

Vision Language Models

Robert Haase

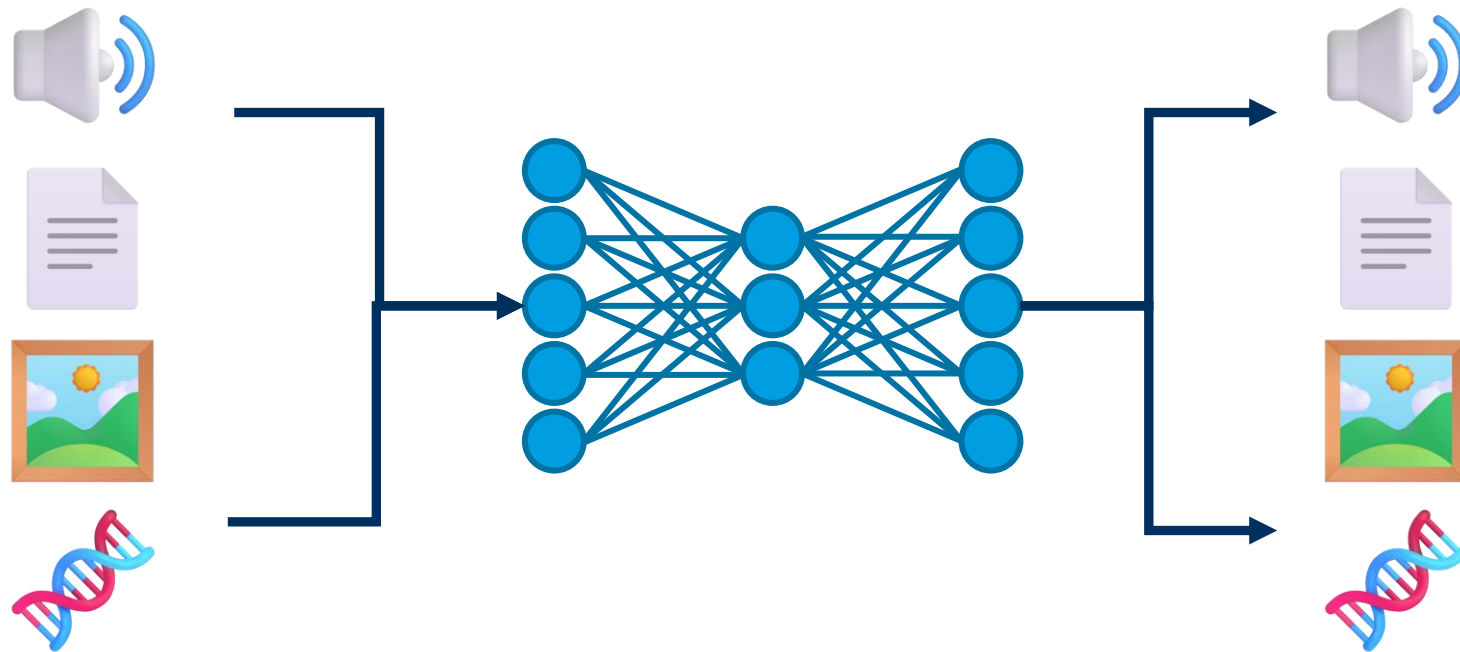


<https://doi.org/10.5281/zenodo.18219167>

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

Multi-modal LLMs

Combining image, text and [...] data



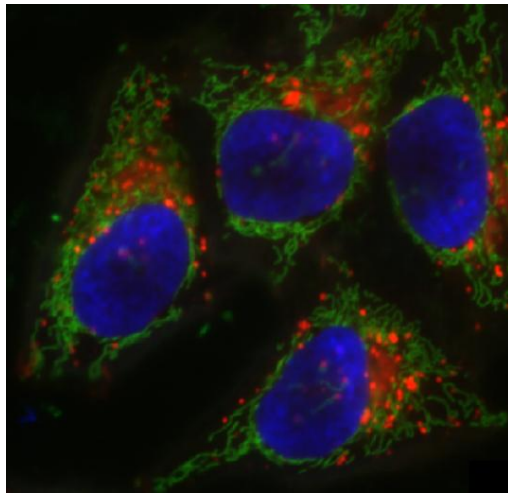
Multi-modal LLMs

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

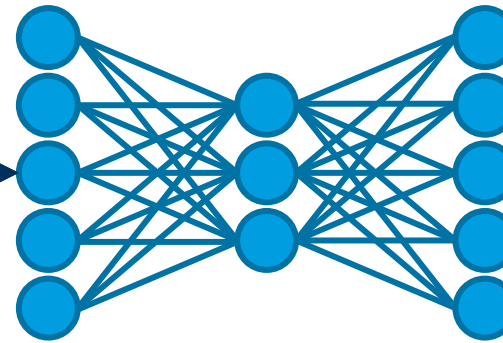
- Chat / translation / text-generation: 📄 -> 📄
 - Voice-chat / -translation: 🔊 -> 🔊
 - Text-to-speech (TTS): 📄 -> 🔊
 - Speech to text (STT): 🔊 -> 📄
 - Vision / image classification / image description / image-to-text: 🖼️ -> 📄
 - Image generation, text-to-image: 📄 -> 🖼️
 - Image variation, inpainting,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

Multi-modal LLMs

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



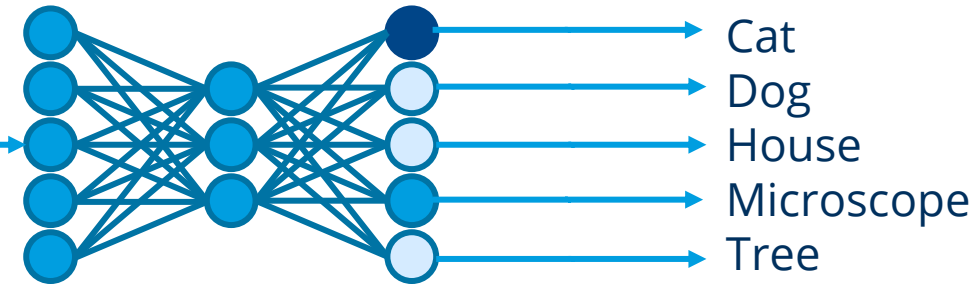
How many cells are there?



There are 4 cells.
I just marked their nuclei.

Vision Models

- Classifying images (decades old research field 🤔)
- Flexibility comes with techniques such as CLIP

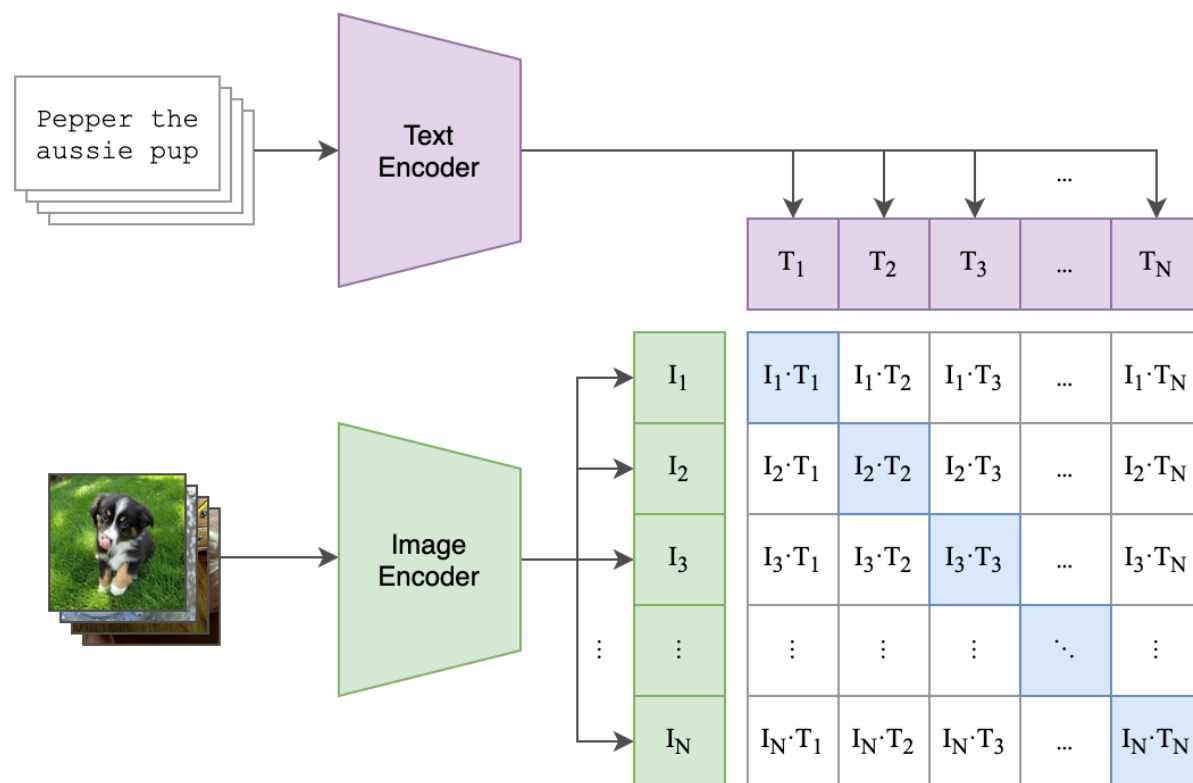


Typically a fixed number of classes, defined during training

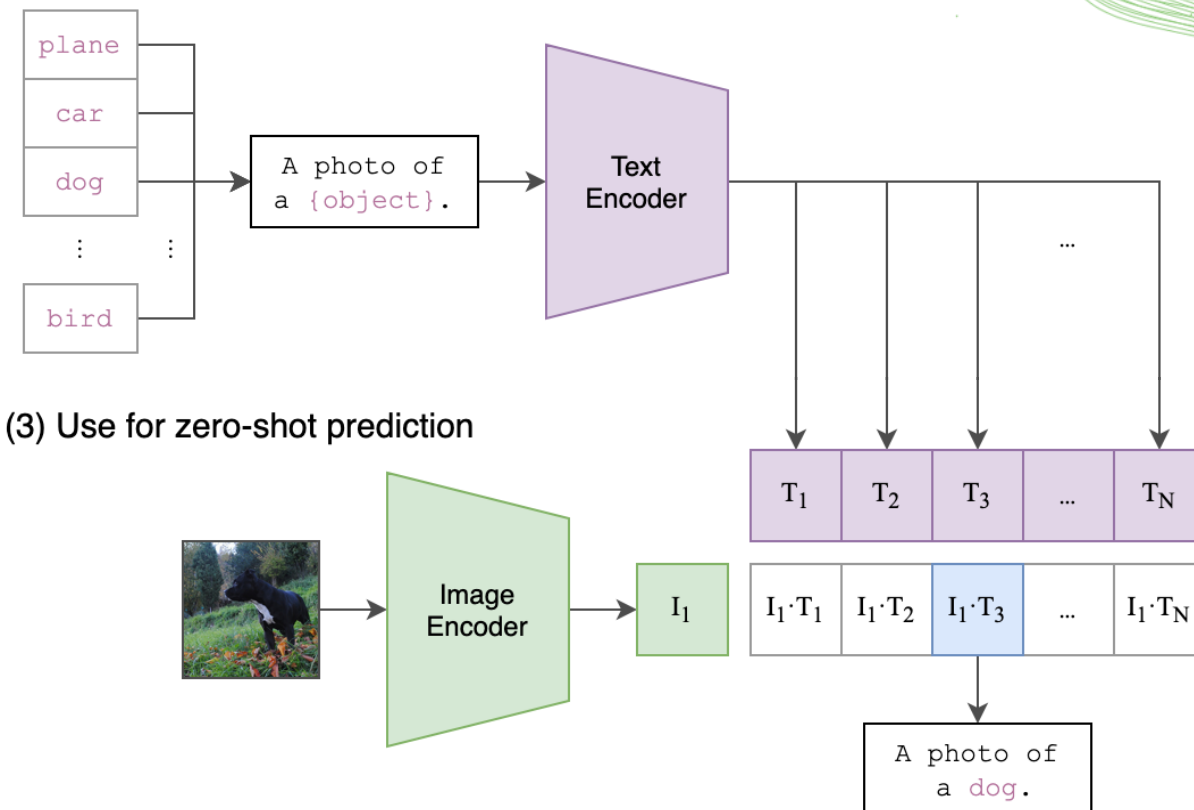
Contrastive Language-Image Pre-Training

„CLIP“ Transformers

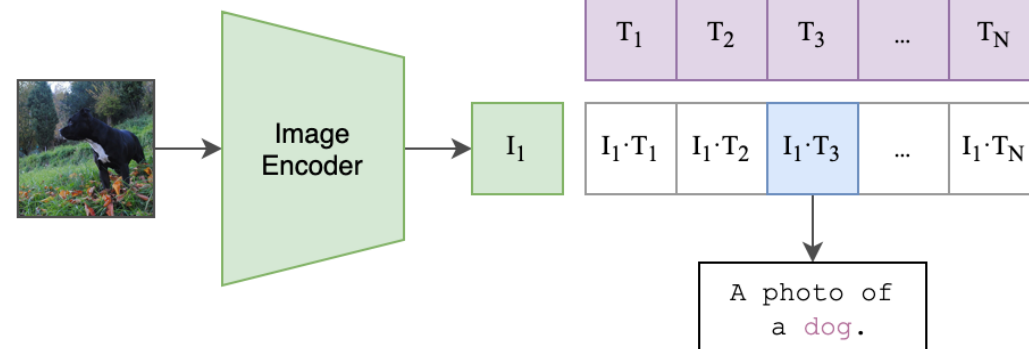
(1) Contrastive pre-training



(2) Create dataset classifier from label text

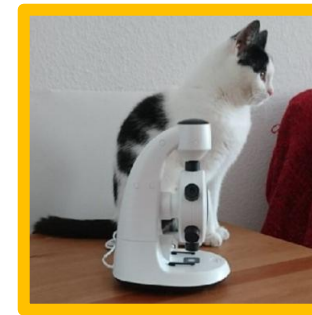


(3) Use for zero-shot prediction



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

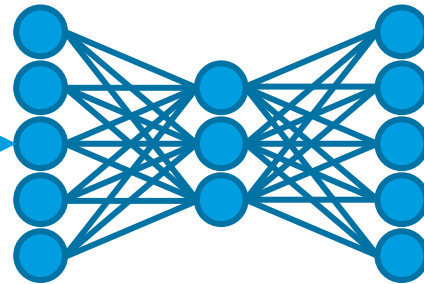
```
{'a photo of a cat': 0.9907298684120178,  
 'a photo of a dog': 0.009270114824175835}
```

label_probabilities

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

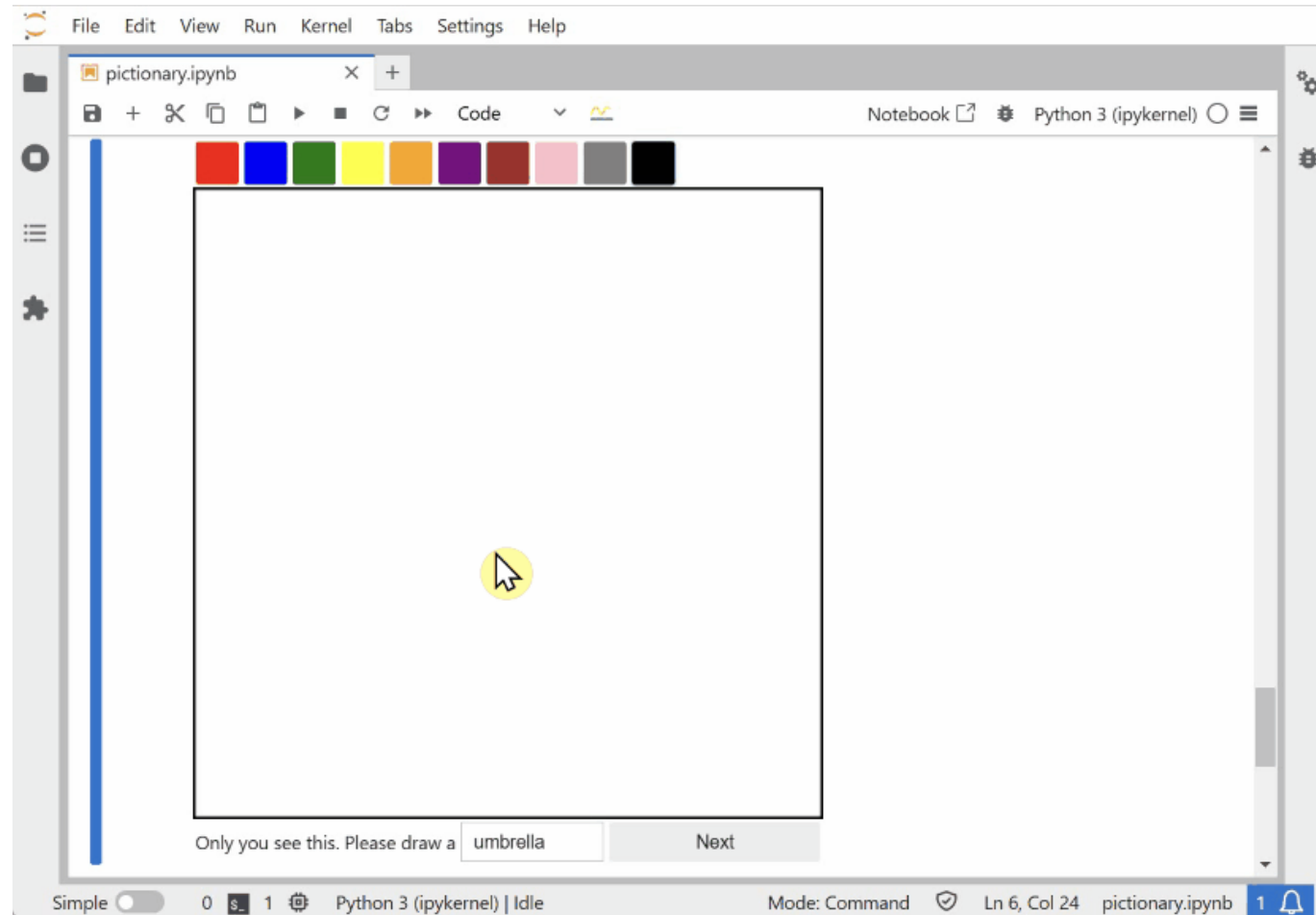

Vision Language Models

- Classifying images
- Describing images







“A picture of
a cat and a
microscope”

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...”

Prompt

Does this wing look normal or abnormal?



Clearly abnormal because...

Prompt with reference image

This is an example of a normal drosophila wing, then I will give you examples of other wings to classify as normal/abnormal.



Ok!



Clearly abnormal because...

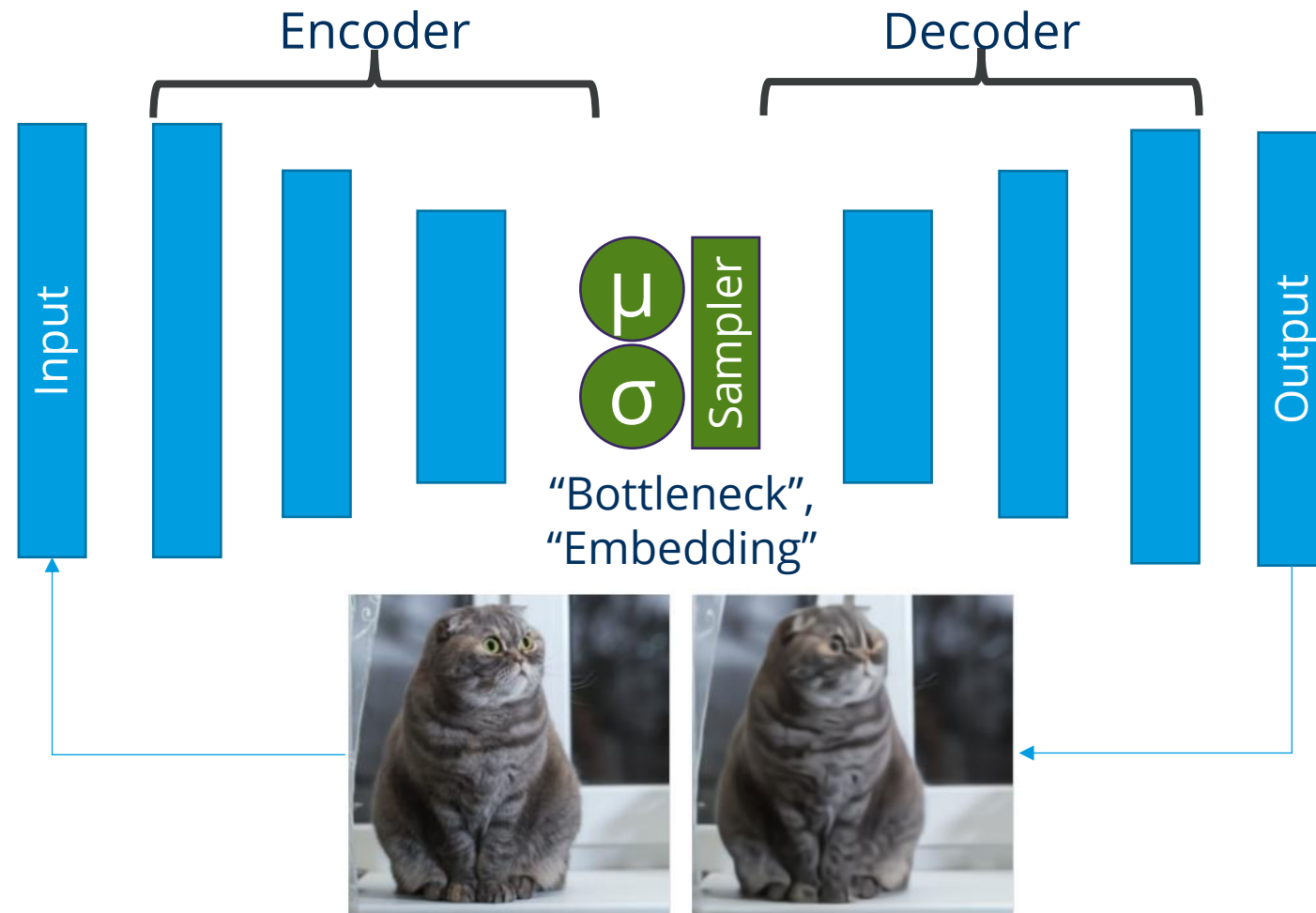


Clearly normal because...

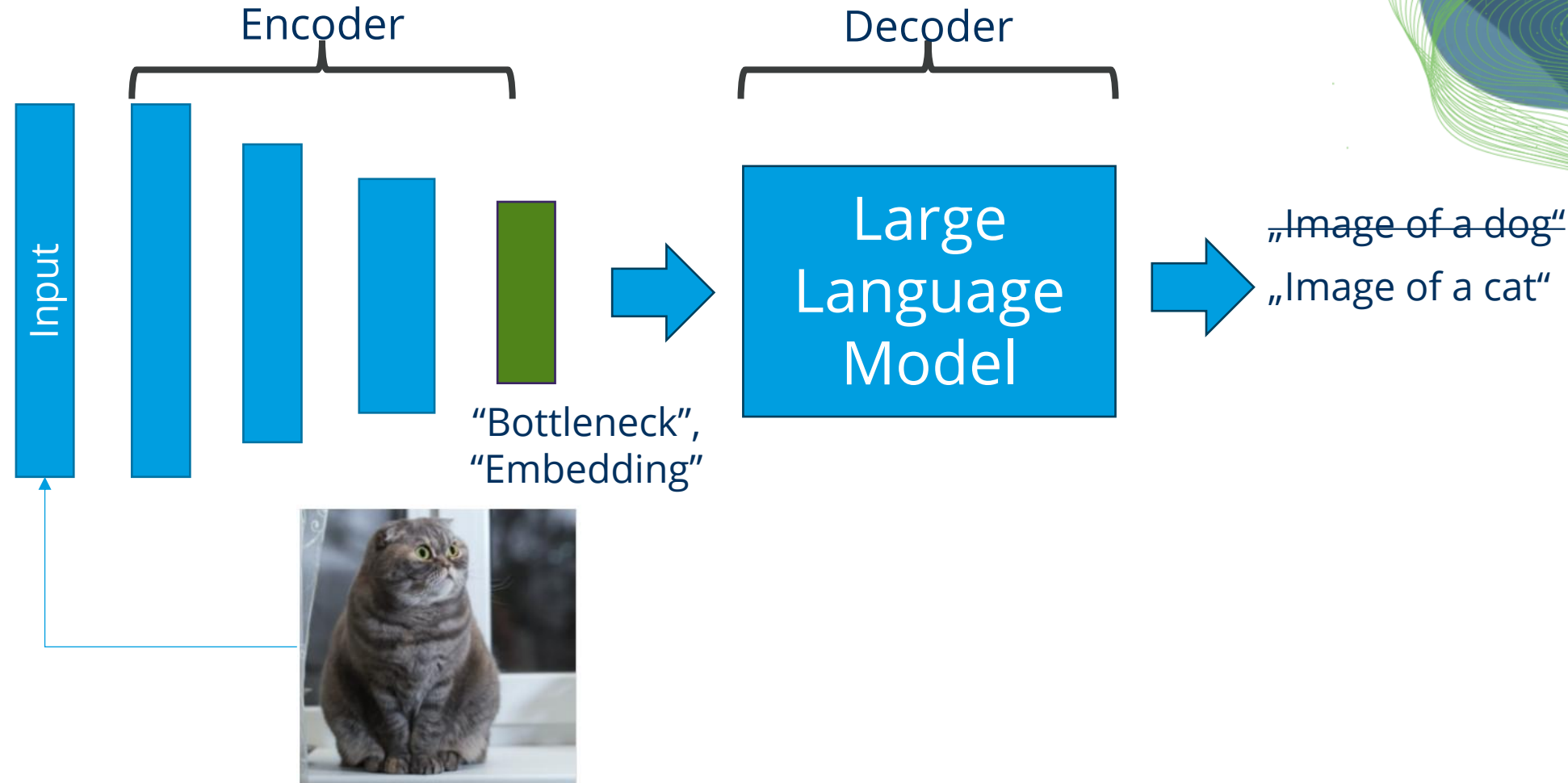
[...]

