ScaDS.AI
DRESDEN LEIPZIG

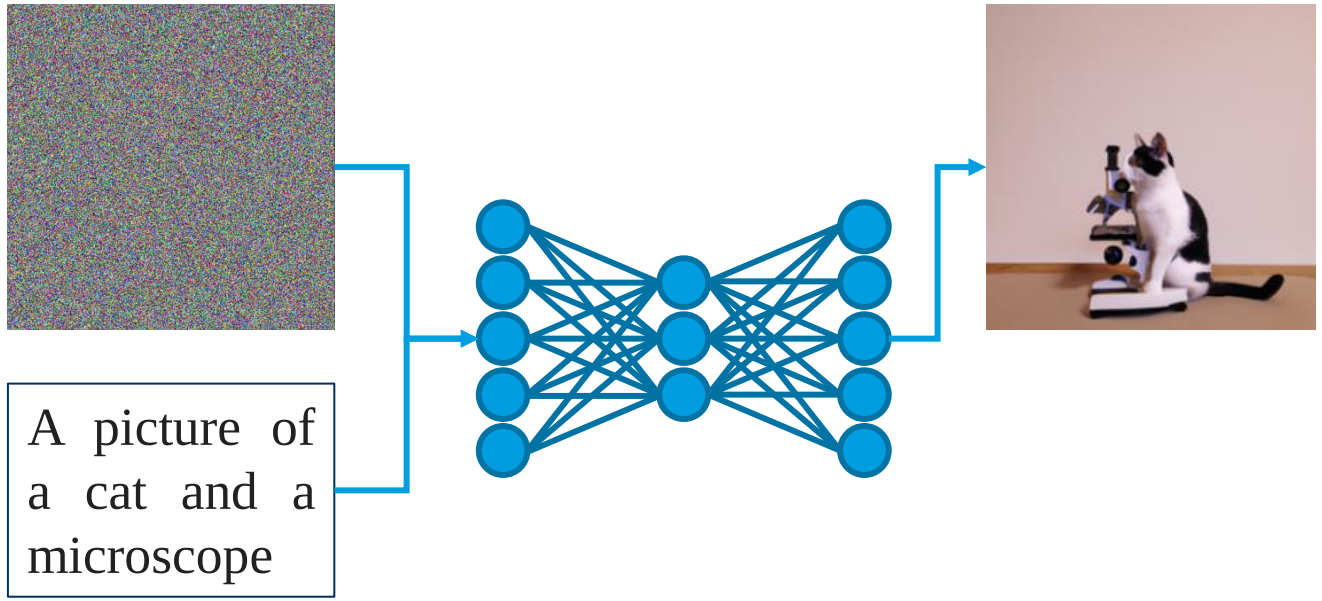
CENTER FOR SCALABLE DATA ANALYTICS AND
ARTIFICIAL INTELLIGENCE

# Image Generation and Vision Language Models

Robert Haase

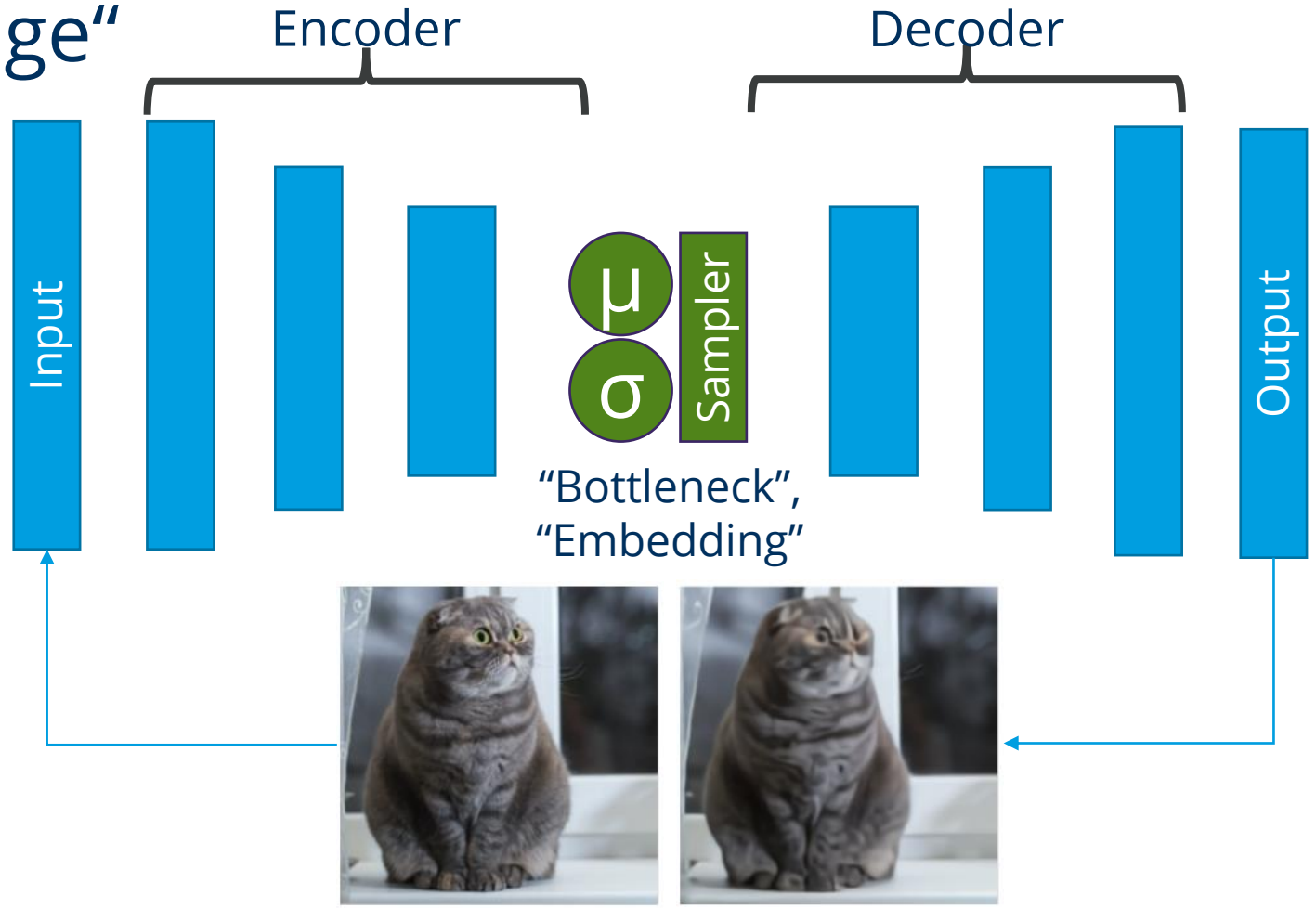
