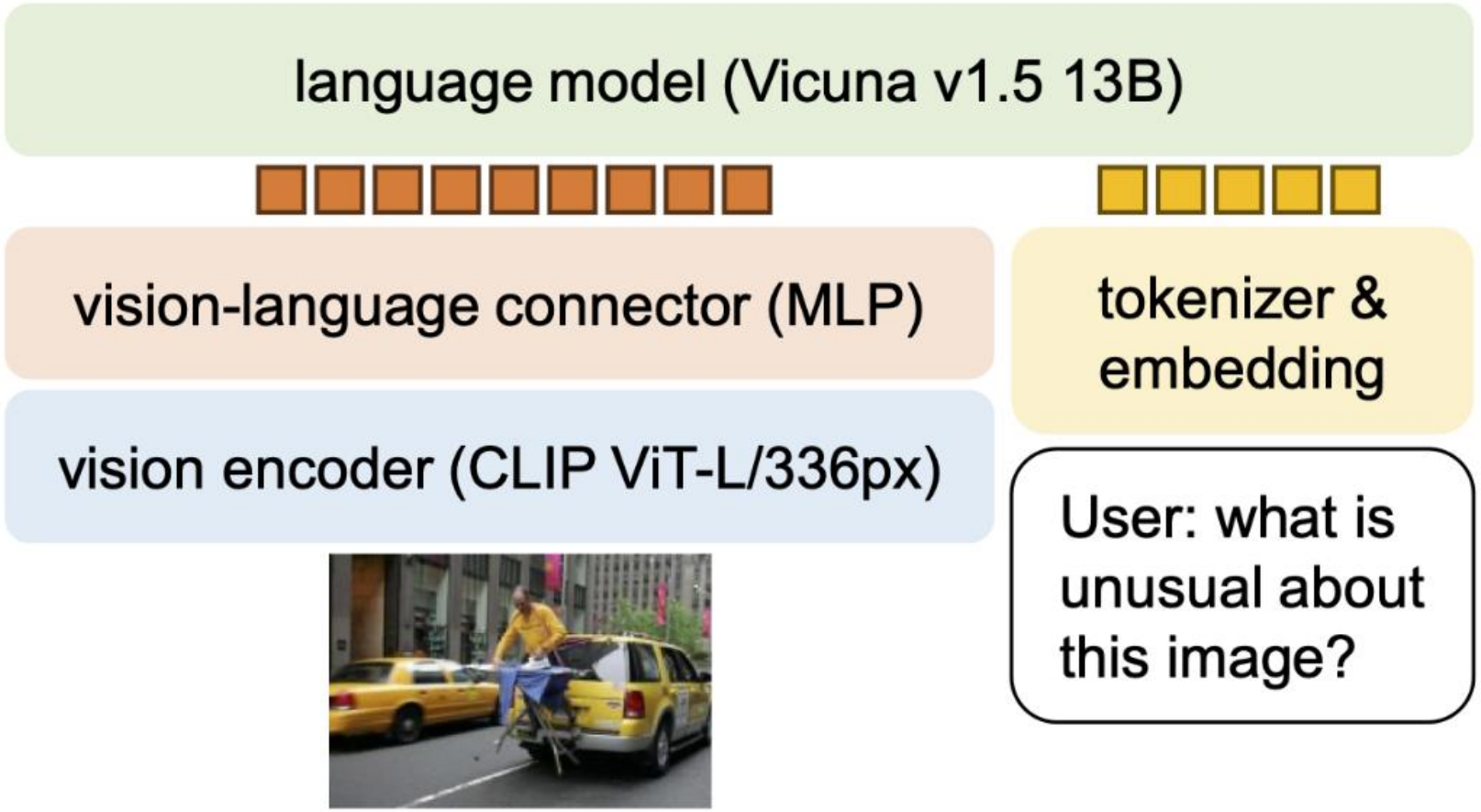
LLAVA

Large Language and Vision Assistant



LLAVA 1.5

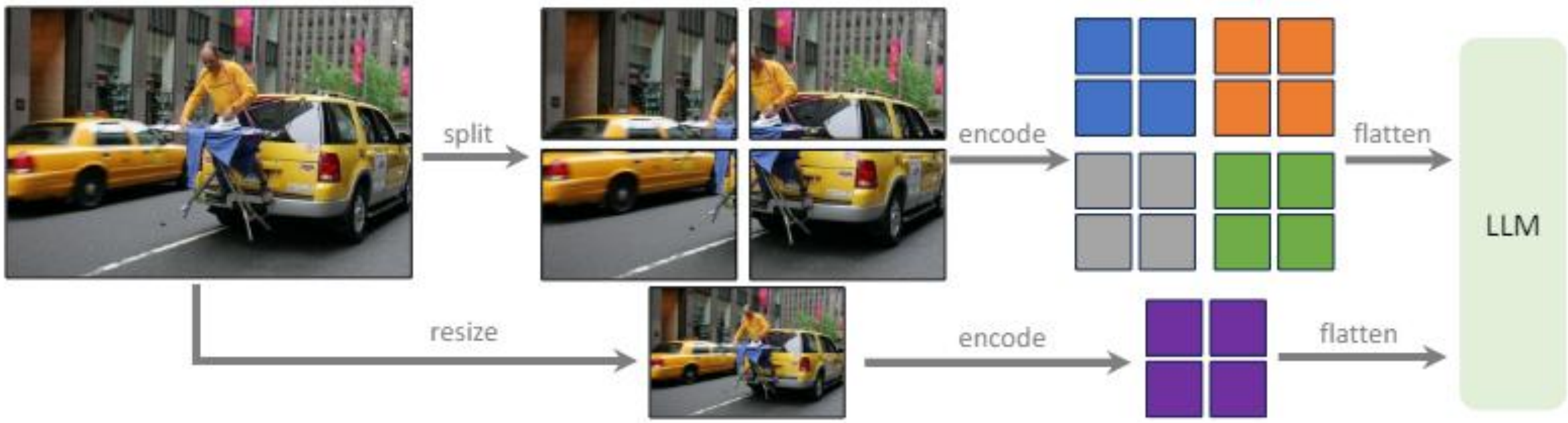
Combining LLAVA with CLIP



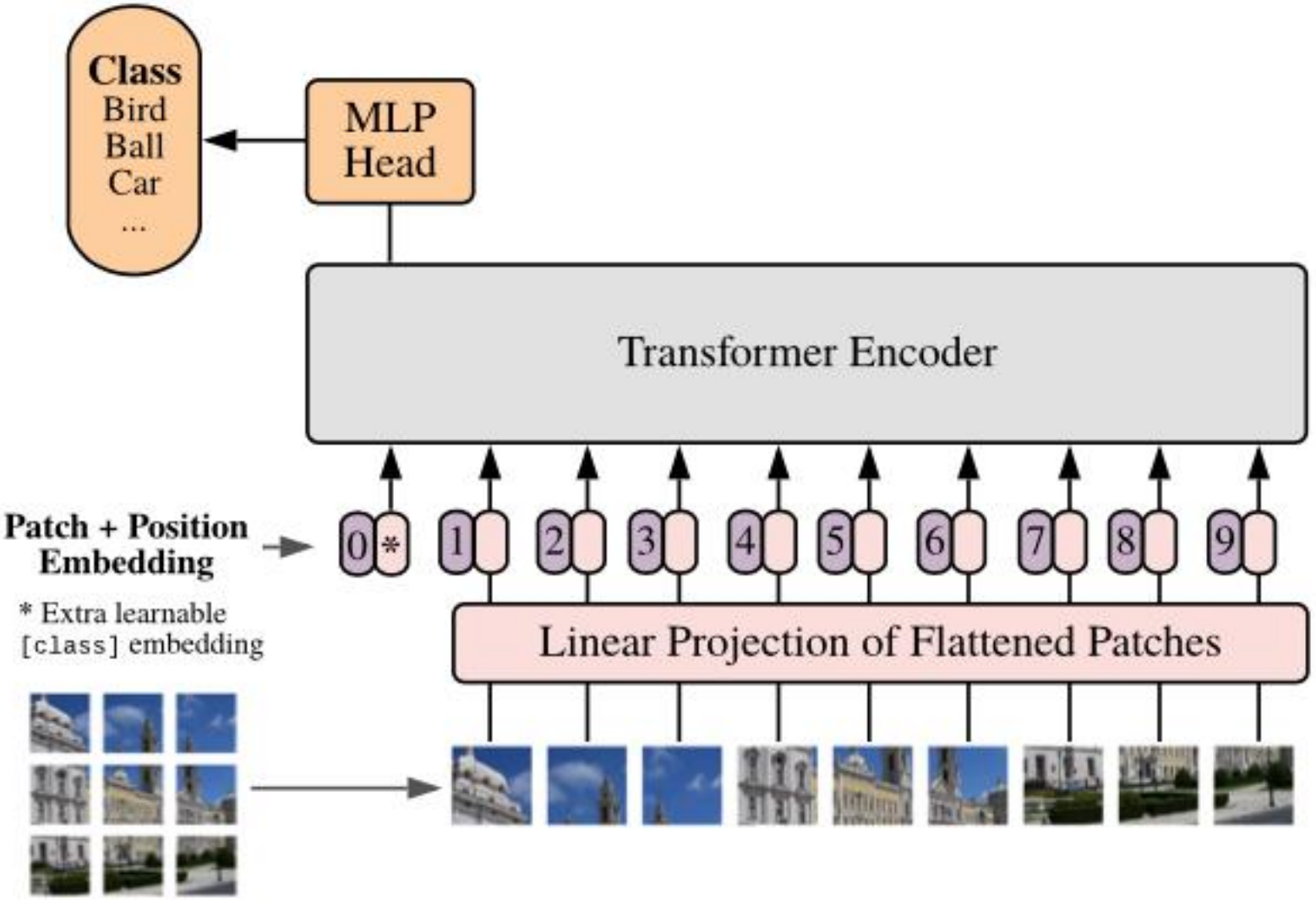
Source: Hui et al 2024, licensed CC-BY 4.0
https://arxiv.org/abs/2310.03744

LLAVA 1.5 HD

Giving the model multiple perspectives on the same scene



Vision-Transformer (ViT)

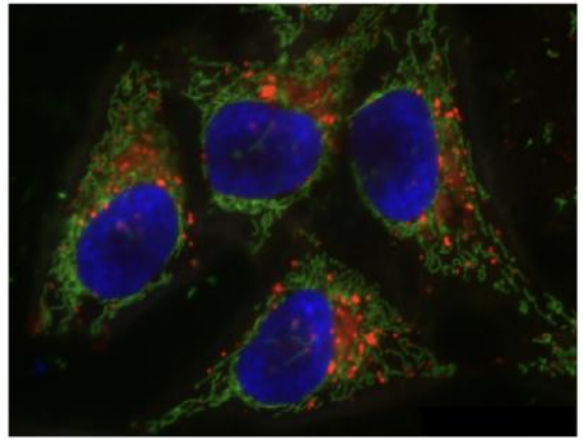


Source: Cropped from Dosovitskiy et al 2021.
https://github.com/google-research/vision_transformer, license Apache 2
See also: https://arxiv.org/abs/2010.11929

Accessing VLMs using Python

E.g. using ScaDS.AI's openai-compatible LLM Server

```python
def prompt_qwen(prompt:str, image, model="Qwen/Qwen2-VL-7B-Instruct"):
    """A prompt helper function that sends a message to the server
    and returns only the text response.
    """
    rgb_image = _img_to_rgb(image)
    byte_stream = numpy_to_bytestream(rgb_image)
    base64_image = base64.b64encode(byte_stream).decode('utf-8')

    message = [{"role": "user", "content": [
        {"type": "text", "text": prompt},
        {
        "type": "image_url",
        "image_url": {
            "url": f"{base64_image}"
        }
    }]}]

    # setup connection to the LLM
    client = openai.OpenAI(base_url="https://llm.scads.ai/v1",
                           api_key=os.environ.get('SCADSAI_API_KEY'))

    # submit prompt
    response = client.chat.completions.create(
        model=model,
        messages=message
    )

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



```python
res = prompt_qwen("what's in this image?", hela_cells)

display(Markdown(res))
```

The image appears to be a microscopic view of biological cells, possibly analyzed using fluorescence microscopy techniques. The cells are stained to highlight different structures:

1. **Blue Areas**: These represent the nuclei of the cells, typically stained with a dye like DAPI or Hoechst that binds to DNA.
2. **Green Structures**: These are likely the plasma membranes of the cells, as visualized by a fluorescence stain like FM1-43, which binds to lipid-bilayer membranes给人。
3. **Red Dots**: These are clusters or puncta that might represent specific signaling molecules or proteins, possibly related to cytokines or receptors participating in cell signaling pathways

Accessing VLMs using Python: No common API

```python
if image is None:
    message = [{"role": "user", "content": message}]
else:
    encoded_image = image_to_url(image)
    message = [{
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": message
            },
            {
                "type": "image",
                "source": {
                    "type": "base64",
                    "media_type": "image/png",
                    "data": encoded_image
                }
            }
        ]
    }]
```

Anthropic / claude

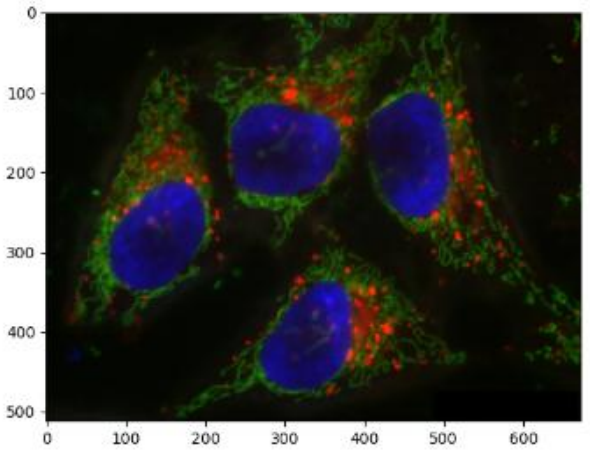
```python
if image is not None:
    response = client.generate_content([image, request])
else:
    response = client.generate_content(request)
```

Google / Gemini

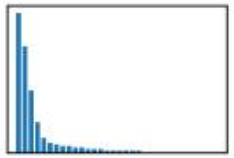
Generating notebooks using vision models

Images have to be turned into text before further processing

```
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

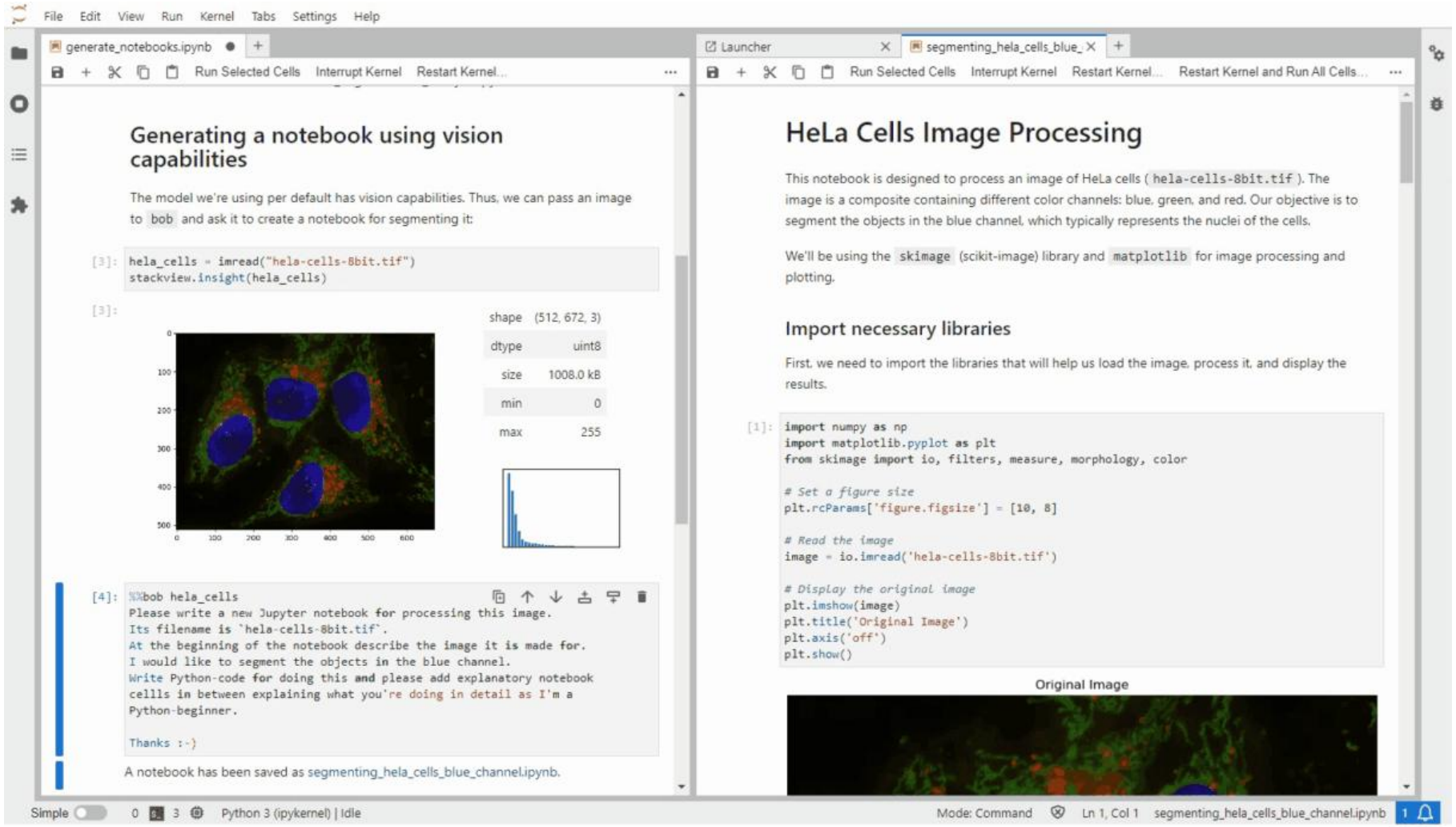


Present Bob an image like this

```
%%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
cellls 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.

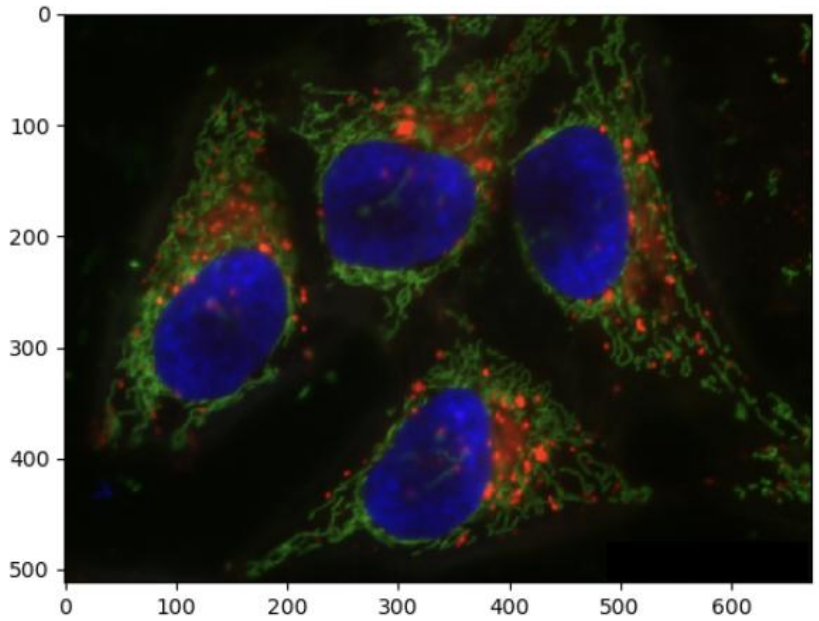
Generating notebooks using vision models



Structured output

Ask VLMs to describe an image and pre-define a structure

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



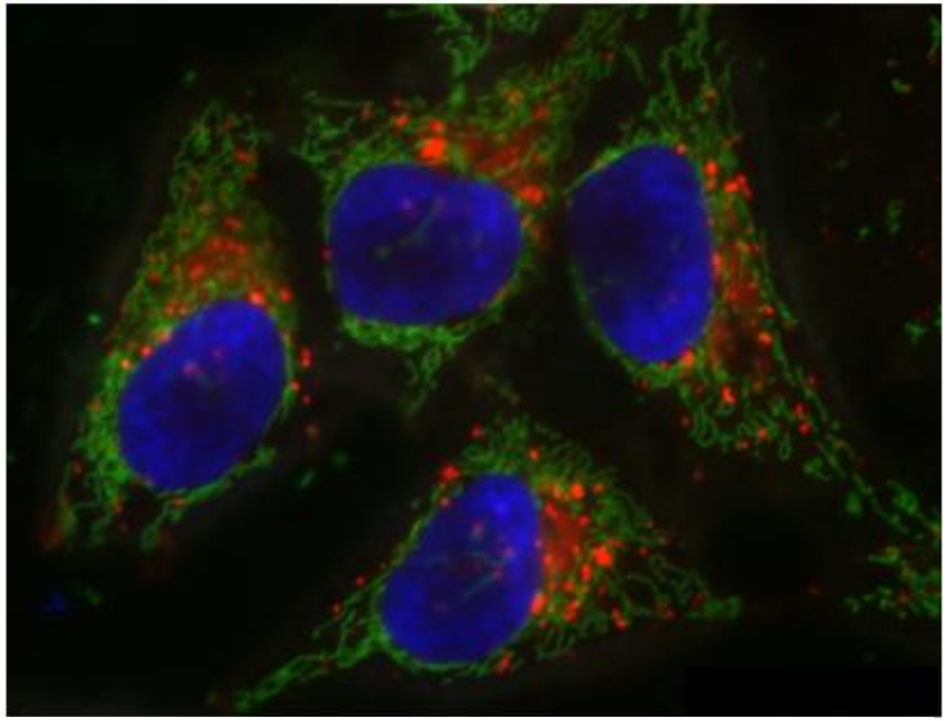