Thread

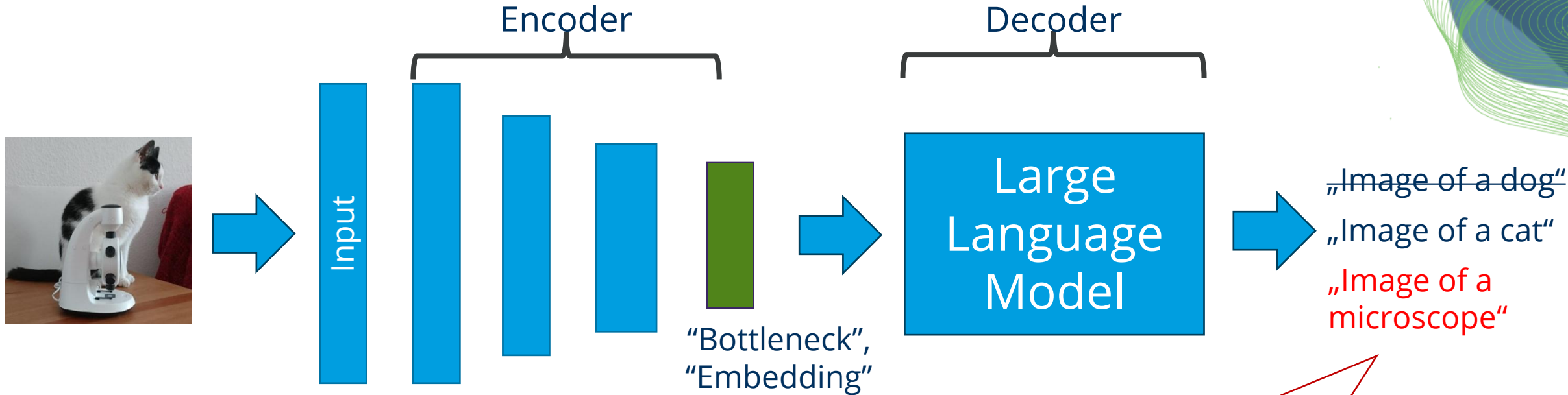
Variational Auto-Encoder



Vision Language Models (VLMs)



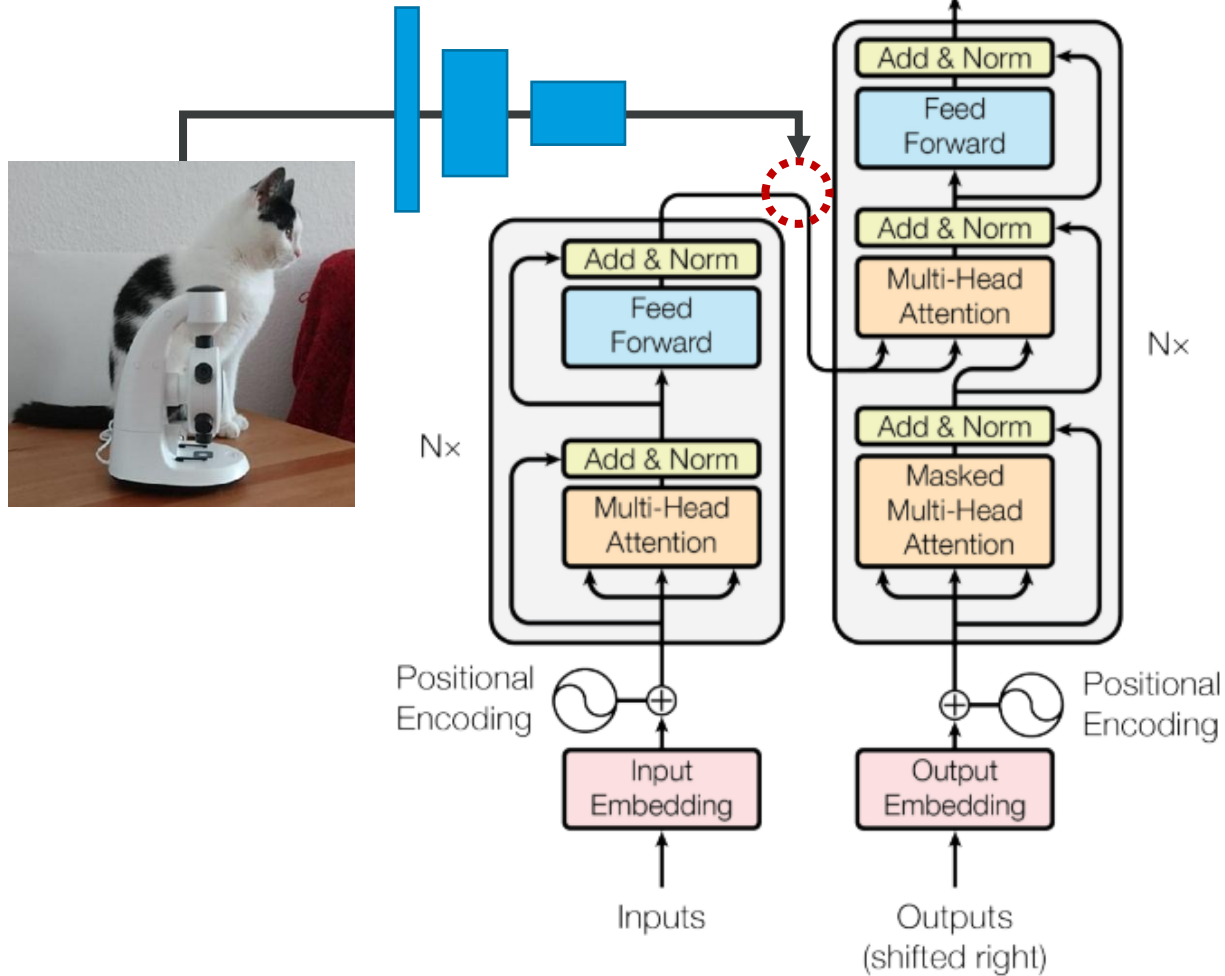
Vision Language Models (VLMs)



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

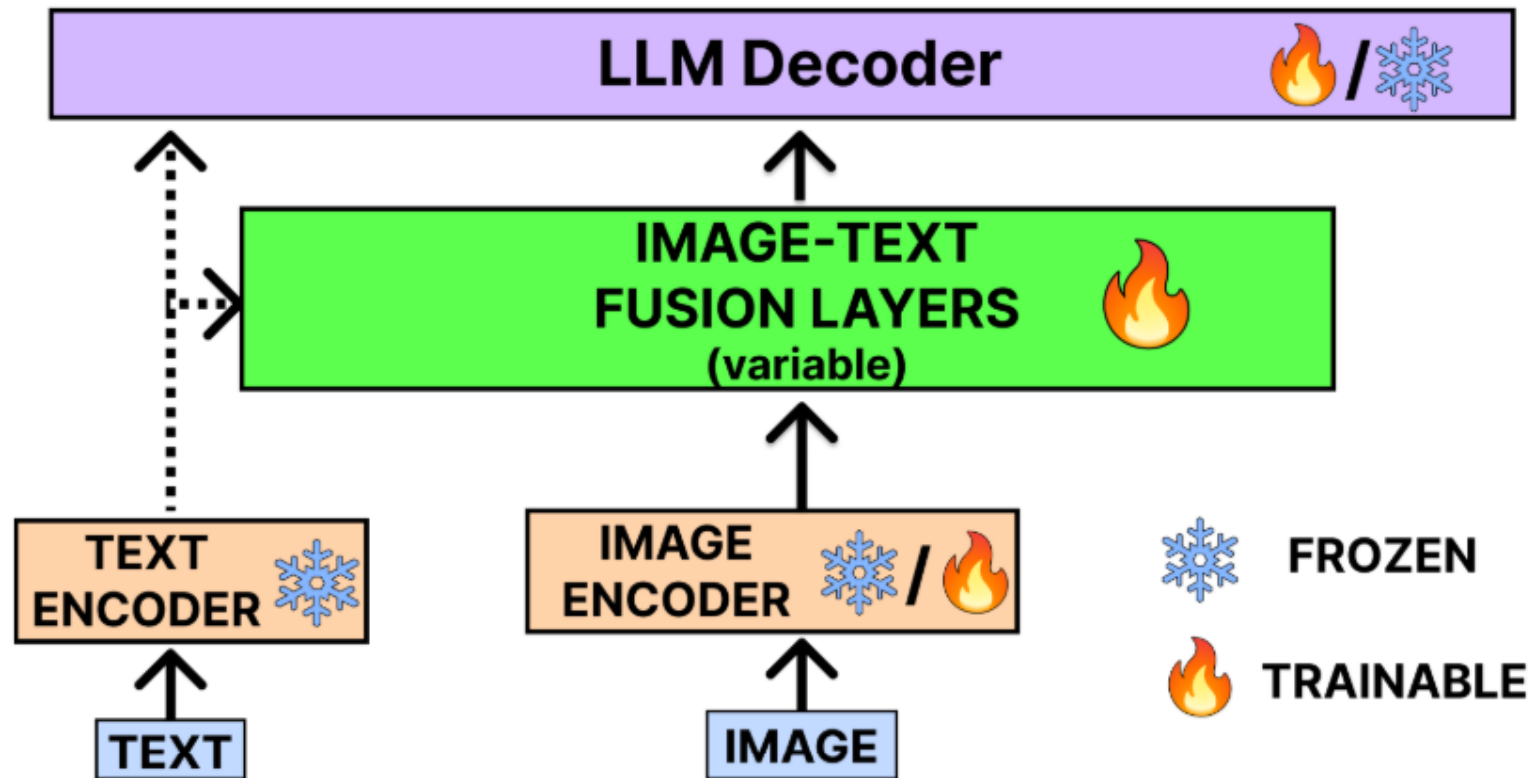
Vision Language Models (VLMs)

Combination of
Image-Encoders with
Text-Decoders



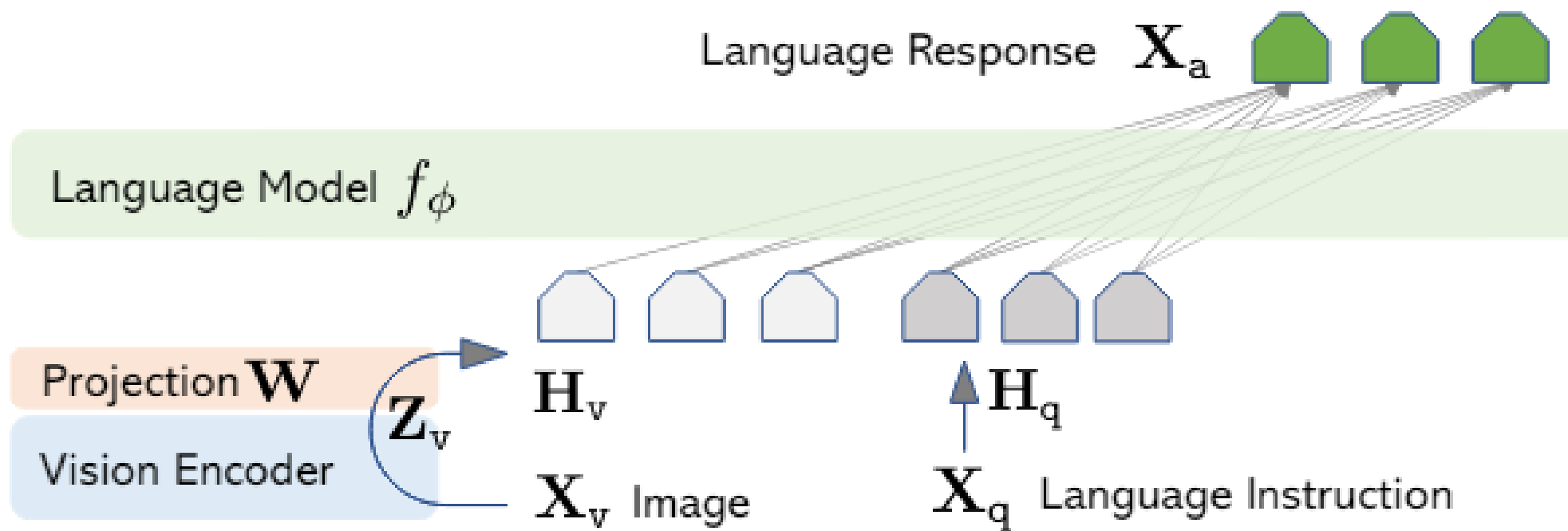
Vision Language Models (VLMs)

Combination of Image-Encoders with Text-Decoders



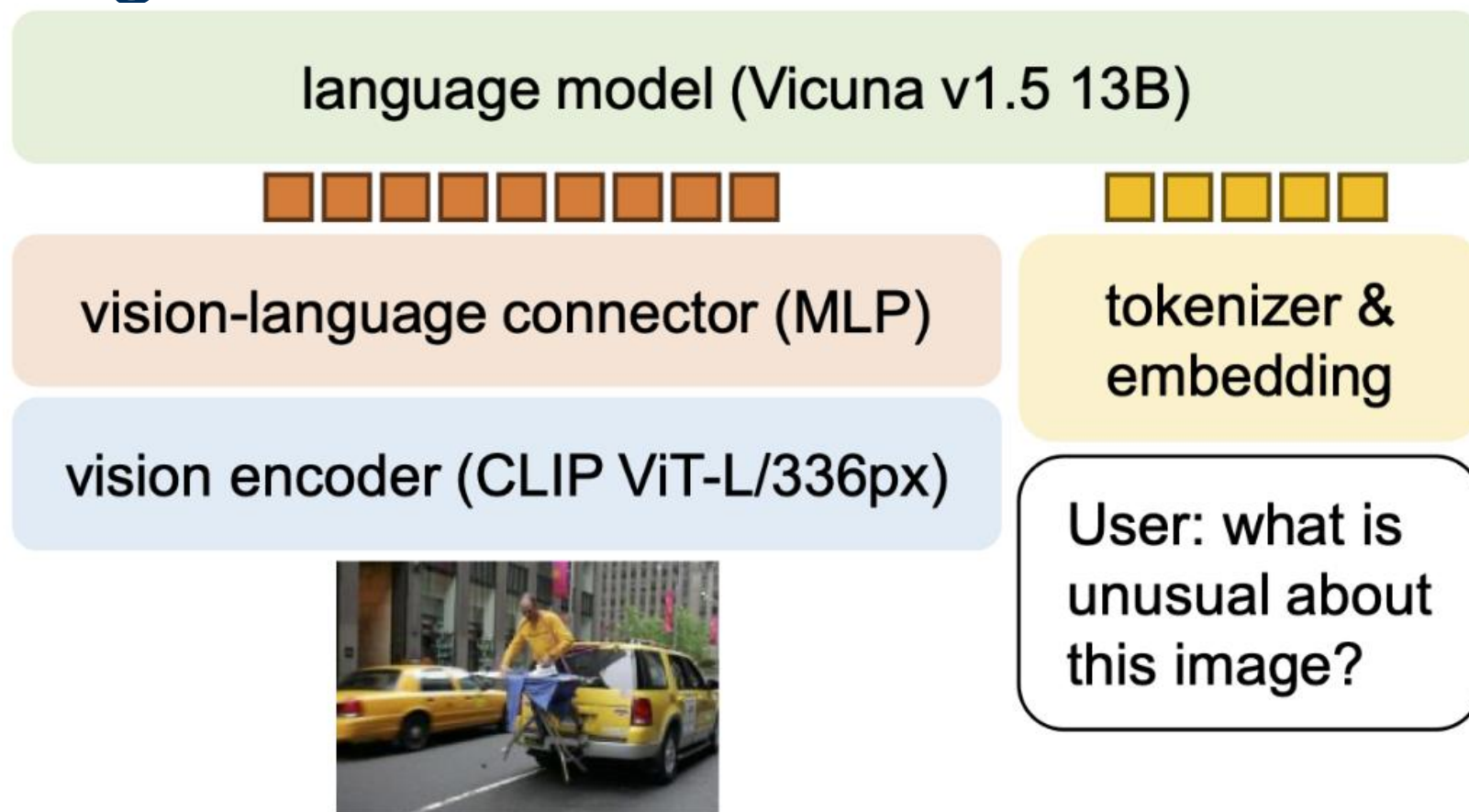
LLAVA

Large Language and Vision Assistant



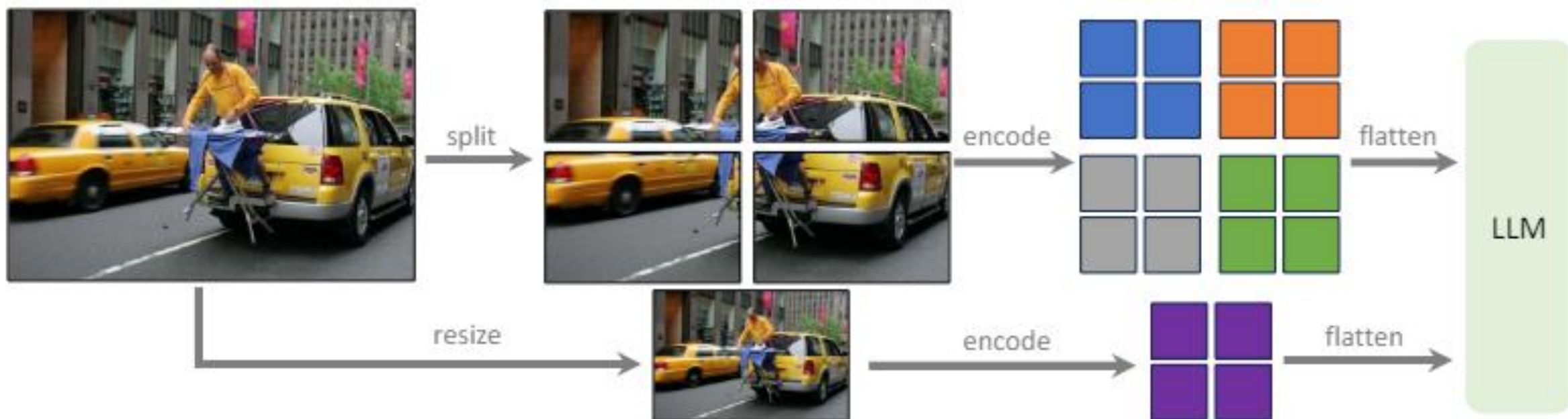
LLAVA 1.5

Combining LLAVA with CLIP



LLAVA 1.5 HD

Giving the model multiple perspectives on the same scene



Accessing VLMs using Python

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

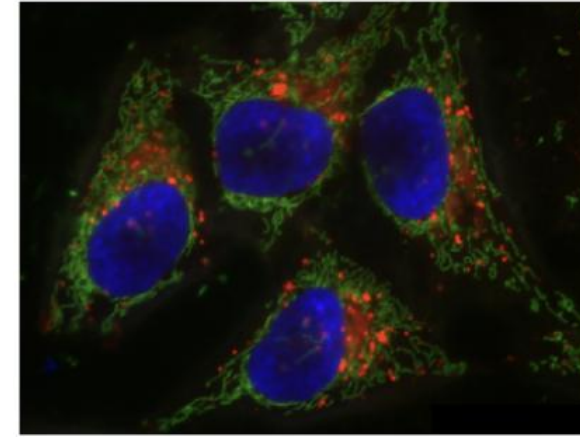
```
def prompt_qwen(prompt:str, image, model="Qwen/Qwen2-VL-7B-Instruct"):
    """A prompt helper function that sends a message to the server
    and returns only the text response.
    """
    rgb_image = _img_to_rgb(image)
    byte_stream = numpy_to_bytestream(rgb_image)
    base64_image = base64.b64encode(byte_stream).decode('utf-8')

    message = [{"role": "user", "content": [
        {"type": "text", "text": prompt},
        {
            "type": "image_url",
            "image_url": {
                "url": f"data:image/png;base64,{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
```



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

display(Markdown(res))
```

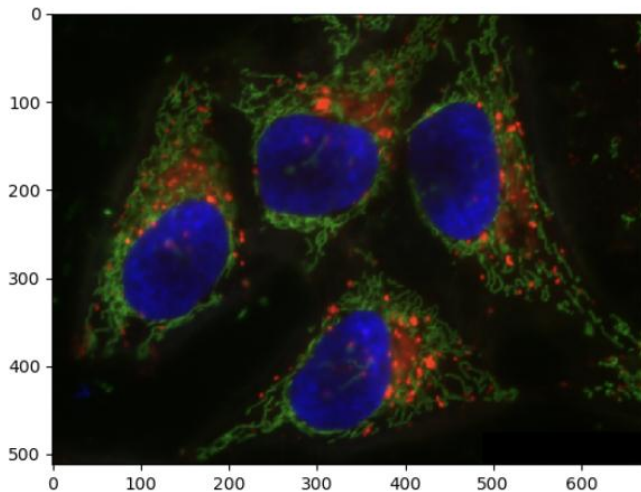
The image appears to be a microscopic view of biological cells, possibly analyzed using fluorescence microscopy techniques. The cells are stained to highlight different structures:

- Blue Areas:** These represent the nuclei of the cells, typically stained with a dye like DAPI or Hoechst that binds to DNA.
- 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.
- 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.

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:
* Nuclei
* 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 structures for the three channels in the format shown above.
""", hela)

Markdown(result)
```