# Image Generation

„text-to-image“



# Variational Auto-Encoder

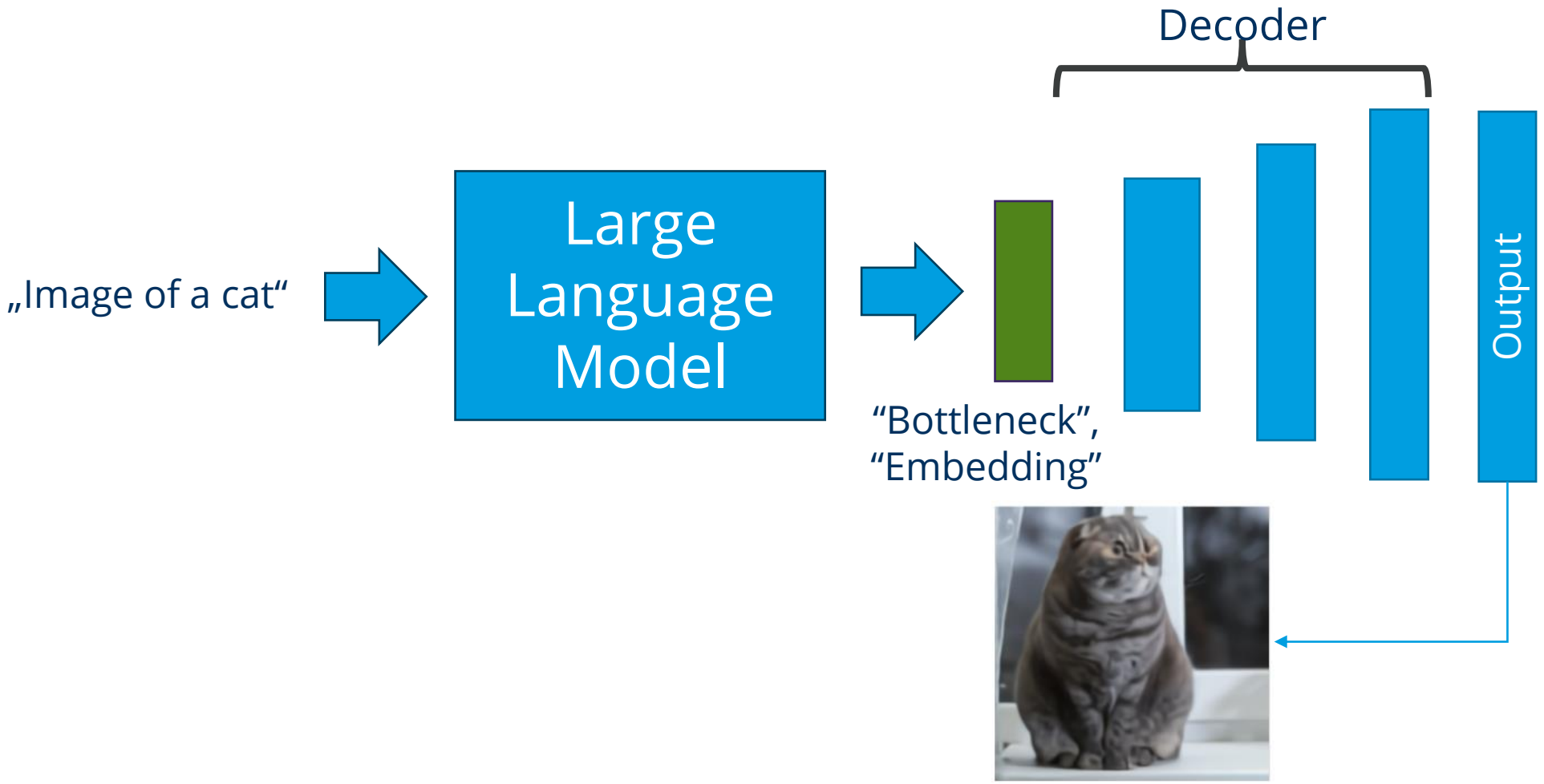
„image-to-image“





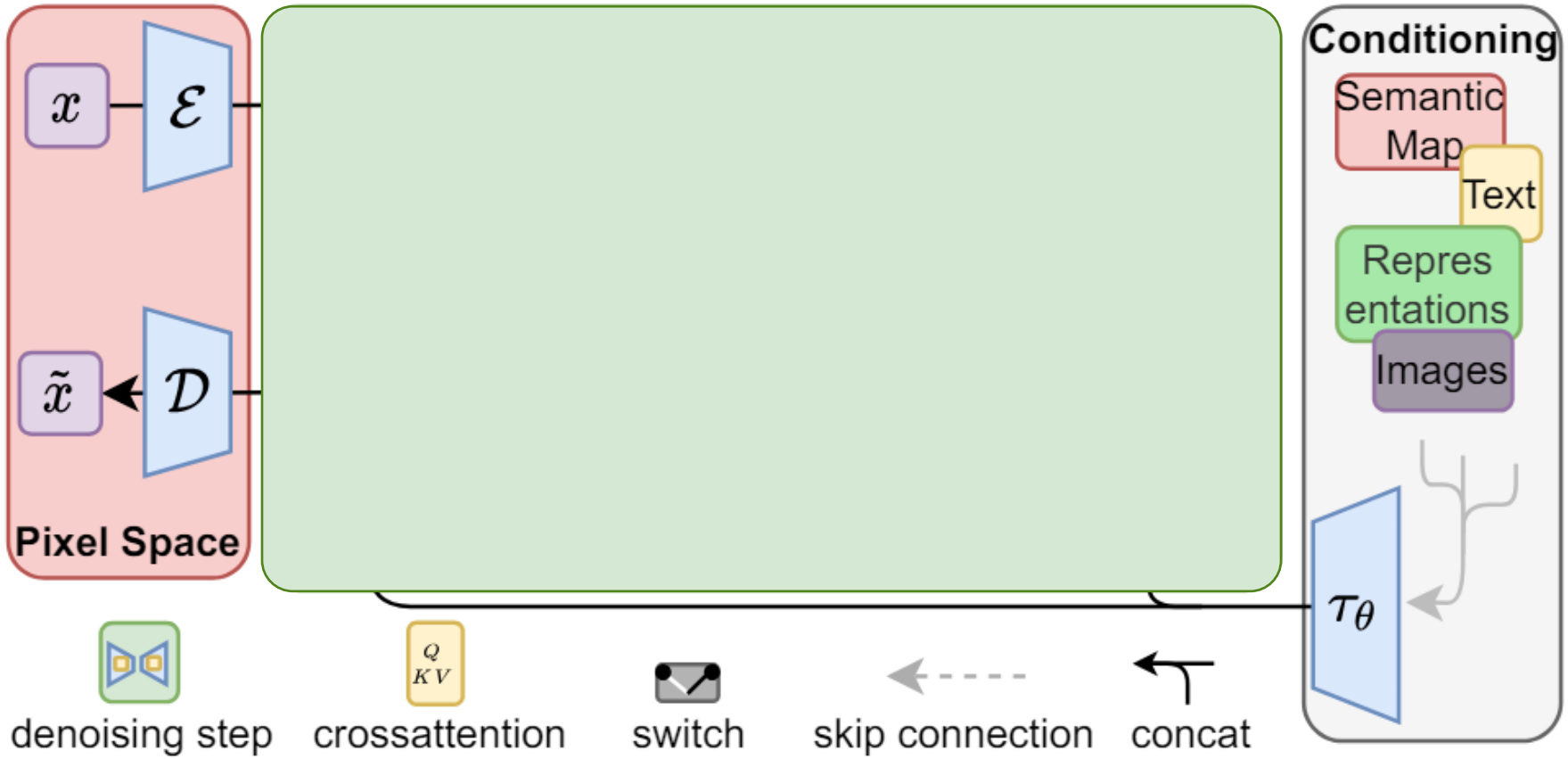
Cat picture source: Ramesh et al. 2021, licensed CC-BY 4.0 https://arxiv.org/pdf/2102.12092.pdf