```
result = prompt_chatGPT("""You are a highly experienced biologist with advanced microscopy skills.

# Task
Name the content of this image. Answer for each channel independently.

# Options
The following structures could be in the image:
* Nulcei
* Membranes
* Cytoplasm
* Cytoskeleton
* Extra-cellular structure
* Other sub-cellular structures

# Output format
* Red channel: <structure>
* Green channel: <structure>
* Blue channel: <structure>

Keep your answer as short as possible.
Only respond with the structres for the three channels in the format shown above.
""", hela)

Markdown(result)
```

- Red channel: Other sub-cellular structures
- Green channel: Cytoskeleton
- Blue channel: Nuclei

Benchmarking vision models

Single attempts... are a trap

You



How many blue nuclei are in this image?

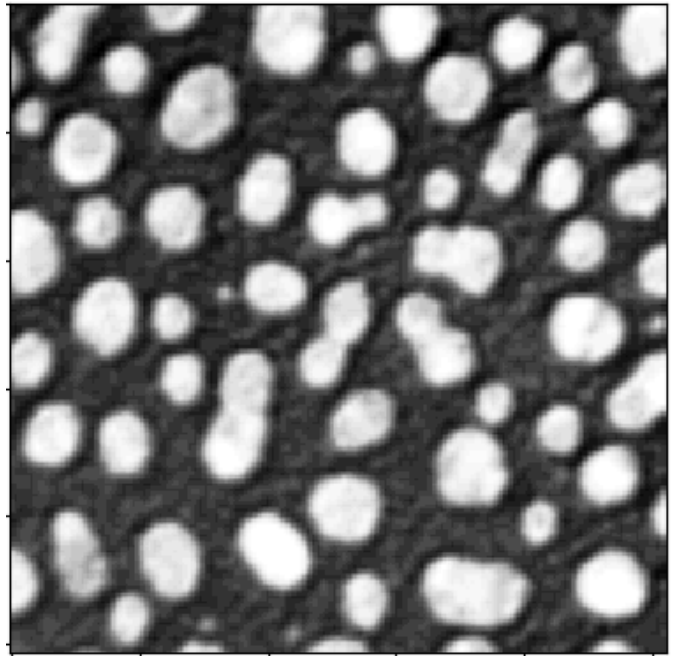
ChatGPT

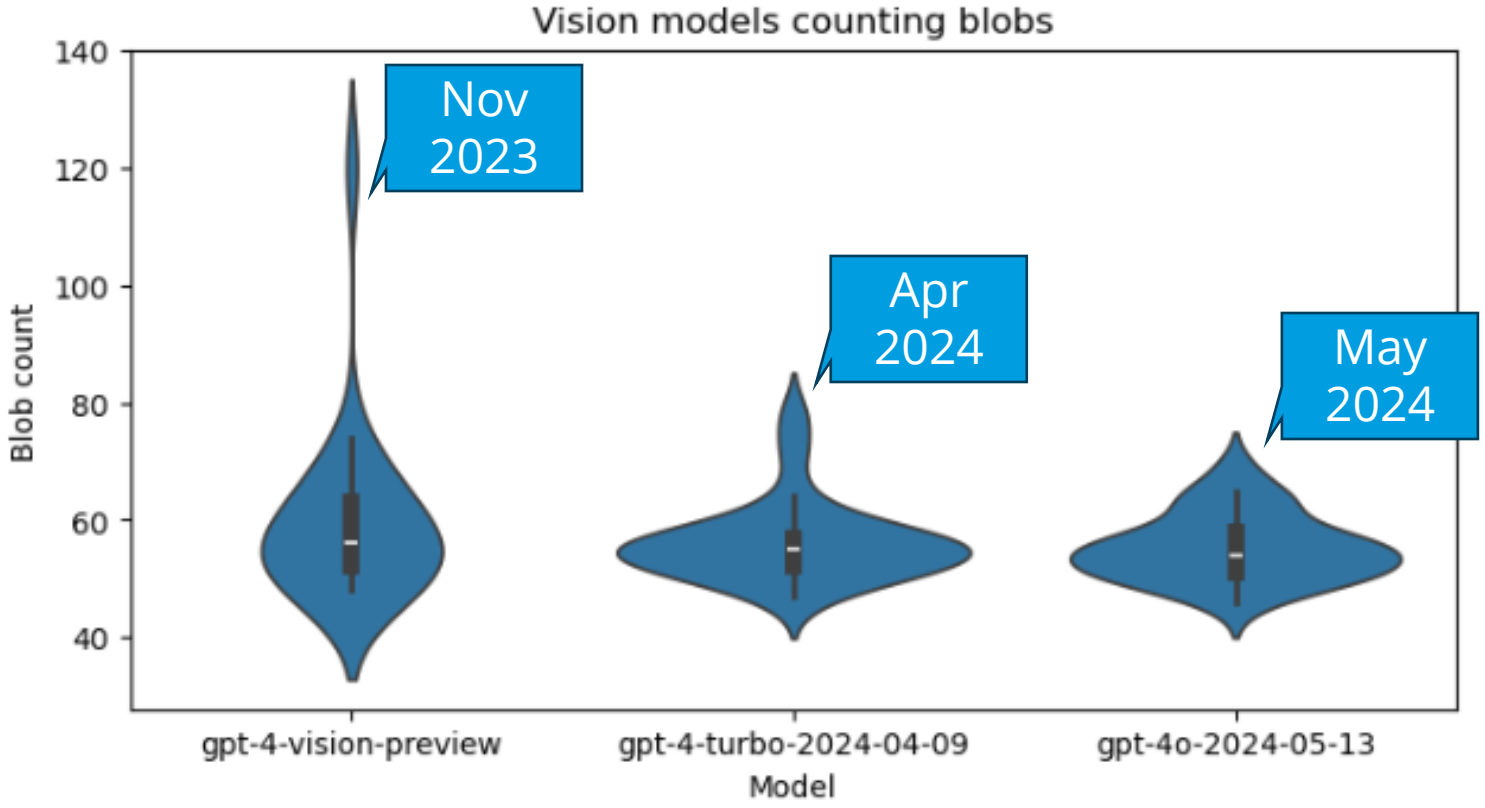
There are three blue nuclei visible in this image.

n=1

Benchmarking vision models

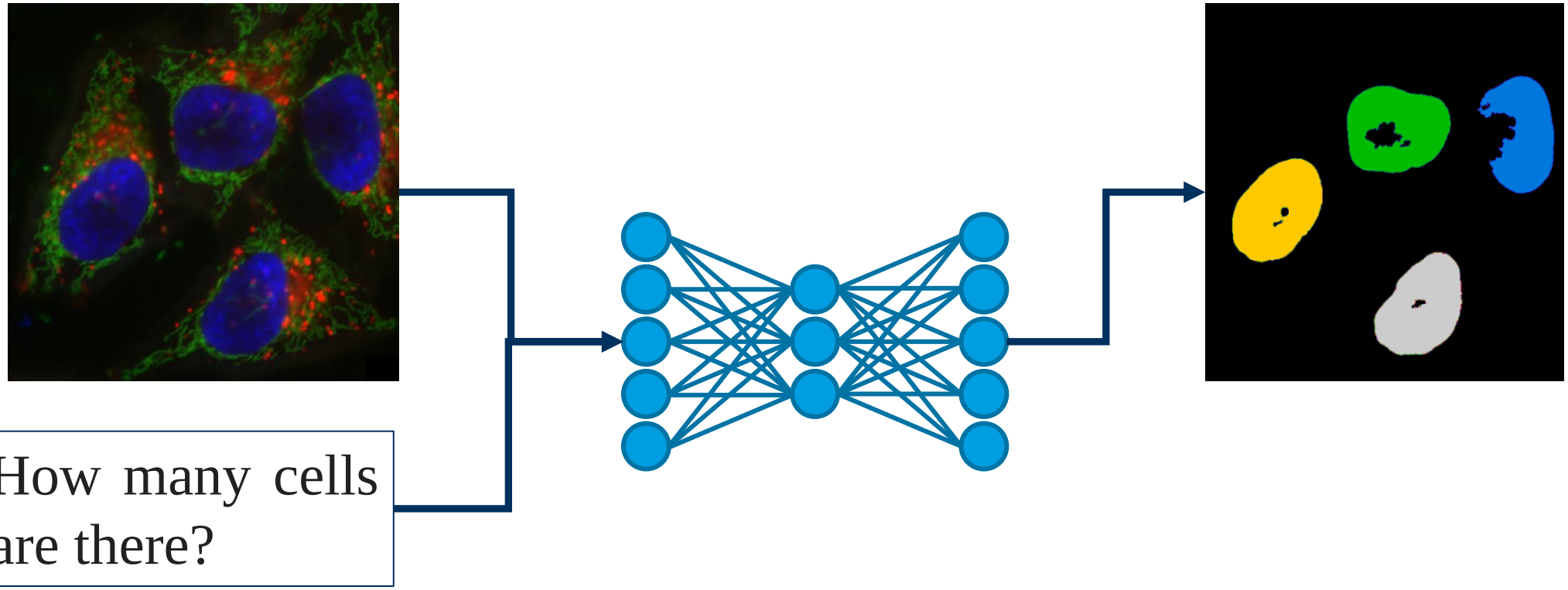
Prompt: „Analyse the following image by counting the bright blobs. Respond with the number only. “ (n=25)





Segment Anything Model

New approach to DL-based image segmentation involving prompts



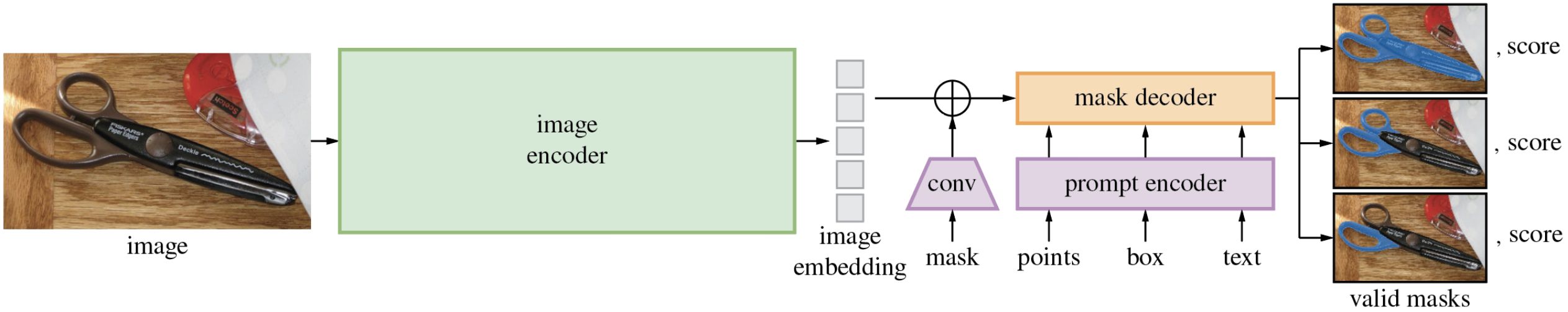
Segment Anything Model

New approach to DL-based image segmentation involving prompts

Source: https://github.com/facebookresearch/segment-anything License: Apache 2