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

Accessing VLMs using Python: No common API

```
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

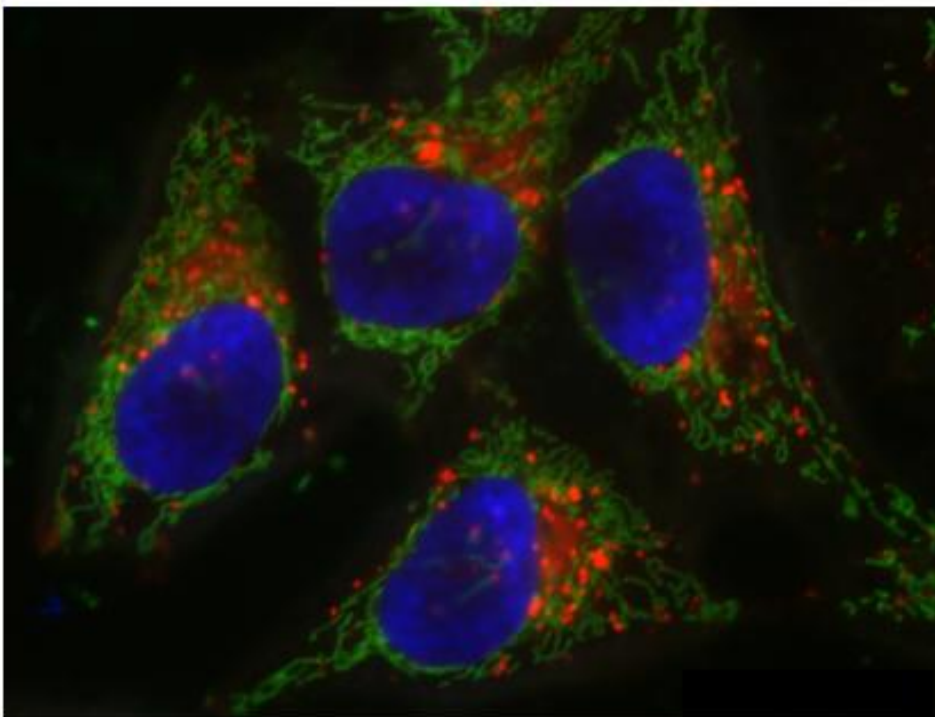
```
if image is not None:
    response = client.generate_content([image, request])
else:
    response = client.generate_content(request)
```

Google / Gemini

Benchmarking vision models

Single attempts... are a trap

You



How many blue nuclei are in this image?



ChatGPT

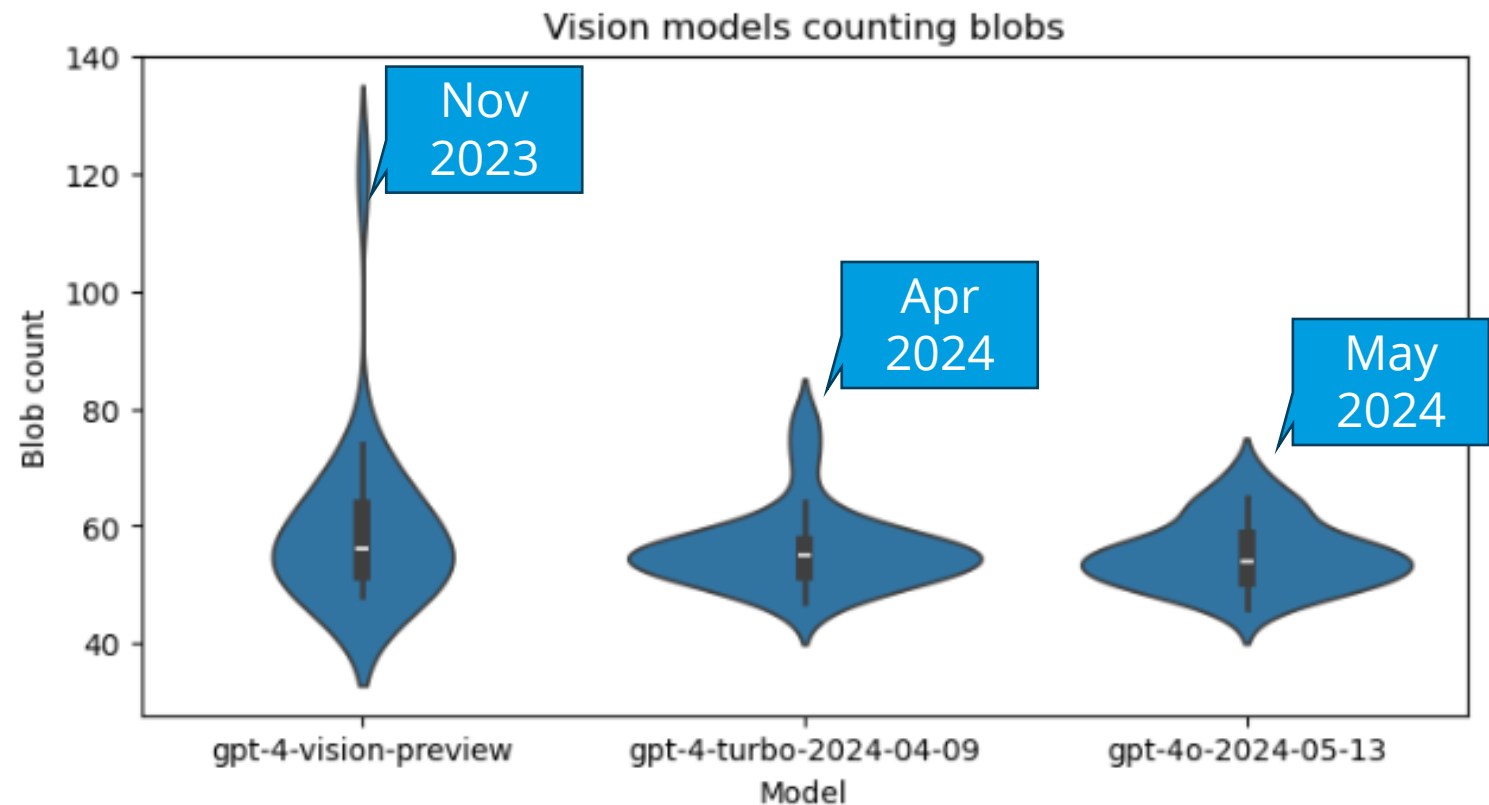
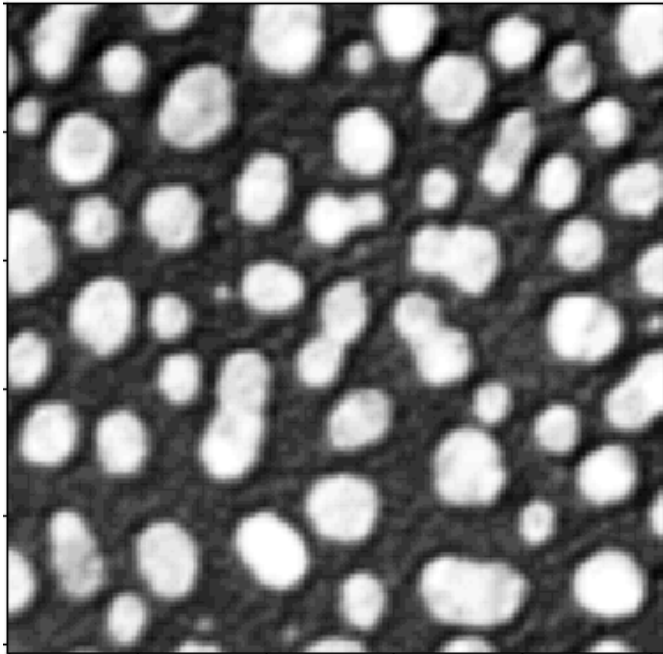
There are three blue nuclei visible in this image.



$n=1$

Vision language models for counting objects

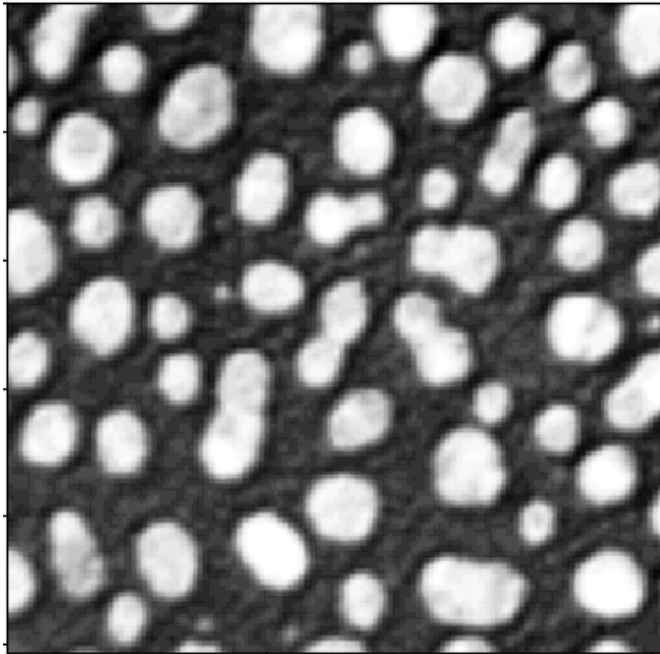
Prompt: „Analyse the following image by counting the bright blobs. Respond with the number only. “ (n=25)



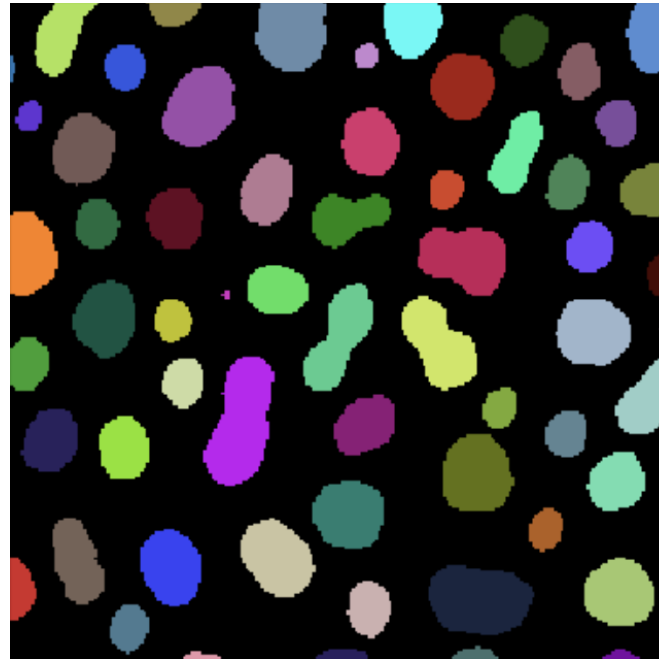
Comparison: Otsu-Thresholding

Classical approach: Otsu-Threshold (1979) + Connected component labelling + excluding objects on edges..

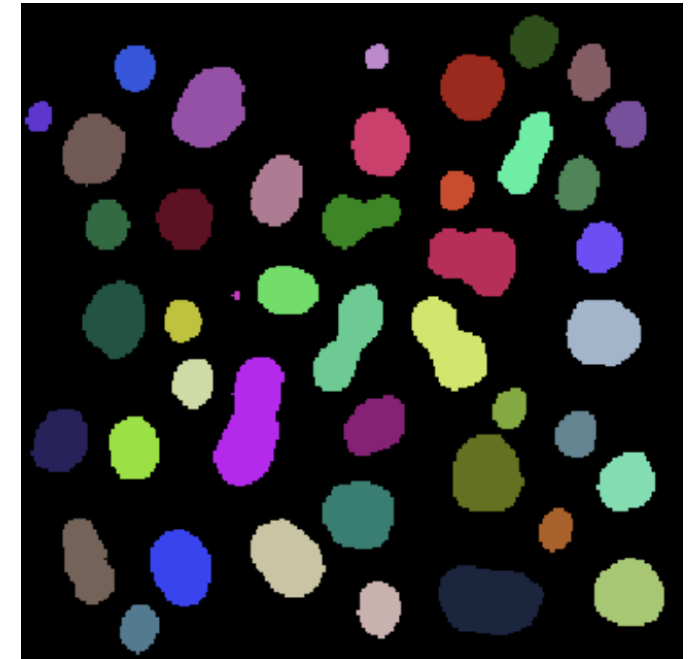
Original image



Raw segmentation (62 objects)



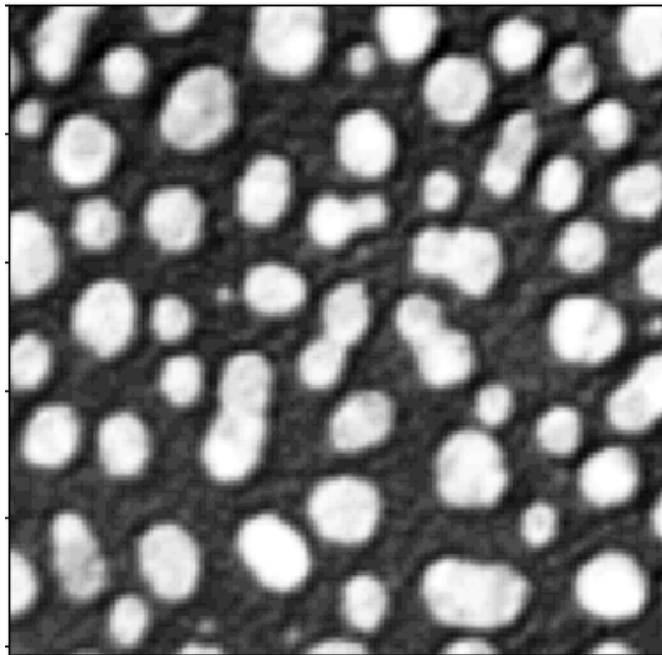
Excluding image border (44 objects)



Vision language models for counting objects

Experiment reproduced June 24th 2025

Lesson learned: We need physical control of models and infrastructure to secure reproducibility!

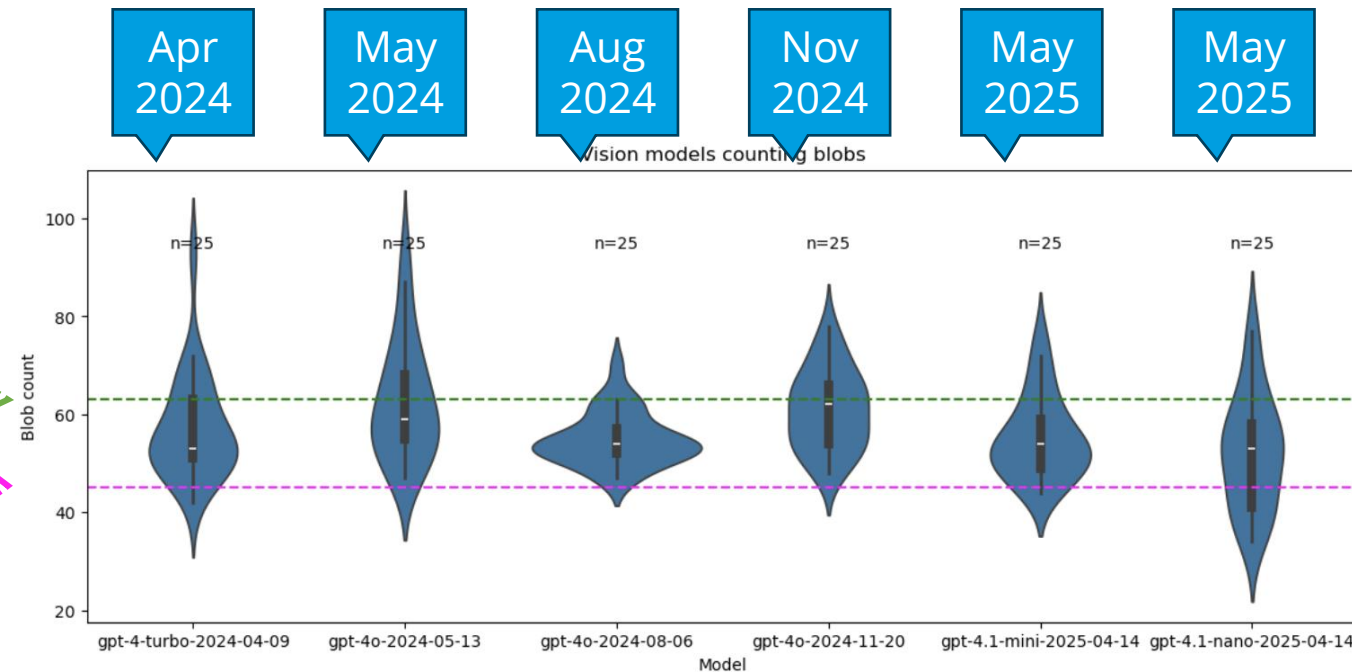


Nov
2023

Model no
longer
available
(Apr. 2025)

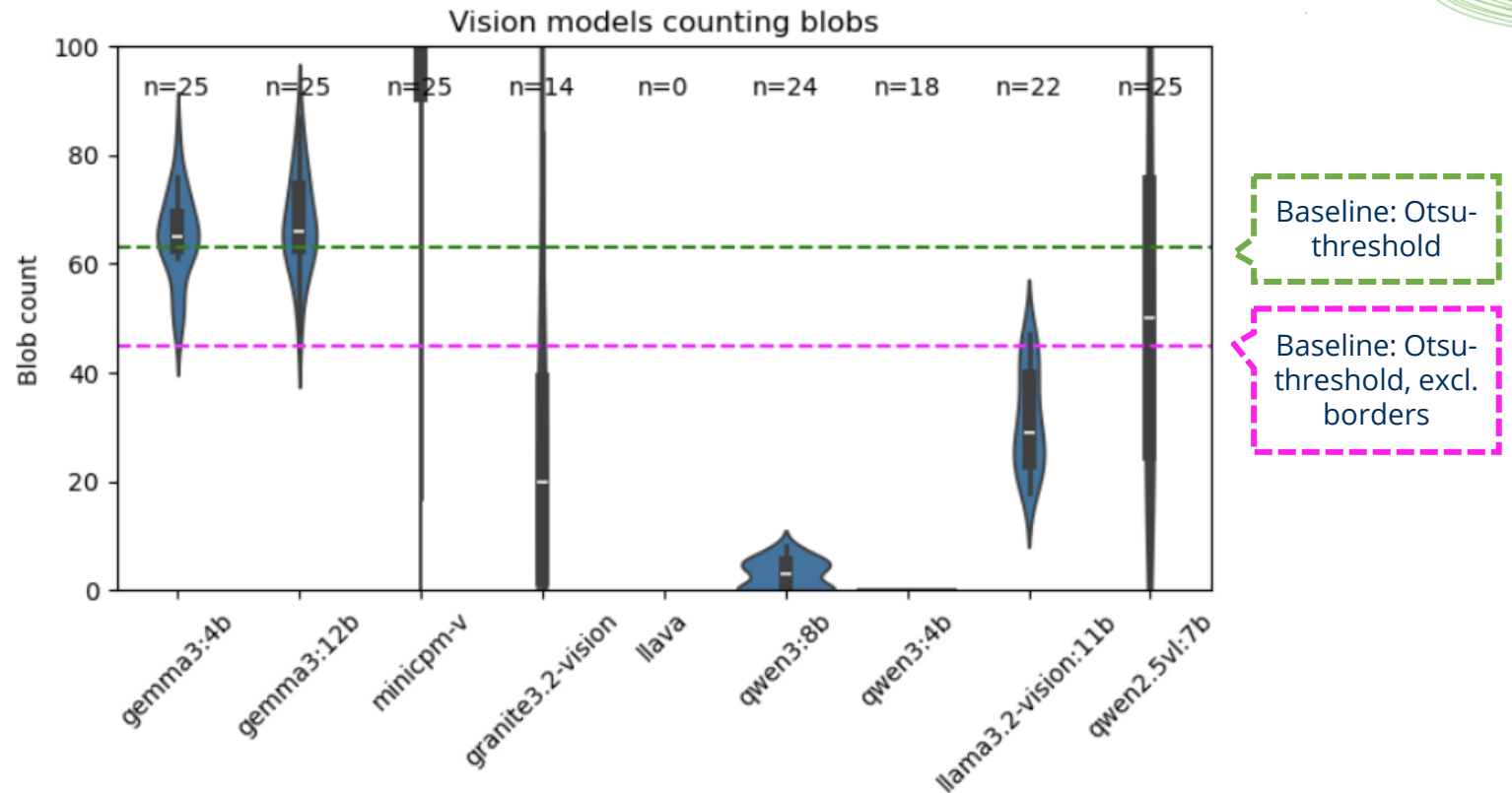
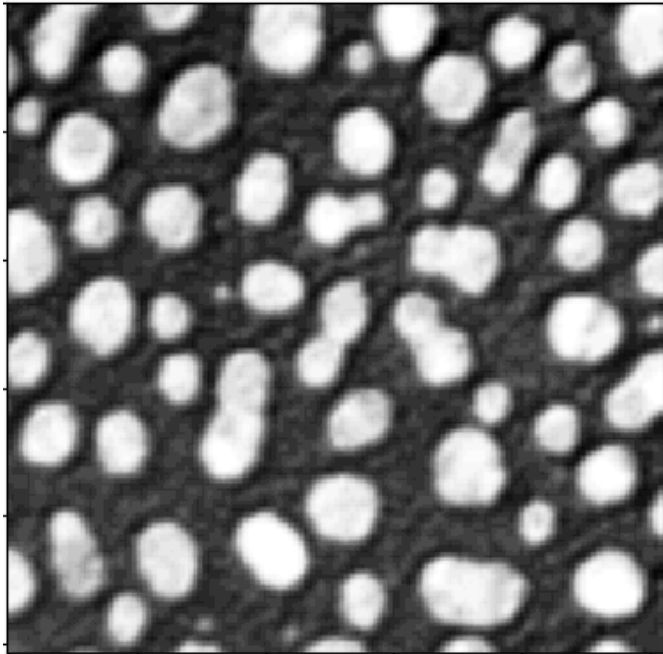
Baseline: Otsu-
threshold

Baseline: Otsu-
threshold, excl.
borders



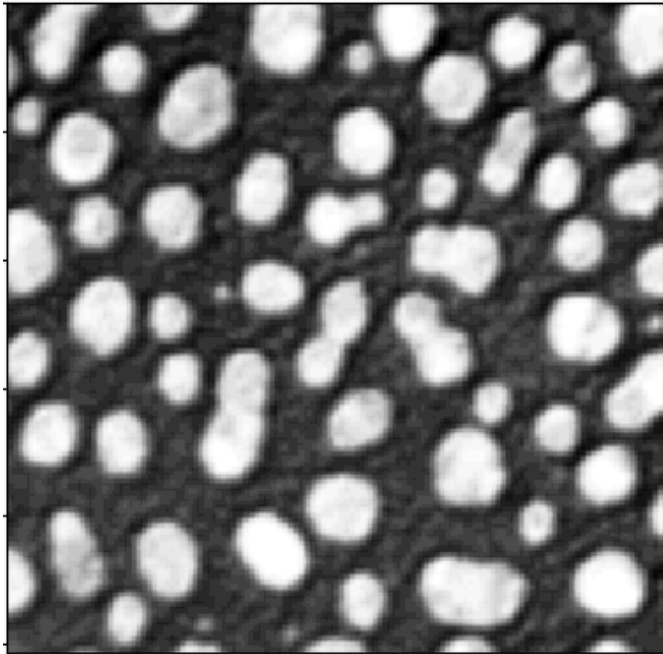
Vision language models for counting objects

Now also possible using open-weight models (via ollama)



Vision language models for counting objects

Prompt-engineering also works with VLMs



Baseline: Otsu-threshold

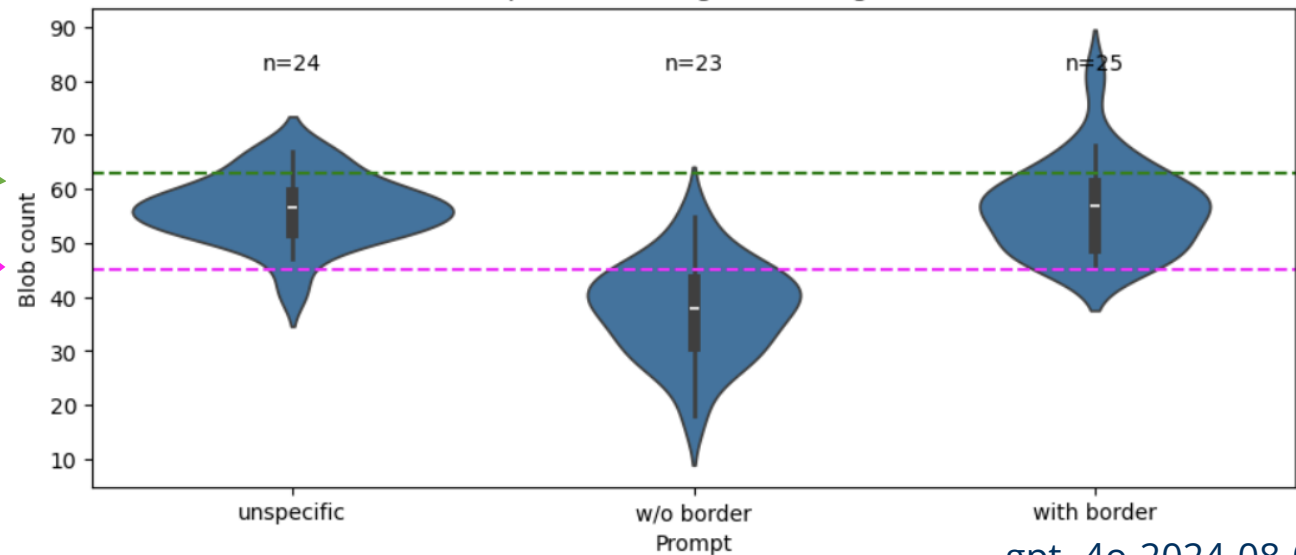
Baseline: Otsu-threshold, excl. borders

"Analyse the following image by counting the bright blobs. Respond ONLY the number."

"Analyse the following image by counting the bright blobs. **Ignore the objects touching the image border.** Respond ONLY the number."

"Analyse the following image by counting the bright blobs, **including the objects touching the image border.** Respond ONLY the number."

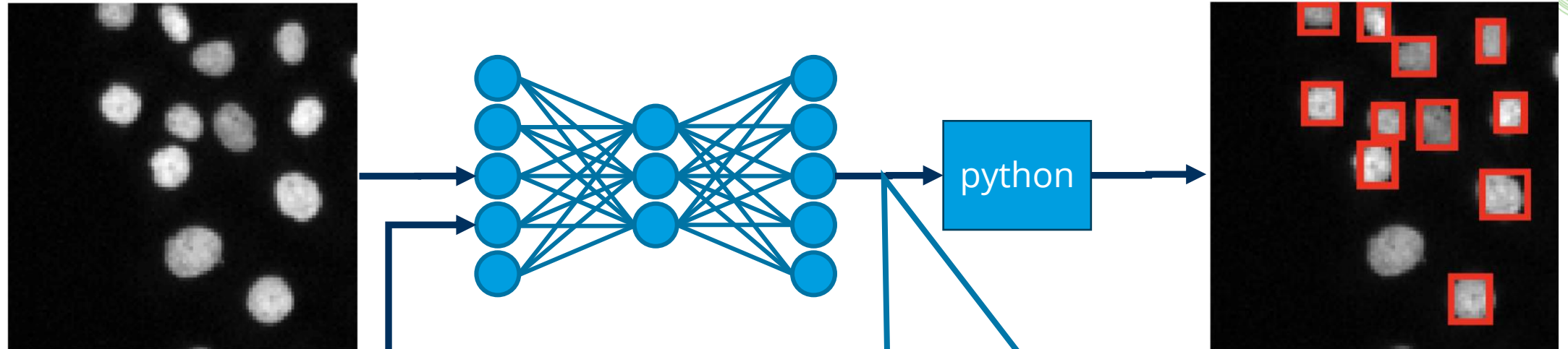
Prompts for counting blobs using VLMs



gpt -4o-2024-08-06

VLMs for bounding box segmentation

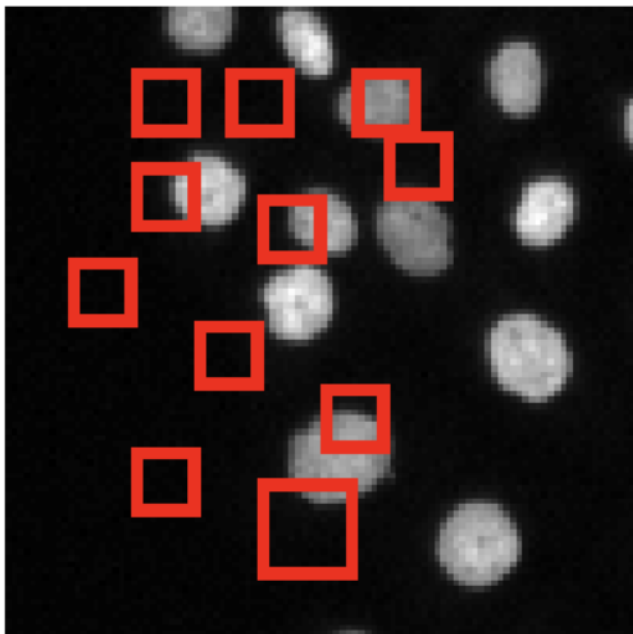
Recap: VLMs combine images + text to produce text



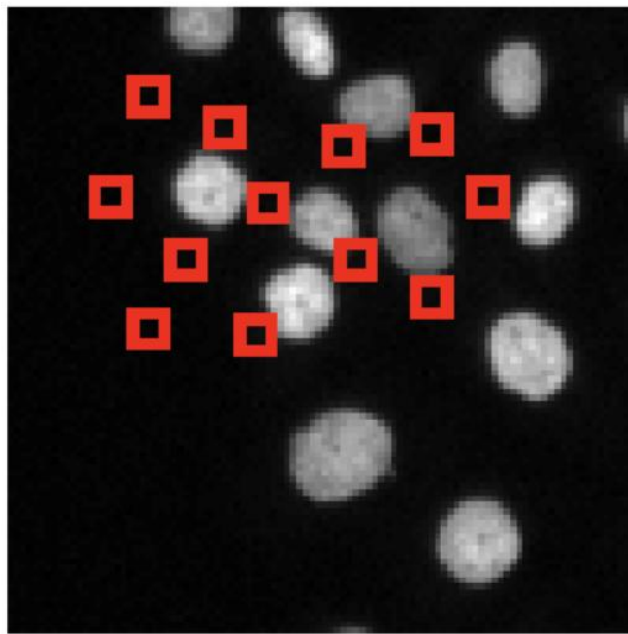
Give me a json object
of bounding boxes
around ALL bright
blobs in this image...