# Image Generation



# Stable Diffusion

Diffusion: reverse denoising

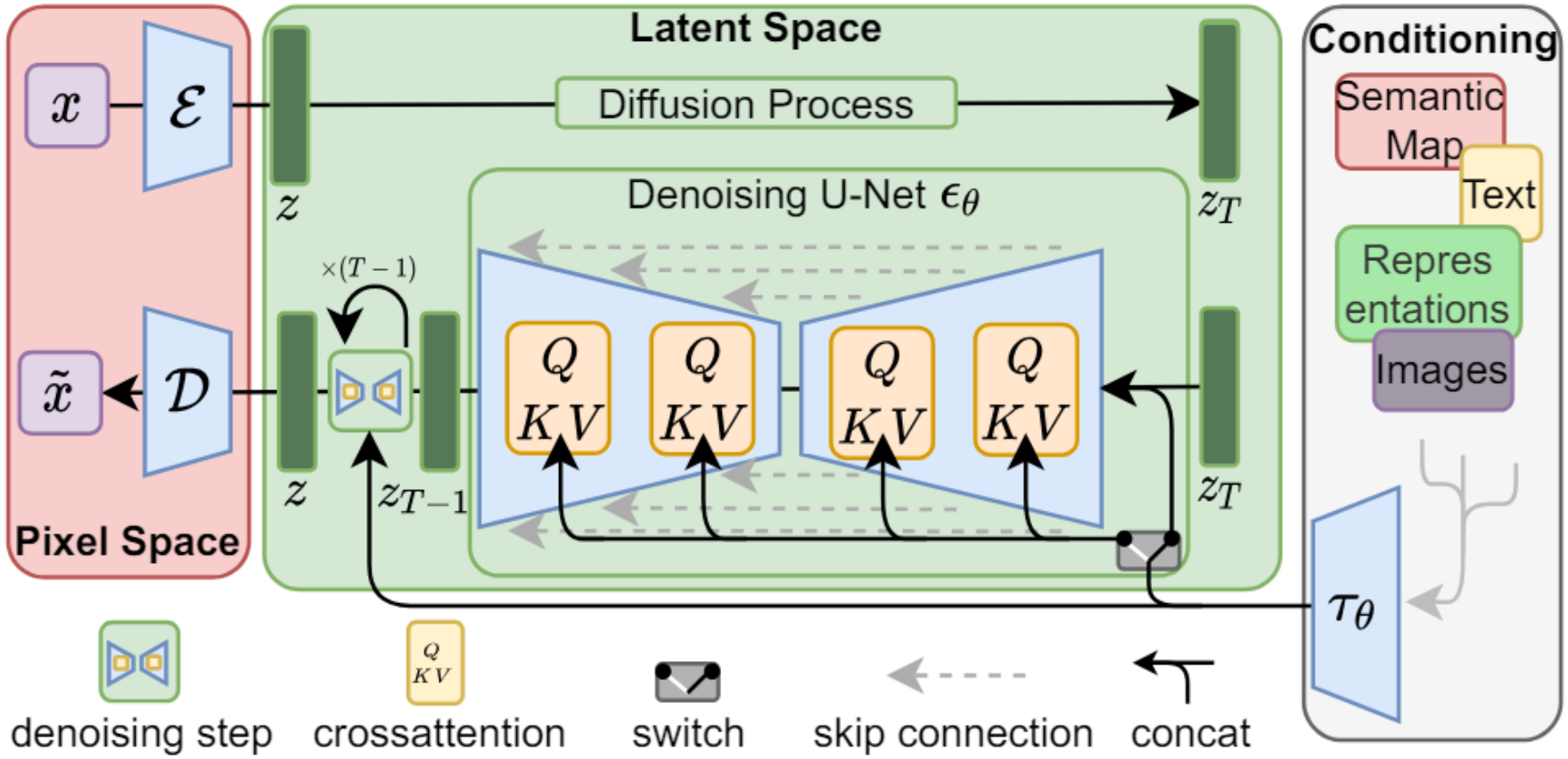





Simplified from: Rombach et al. 2021