Segment Anything Model

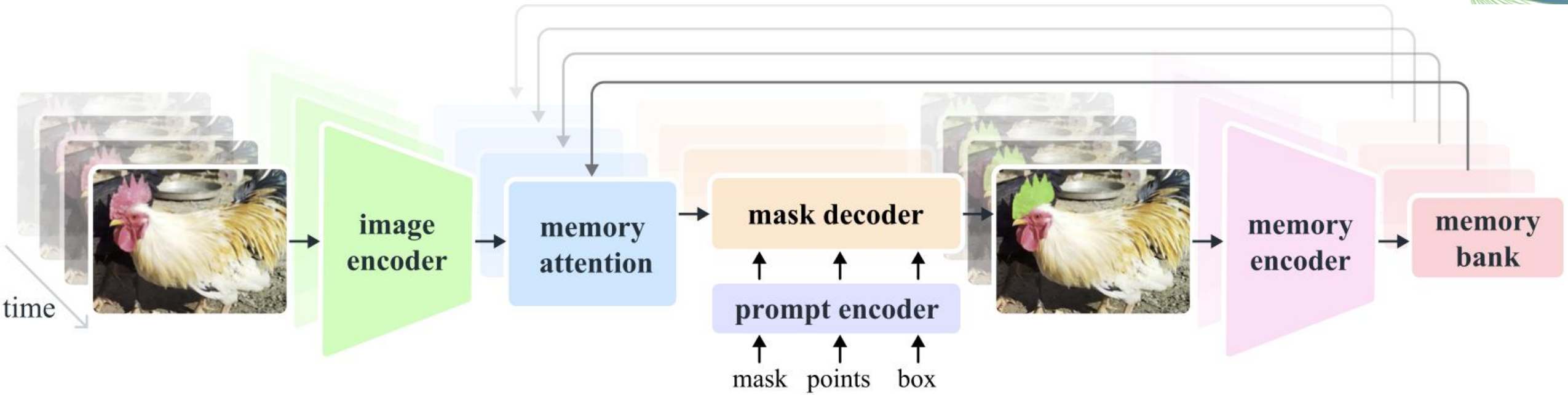
New approach to DL-based image segmentation involving prompts



Source: https://github.com/facebookresearch/segment-anything License: Apache 2

Segment Anything Model 2

Extending the approach to videos by introducing memory



Source: https://github.com/facebookresearch/segment-anything License: Apache 2

Segment Anything Model

Mostly natural images used for benchmarking

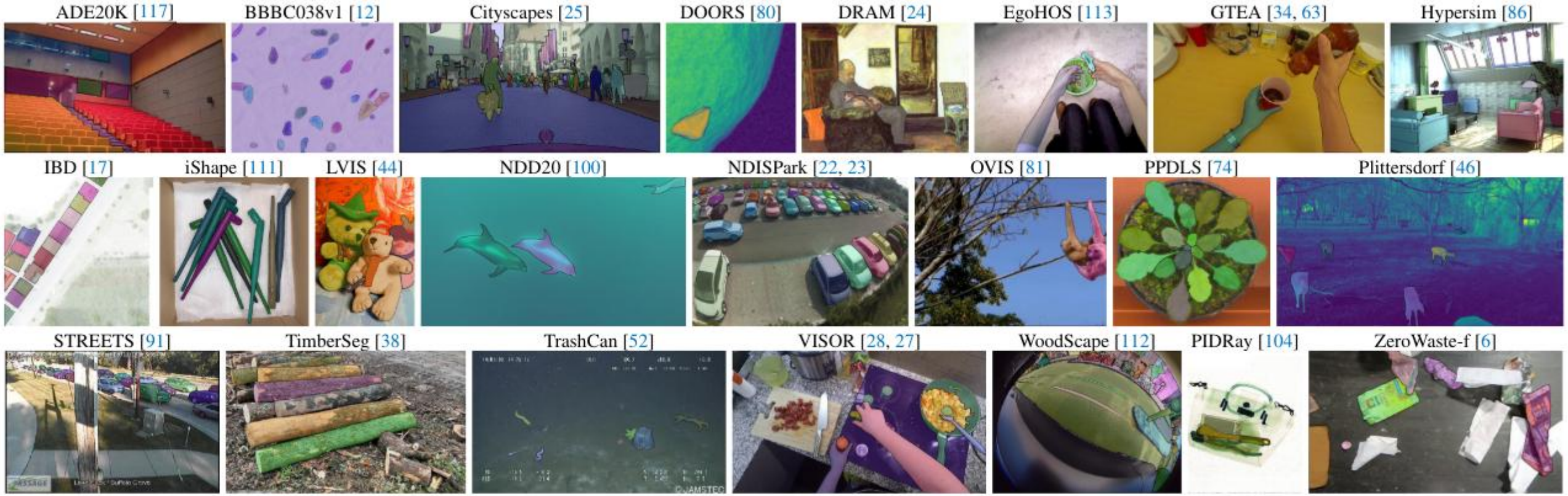
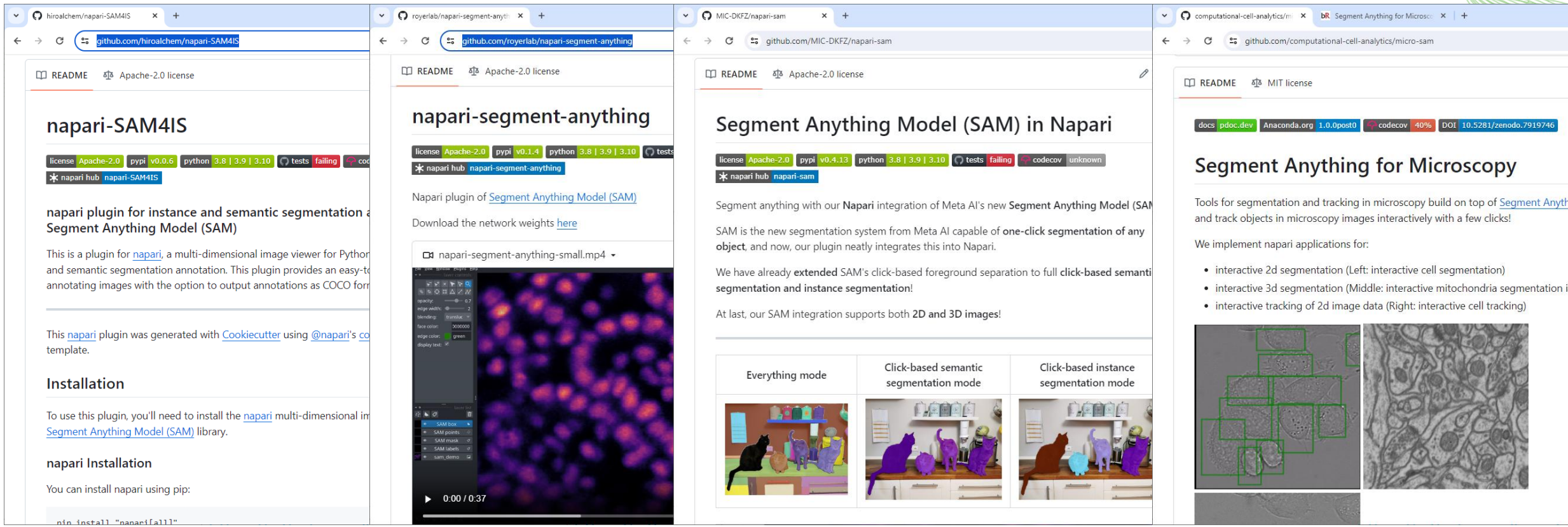


Figure 8: Samples from the 23 diverse segmentation datasets used to evaluate SAM's zero-shot transfer capabilities.

Segment Anything for Microscopy

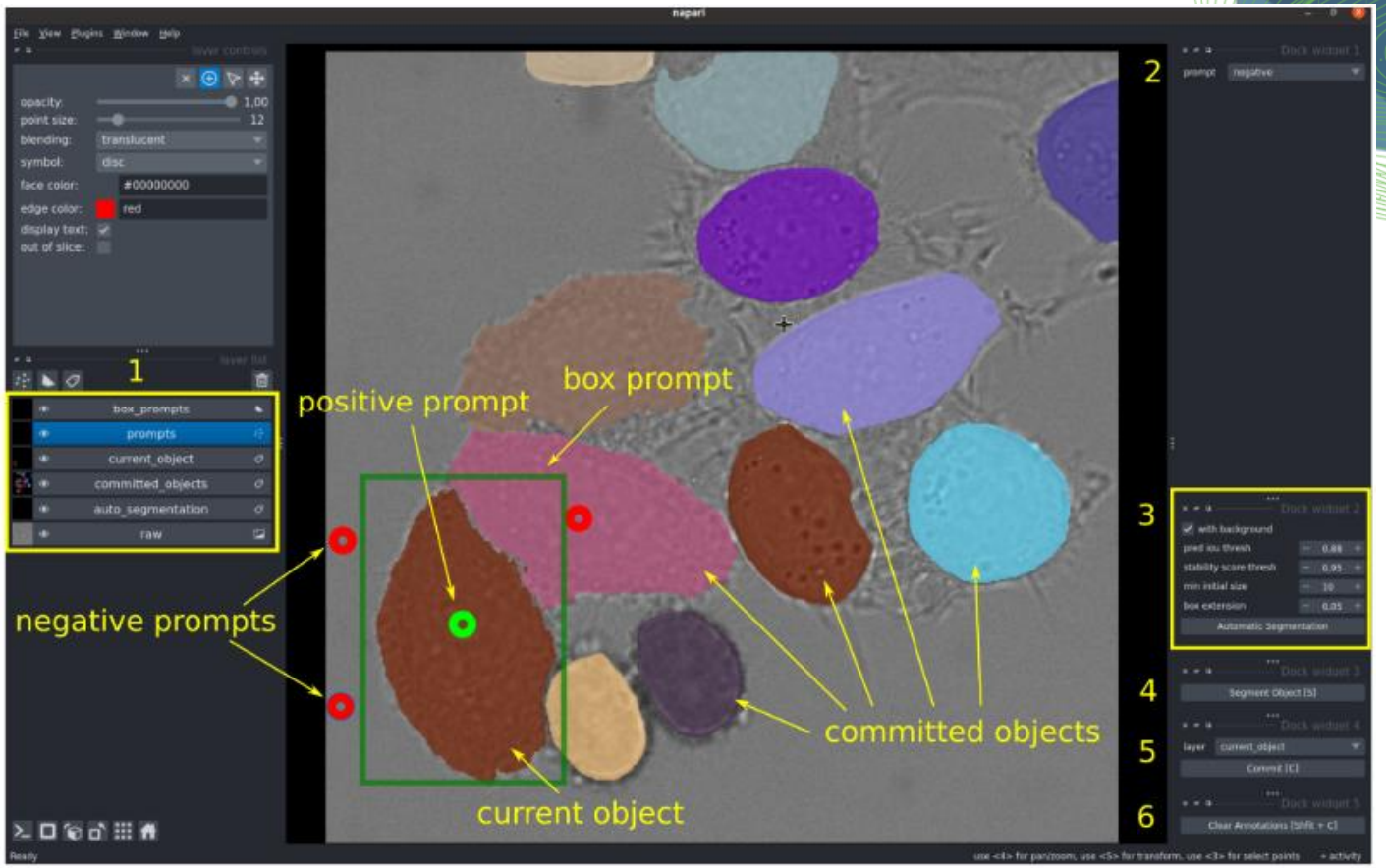
Popping up napari plugins, some within 24h after SAM was published



Segment Anything for Microscopy

Downsides:

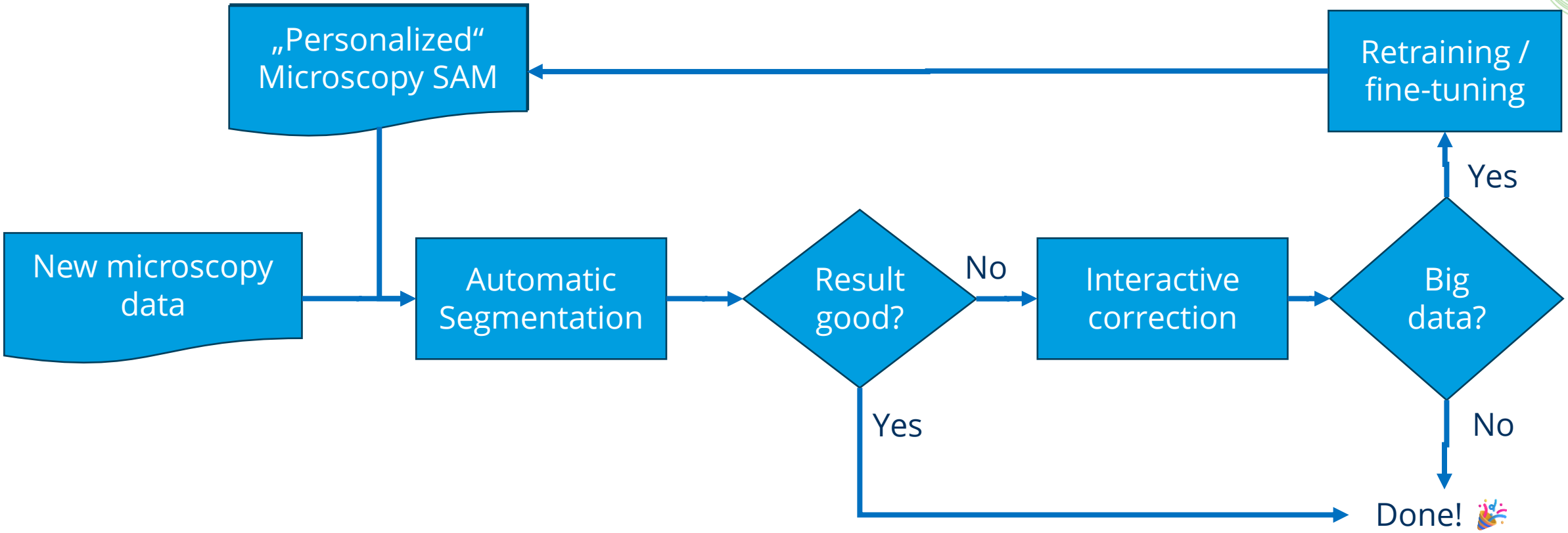
- Original code did not contain the procedure for iterative training
- Instance segmentation not ideal (watershed-implementation added in micro-sam)
- Fine-tuning for microscopy data neccessary



https://www.biorxiv.org/content/10.1101/2023.08.21.554208v1.article-info
https://github.com/computational-cell-analytics/micro-sam

Segment Anything for Microscopy

Real-world scenarios: human-in-the-loop



Segment Anything for Microscopy

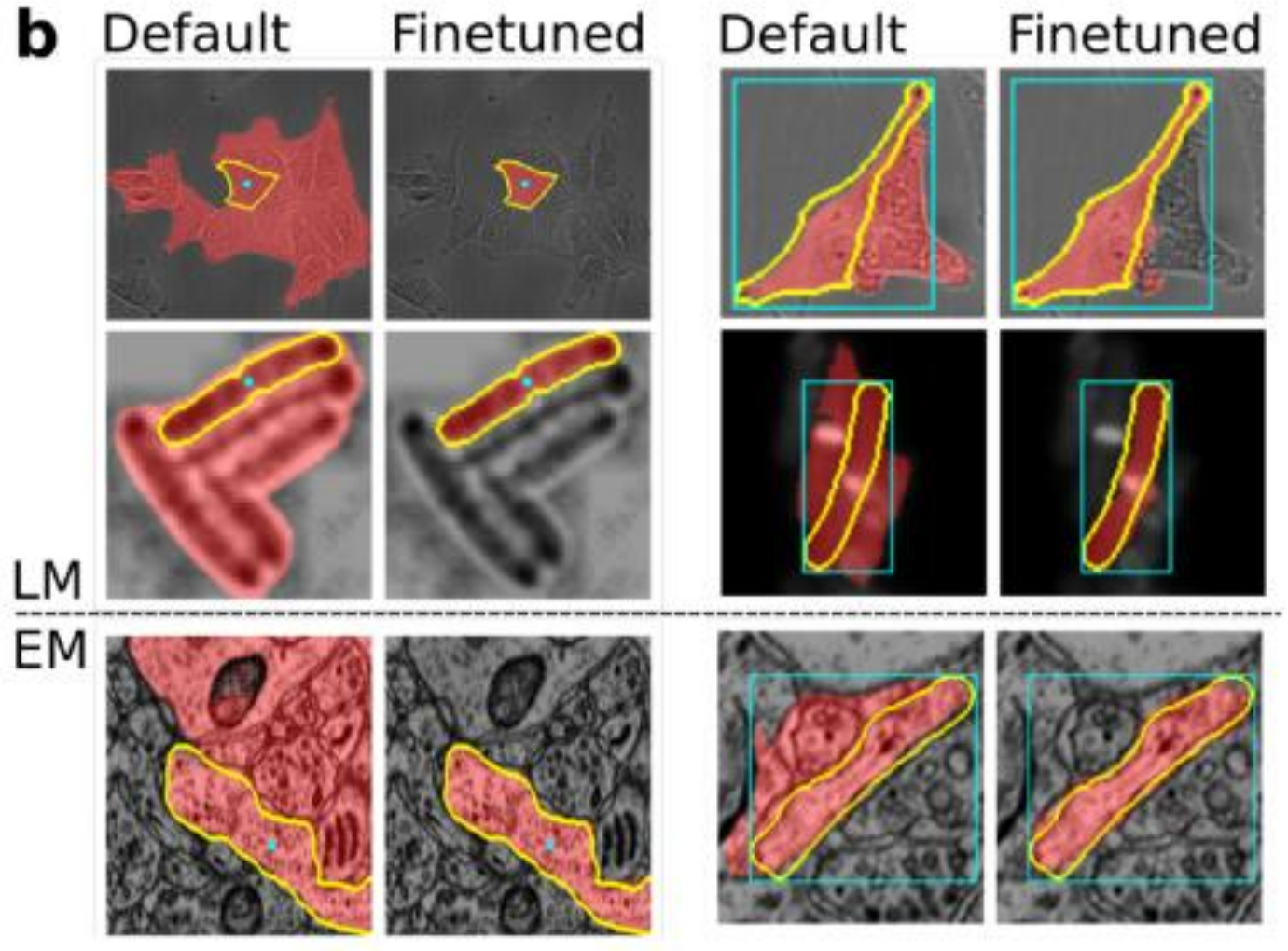
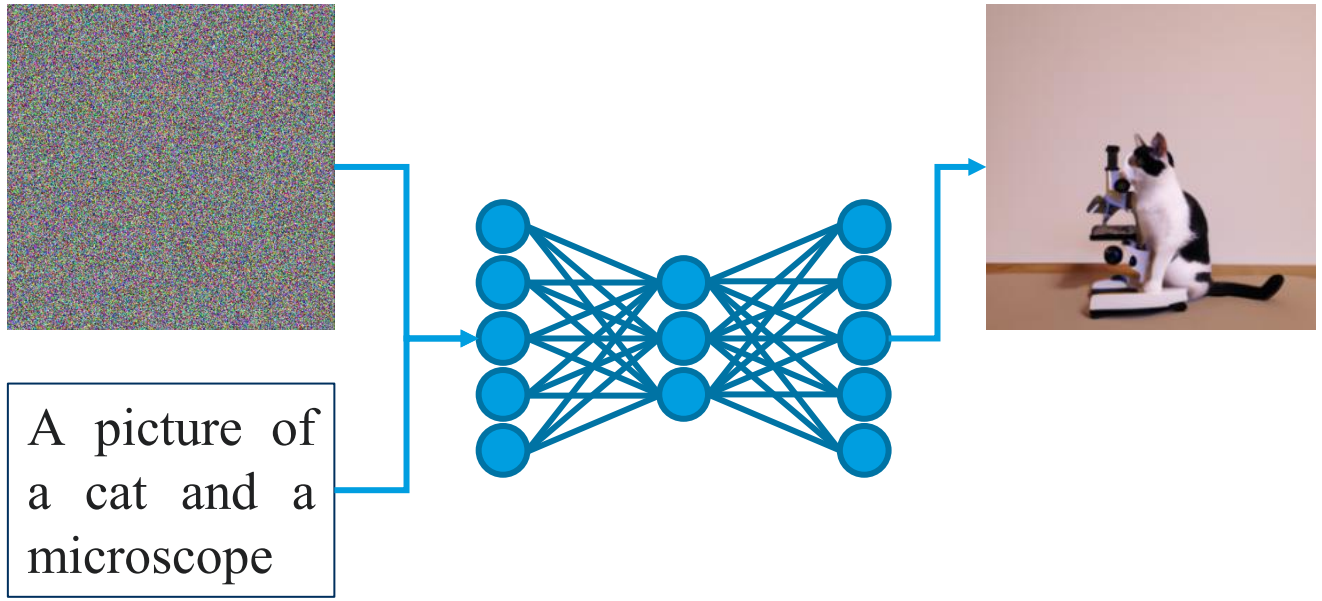


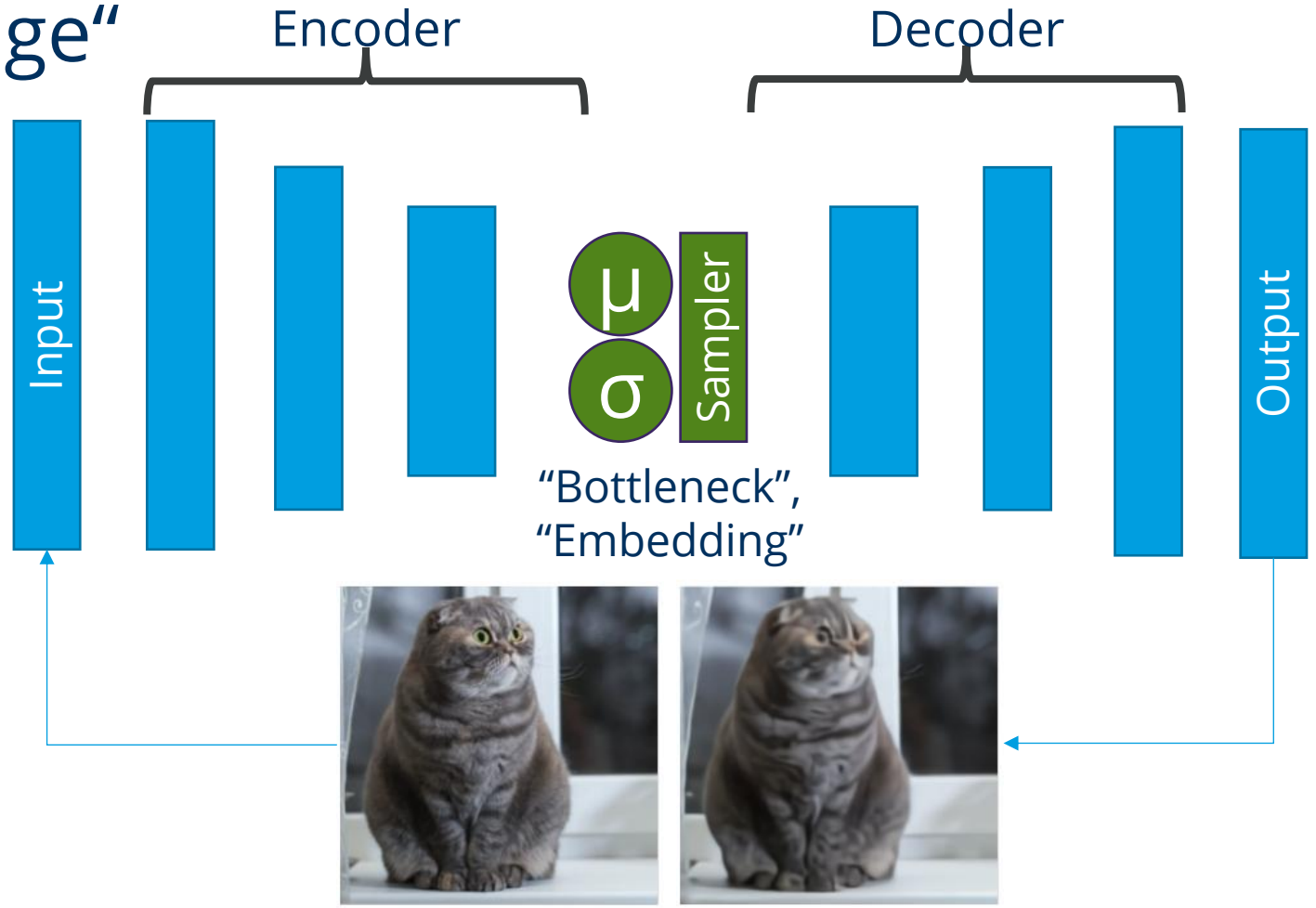