```
[{"x": 0.191, "y": 0.111,...},  
{"x": 0.313, "y": 0.161,...},...]
```

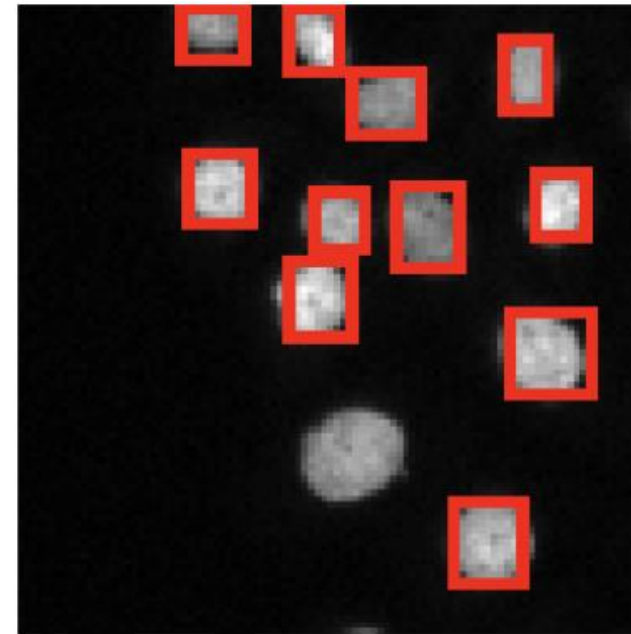
VLMs for bounding box segmentation



GPT-4o



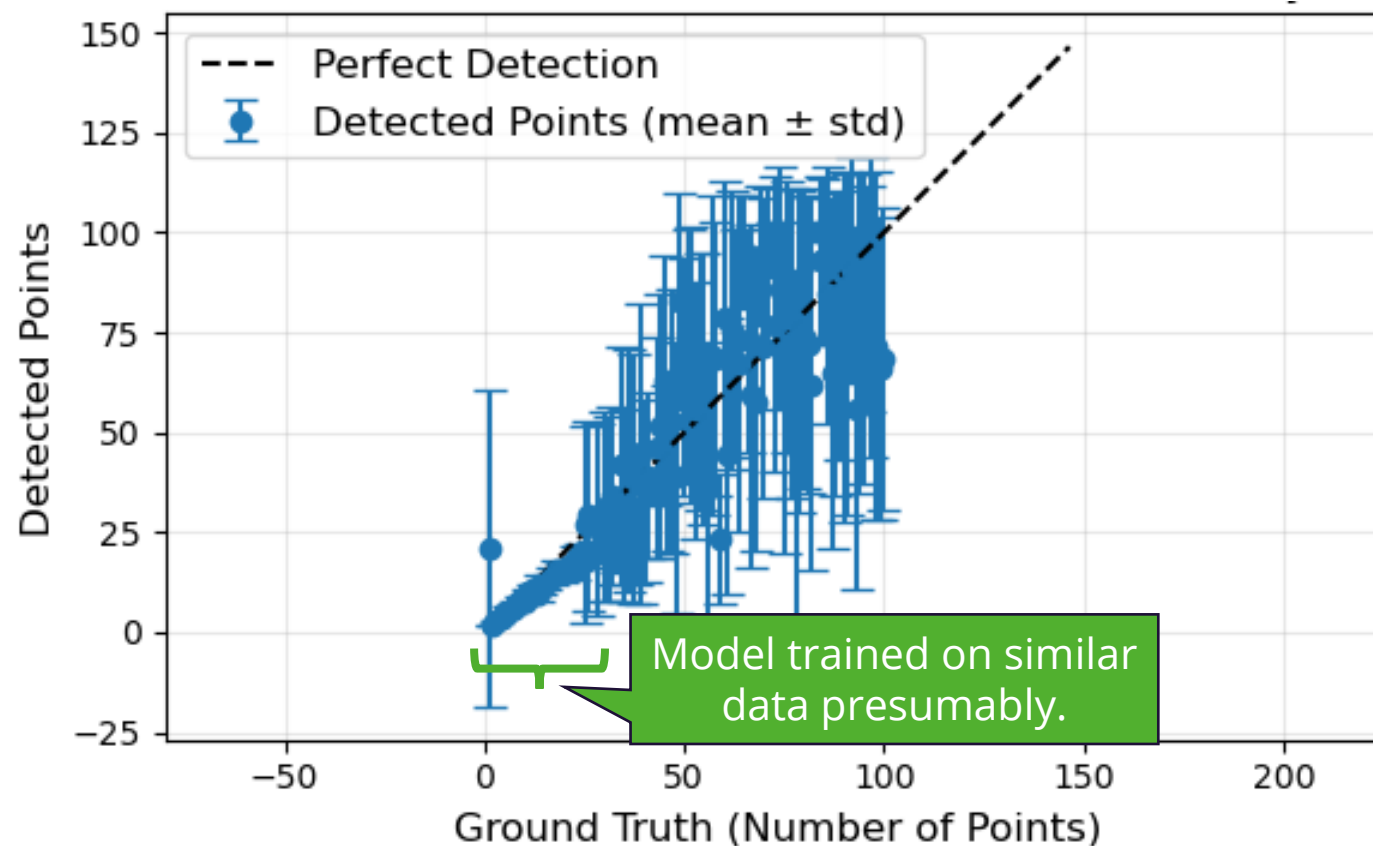
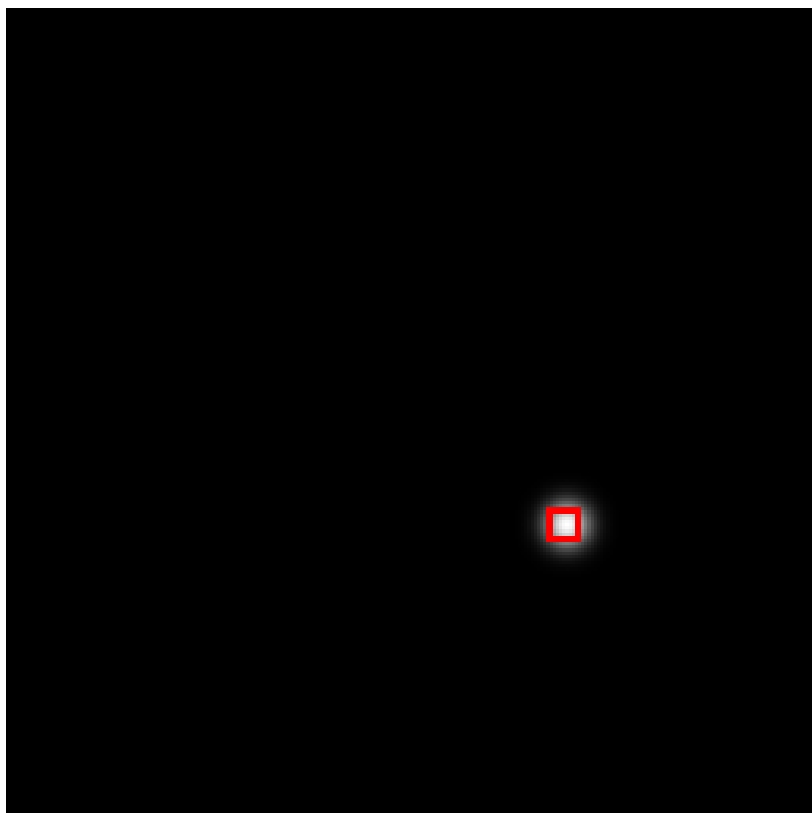
Claude 3.7 Sonnet



Moondream 2B

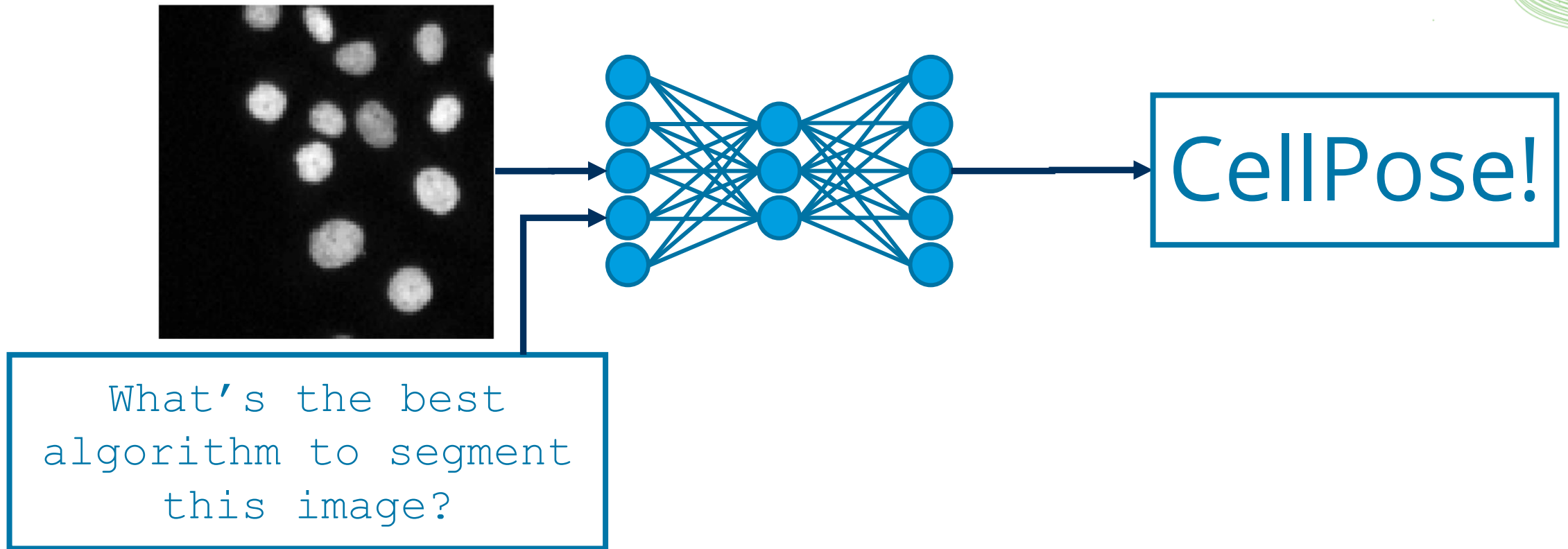
Limits of VLMs

Example: Moondream 2B applied to simulated data, just counting bounding boxes



VLMs *guessing* segmentation algorithms

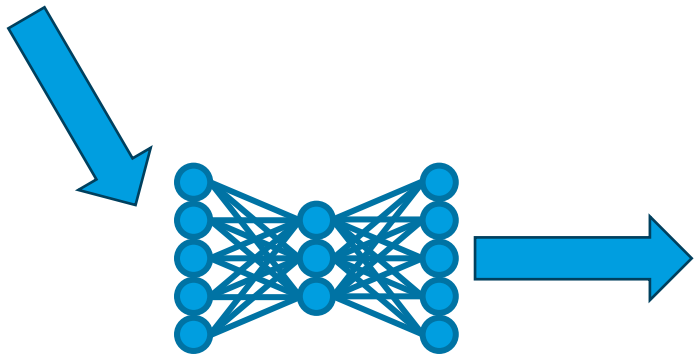
Combine image + text to generate text



LLMs *guessing* segmentation algorithms

```
prompt = """You are a bioimage-analysis expert.  
What is the best image processing algorithm to segment microscopy images?  
Answer the algorithm name only. No explanations.  
"""
```

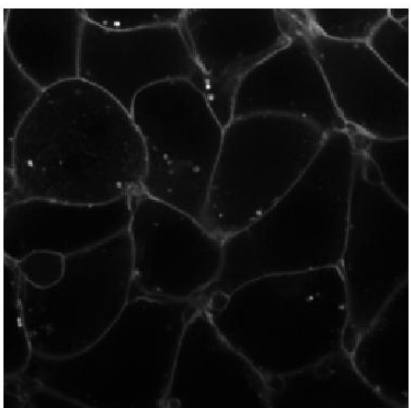
+ no
image



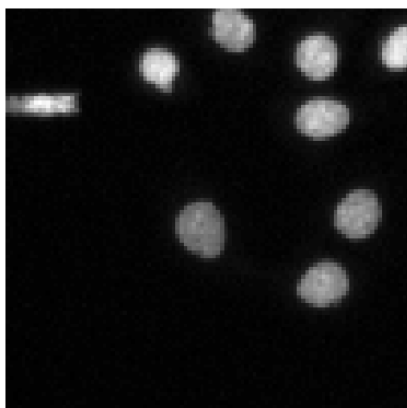
learning
cellpose
deep g
deeplabunet
e

VLMs guessing segmentation algorithms

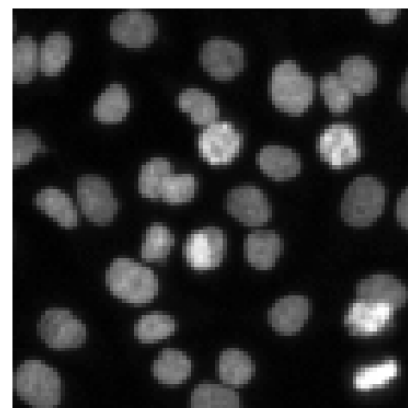
```
prompt = """You are a bioimage-analysis expert.  
What is the best image processing algorithm to segment this microscopy image?  
Answer the algorithm name only. No explanations.  
"""
```



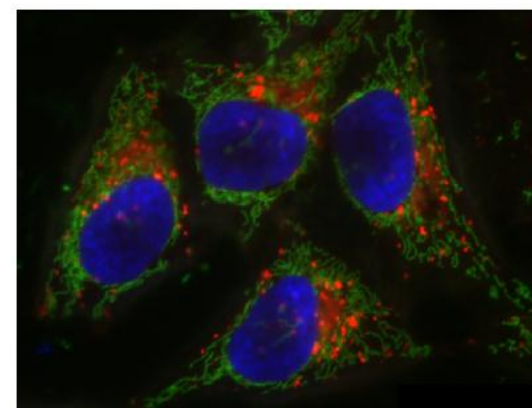
watershed



otsu thresholding
watershed



watershed
otsu thresholding



otsu thresholding
learning deep u
cellpose
e watershed
g

VLMs guessing segmentation algorithms

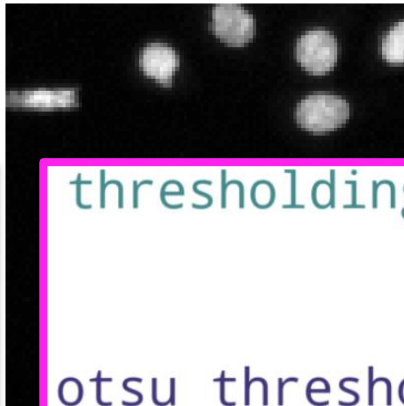
```
prompt = """You are a bioimage-analysis expert. You have a rule-book what alogrithms to use for specific images.
```

```
## Rules
```

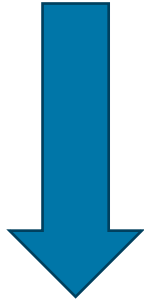
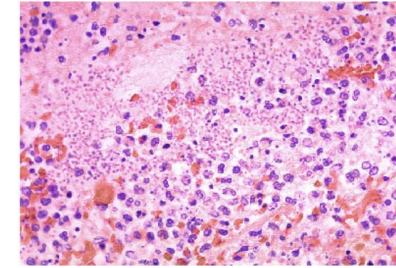
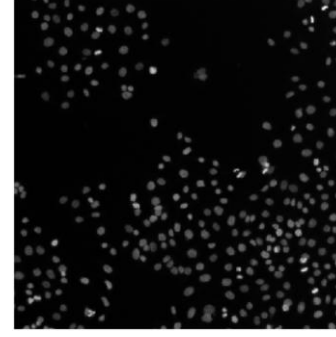
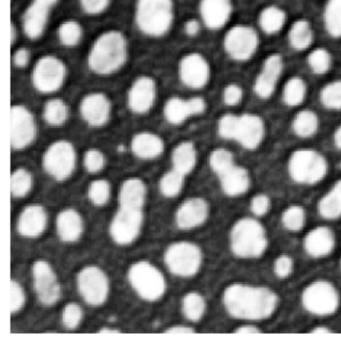
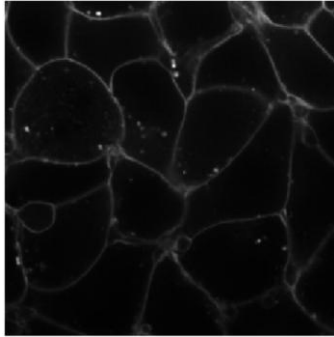
- * If an image shows sparse objects such as nuclei, use Otsu-thresholding for segmenting them.
- * If an image shows dense, partially overlapping objects such as nuclei, use StarDist.
- * If an image shows large cell-like structures with bright membranes, use the Watershed algorithm.
- * In case of doubt, use CellPose.

```
## The task
```

```
What is the best image processing algorithm to segment this microscopy image?  
Answer the algorithm name only. No explanations.  
"""
```

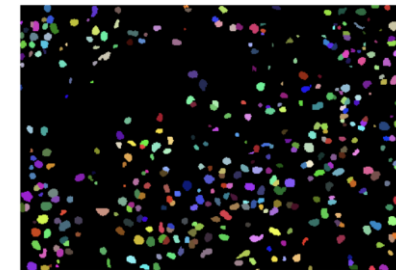
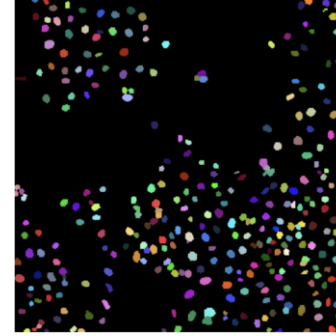
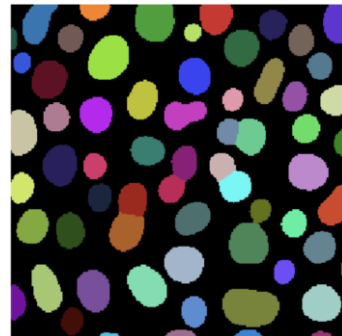


Speaking of Cellpose[-SAM]



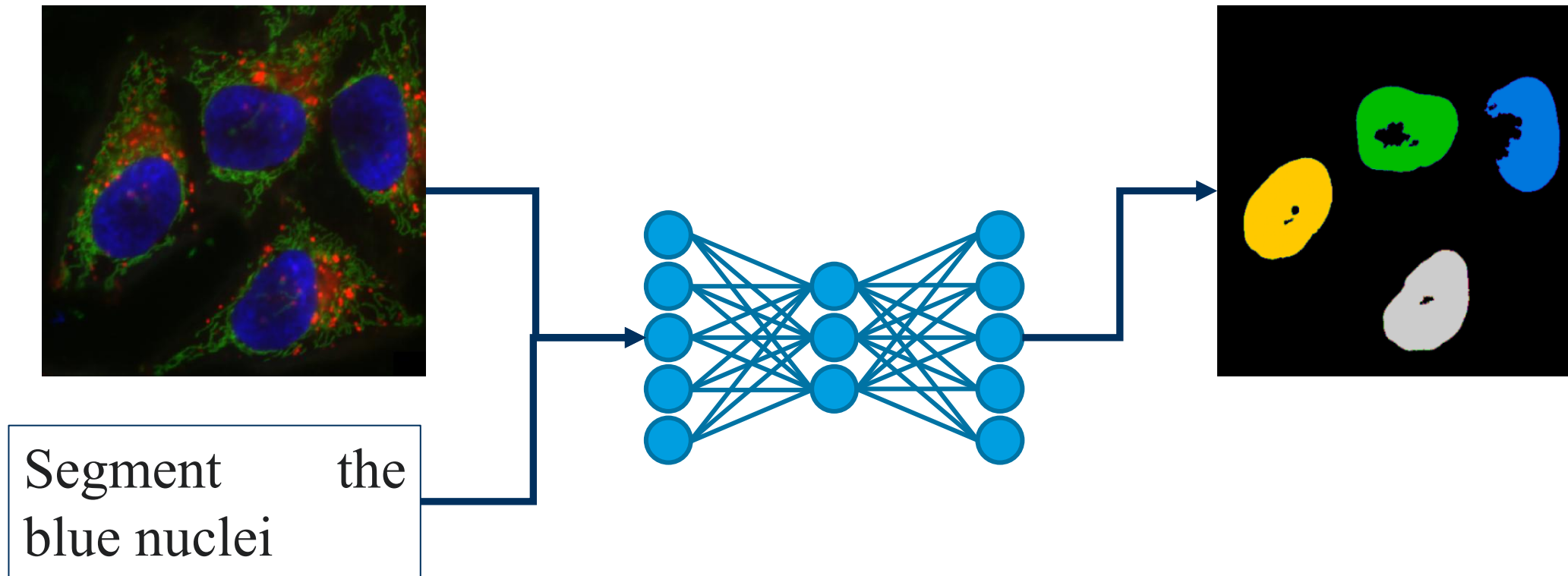
```
masks, flows, styles = model.eval(image,  
    batch_size=32,  
    flow_threshold=0.4,  
    cellprob_threshold=0.0,  
    normalize={"tile_norm_blocksize": 0})  
  