https://arxiv.org/abs/2112.10752

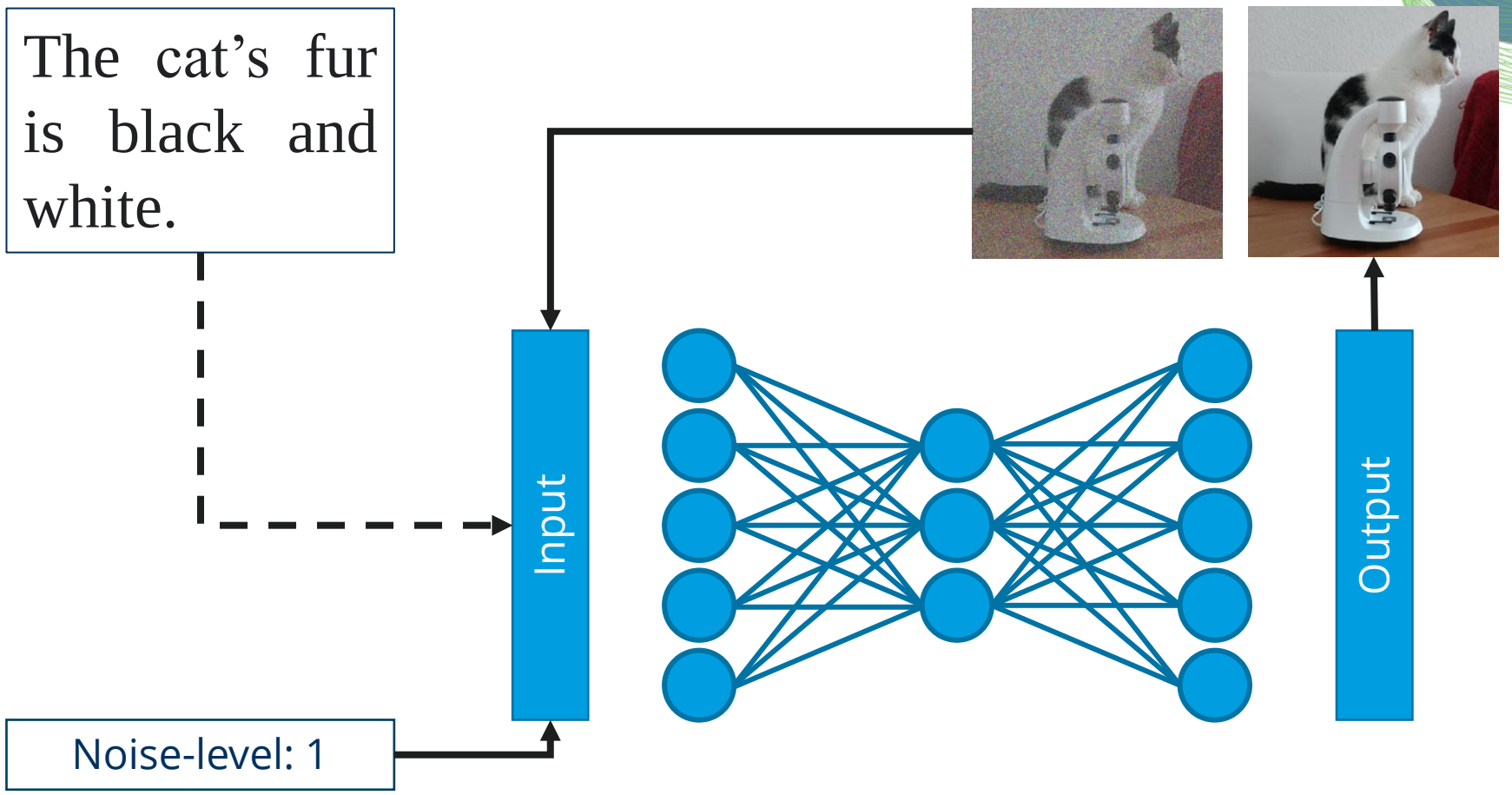
# Stable Diffusion

Diffusion: reverse denoising



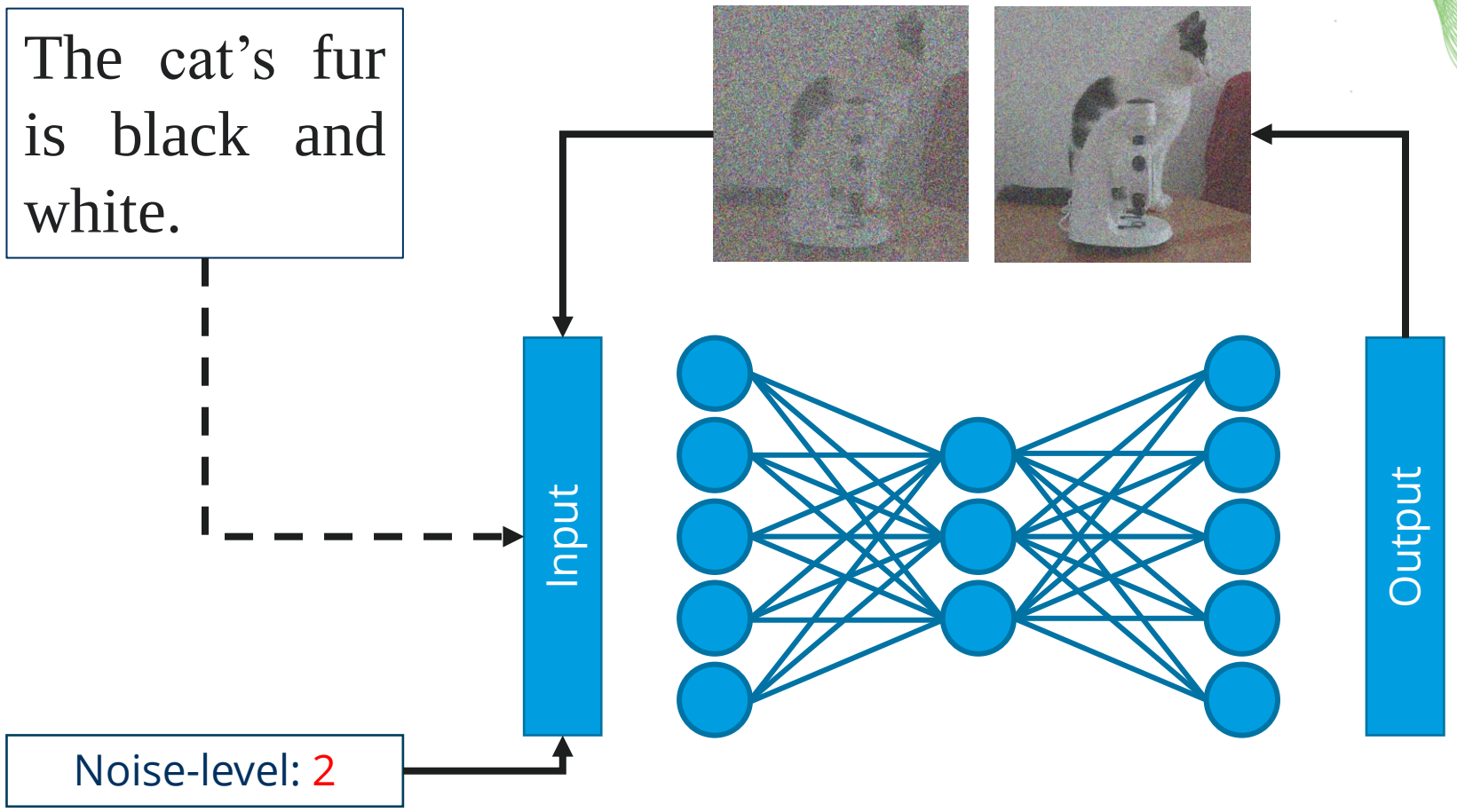