Image Generation

„text-to-image“



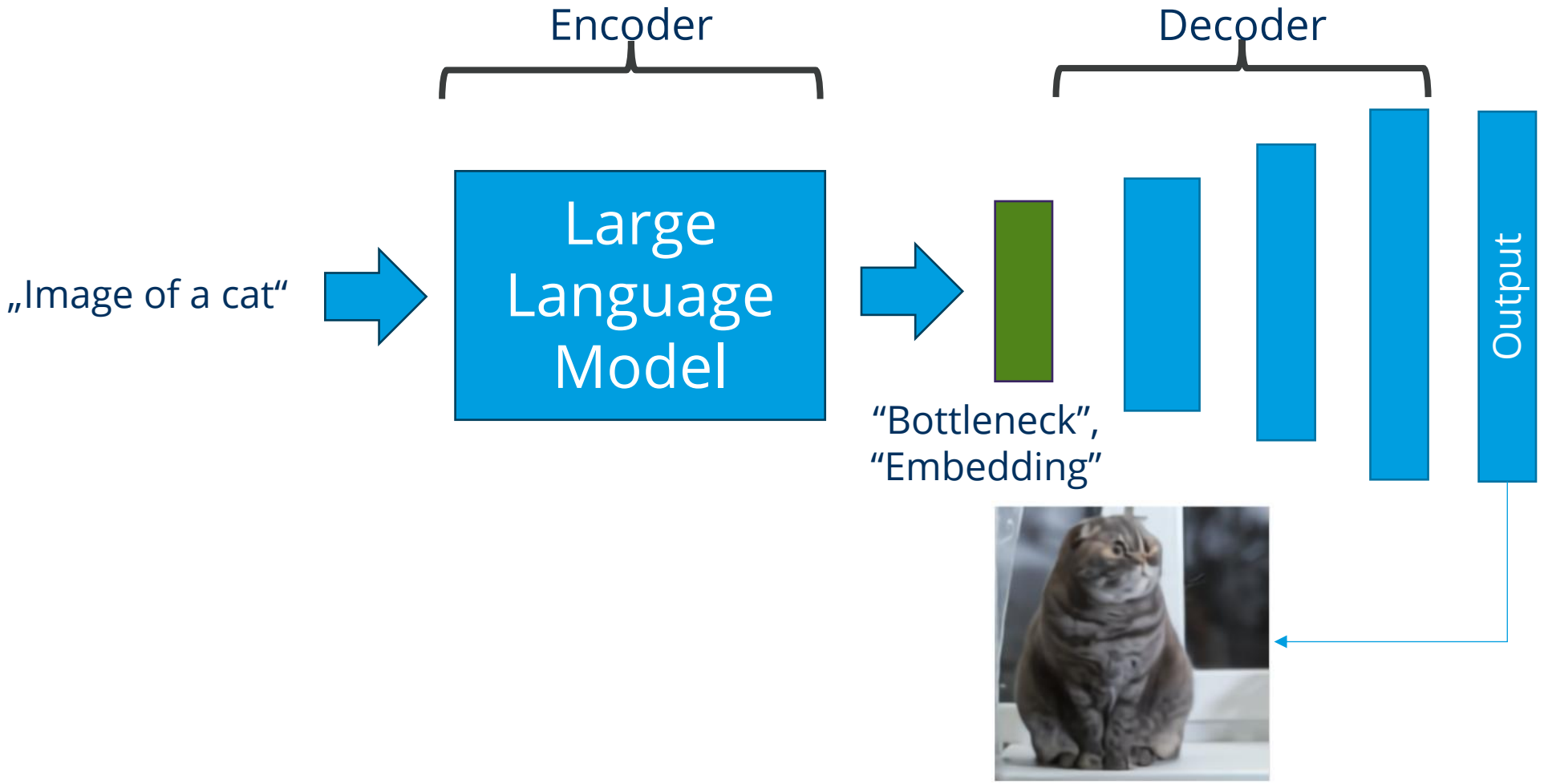
Variational Auto-Encoder

„image-to-image“



Cat picture source: Ramesh et al. 2021, licensed CC-BY 4.0 https://arxiv.org/pdf/2102.12092.pdf

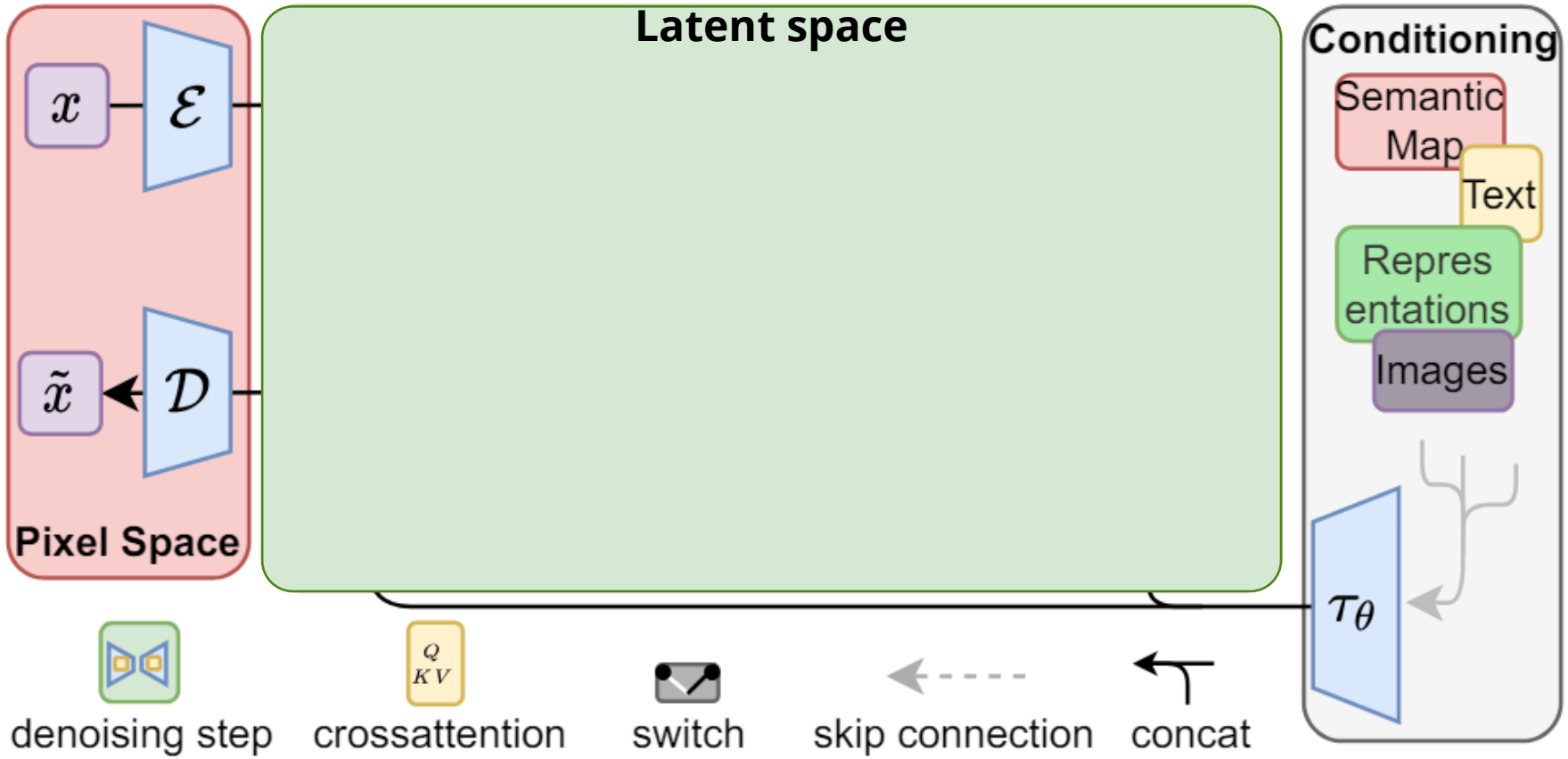
Image Generation



Cat picture source: Ramesh et al. 2021, licensed CC-BY 4.0 https://arxiv.org/pdf/2102.12092.pdf

Stable Diffusion

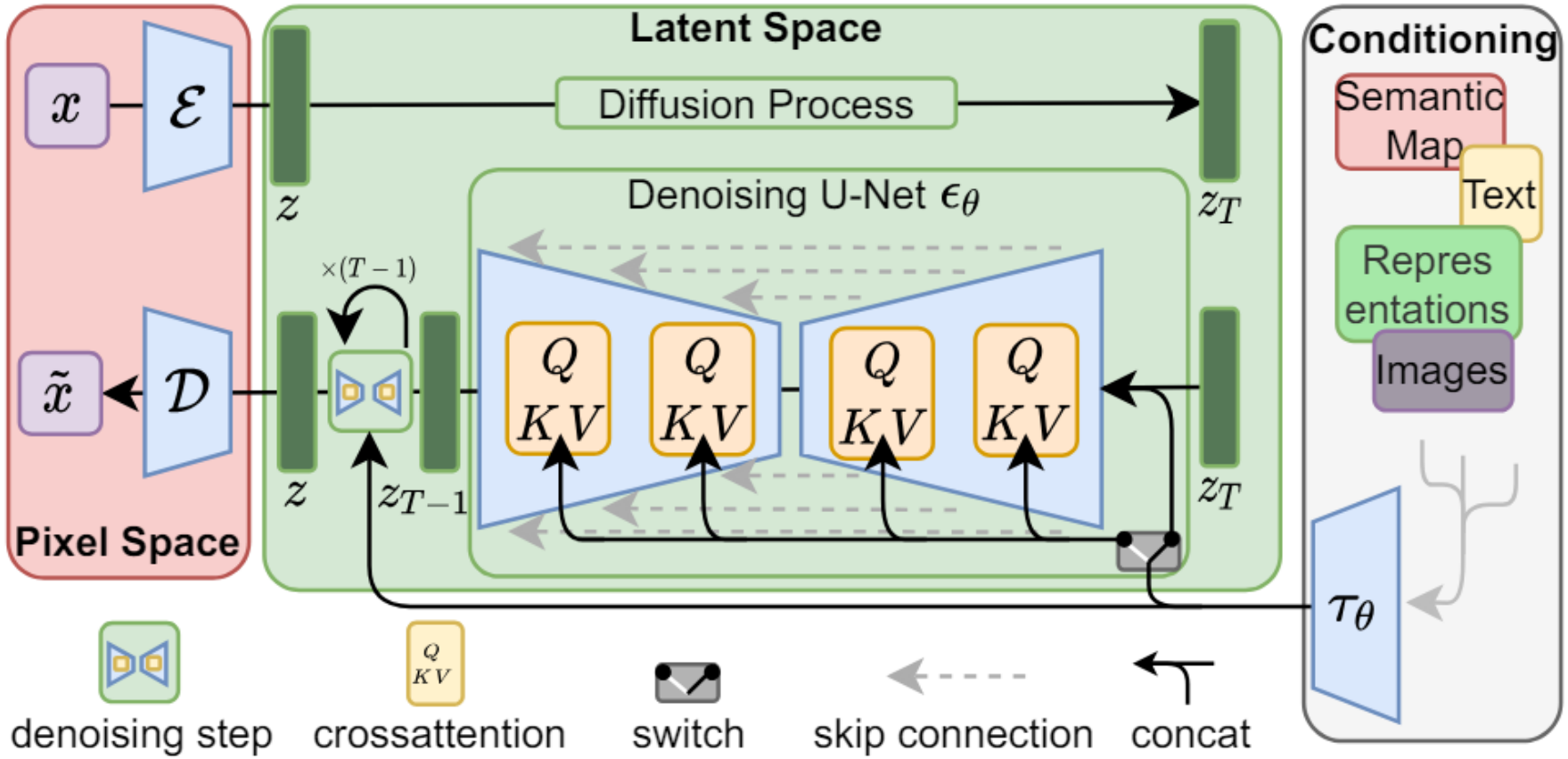
Diffusion: iterative, reverse denoising



Simplified from: Rombach et al. 2021
https://arxiv.org/abs/2112.10752

Stable Diffusion

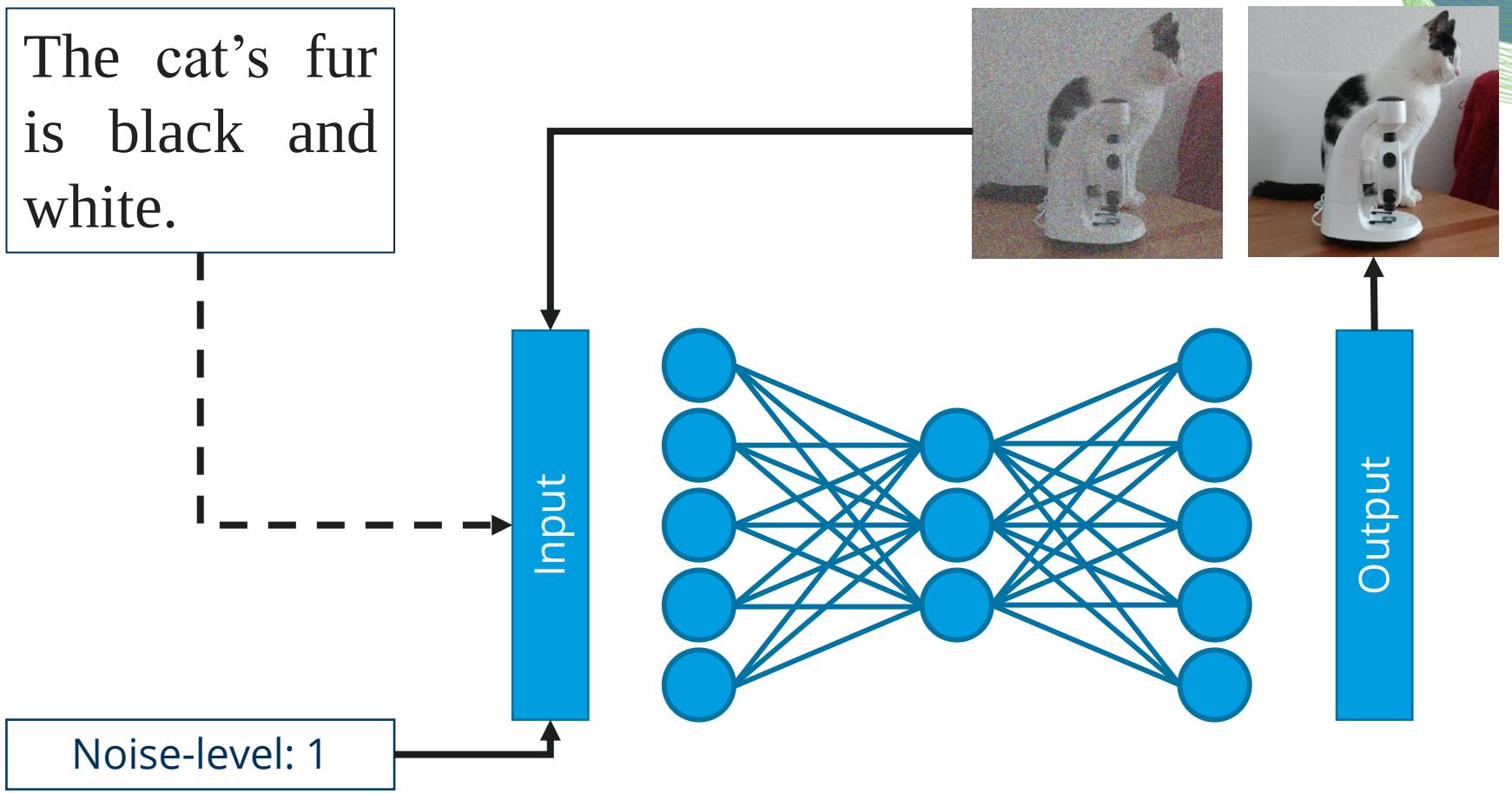
Diffusion: iterative, reverse denoising



Source: Rombach et al. 2021
https://arxiv.org/abs/2112.10752

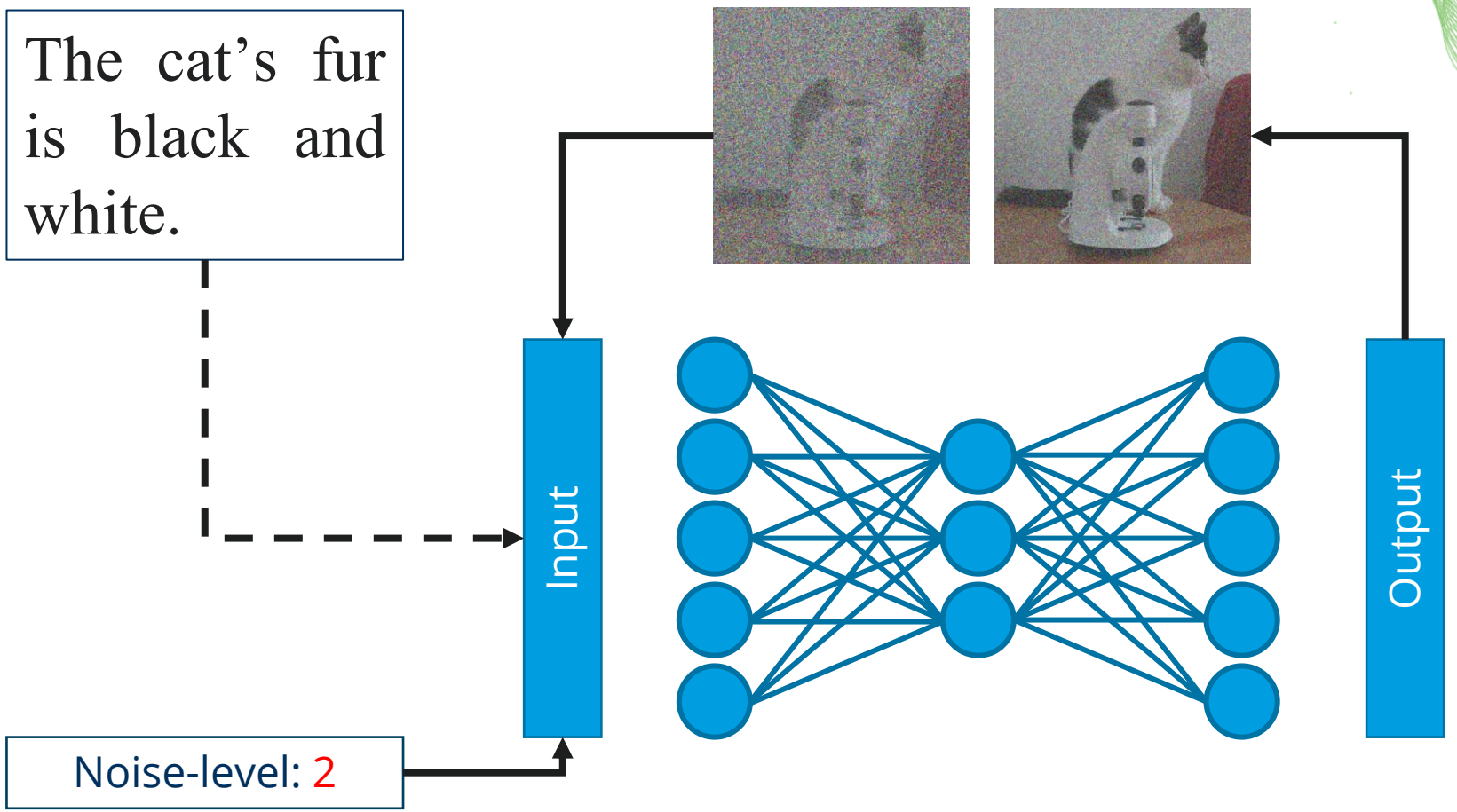
How does it work?

Train a U-Net on data: image + noisy image + description + noise-level



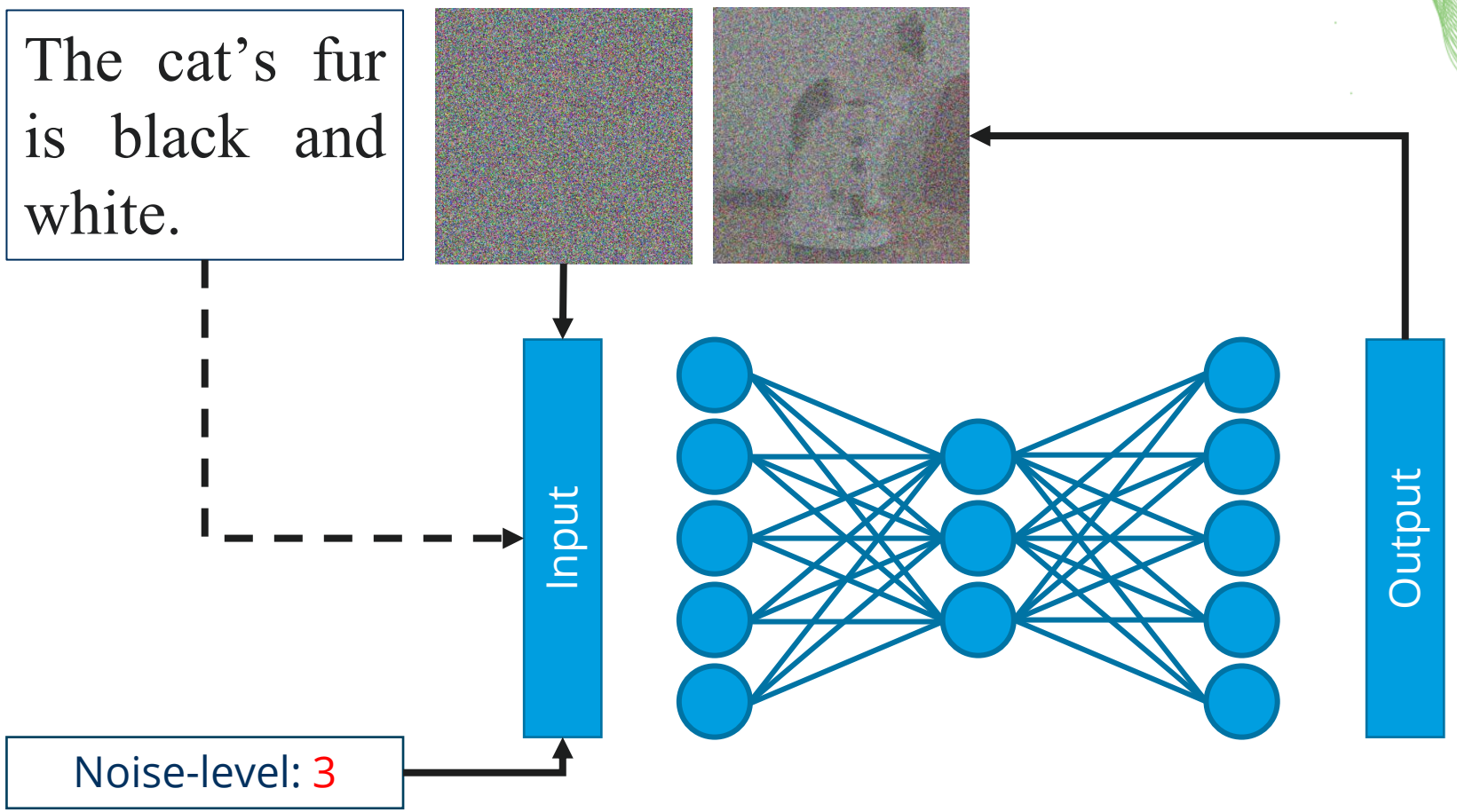
How does it work?

Train a U-Net on data: image + noisy image + description + noise-level



How does it work?

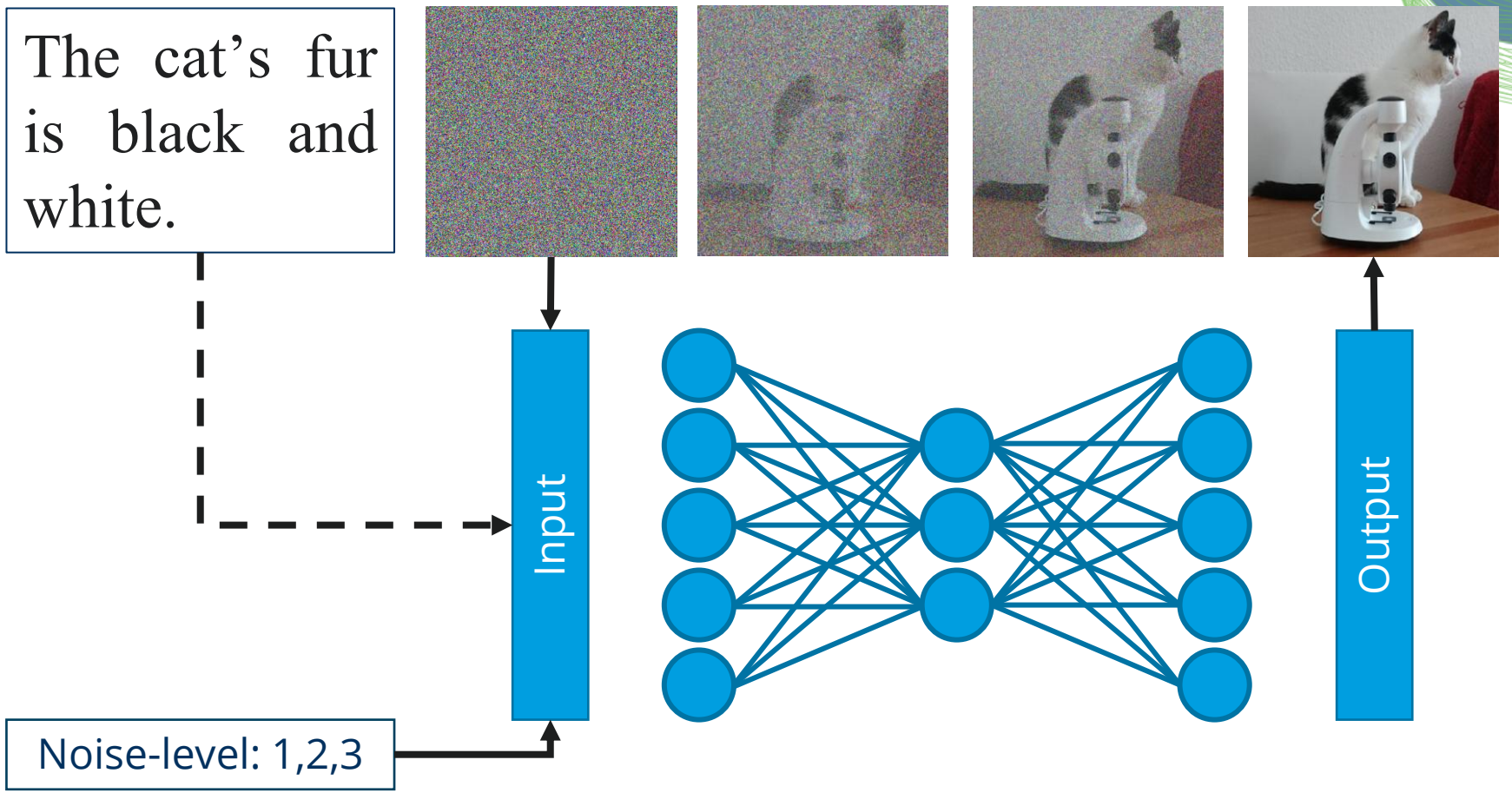
Train a U-Net on data: image + noisy image + description + noise-level



How does it work?

Prediction is iterative denoising of:

Pure noise + text prompt



How does it work?

Reminder:

- Word embeddings
- Attention

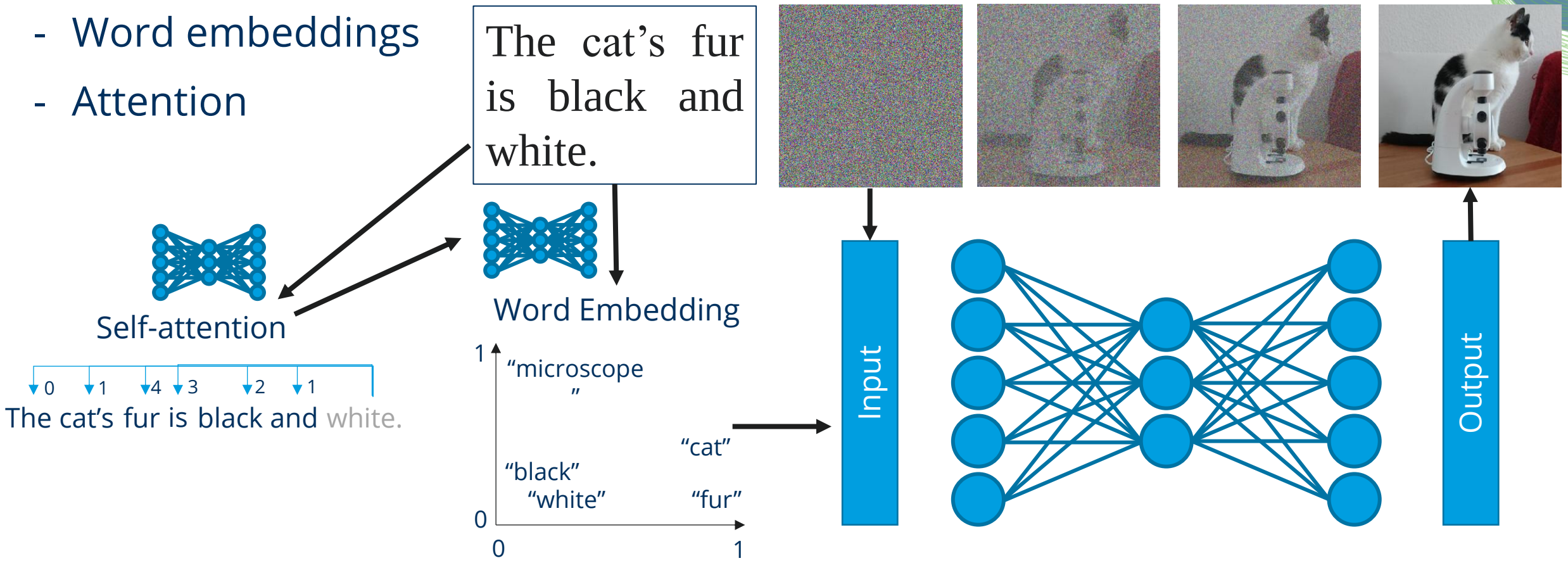


Image generation in Python: Huggingface

Most [Huggingface] image-generation models require a GPU.