stackview.insight(masks.astype(np.uint32))
```

“Foundation
model”



Segment Anything Model

New approach to DL-based image segmentation involving prompts



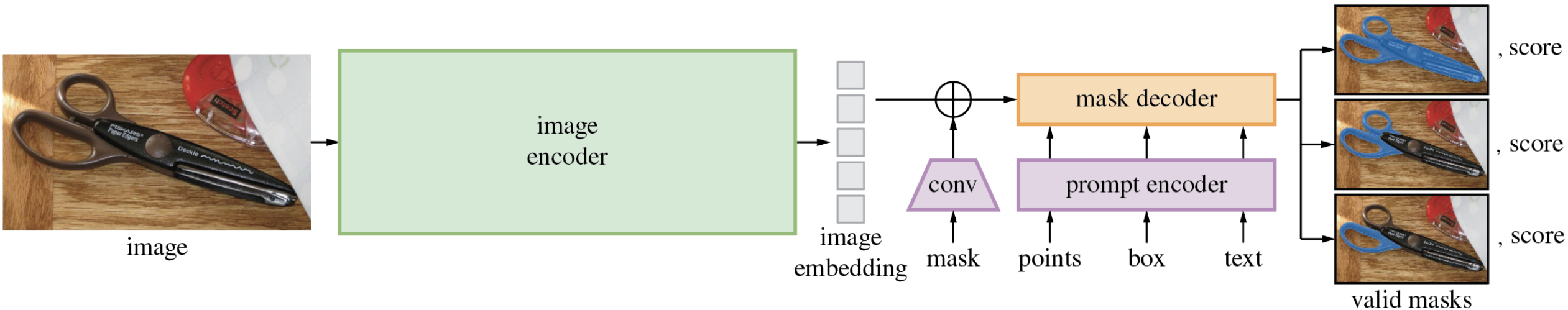
Segment Anything Model

New approach to DL-based image segmentation involving prompts



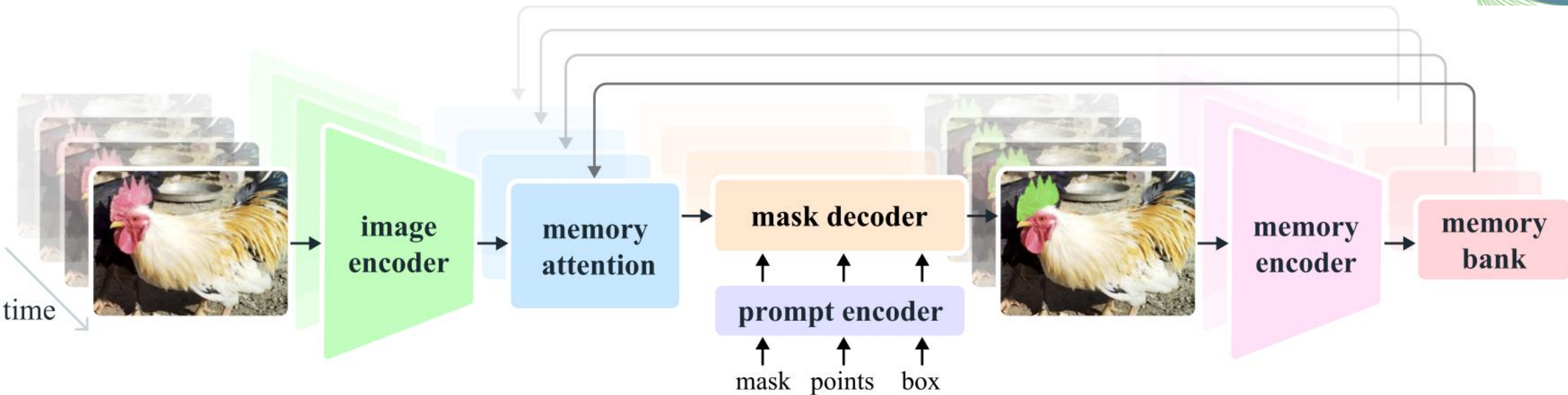
Segment Anything Model

New approach to DL-based image segmentation involving prompts
Effectively image-to-image model



Segment Anything Model 2

Extending the approach to videos by introducing memory



Segment Anything Model

Mostly natural images used for benchmarking

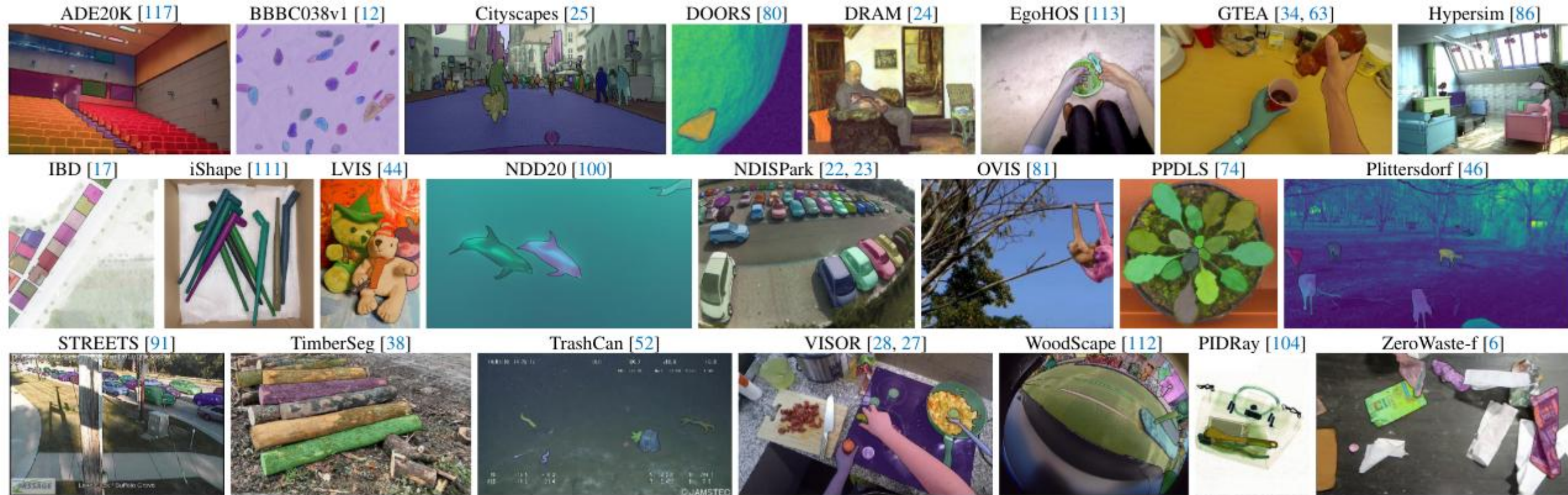


Figure 8: Samples from the 23 diverse segmentation datasets used to evaluate SAM's zero-shot transfer capabilities.

Segment Anything for Microscopy

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

The image displays four browser windows showing GitHub repositories for napari plugins related to Segment Anything (SAM). Each window shows the repository name, license, version, and a brief description of the plugin's functionality. The first window shows 'napari-SAM4IS' by hiroalchem, which is a plugin for instance and semantic segmentation. The second window shows 'napari-segment-anything' by royerlab, which is a Napari plugin of the Segment Anything Model (SAM). The third window shows 'Segment Anything Model (SAM) in Napari' by MIC-DKFZ, which integrates Meta AI's SAM into Napari. The fourth window shows 'Segment Anything for Microscopy' by computational-cell-analytics, which provides tools for segmentation and tracking in microscopy images.

napari-SAM4IS
license: Apache-2.0 | pypi: v0.0.6 | python: 3.8 | 3.9 | 3.10 | tests: failing | codecov: unknown
* napari hub: napari-SAM4IS
napari plugin for instance and semantic segmentation using Segment Anything Model (SAM)
This is a plugin for [napari](#), a multi-dimensional image viewer for Python and semantic segmentation annotation. This plugin provides an easy-to-annotating images with the option to output annotations as COCO format.
This [napari](#) plugin was generated with [Cookiecutter](#) using [@napari's](#) template.
Installation
To use this plugin, you'll need to install the [napari](#) multi-dimensional image viewer and the [Segment Anything Model \(SAM\)](#) library.
napari Installation
You can install napari using pip:
`pip install "napari[sam]"`

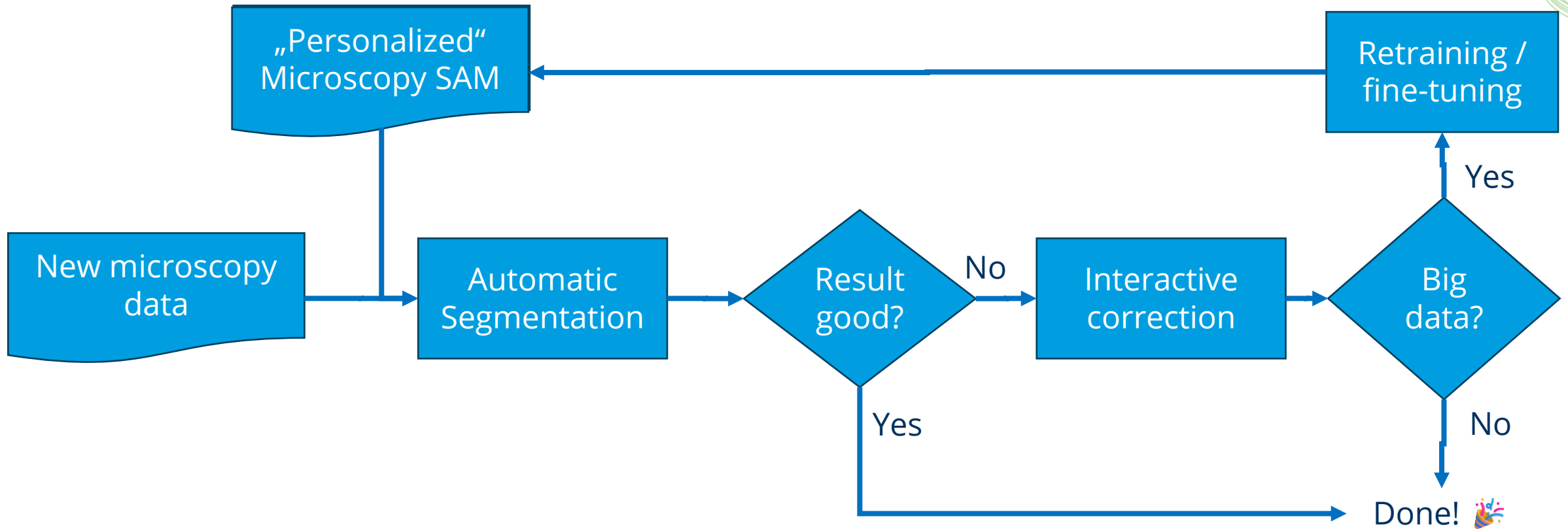
napari-segment-anything
license: Apache-2.0 | pypi: v0.1.4 | python: 3.8 | 3.9 | 3.10 | tests: failing | codecov: unknown
* napari hub: napari-segment-anything
Napari plugin of [Segment Anything Model \(SAM\)](#)
Download the network weights [here](#)
napari-segment-anything-small.mp4

Segment Anything Model (SAM) in Napari
license: Apache-2.0 | pypi: v0.4.13 | python: 3.8 | 3.9 | 3.10 | tests: failing | codecov: unknown
* napari hub: napari-sam
Segment anything with our [Napari](#) integration of Meta AI's new Segment Anything Model (SAM)
SAM is the new segmentation system from Meta AI capable of **one-click segmentation of any object**, and now, our plugin neatly integrates this into Napari.
We have already **extended** SAM's click-based foreground separation to full **click-based semantic segmentation and instance segmentation!**
At last, our SAM integration supports both **2D and 3D images!**

Segment Anything for Microscopy
license: MIT
docs: pdoc.dev | Anaconda.org | 1.0.0post0 | codecov: 40% | DOI: 10.5281/zenodo.7919746
Tools for segmentation and tracking in microscopy build on top of [Segment Anything](#) and track objects in microscopy images interactively with a few clicks!
We implement napari applications for:
• interactive 2d segmentation (Left: interactive cell segmentation)
• interactive 3d segmentation (Middle: interactive mitochondria segmentation)
• interactive tracking of 2d image data (Right: interactive cell tracking)

Segment Anything for Microscopy

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



Segment Anything for Microscopy

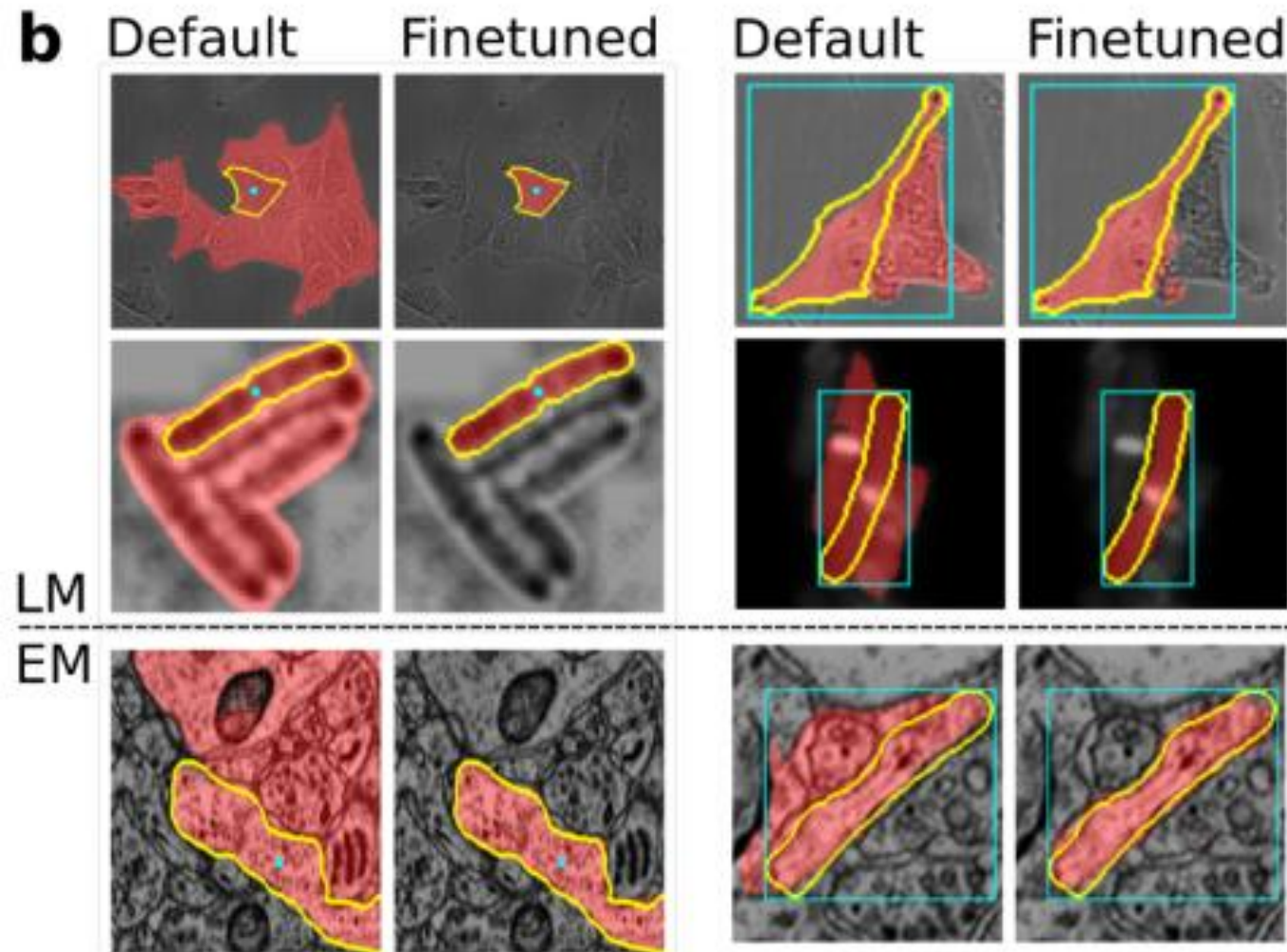
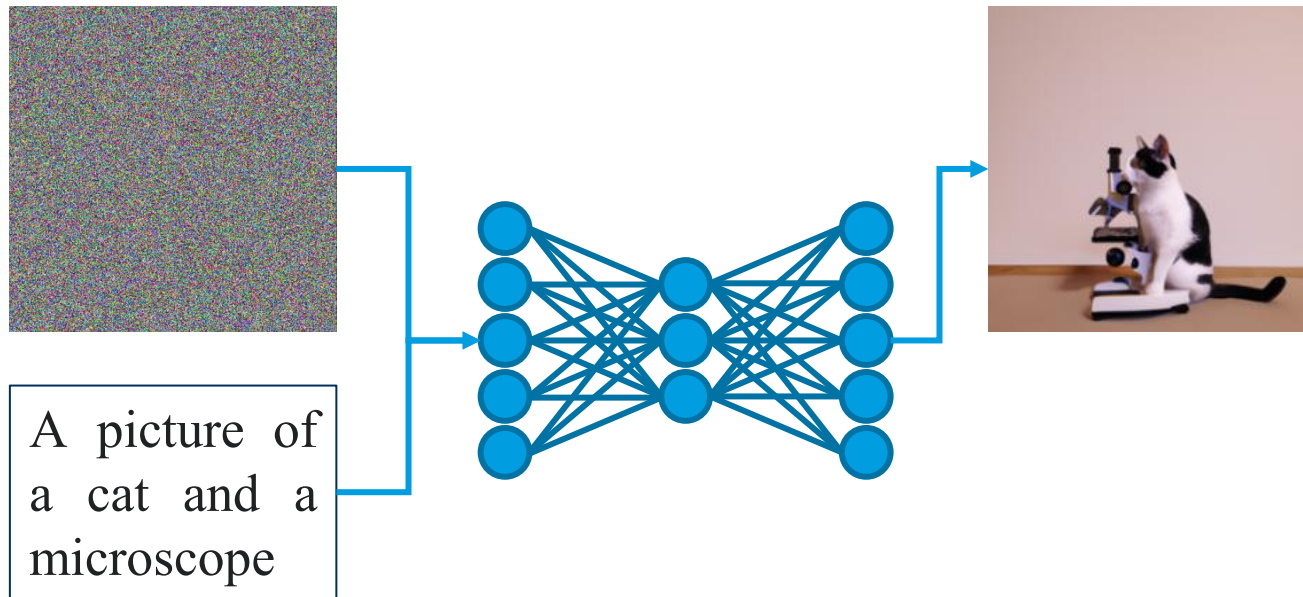


Image Generation

„text-to-image“



Variational Auto-Encoder

„image-to-image“

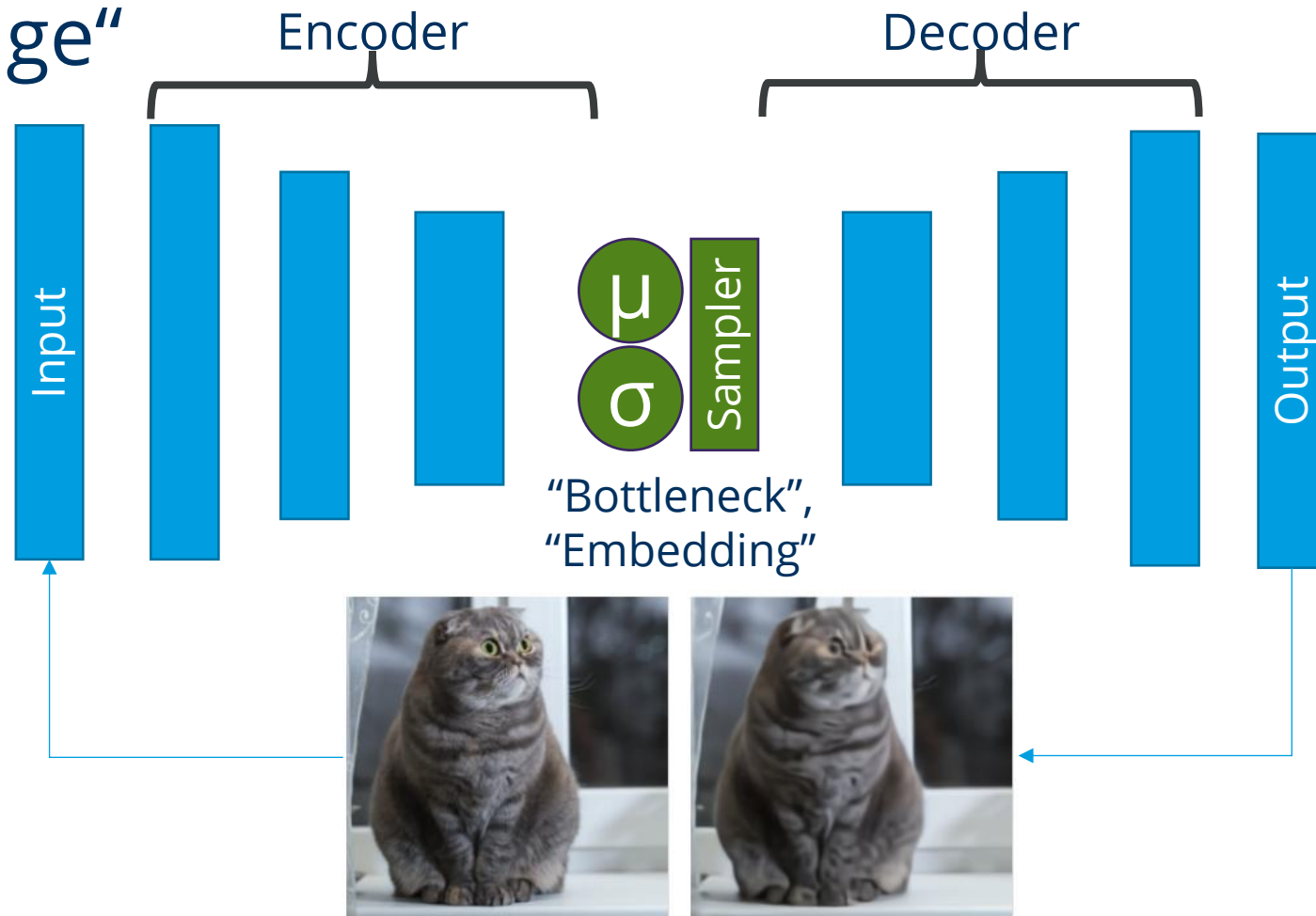
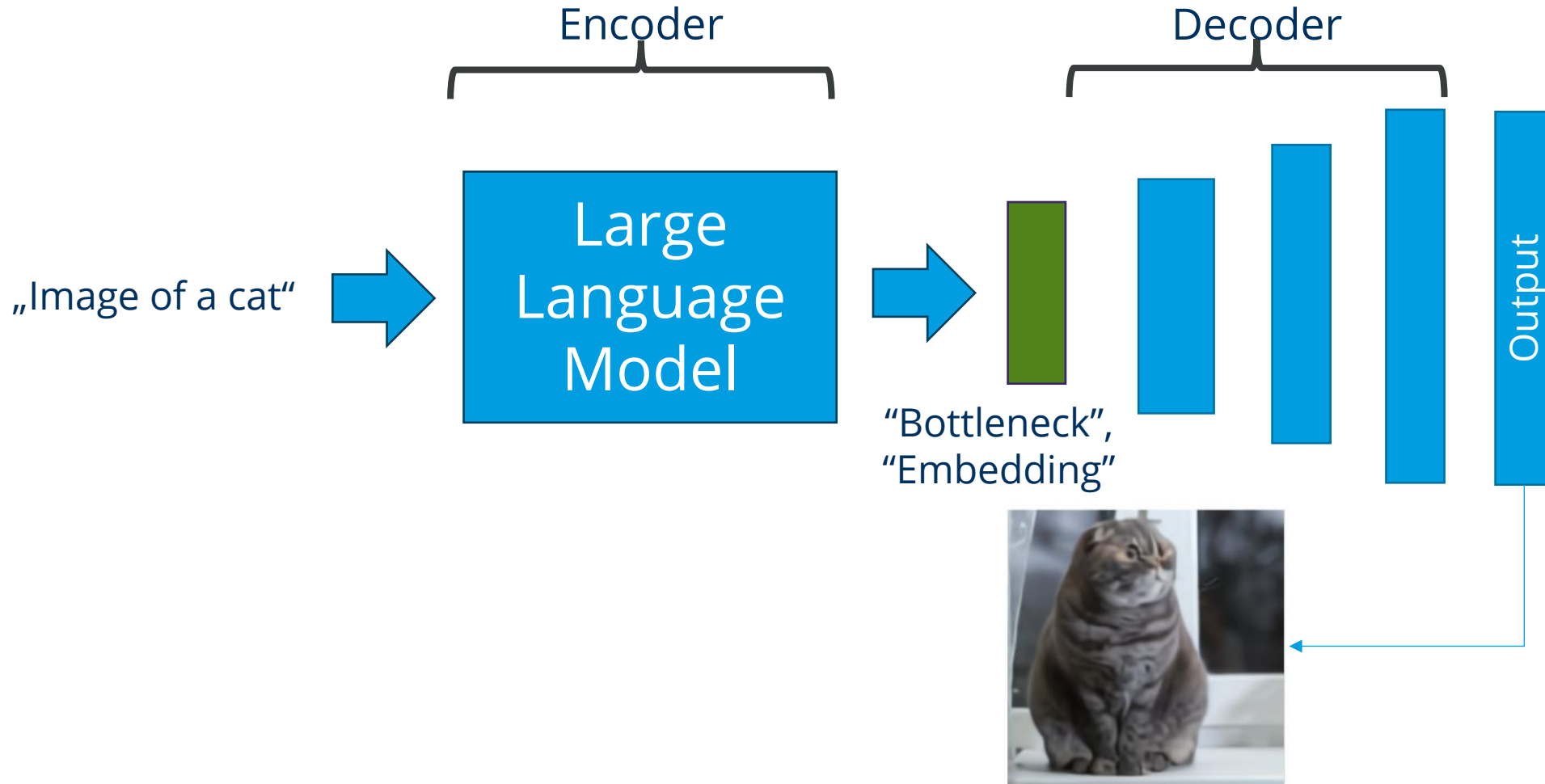
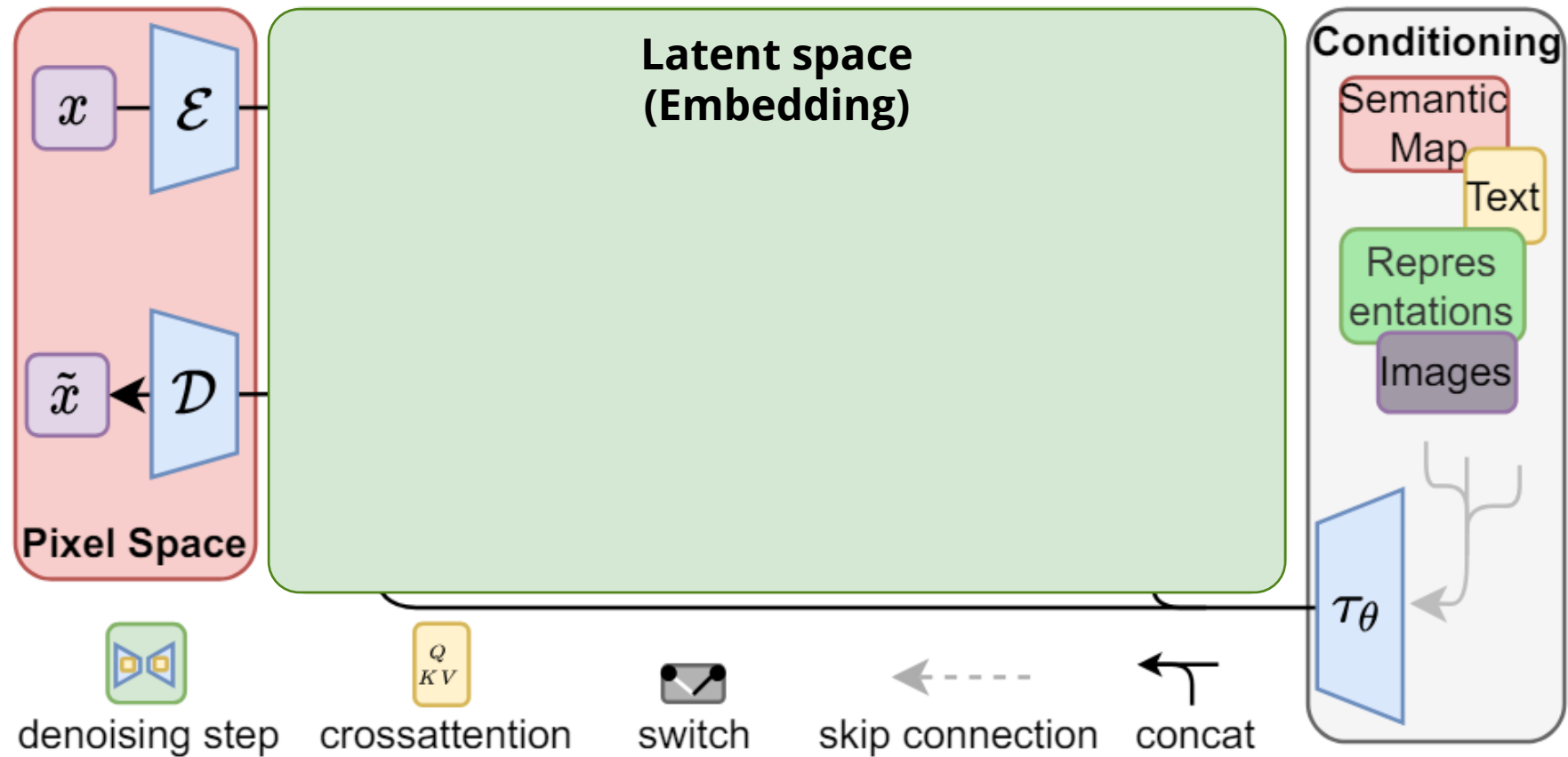


Image Generation



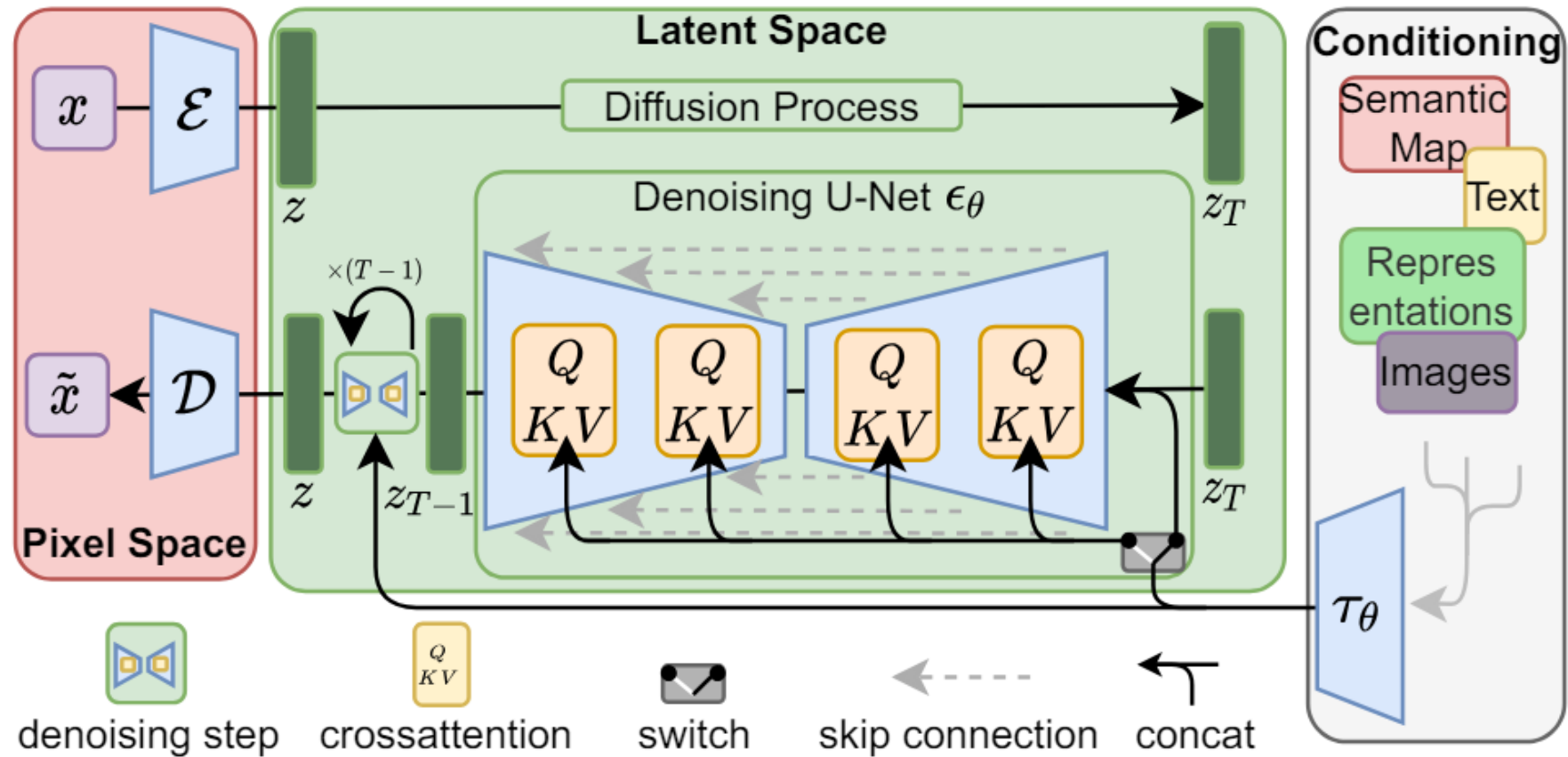
Stable Diffusion

Diffusion: iterative, reverse denoising



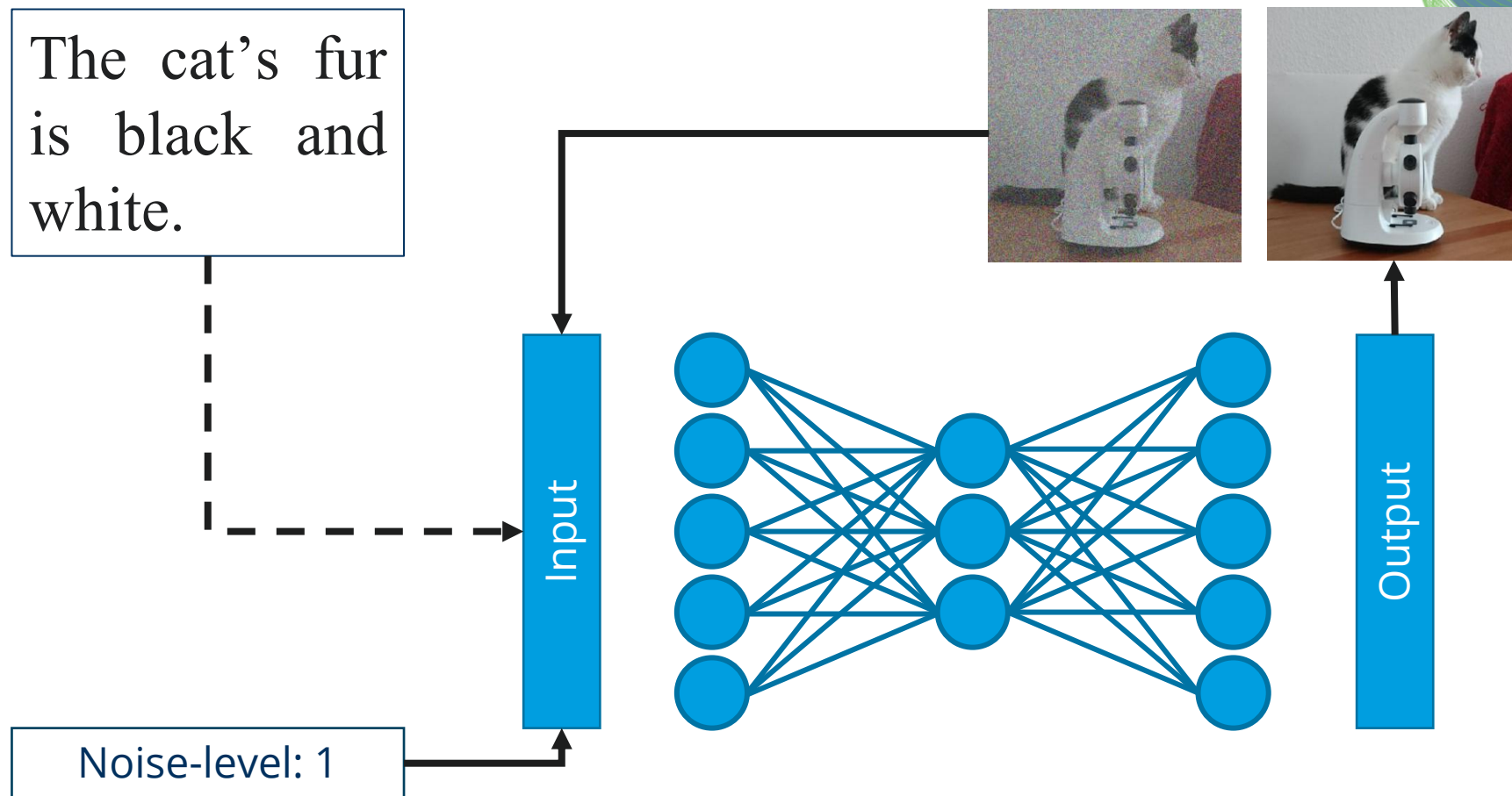
Stable Diffusion

Diffusion: iterative, reverse denoising



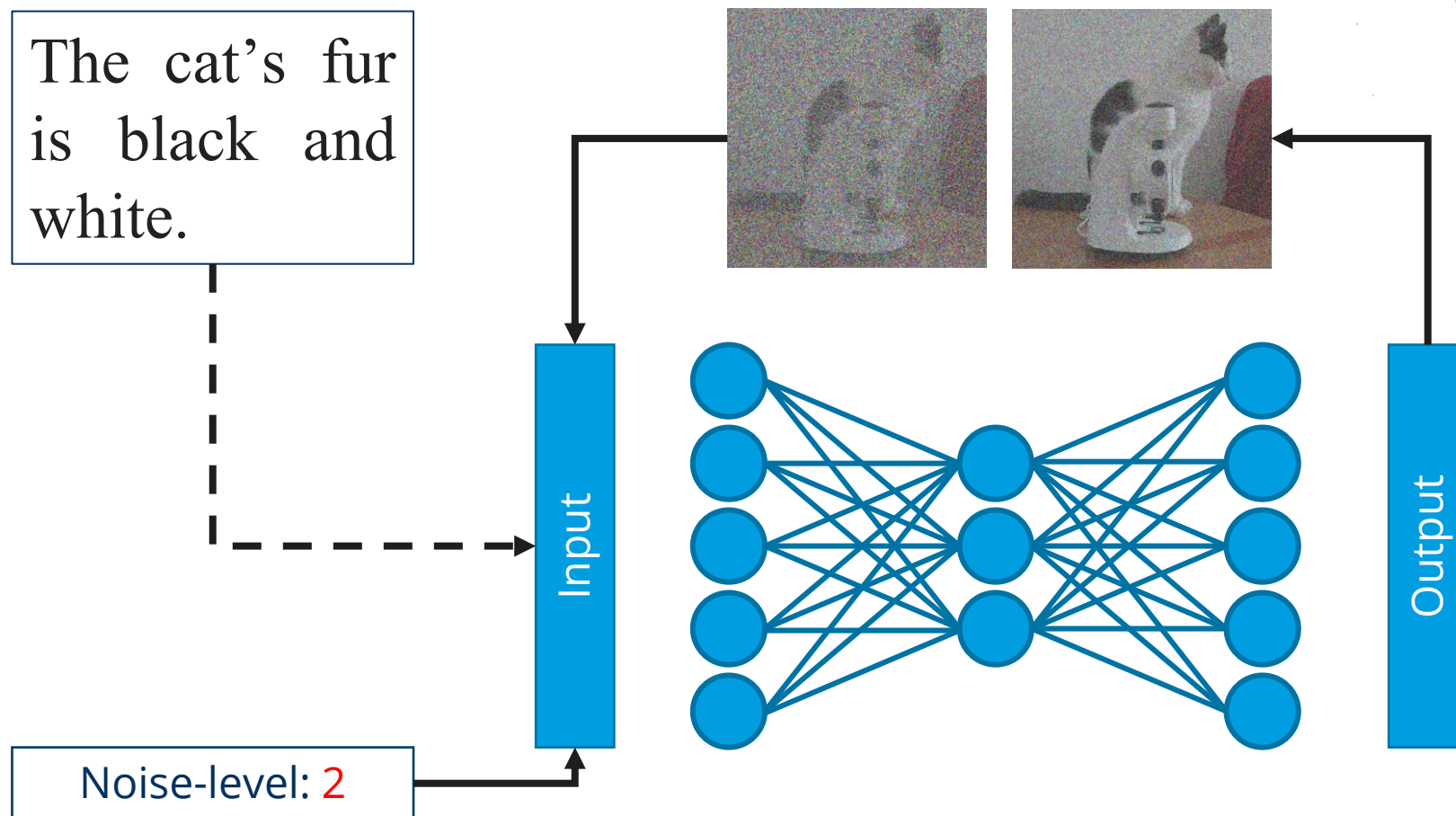
How does it work?

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



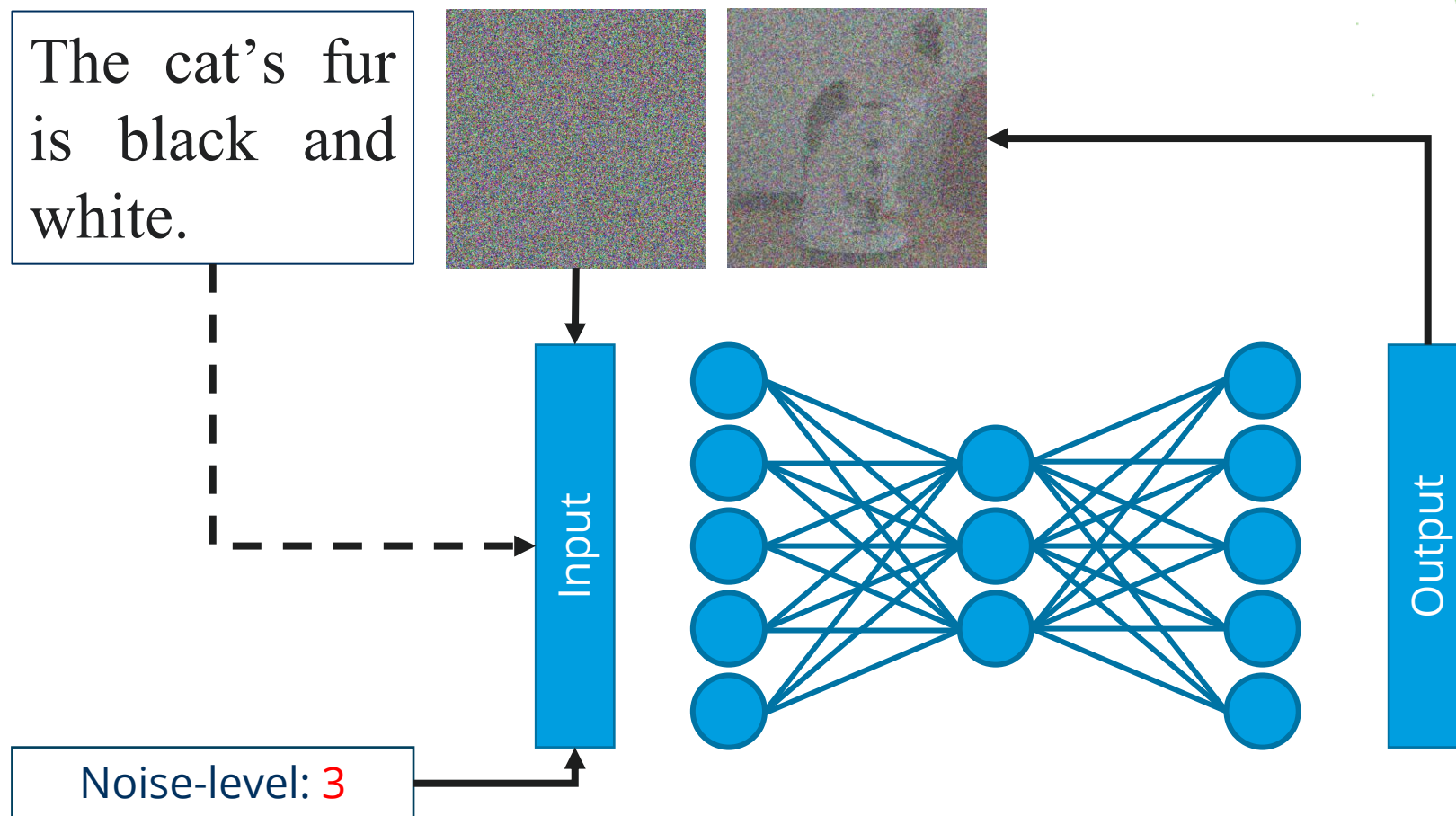
How does it work?

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



How does it work?

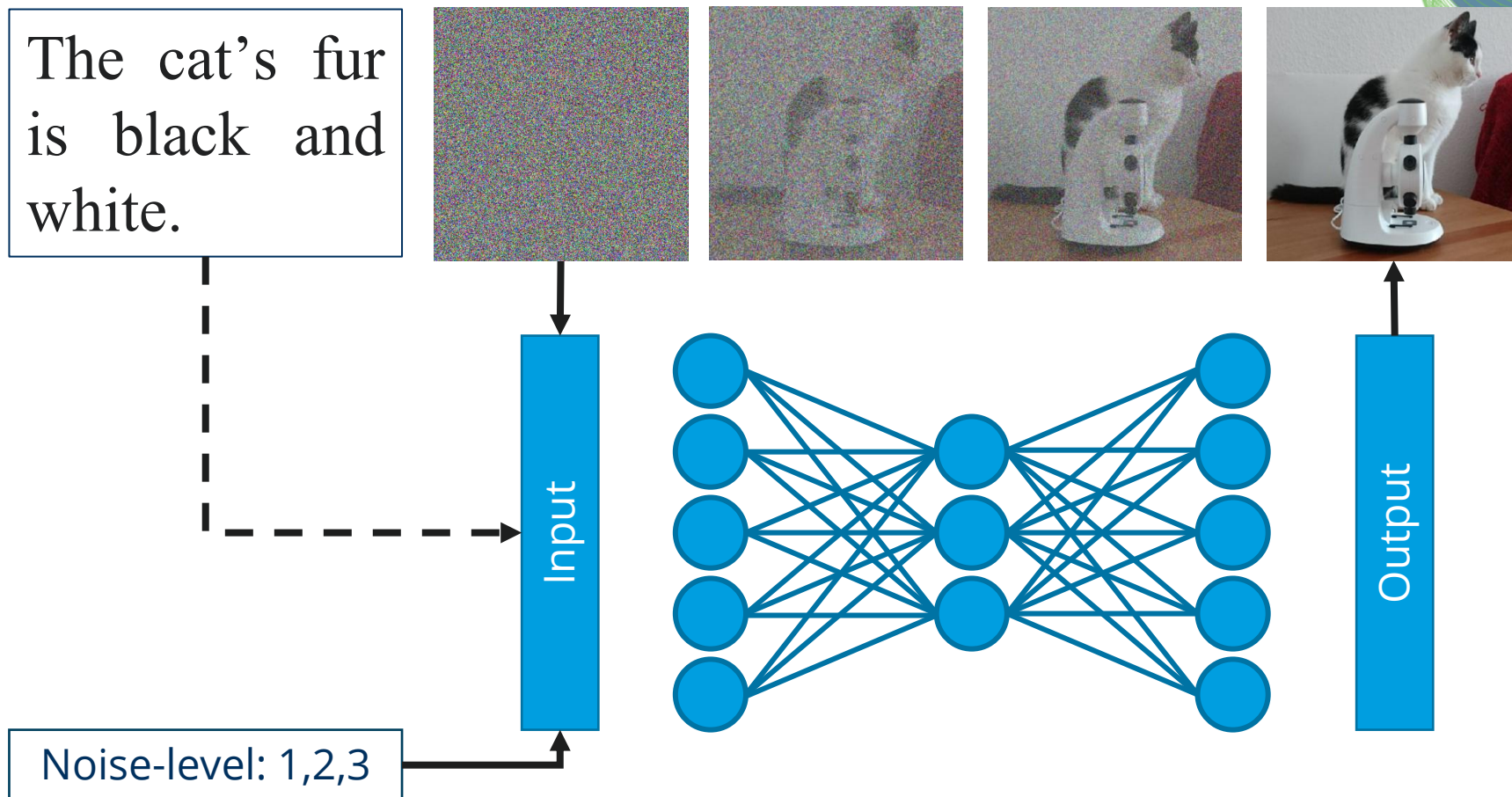
Train a U-Net on
data: image +
noisy image +
description +
noise-level



How does it work?

Prediction is
iterative denoising
of:

Pure noise +
text prompt



How does it work?

Reminder:

- Word embeddings
- Attention

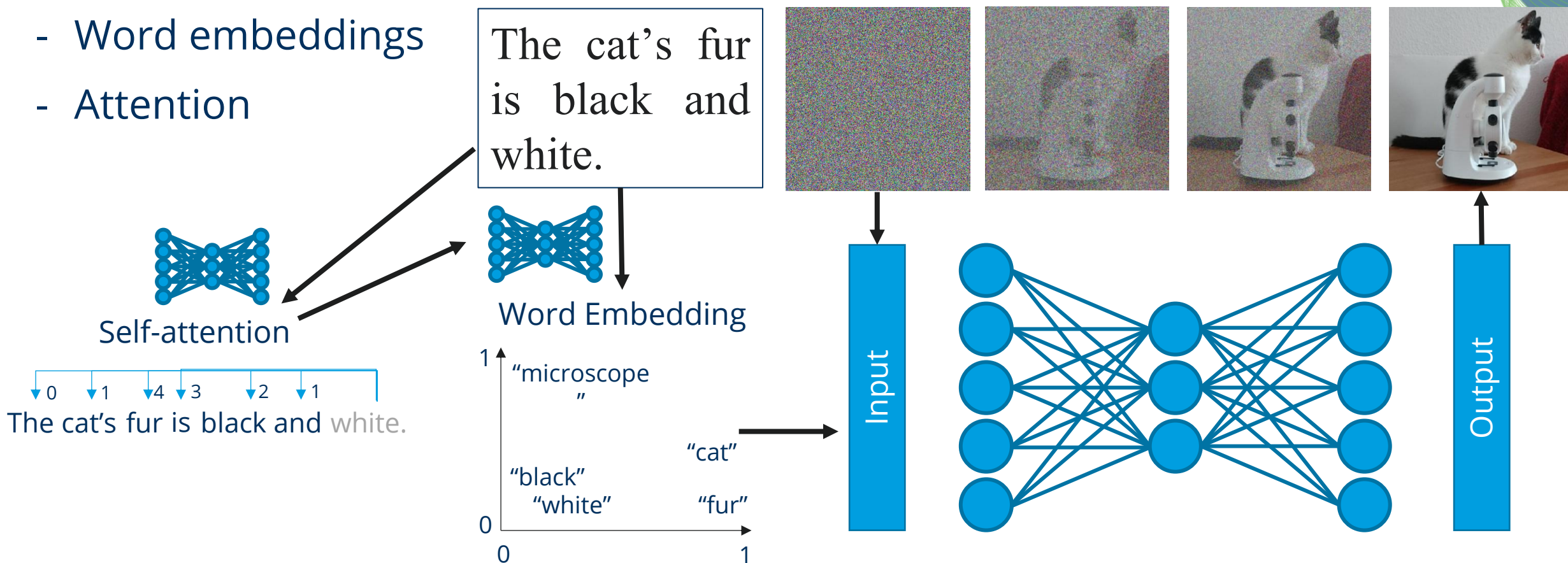


Image generation in Python: Huggingface

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
```