# How does it work?

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



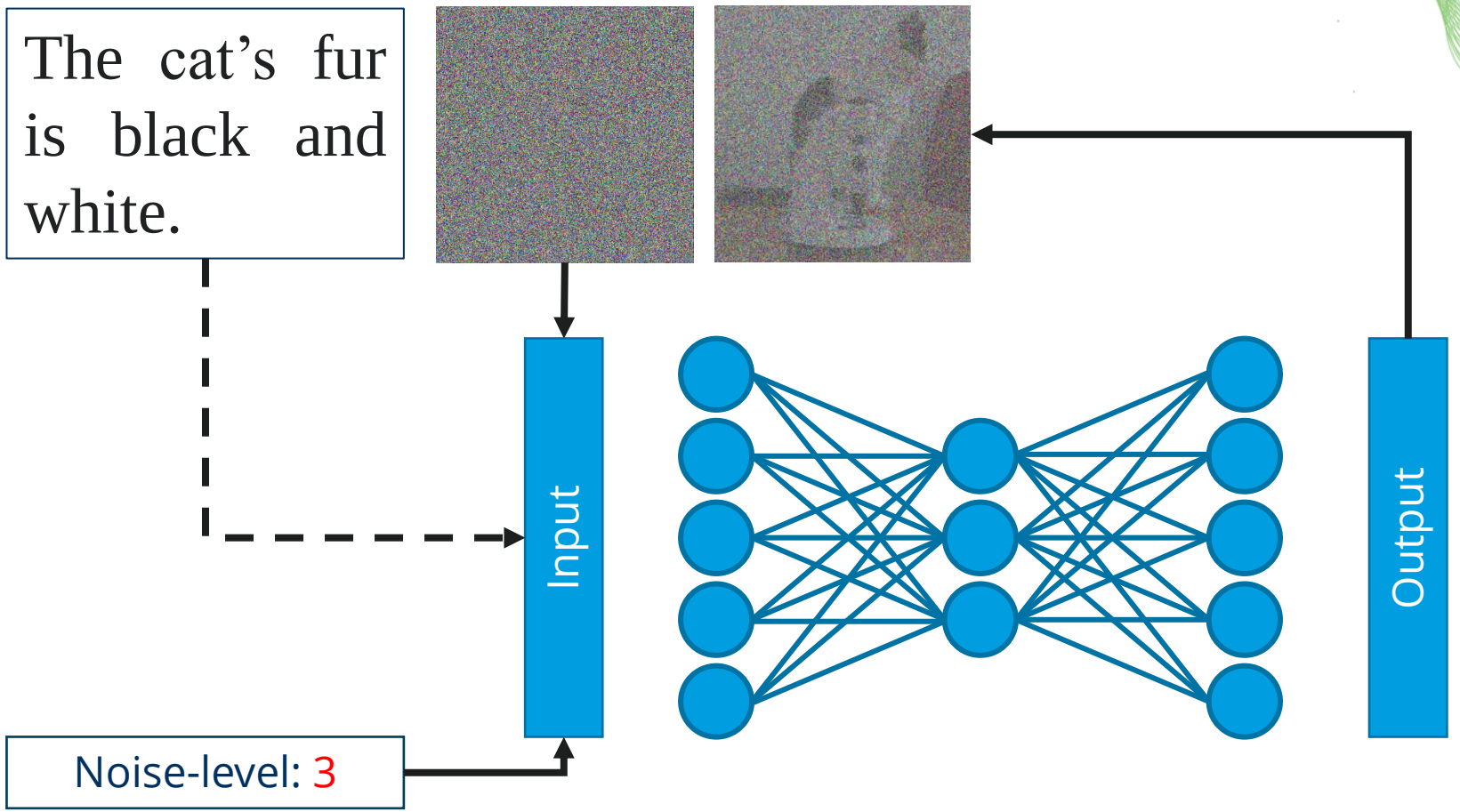
# How does it work?