```
pipe = DiffusionPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-1-base", torch_dtype=torch.float16
)
```

Downloads 4.8 GB

```
pipe = pipe.to("cuda")
```

Needs Nvidia GPU

```
prompt = """
Draw a realistic photo of an astronaut riding a horse.
"""

astronaut = pipe(prompt).images[0]
astronaut
```

Image generation in Python: Huggingface

Works well if the prompt overlaps with training data, potentially huge variation between attempts

```
prompt = """
Draw a realistic photo of an astronaut riding a horse.
"""

astronaut = pipe(prompt).images[0]
astronaut
```

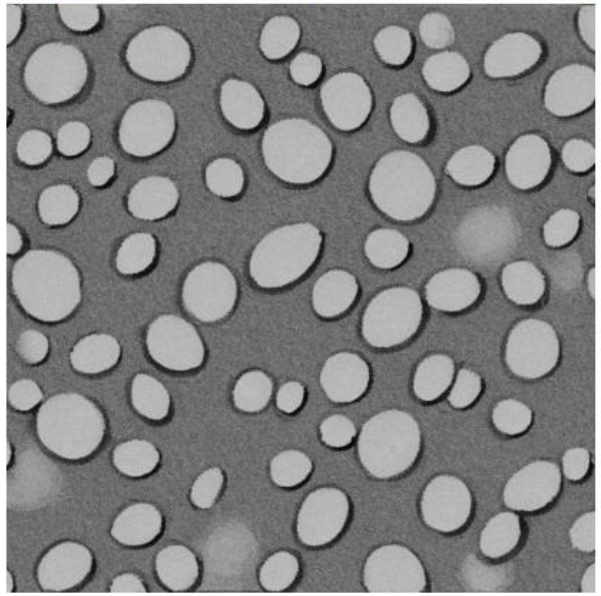
```
100%  50/50 [00:43<00:00,  1.24s/it]
```

```
prompt = """
Draw a realistic photo of a lecture hall with an
ongoing lecture about vision language models.
"""

photo = pipe(prompt).images[0]
photo
```

```
100%  50/50 [01:30<00:00,  1.40s/it]
```

```
prompt = """
Draw a greyscale picture of sparse bright blobs on dark
background. Some of the blobs are roundish, some are a
bit elongated.
"""

image = pipe(prompt).images[0]
image
```

```
100%  50/50 [01:16<00:00,  1.18s/it]
```



```
image = pipe(prompt,
             num_inference_steps=10,
             width=512,
             height=512).images[0]
image
```

```
100%  10/10 [00:17<00:00,  1.88s/it]
```

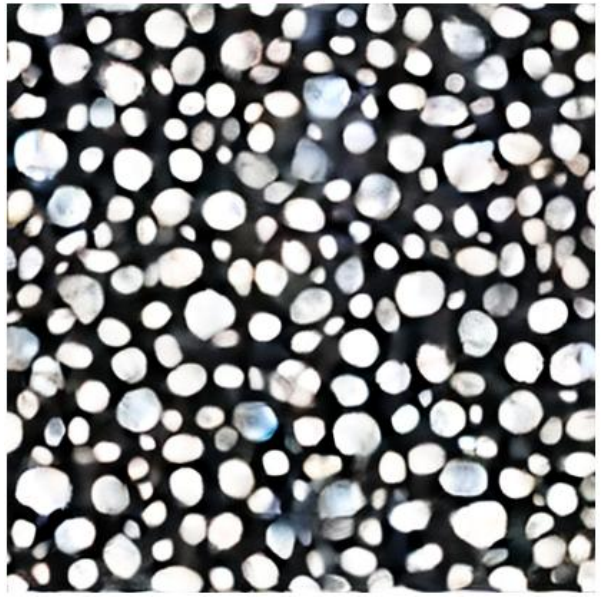


Image generation in Python: Dall-E

No need for a GPU, but costs 💸 💸 💸

```python
def prompt_image(message:str, width:int=1024, height:int=1024, model='dall-e-3'):
    client = openai.OpenAI()
    response = client.images.generate(
      prompt=message,
      model=model,
      n=1,
      size=f"{width}x{height}"
    )
    image_url = response.data[0].url
    image = imread(image_url)

    return image
```

Works with Dall-E 2 and 3

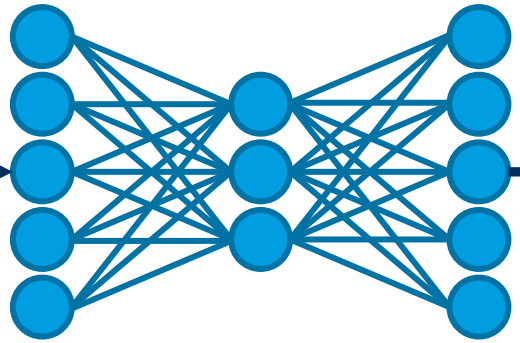
May soon also work with gpt-4o

Image Generation LLMs

Challenges: fake-images

Interesting challenges for our community ahead

a histology image of lung cancer cells and some healthy tissue



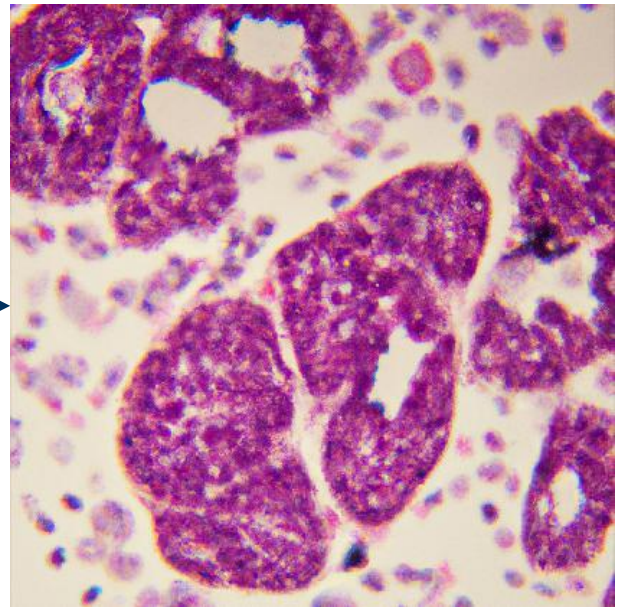
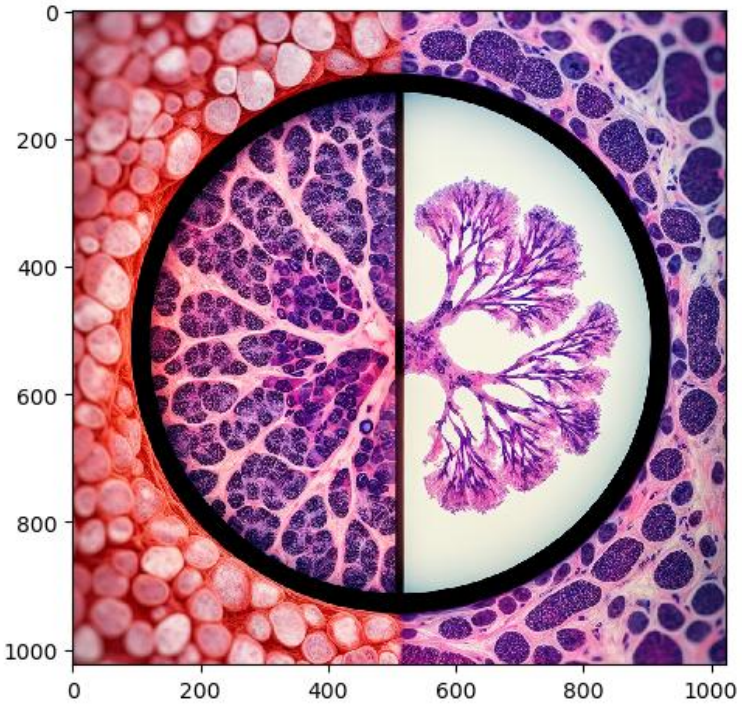


Image generation in Python: Dall-E

Is Dall-E 2 more capable of creating realistic microscopy images than Dall-E 3?