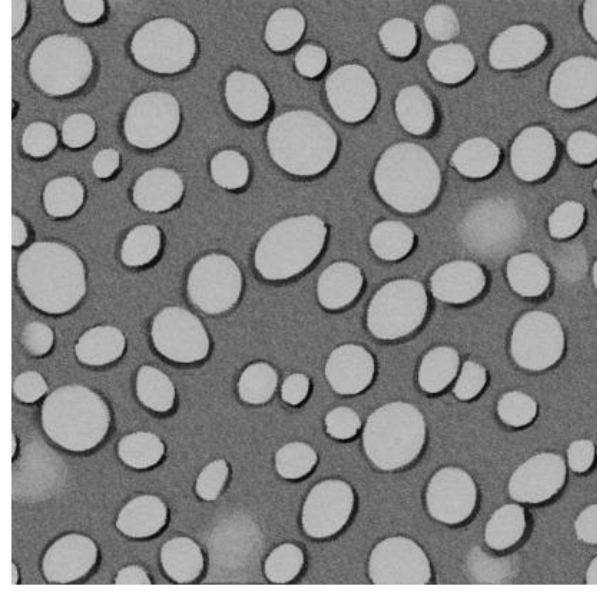
100%  50/50 [01:30<00:00, 1.40s/it]



```
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 