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



# How does it work?

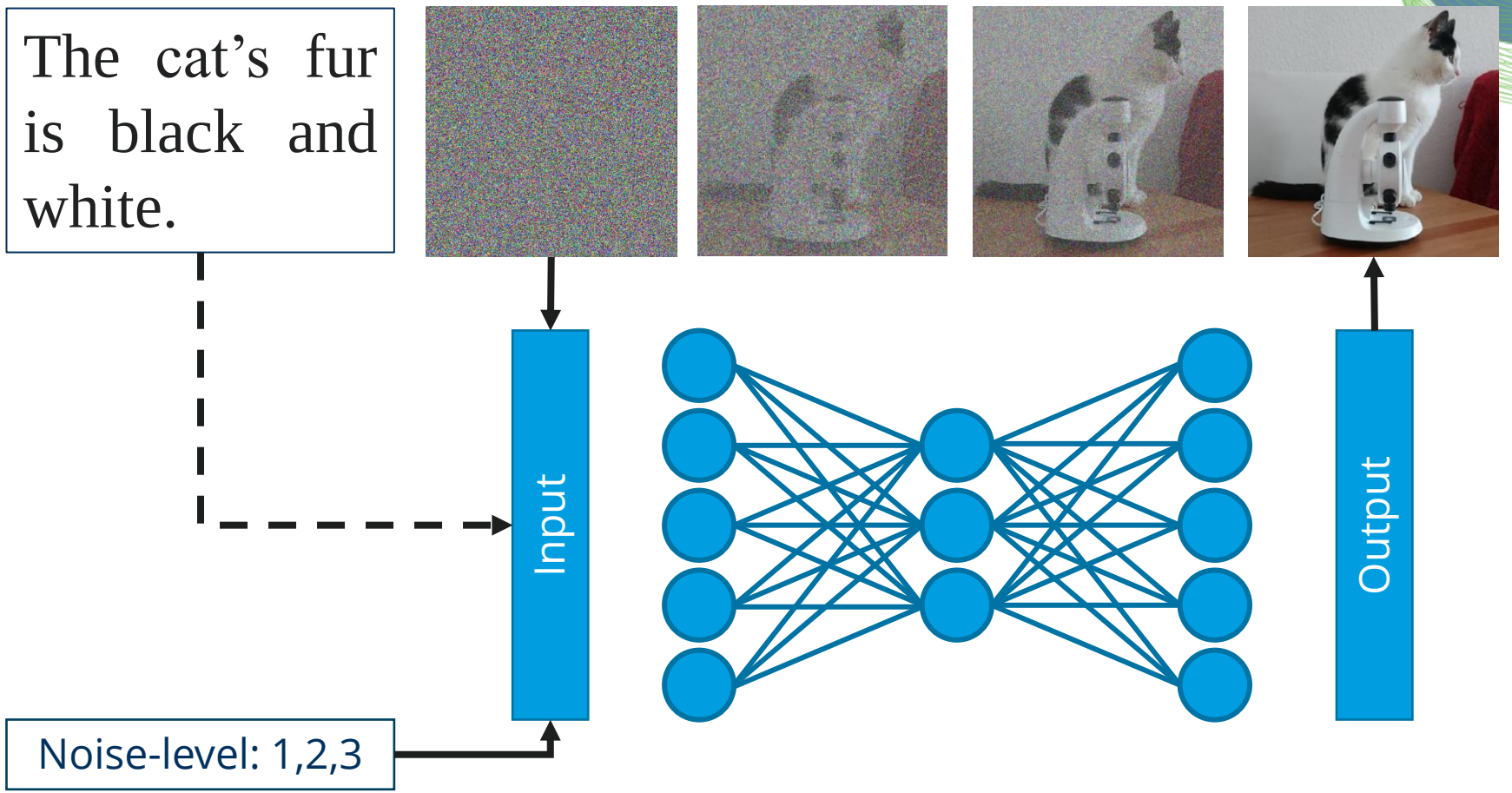
Train a U-Net on data: image + noisy image + description + noise-level



# How does it work?

Prediction is iterative denoising of:

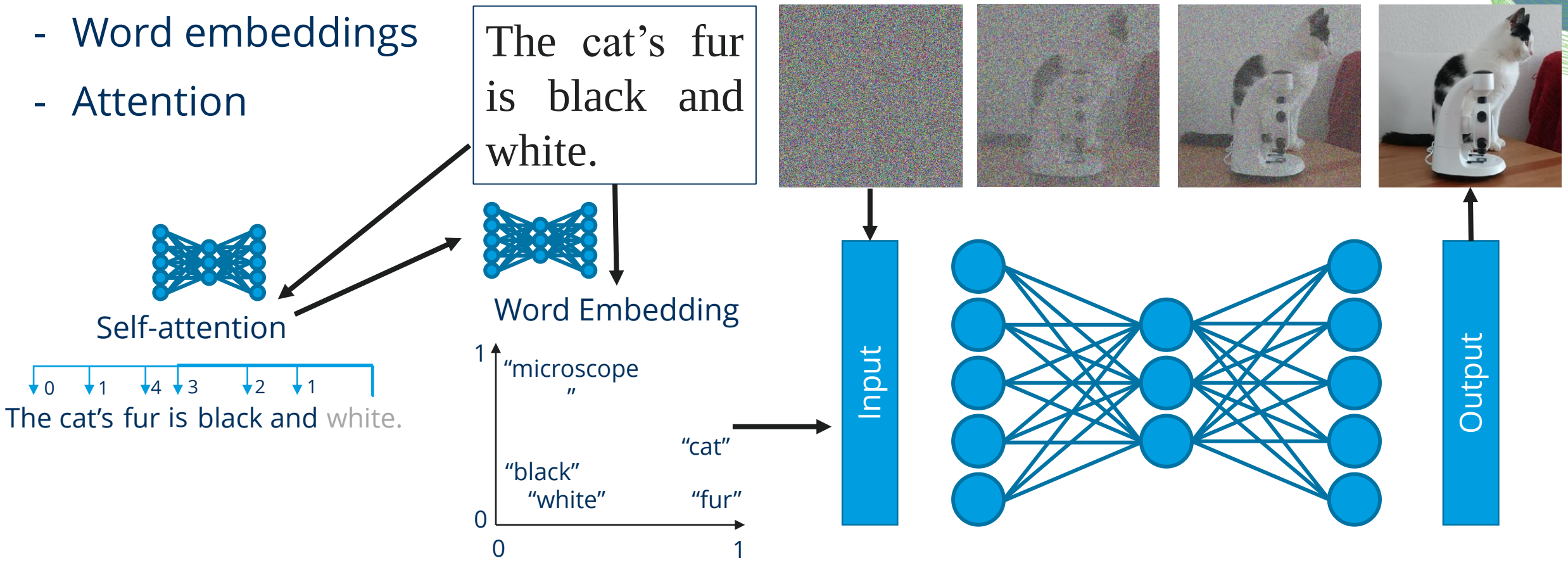
Pure noise + text prompt



# 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

# 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
```

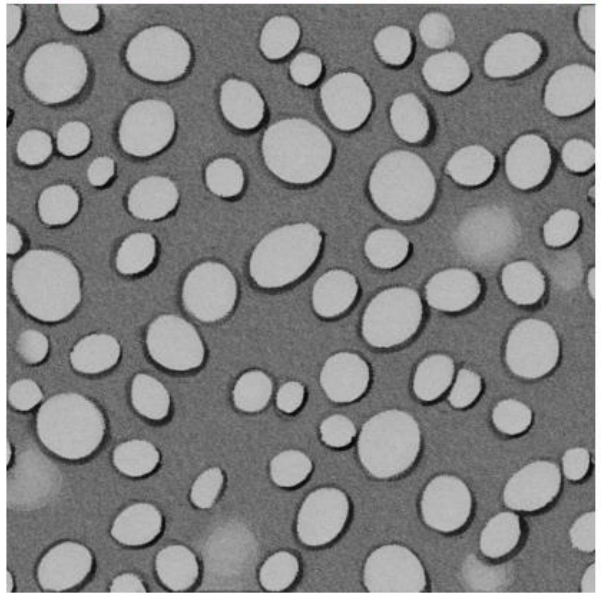
```
100%  50/50 [00:43<00:00,  1.24s/it]
```

```
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]
```

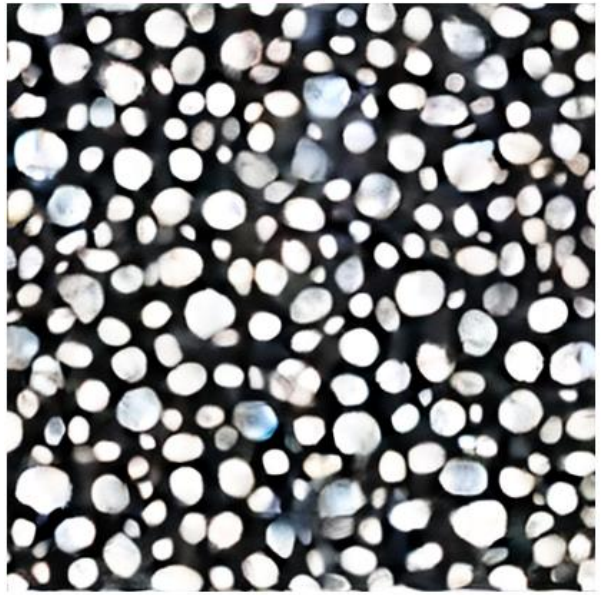
```
prompt = """
Draw a greyscale picture of sparse bright blobs on dark
background. Some of the blobs are roundish, some are a
bit elongated.
"""