```
histology = prompt_image('a histology image of lung cancer cells and some healthy tissue')
imshow(histology)
```

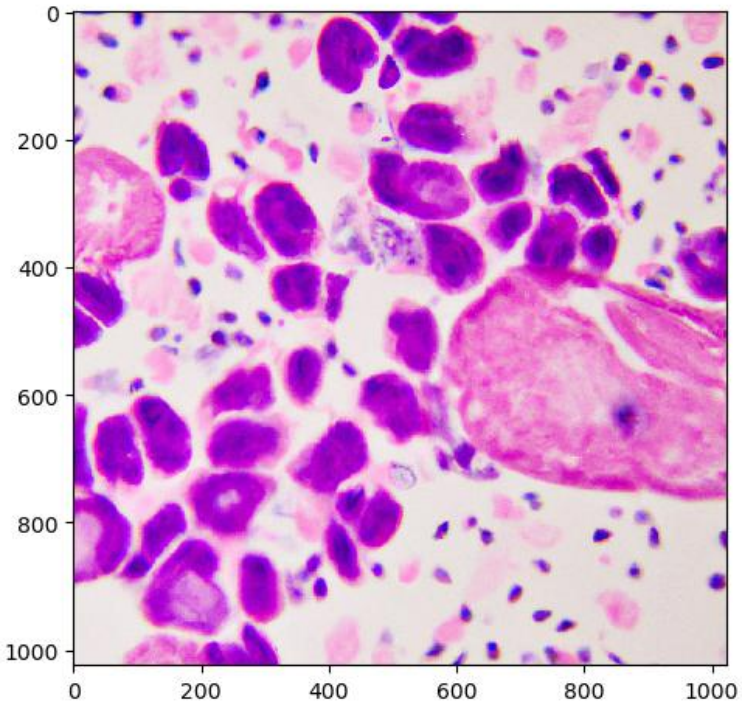
```
<matplotlib.image.AxesImage at 0x1d9eda5bd90>
```



Dall-E 3

```
histology = prompt_image('a histology image of lung cancer cells and some healthy tissue',
                         model='dall-e-2')
imshow(histology)
```

```
<matplotlib.image.AxesImage at 0x1d9edac6fd0>
```



Dall-E 2

Image Generation LLMs

Challenges: physical / physiological hallucinations

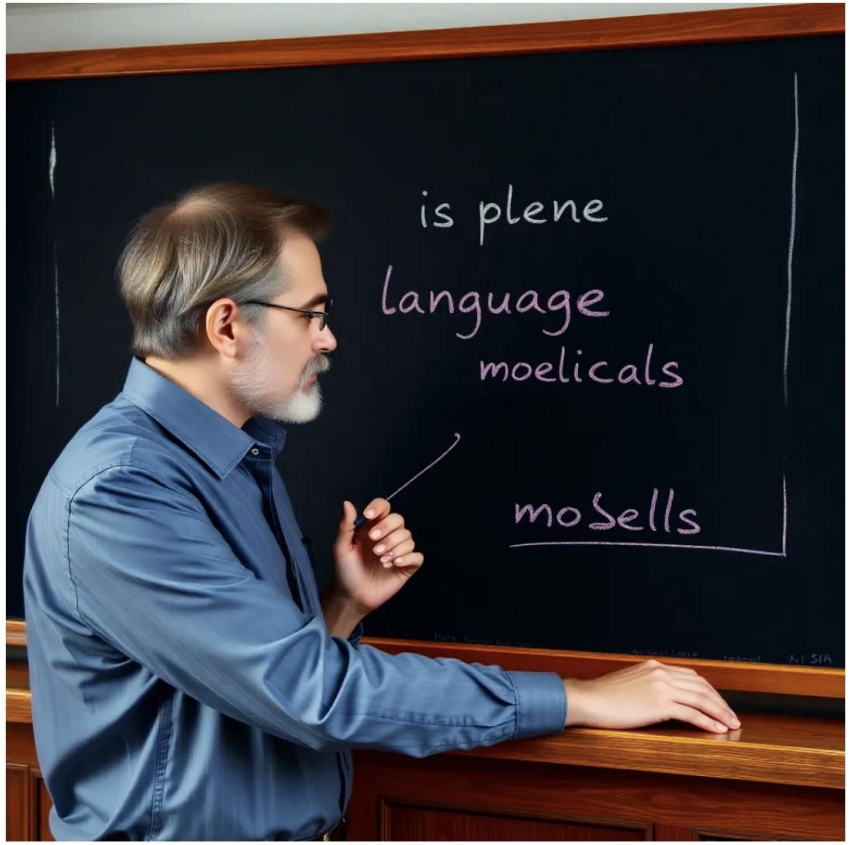
Prompt: “Draw a close-up picture of people's hands”

Source: https://replicate.com/black-forest-labs/flux-schnell-lora

Image Generation LLMs

Challenges: Text in images

Prompt: ‘please draw a picture of a professor writing "large language models" on a blackboard’



Source: https://replicate.com/black-forest-labs/flux-schnell-lora

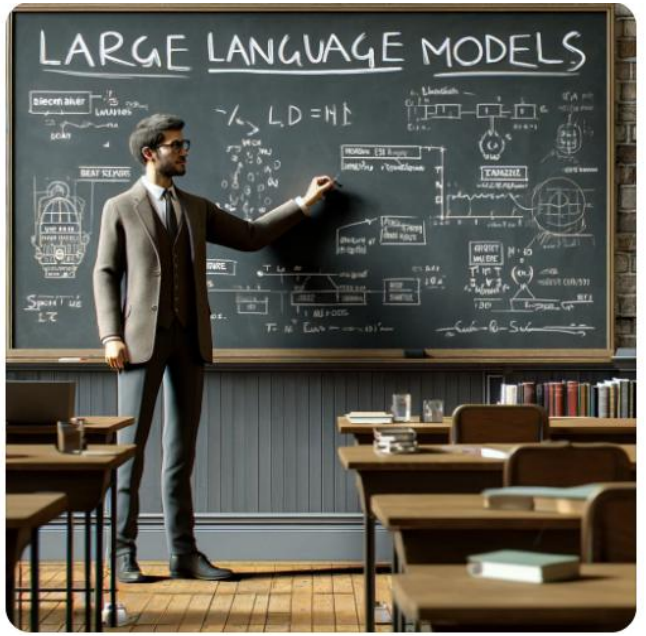
Image Generation LLMs

Challenges: Bias

Prompt: “a nice photo of a human”

Source: https://replicate.com/stability-ai/stable-diffusion

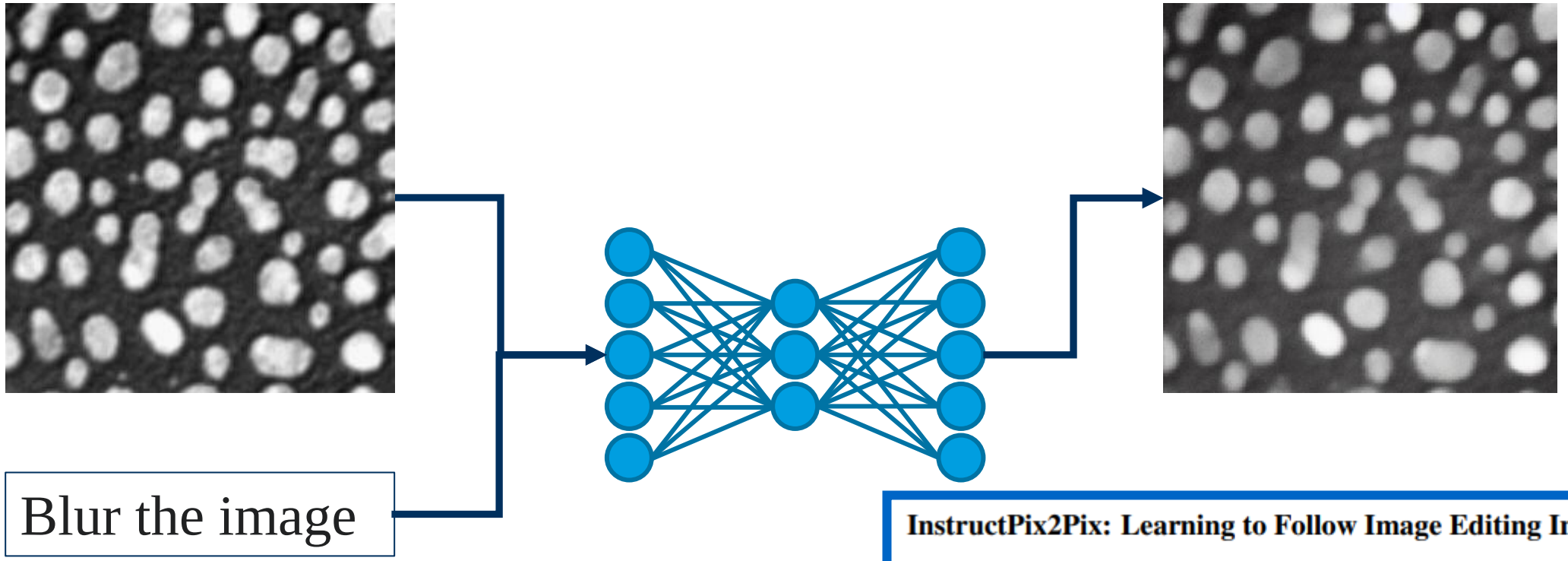
Prompt: ‘please draw a picture of a professor writing "large language models" on a blackboard’



Source: https://chatgpt.com/

Image Generation LLMs

Image-to-image prompting, image variations



InstructPix2Pix: Learning to Follow Image Editing Instructions

Tim Brooks* Aleksander Holynski* Alexei A. Efros

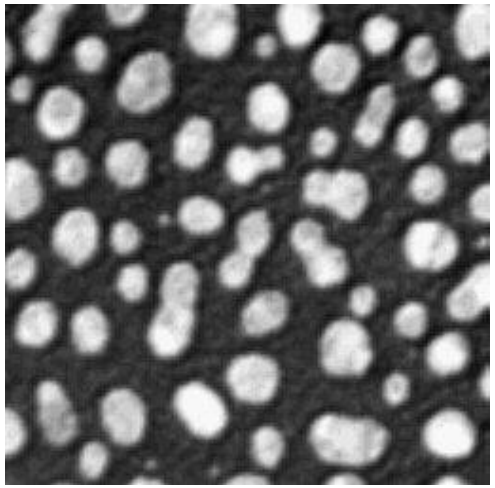
University of California, Berkeley

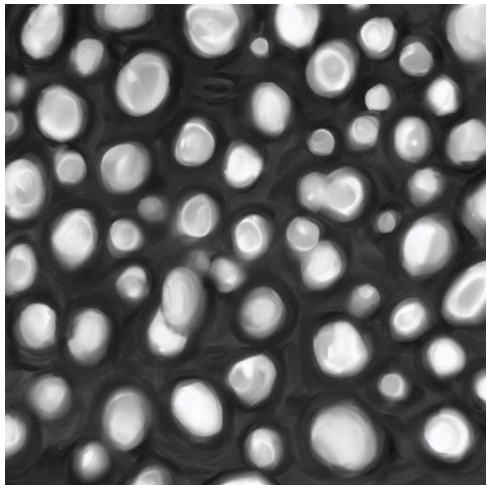
https://arxiv.org/abs/2211.09800
https://scads.github.io/generative-ai-notebooks/41_image_generation_huggingface/60_image_variations_pix2pix.html

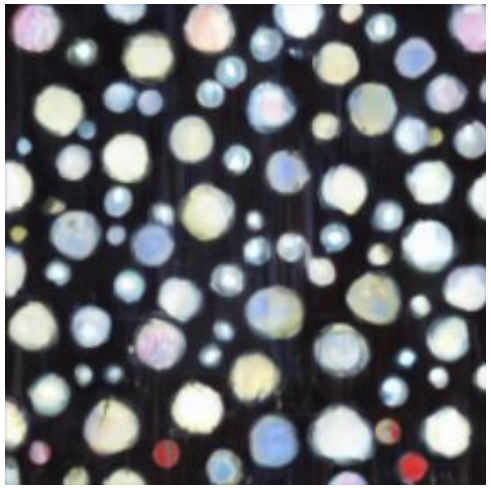
Image variation

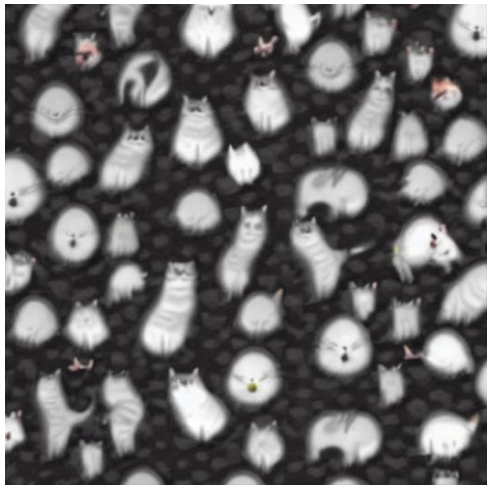
Generate images, e.g. for augmenting data

Potentially useful to make algorithms more robust





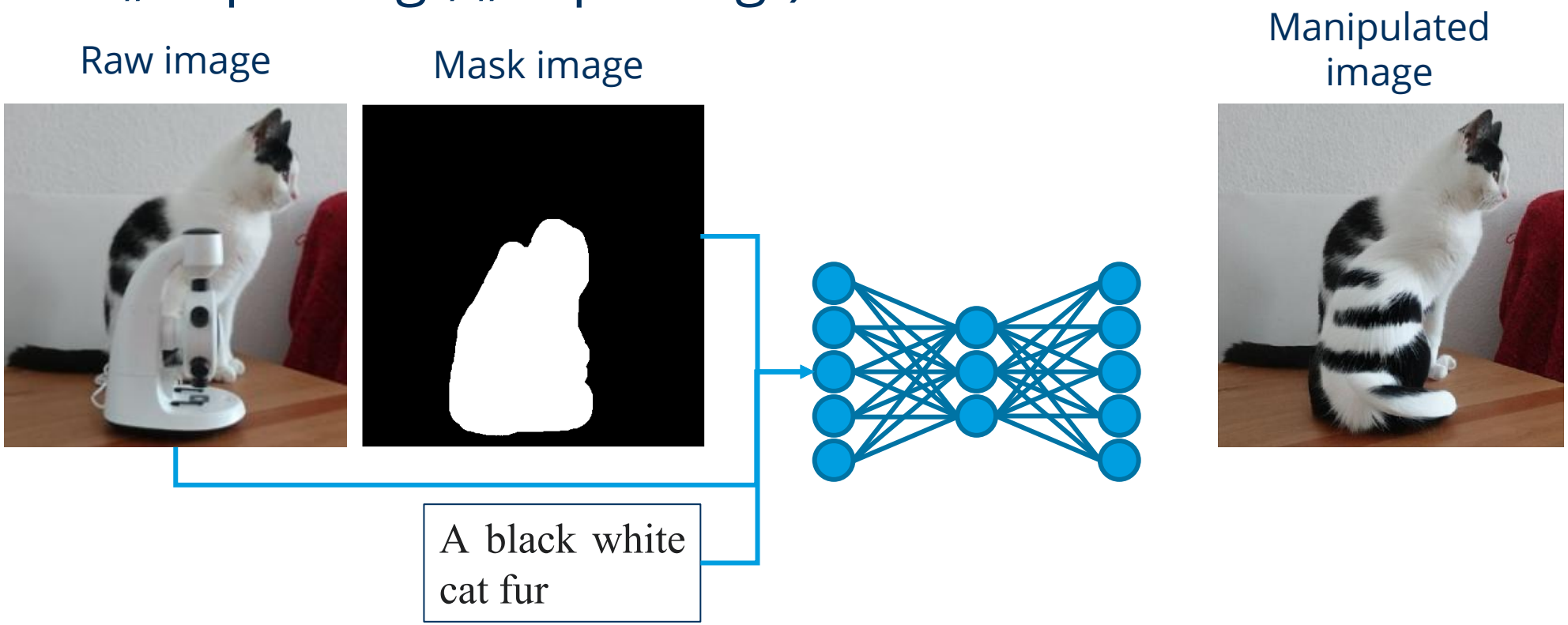




Inpainting

Replacing regions in images

(also „Gap-filling“, „Replacing“)

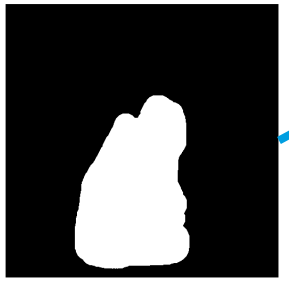


Inpainting in Python: Huggingface