```
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

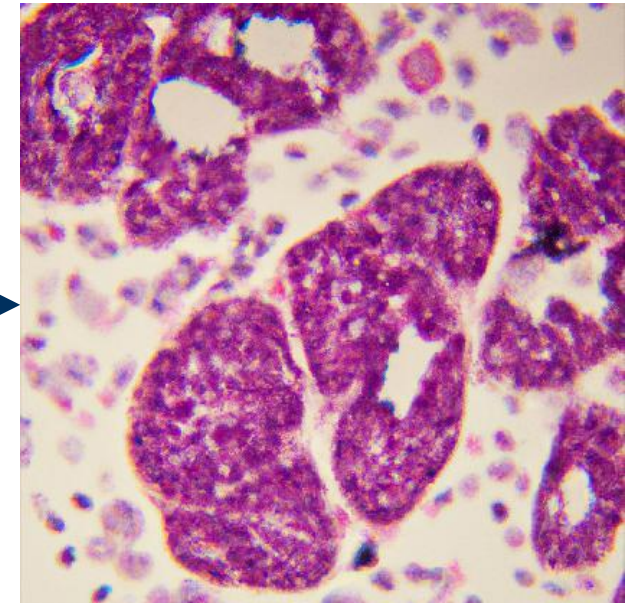
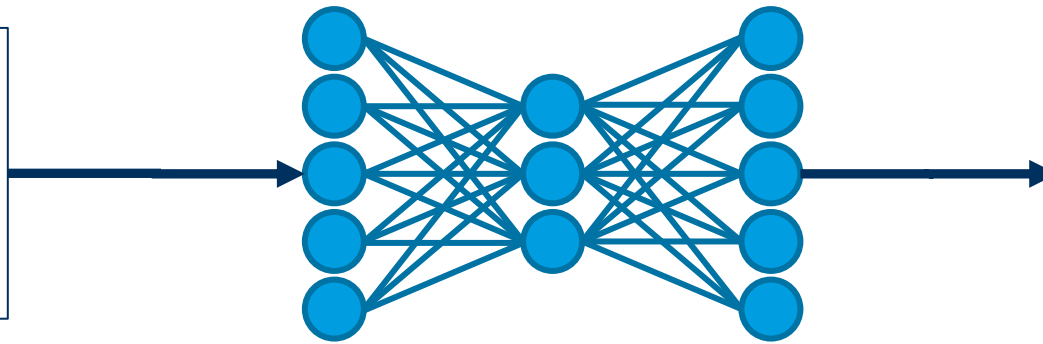
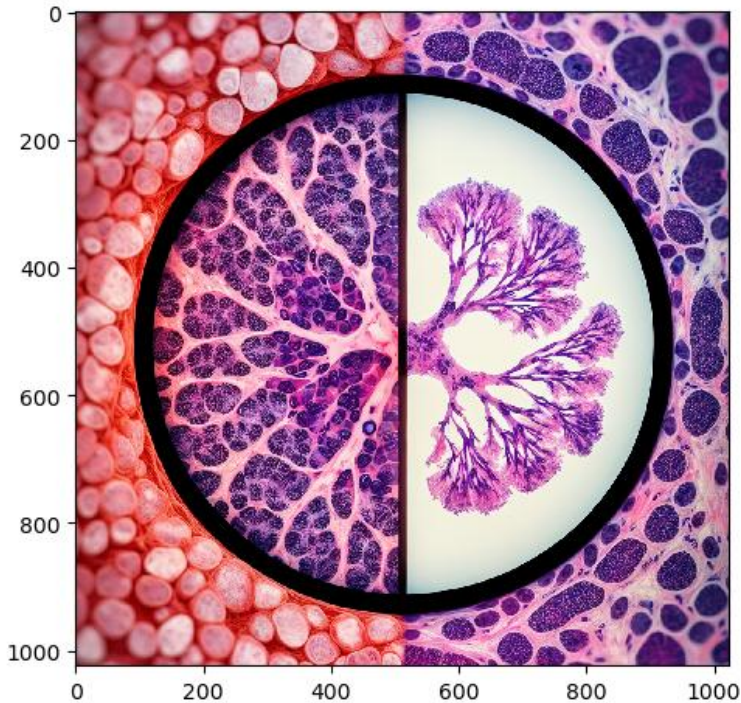


Image generation in Python: Dall-E

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)
```

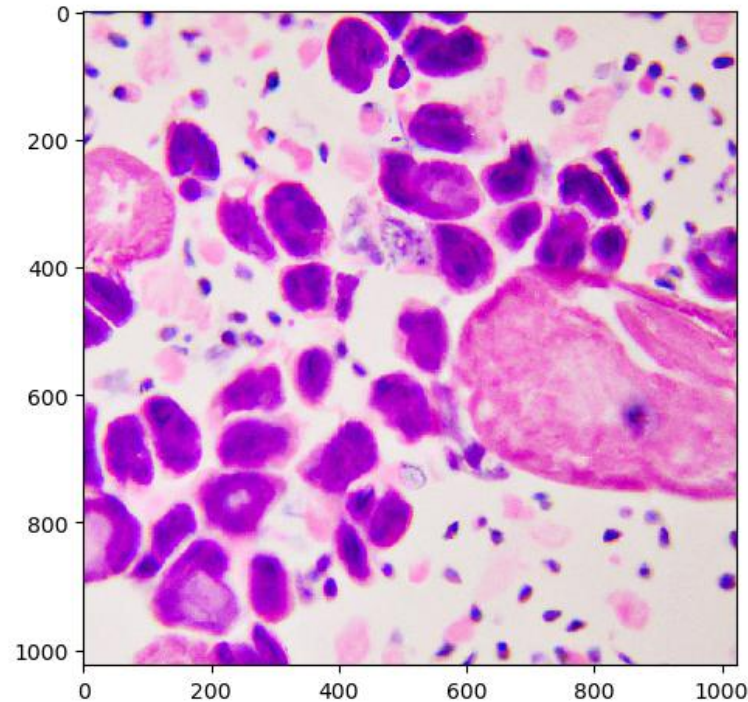
<matplotlib.image.AxesImage at 0x1d9eda5bd90>



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"

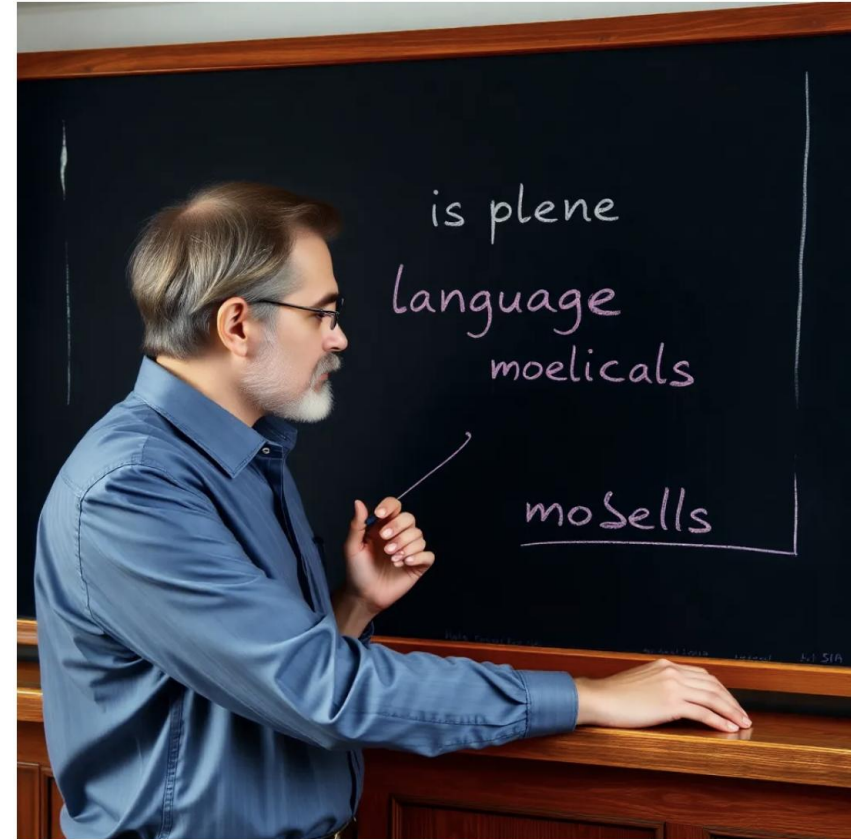


Source: <https://replicate.com/black-forest-labs/flux-schnell-lora>

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

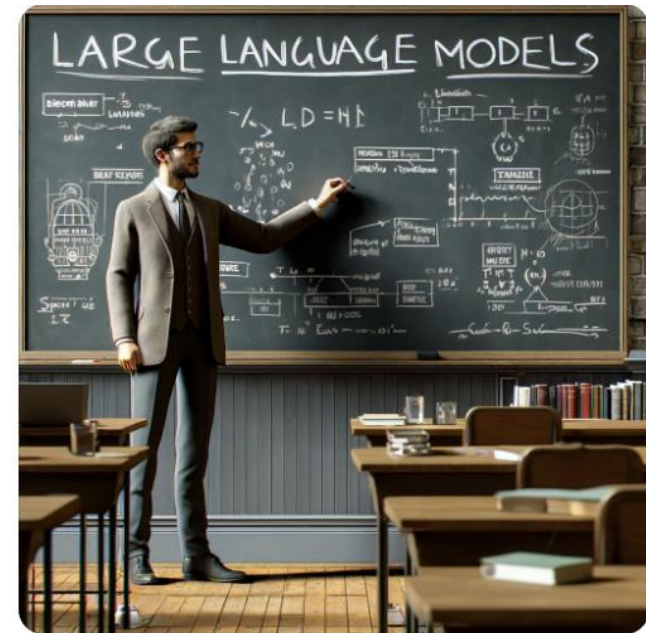
Challenges: Bias

Prompt: "a nice photo of a human"



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

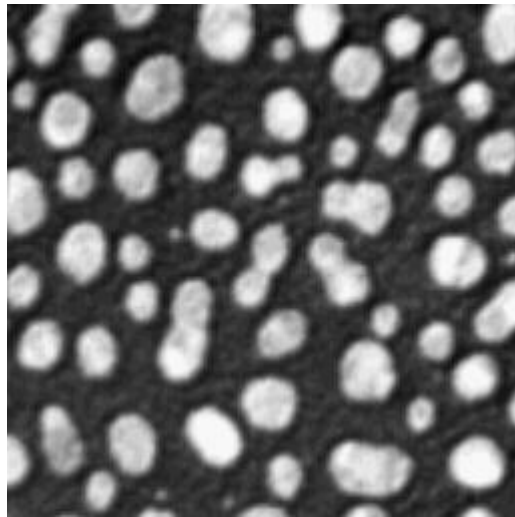
Prompt: 'please draw a picture of a professor writing "large language models" on a blackboard'



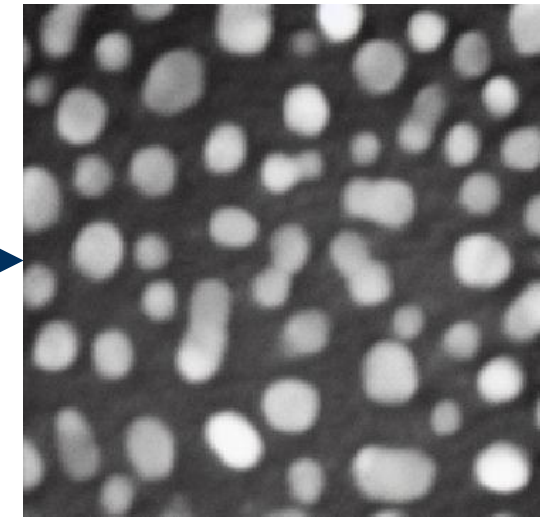
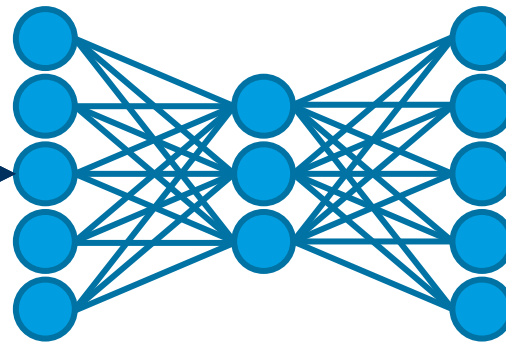
Source: <https://chatgpt.com/>

Image Generation LLMs

Image-to-image prompting, image variations



Blur the image



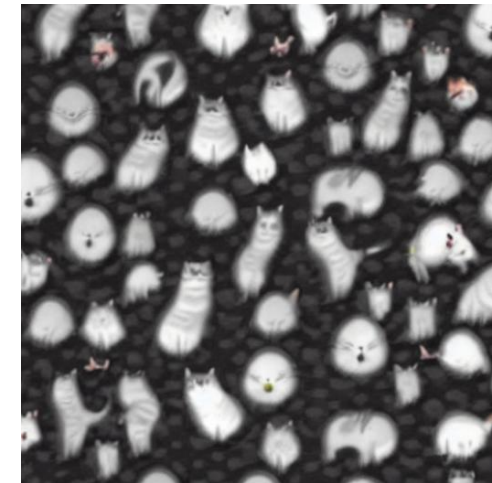
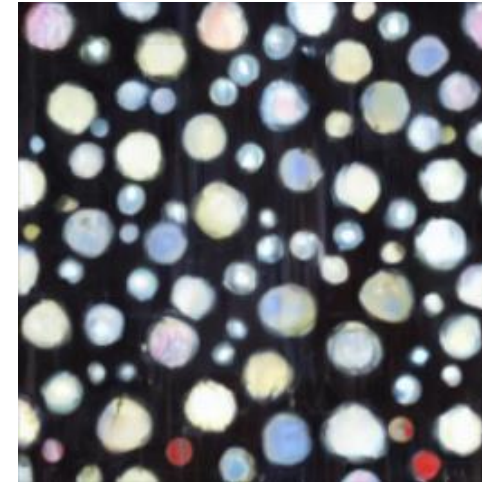
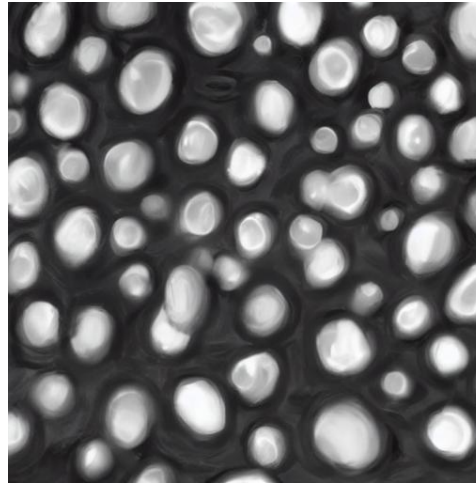
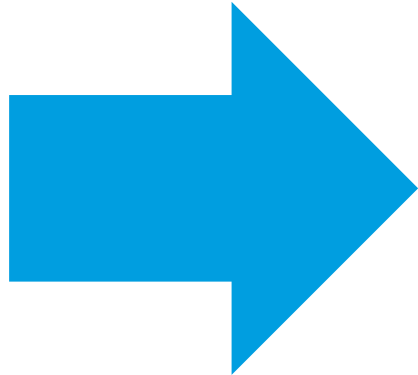
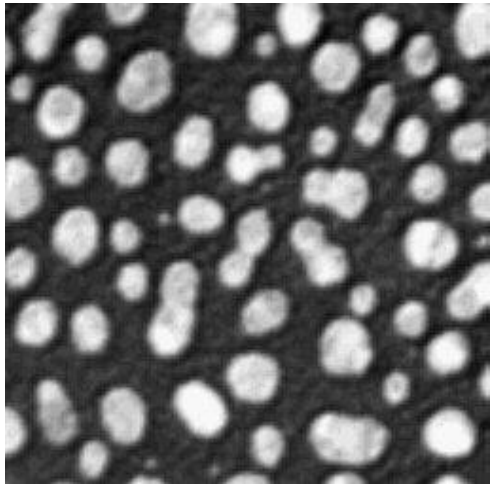
InstructPix2Pix: Learning to Follow Image Editing Instructions

Tim Brooks* Aleksander Holynski* Alexei A. Efros
University of California, Berkeley

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“)

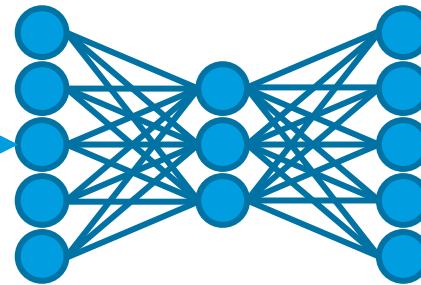
Raw image



Mask image



A black white
cat fur



Manipulated
image

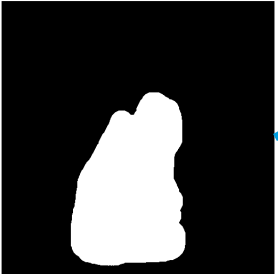


Inpainting in Python: Huggingface

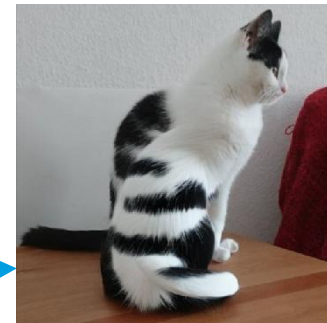
```
pipe = StableDiffusionInpaintPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-inpainting",
    torch_dtype=torch.float16
)
pipe = pipe.to("cuda")
```

Downloads
4.8 GB

Needs
Nvidia GPU



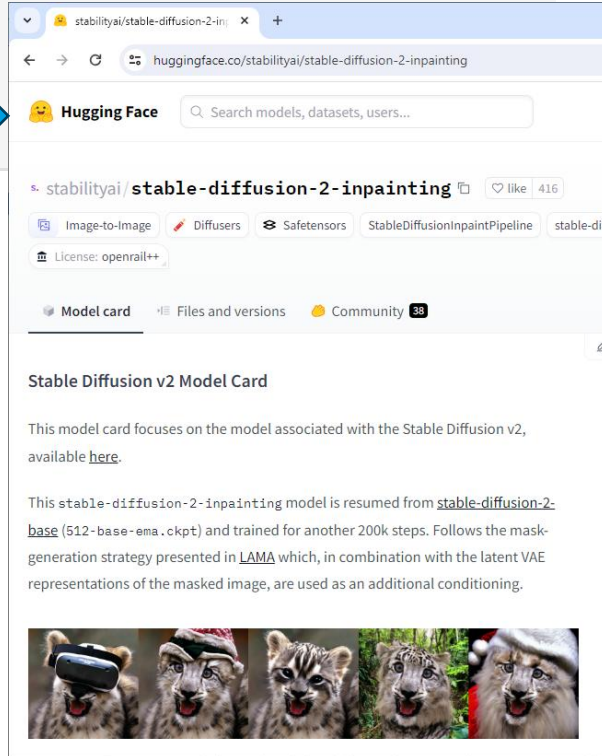
```
prompt = "A black white cat fur"
image = pipe(prompt=prompt,
    image=init_image,
    mask_image=mask_image,
    num_inference_steps=50,
    width=512,
    height=512,
    num_images_per_prompt=1,
).images[0]
```



Inpainting in Python: Huggingface

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

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



Hugging Face Search models, datasets, users...

stabilityai/stable-diffusion-2-inpainting 416

Image-to-Image Diffusers Safetensors StableDiffusionInpaintPipeline stable-diff

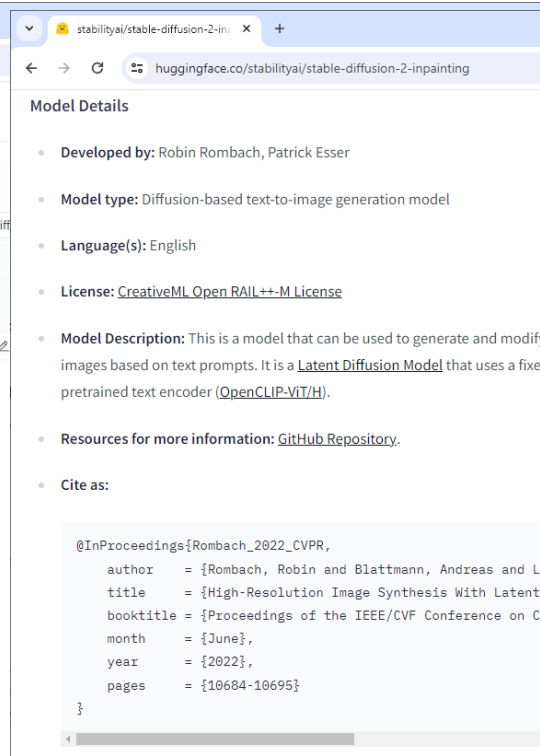

License: openrail++

Model card Files and versions Community 38

Stable Diffusion v2 Model Card

This model card focuses on the model associated with the Stable Diffusion v2, available [here](#).

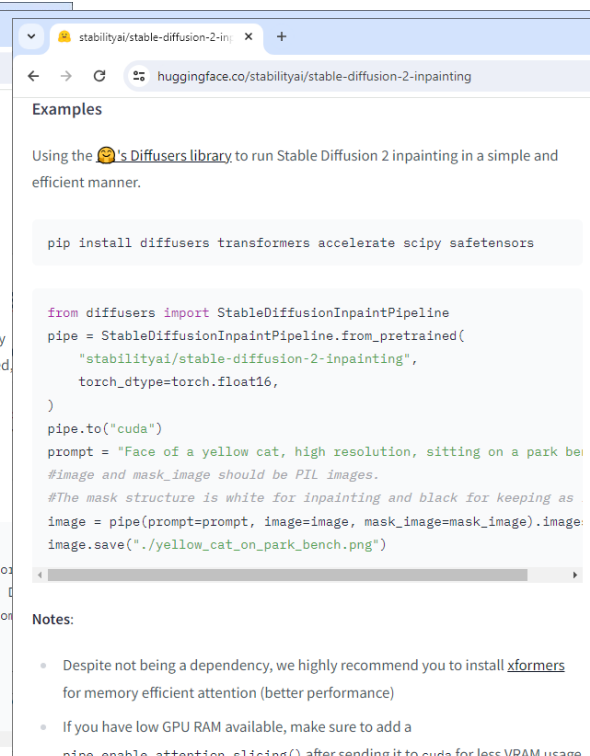
This stable-diffusion-2-inpainting model is resumed from [stable-diffusion-2-base](#) (512-base-ema.ckpt) and trained for another 200k steps. Follows the mask-generation strategy presented in [LAMA](#) which, in combination with the latent VAE representations of the masked image, are used as an additional conditioning.



Model Details

- Developed by: Robin Rombach, Patrick Esser
- Model type: Diffusion-based text-to-image generation model
- Language(s): English
- License: [CreativeML Open RAIL++-M License](#)
- Model Description: This is a model that can be used to generate and modify images based on text prompts. It is a [Latent Diffusion Model](#) that uses a fixed pretrained text encoder ([OpenCLIP-ViT/H](#)).
- Resources for more information: [GitHub Repository](#).
- Cite as:

```
@InProceedings{Rombach_2022_CVPR,
  author    = {Rombach, Robin and Blattmann, Andreas and Lorenz, David and
  title     = {High-Resolution Image Synthesis With Latent Diffusion Models},
  booktitle = {Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)},
  month     = {June},
  year      = {2022},
  pages     = {10684-10695}}
```



Examples

Using the [Diffusers](#) library to run Stable Diffusion 2 inpainting in a simple and efficient manner.

```
pip install diffusers transformers accelerate scipy safetensors
```

```
from diffusers import StableDiffusionInpaintPipeline
pipe = StableDiffusionInpaintPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-inpainting",
    torch_dtype=torch.float16,
)
pipe.to("cuda")
prompt = "Face of a yellow cat, high resolution, sitting on a park bench"
#image and mask_image should be PIL images.
#The mask structure is white for inpainting and black for keeping as is
image = pipe(prompt=prompt, image=image, mask_image=mask_image).image
image.save("./yellow_cat_on_park_bench.png")
```

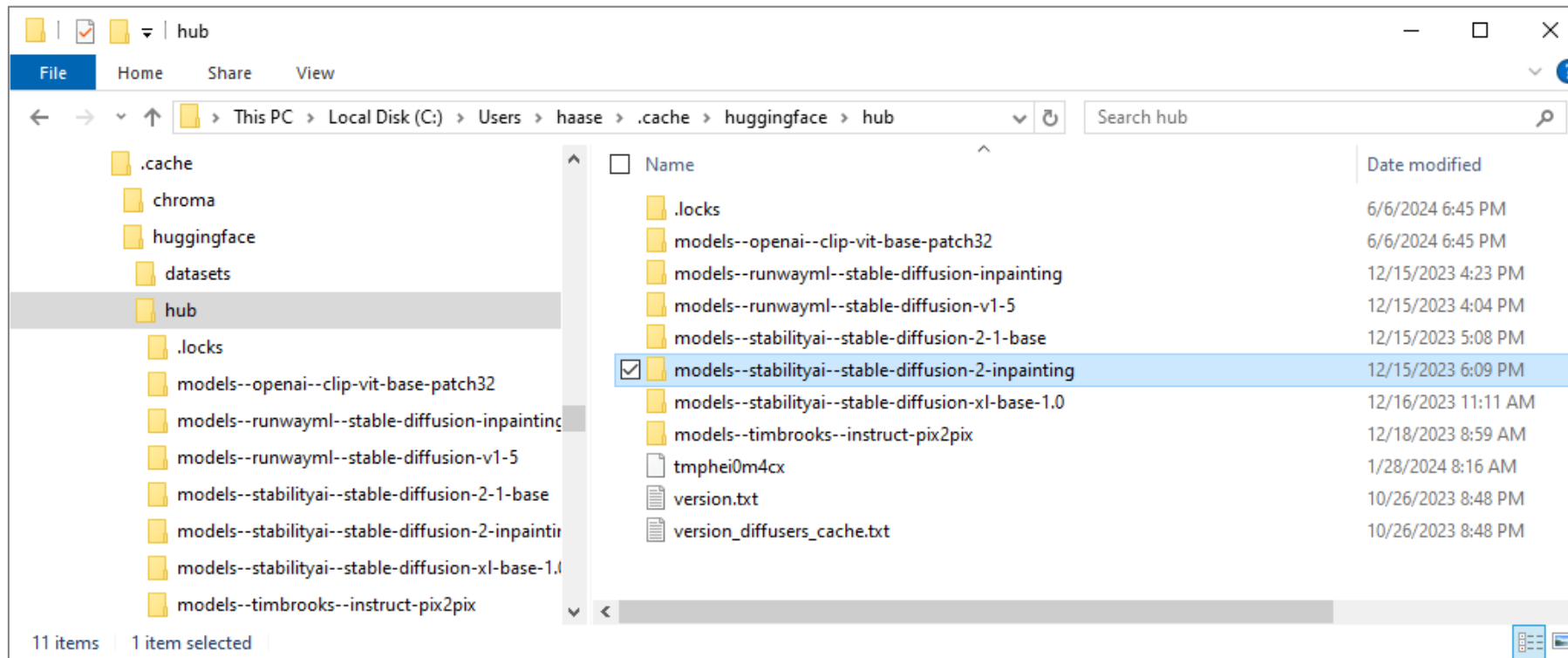
Notes:

- Despite not being a dependency, we highly recommend you to install [xformers](#) for memory efficient attention (better performance)
- If you have low GPU RAM available, make sure to add a `pipe.enable_attention_slicing()` after sending it to cuda for less VRAM usage

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

Supported: 256,
512, 1024 pixels

Size must match

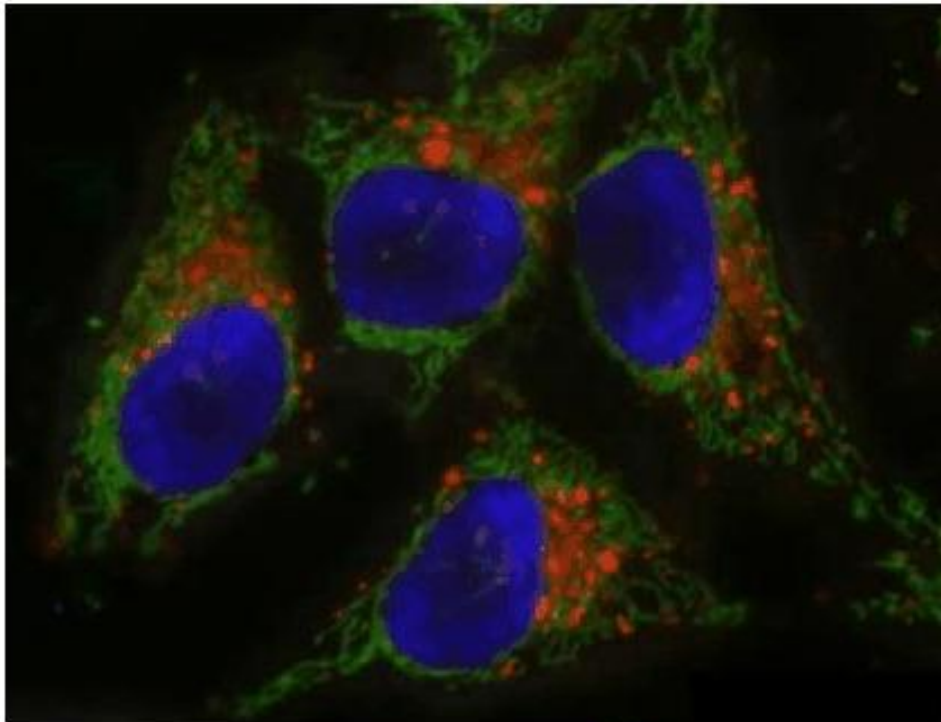


Result: List of URL(s)

New technologies bring new risks...

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

[6]:



Interesting
challenges for our
community ahead

Image manipulation detection

The noise pattern differs between raw and processed images...

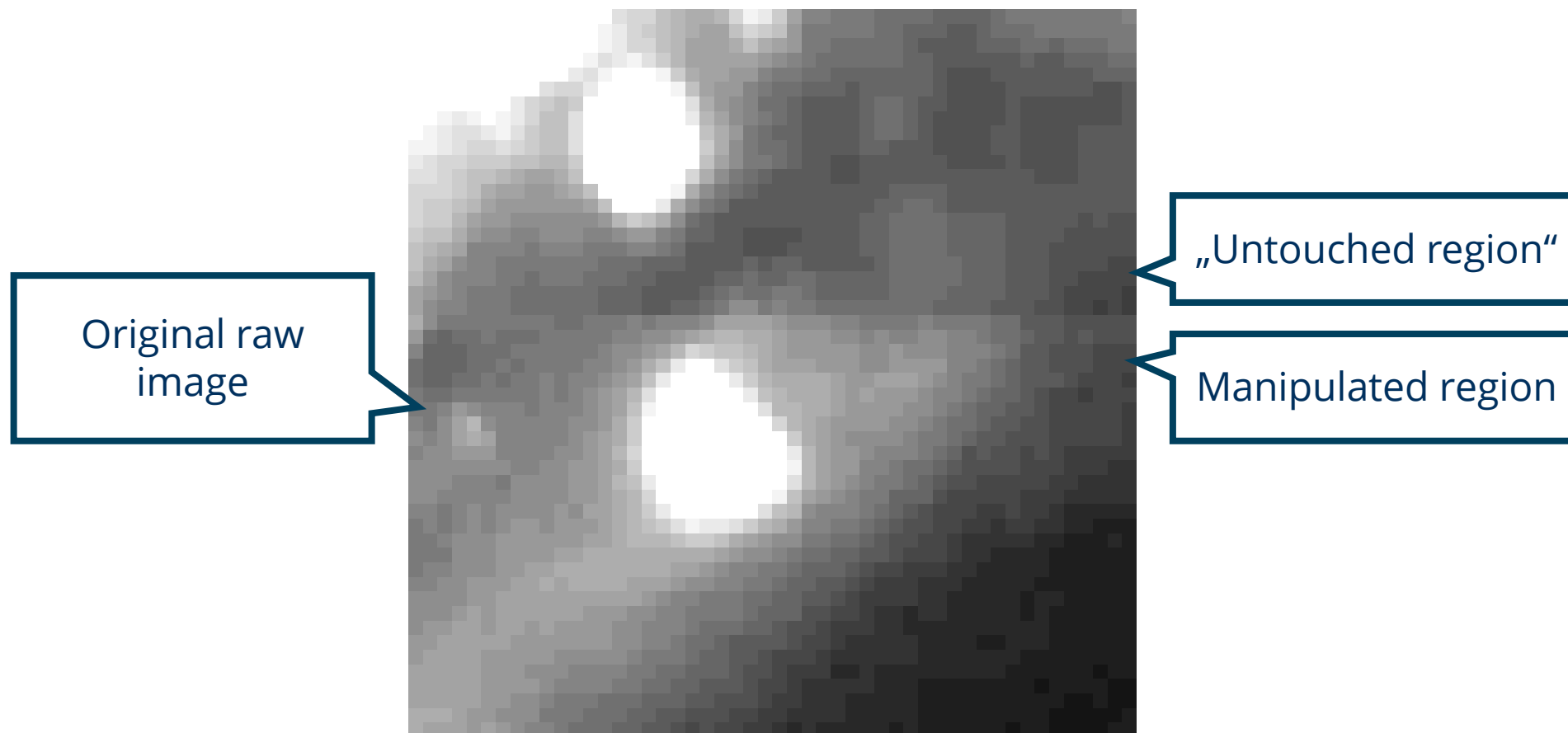
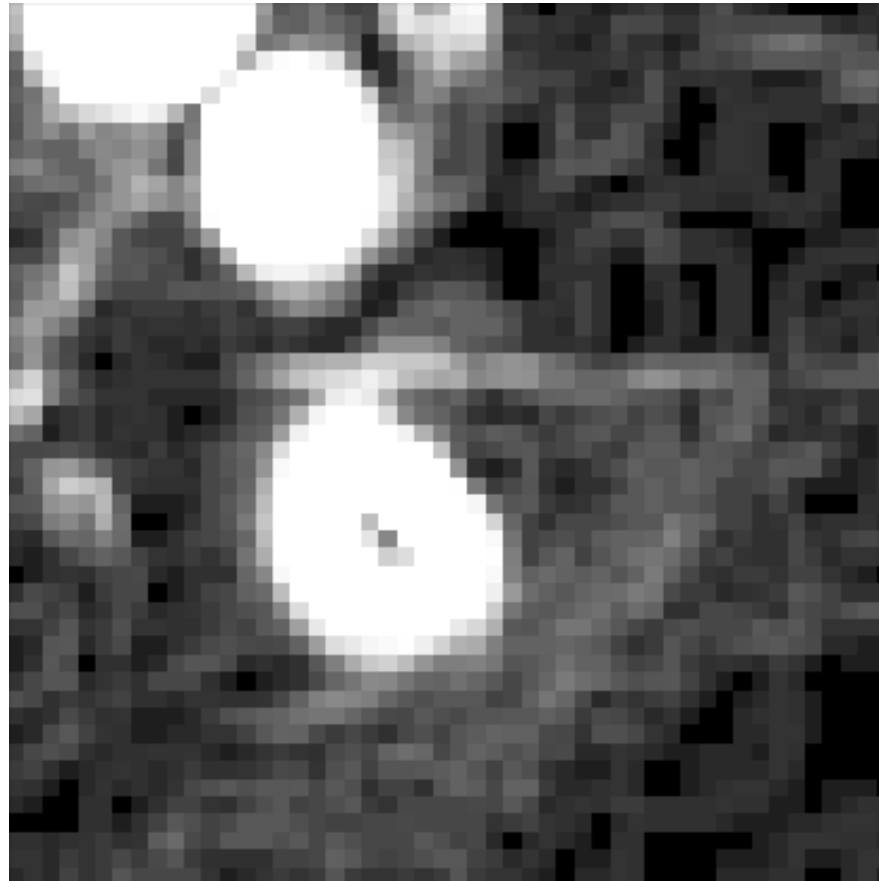


Image manipulation detection

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“

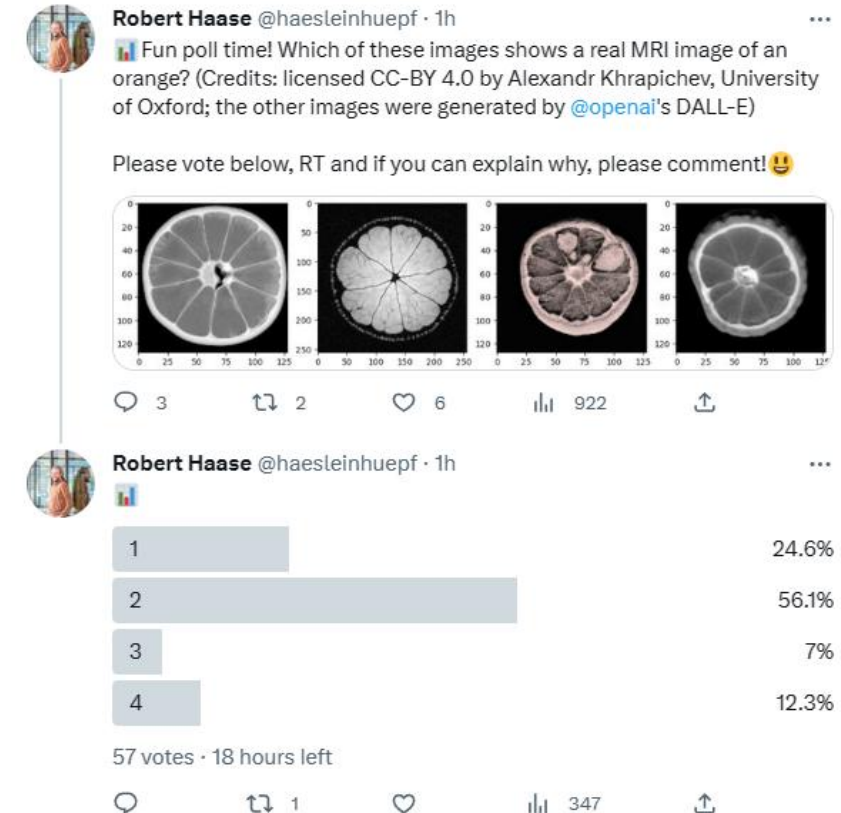
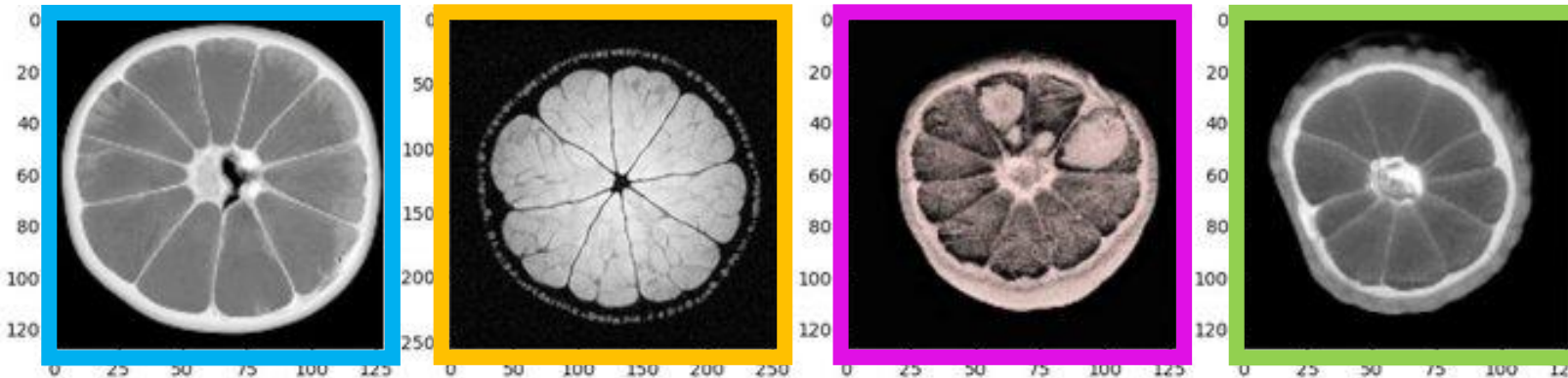
Manipulated region

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).  
""
```

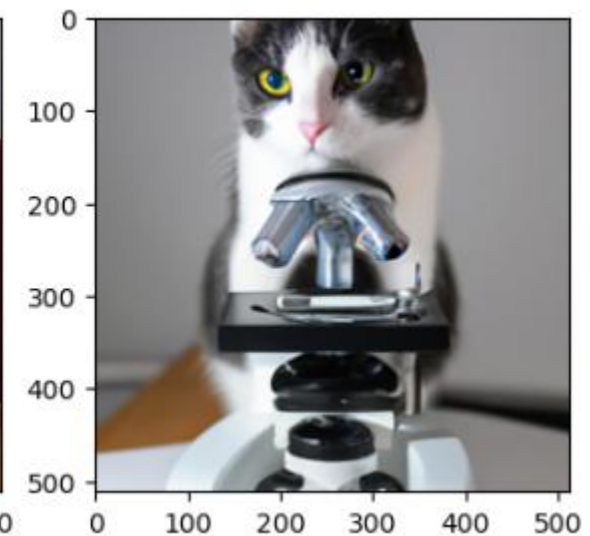
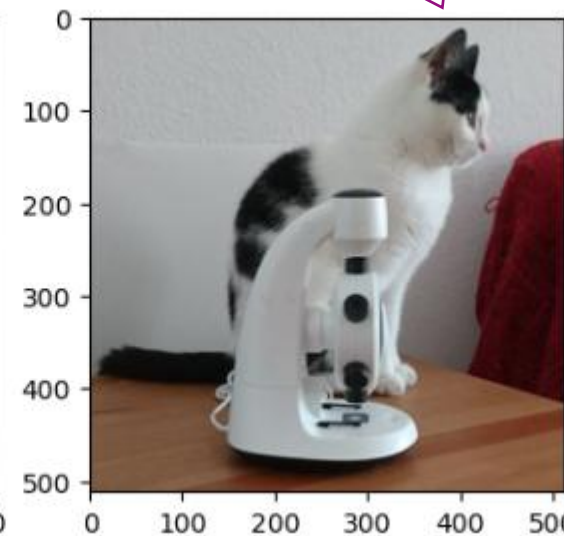
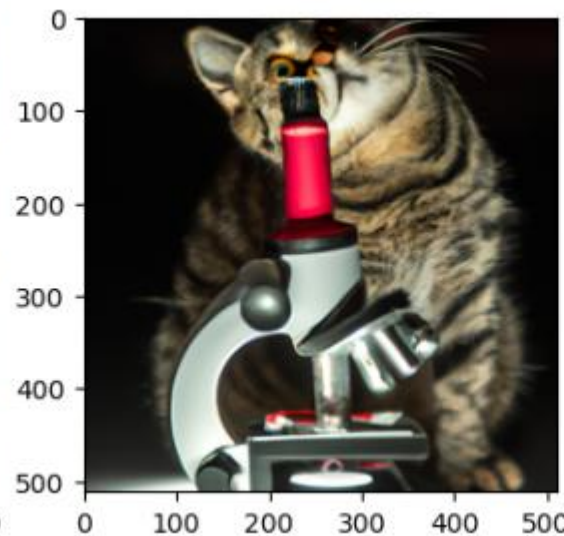
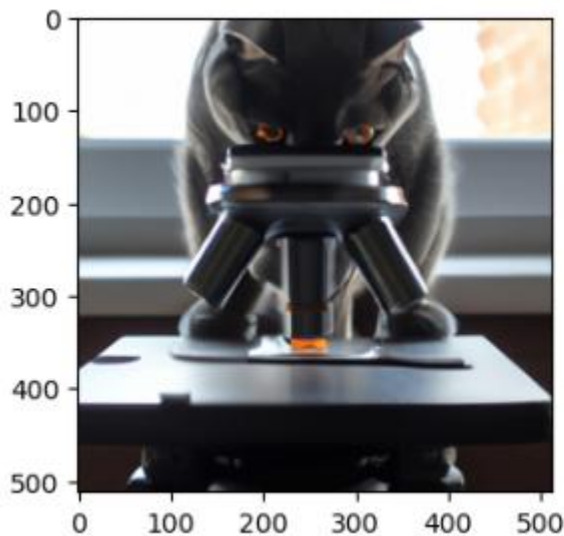


Benchmarking image generation

Prompt engineering to optimize images

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

One cat
is real.

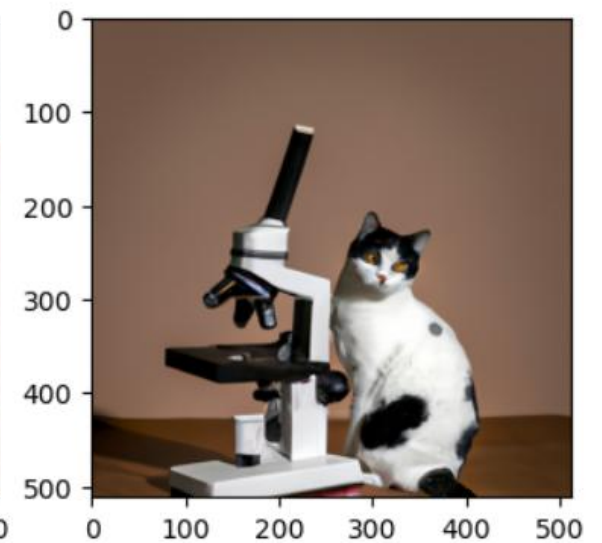
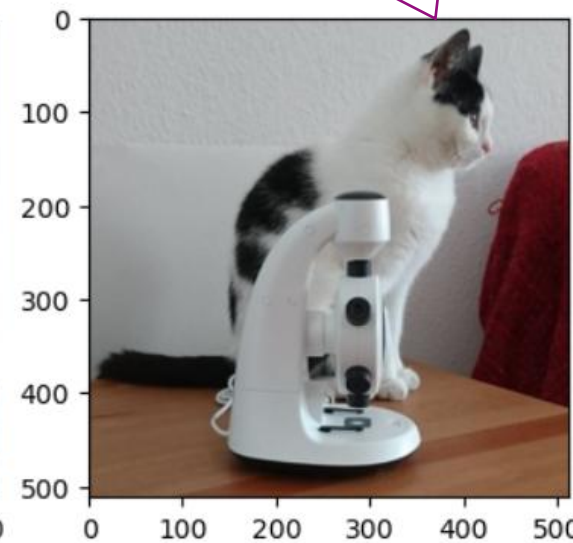
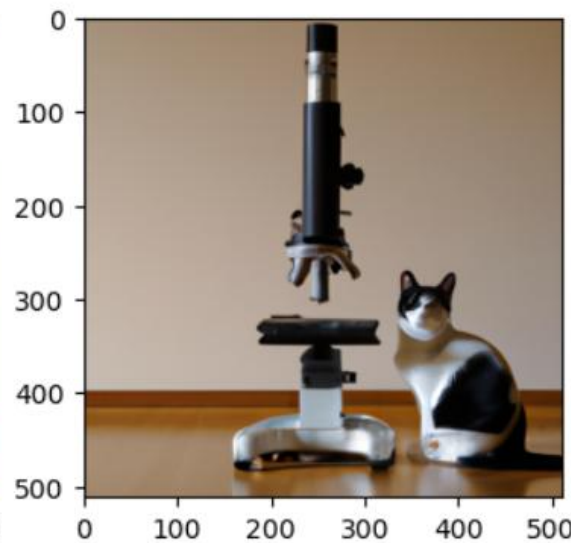
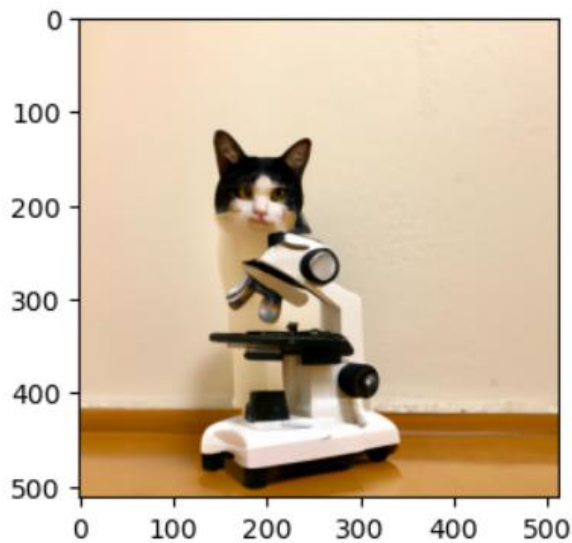


Benchmarking image generation

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 microscope.  
"""
```

One cat
is real.




CLIP scores

Contrastive Language-Image Pre-Training (CLIP)

- For image describing

Here: Similarity between image and prompt

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



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

tensor(25.3473)

```
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]

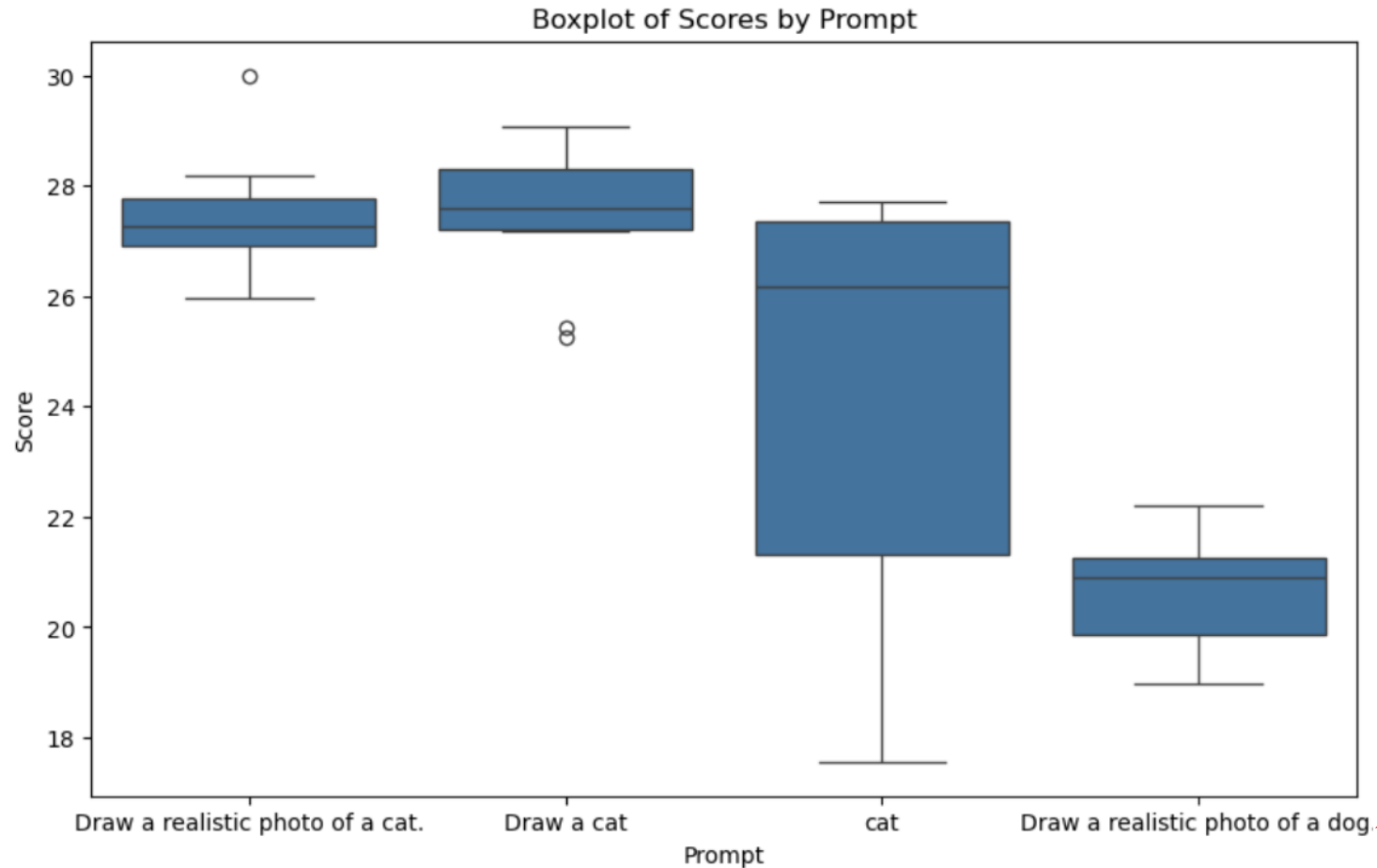
        score = metric(torch.as_tensor(np.array(image)), "cat")
        data["score"].append(float(score.detach()))
        data["prompt"].append(prompt)
```

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