```python
pipe = StableDiffusionInpaintPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-inpainting",
    torch_dtype=torch.float16
)
pipe = pipe.to("cuda")
```

Downloads 4.8 GB

Needs Nvidia GPU



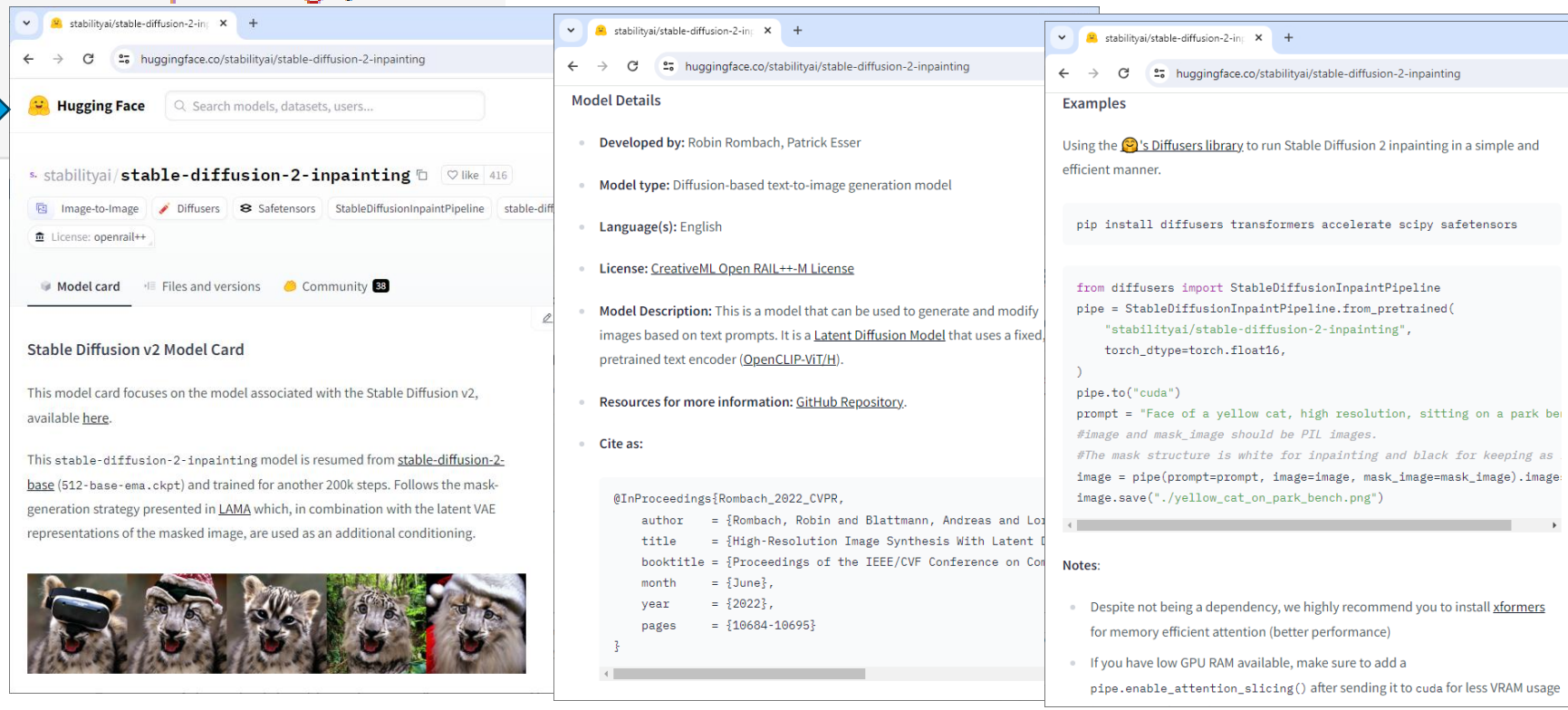
```python
prompt = "A black white cat fur"
image = pipe(prompt=prompt,
             image=init_image,
             mask_image=mask_image,
             num_inference_steps=50,
             width=512,
             height=512,
             num_images_per_prompt=1,
            ).images[0]
```

Read more:
https://huggingface.co/docs/diffusers/api/pipelines/stable_diffusion/inpaint

Inpainting in Python: Huggingface

Check out the *model cards* online in the Huggingface hub.

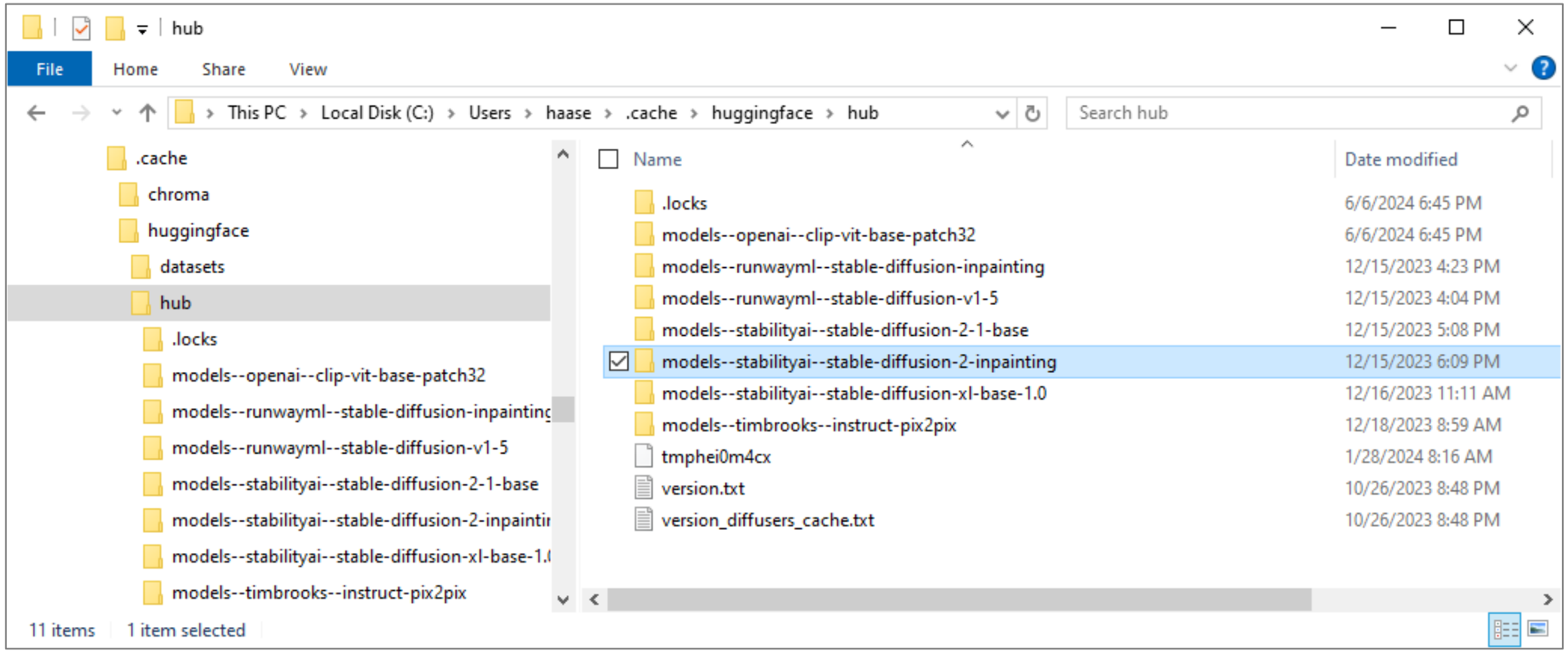
```python
pipe = StableDiffusionInpaintPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-inpainting",
    torch_dtype=torch.float16
)
pipe = pipe.to("cuda")
```



Inpainting in Python: Huggingface

You find the downloaded models cached in your home directory

They are big! Clean up here from time to time.



Inpainting in Python: Dall-E

No need for a GPU, but costs 💸 💸 💸

```
client = OpenAI()

response = client.images.edit(
  image=numpy_to_bytestream(resized_image_rgb),
  mask=numpy_to_bytestream(masked_rgba),
  prompt=prompt,
  n=1,
  size=f"{image_width}x{image_height}",
  model=model
)
```

2D RGB images only

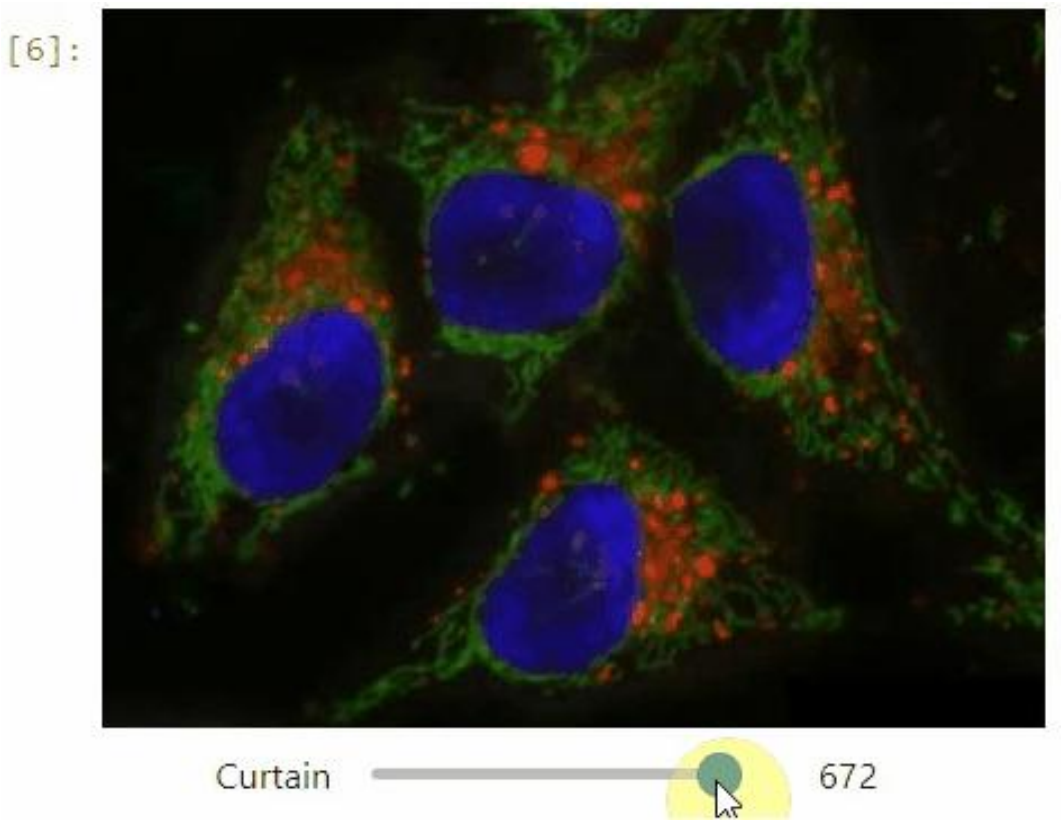
Size must match

Supported: 256, 512, 1024 pixels

Result: List of URL(s)

New technologies bring new risks...

If you can generate images,
you can also generate parts of images....



Interesting challenges for our community ahead

Image manipulation detection

The noise pattern differs between raw and processed images…

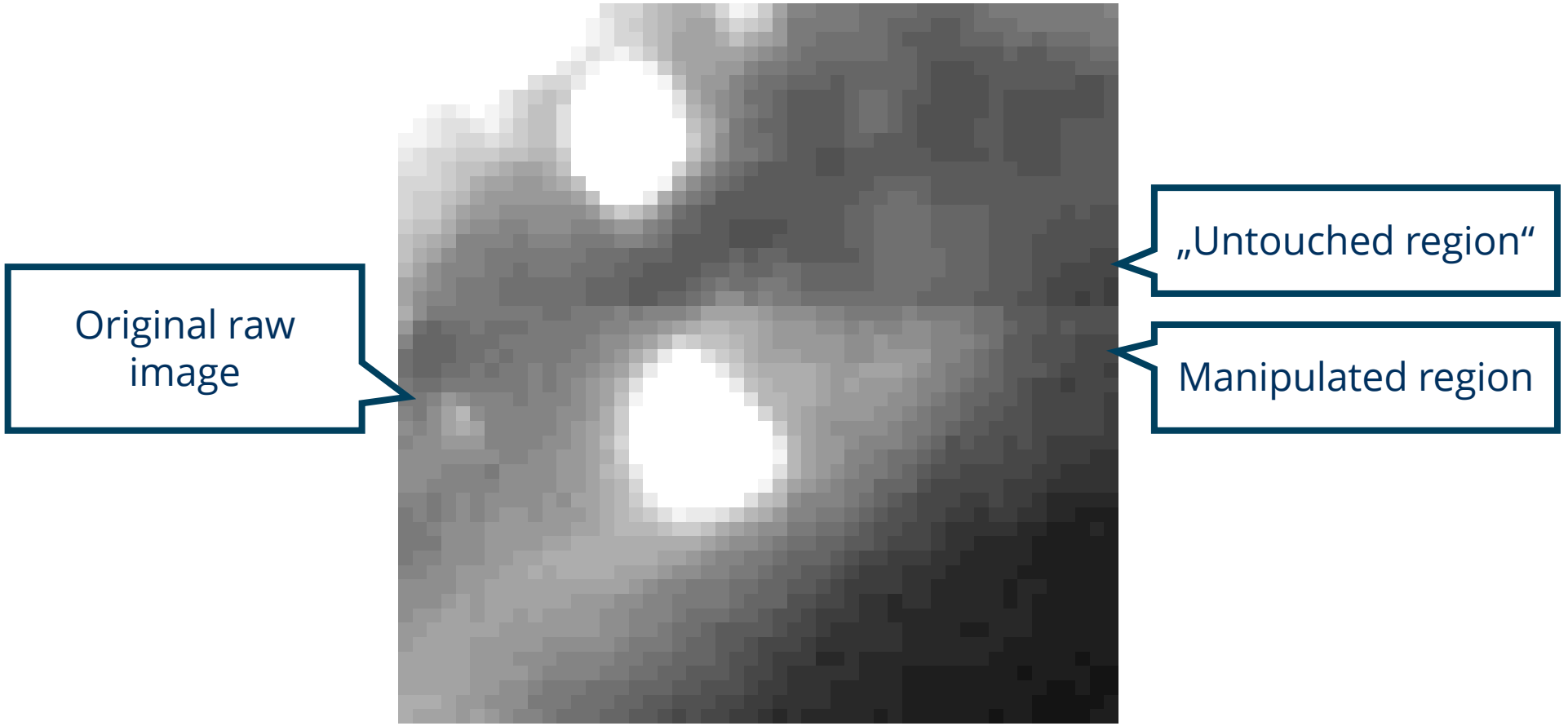
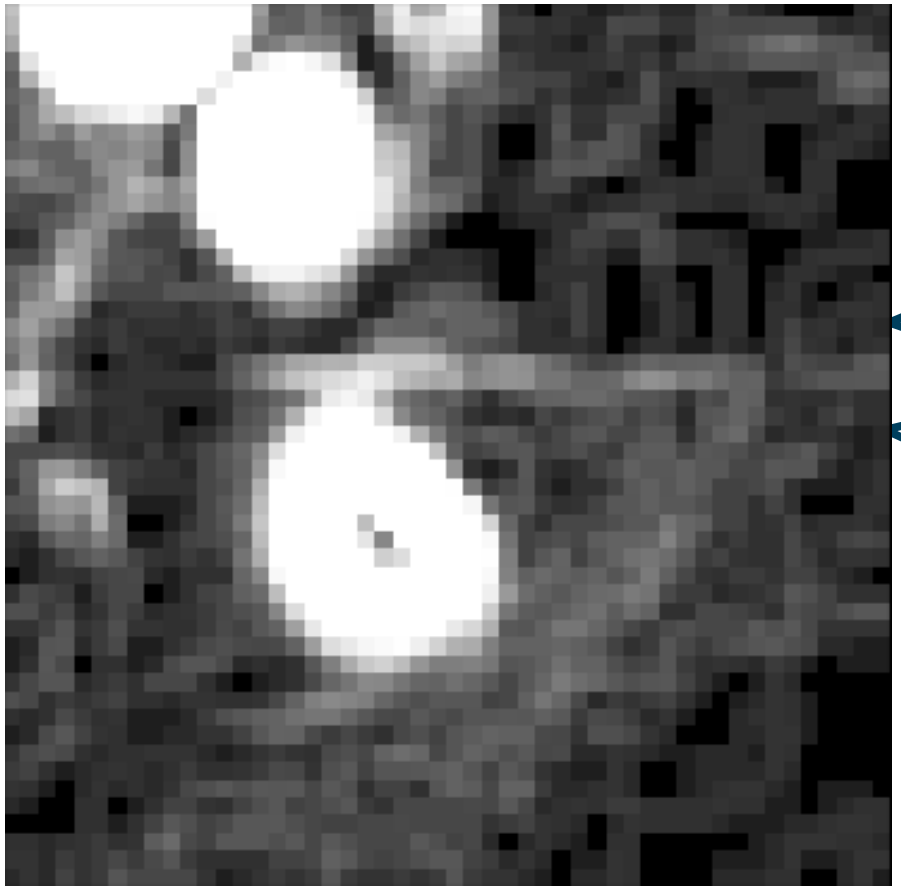


Image manipulation detection

e.g. by studying noise-patterns

Local standard deviation filter



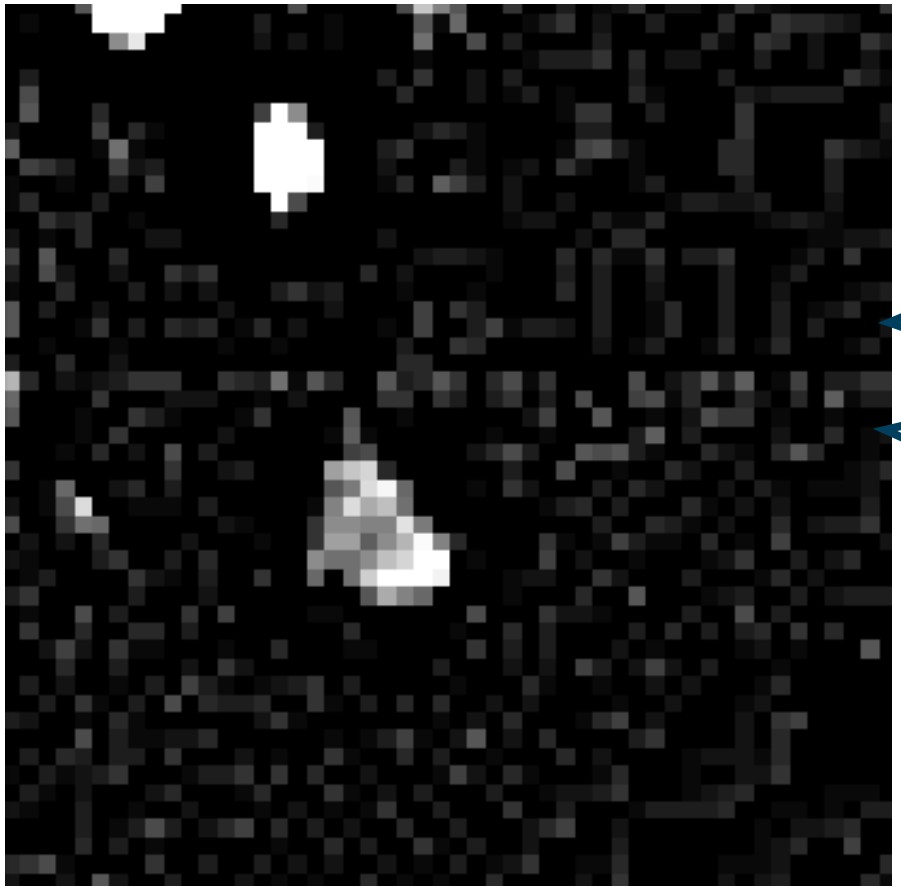
„Untouched region“

Manipulated region

Image manipulation detection

e.g. by studying noise-patterns

Sobel filter



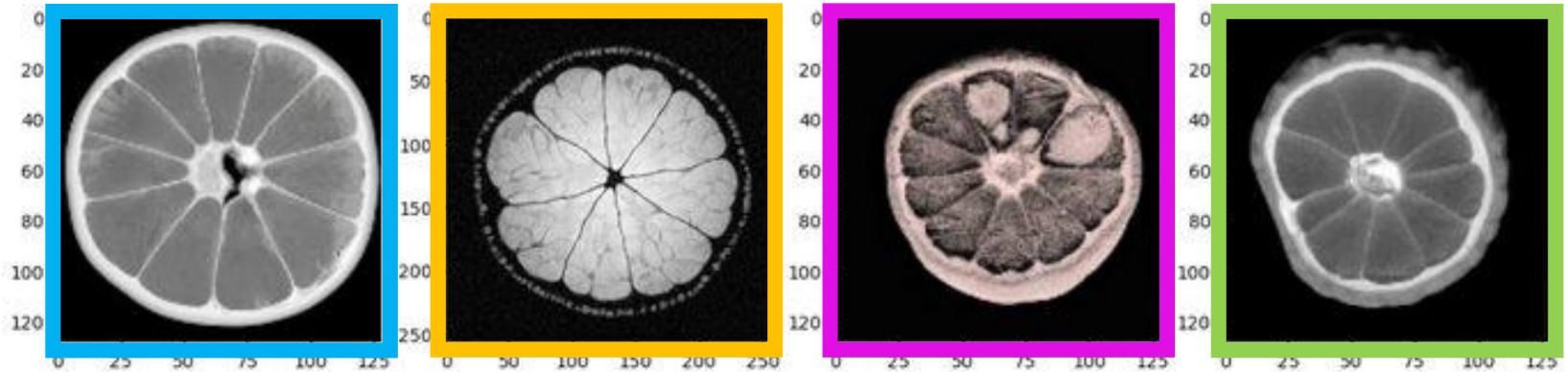
„Untouched region“

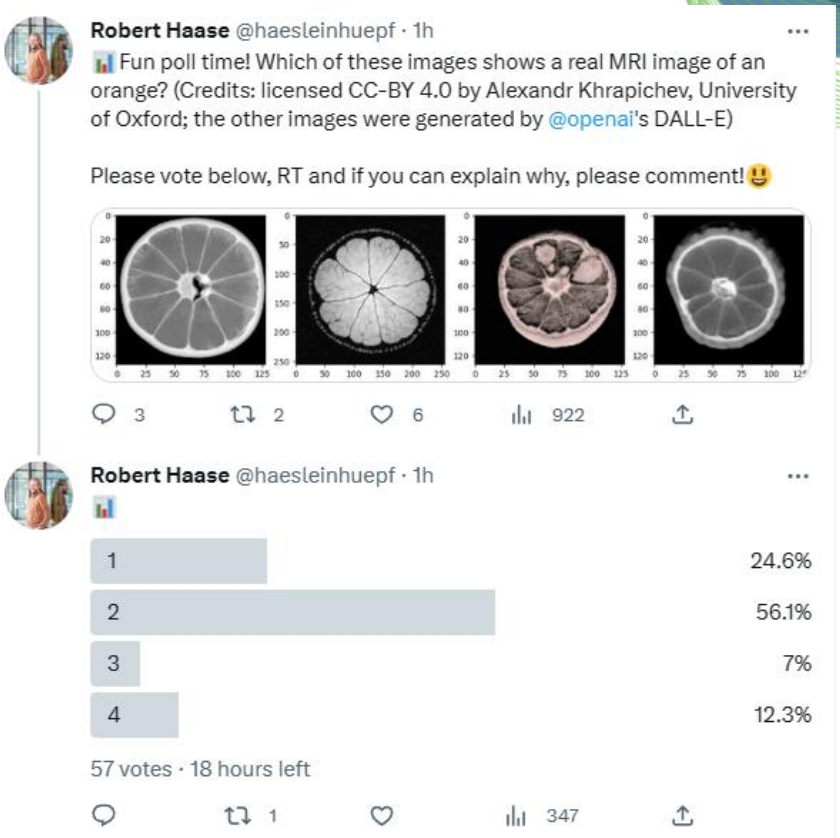
Manipulated region

Benchmarking image generation

When asking humans to evaluate results, make sure they are the right target audience

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





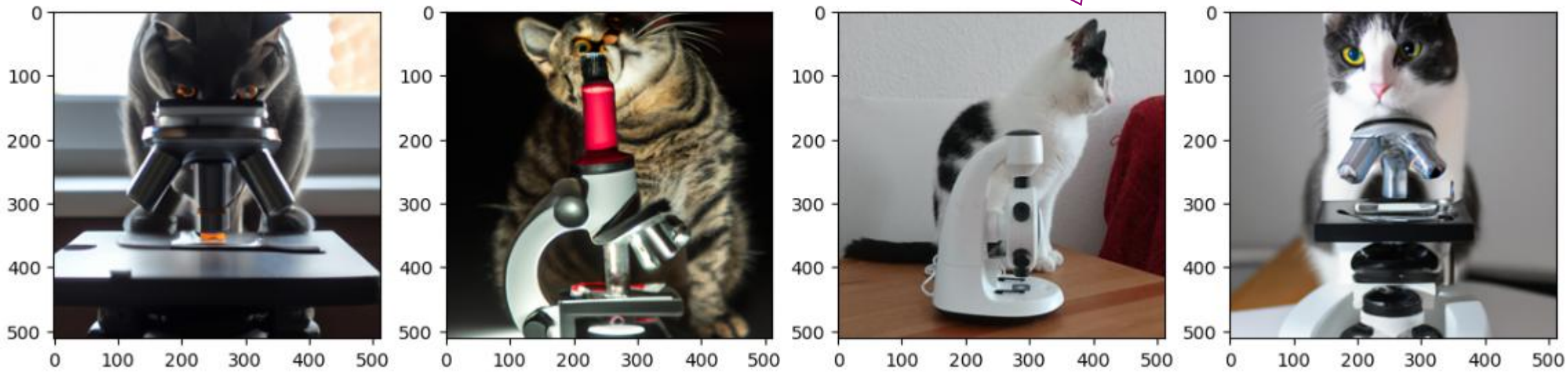
https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/07_prompt_engineering/02_generating_mri_images.html
https://twitter.com/haesleinhuepf/status/1659138282641342466

Benchmarking image generation

Prompt engineering to optimize images

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

One cat is real.

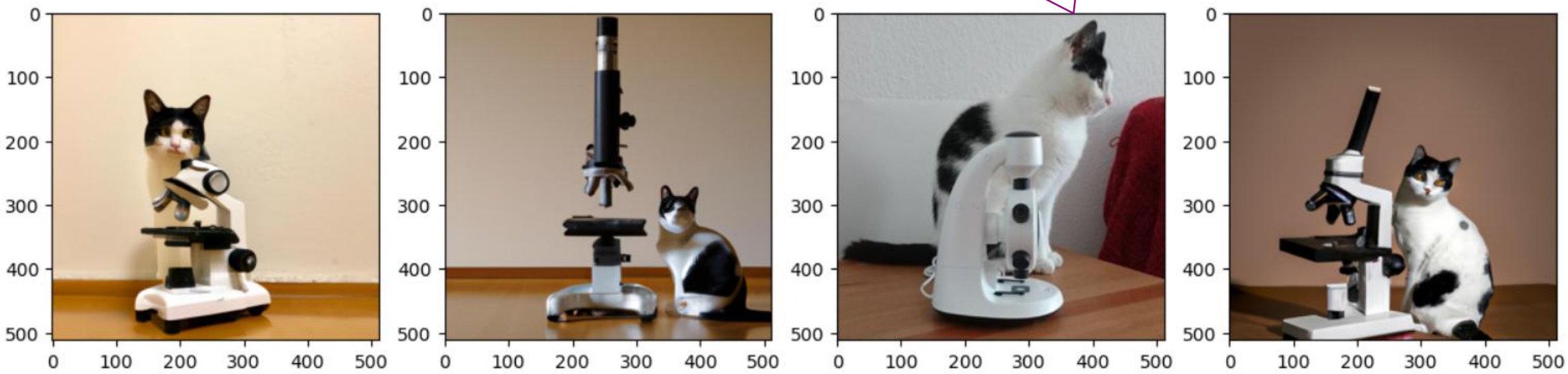


Benchmarking image generation

Prompt engineering to optimize images

[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 microsope.
"""
```

One cat is real.



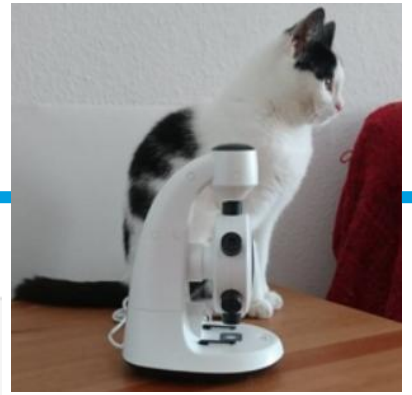
CLIP scores

Contrastive Language-Image Pre-Training (CLIP)

- For image describing

Here: Similarity between image and prompt

```python
from torchmetrics.multimodal.clip_score import CLIPScore
metric = CLIPScore(model_name_or_path="openai/clip-vit-base-patch16")
```



```python
score = metric(torch.as_tensor(image), "cat")
score.detach()
```

```
tensor(25.3473)
```

```python
score = metric(torch.as_tensor(image), "microscope")
float(score.detach())
```

```
30.786287307739258
```

CLIP scores

Example: Prompt optimization

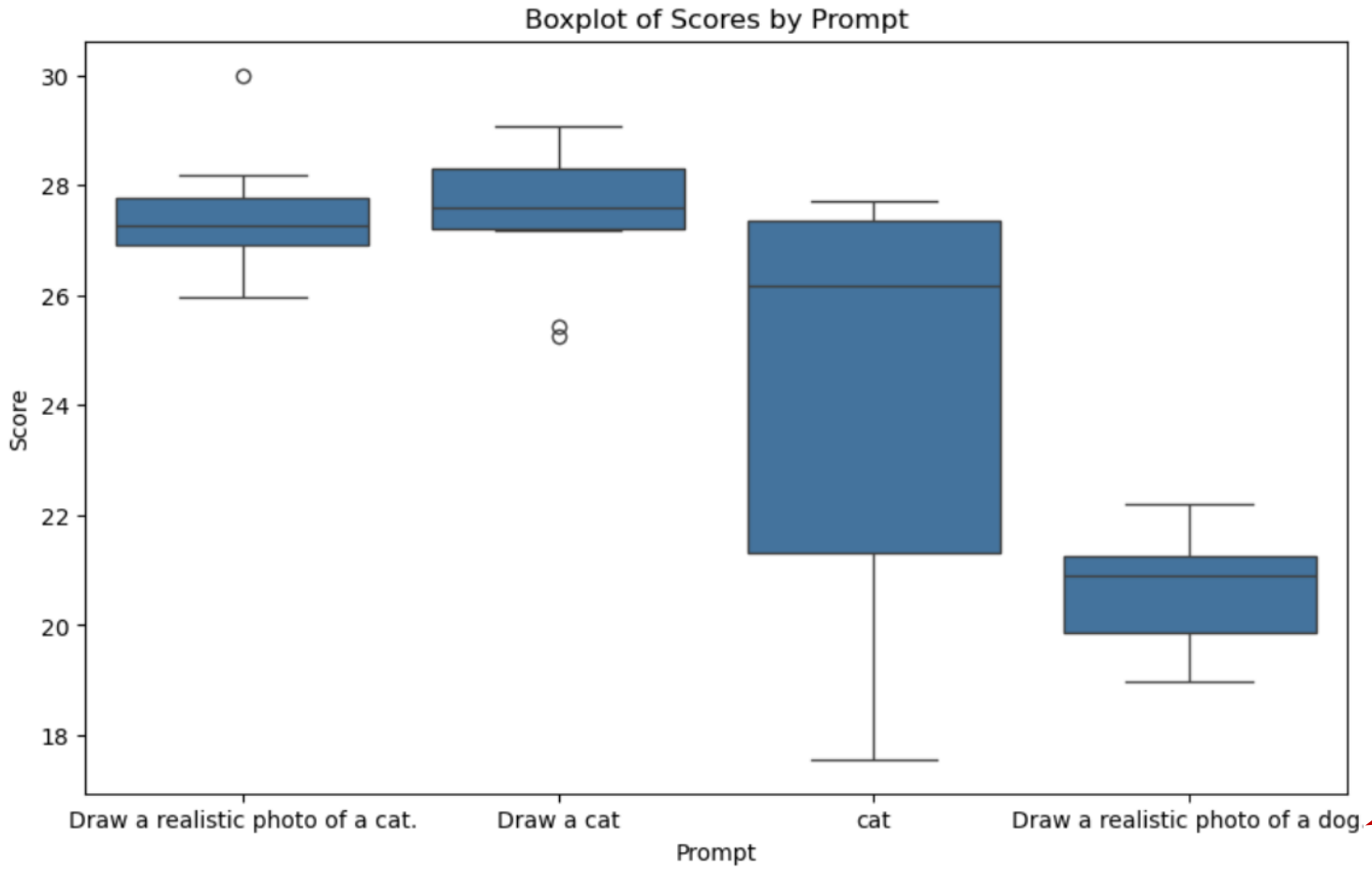
```
num_attempts = 10
prompts = ["Draw a realistic photo of a cat.",
           "Draw a cat",
           "cat",
           "Draw a realistic photo of a dog."]

data = {"prompt":[],
        "score":[]}
for prompt in prompts:
    for i in range(num_attempts):
        image = pipe(prompt, disable_tqdm=True).images[0]

        score = metric(torch.as_tensor(np.array(image)), "cat")
        data["score"].append(float(score.detach()))
        data["prompt"].append(prompt)
```

Image generation

Quality measurement

CLIP scores

Example: Prompt optimization



Always have a control experiment!

Summary

- Multi-modal models combine image, text and [...] data
- Combination of pre-existing model architectures
- APIs not standardized (yet)

Trade-off:

- LLM capacity / size limited
- Multi-modal LLMs presumably perform worse compared to specialized models.
- Example: Ask a vision-language model to write code
- Potential for mixture-of-experts and multi-agent systems

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Funded by

Bundesministerium für Bildung und Forschung

SACHSEN

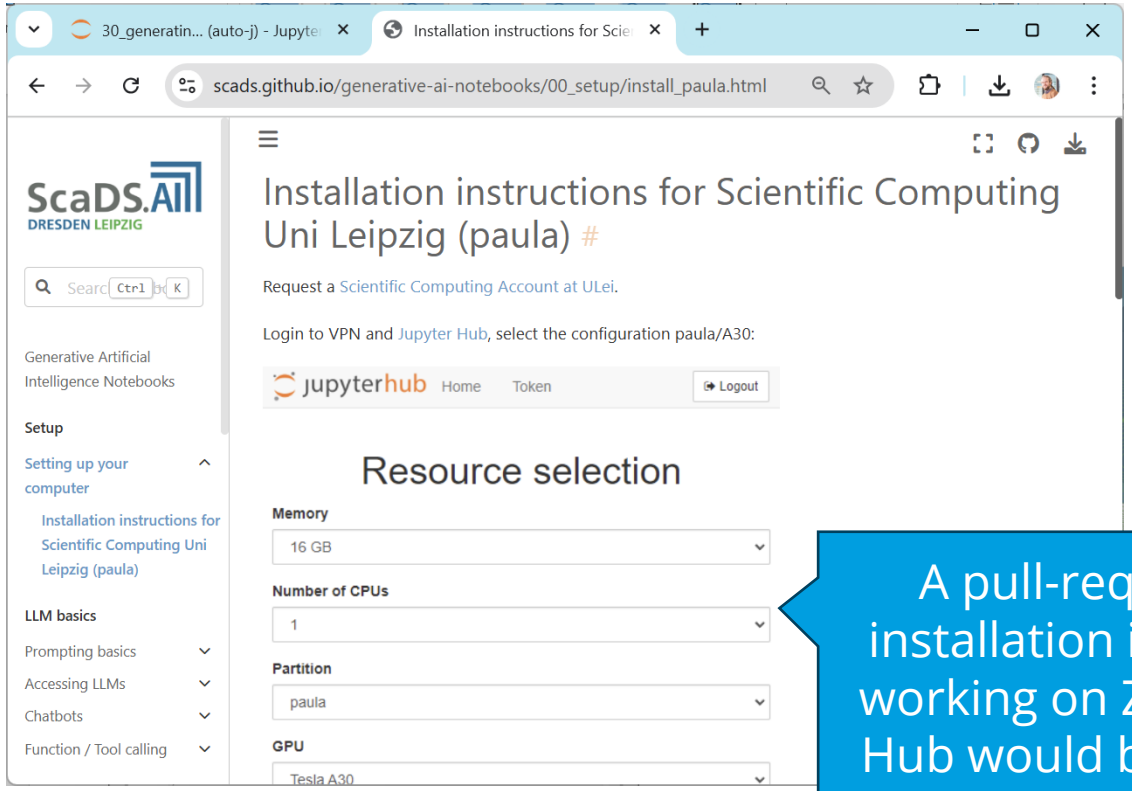
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Exercises

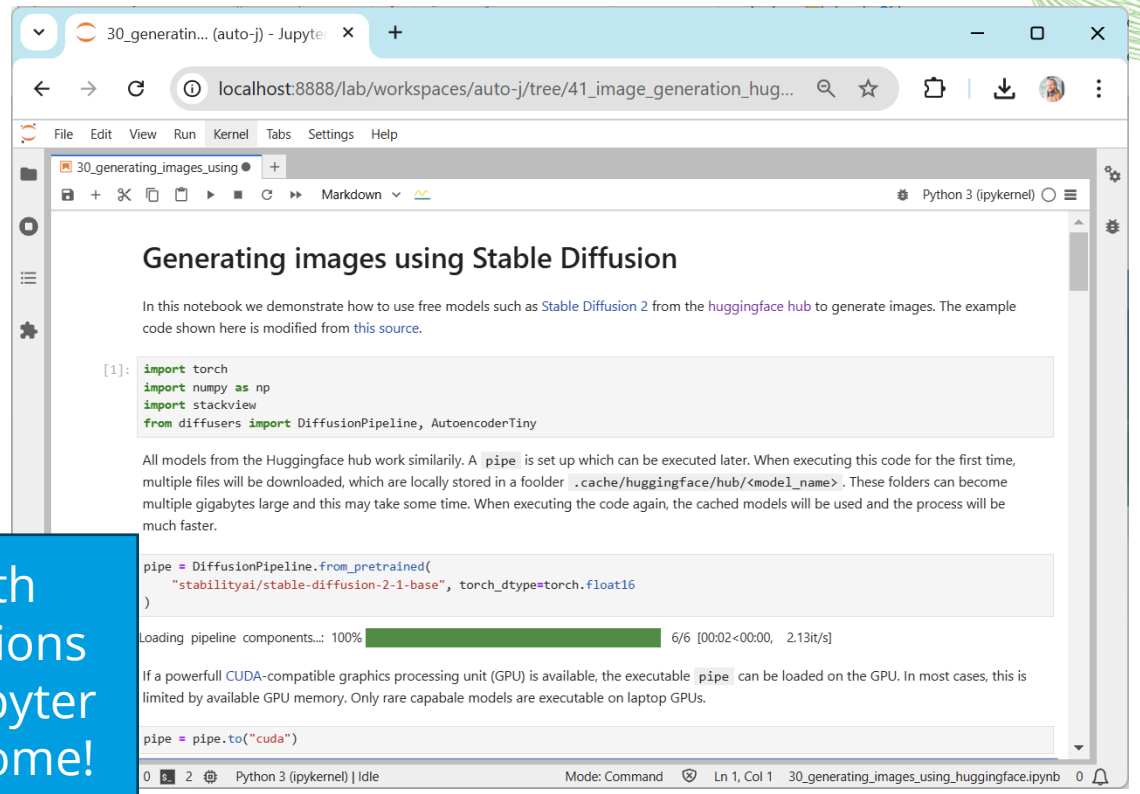
Robert Haase

Image Generation

Challenge: Install everything necessary to generate image on your PC, or on ZIH`s JupyterHub



A pull-request with installation instructions working on ZIH's Jupyter Hub would be welcome!



CLIP Scores

Improve given prompts and measure the improvement