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

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



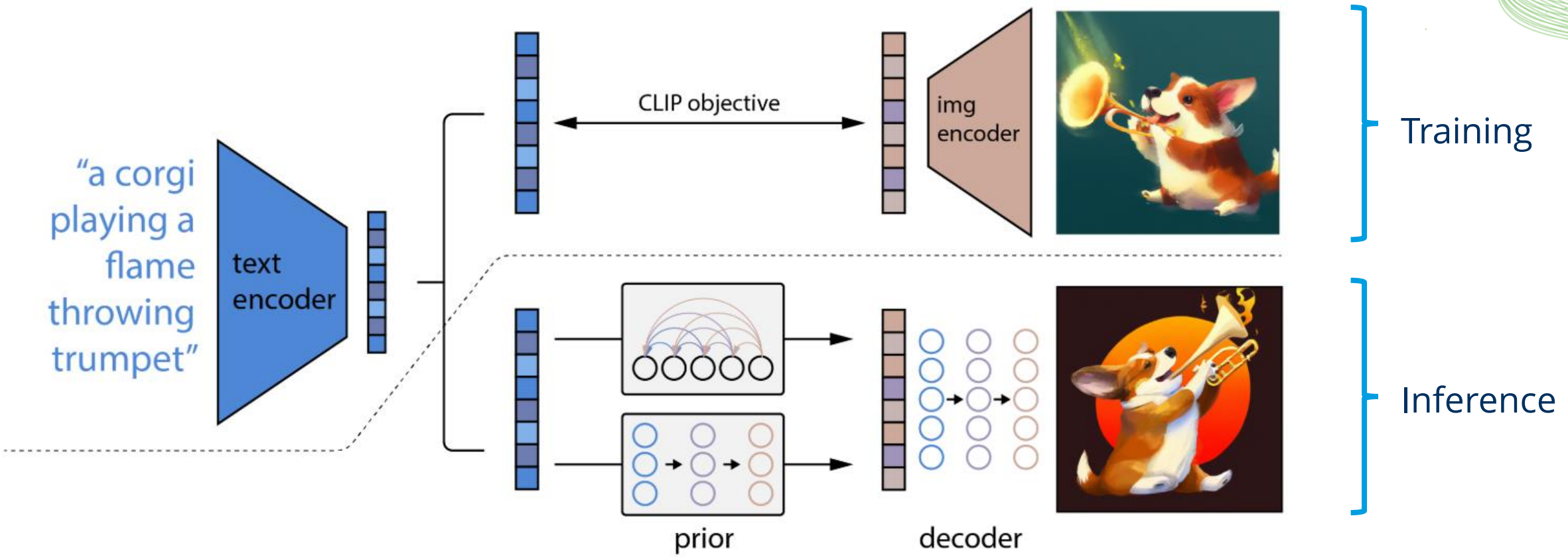
```
image = pipe(prompt,
             num_inference_steps=10,
             width=512,
             height=512).images[0]
image
```

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



# Dall-E

OpenAI's model for image geneation based on diffusion models + CLIP transformer



# 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

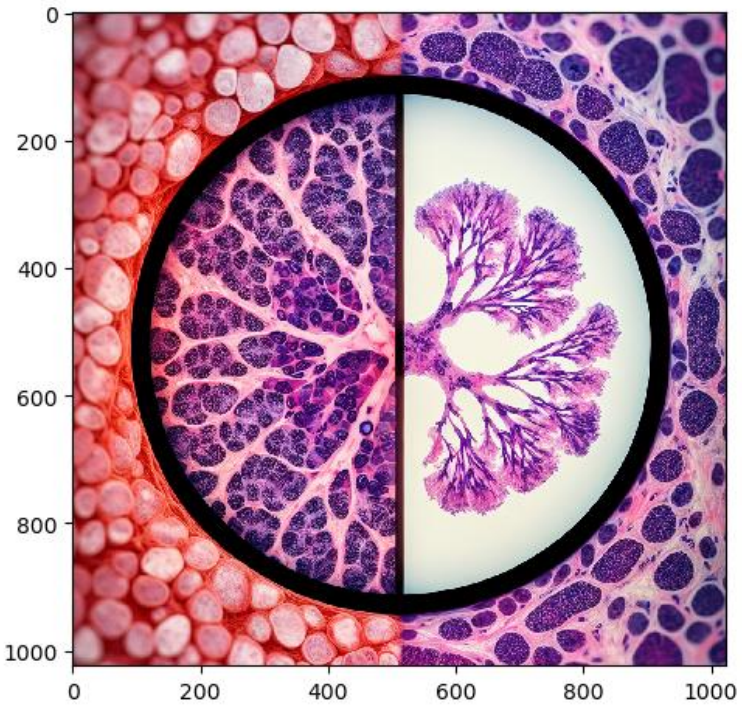
May soon also work with gpt-4o

# Image generation 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)
```

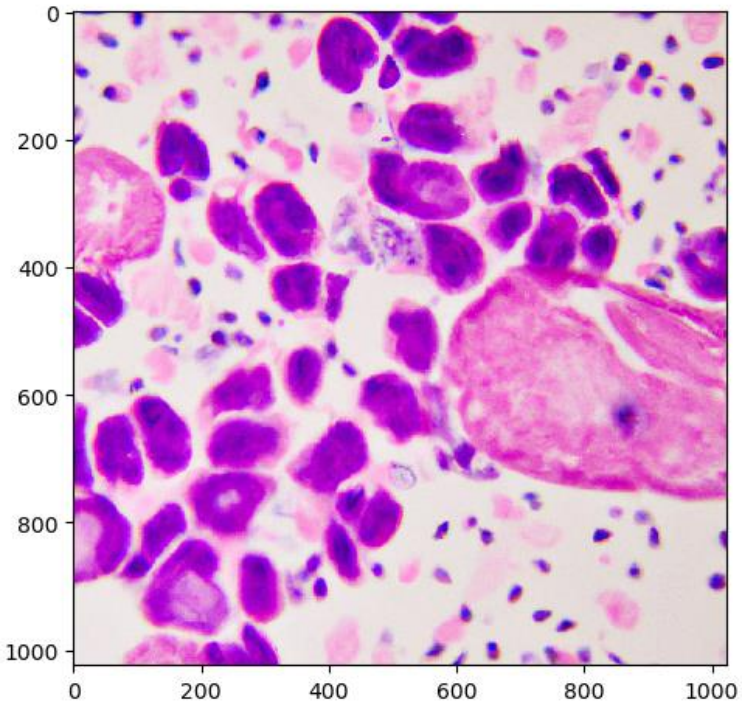
```
<matplotlib.image.AxesImage at 0x1d9eda5bd90>
```




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>
```

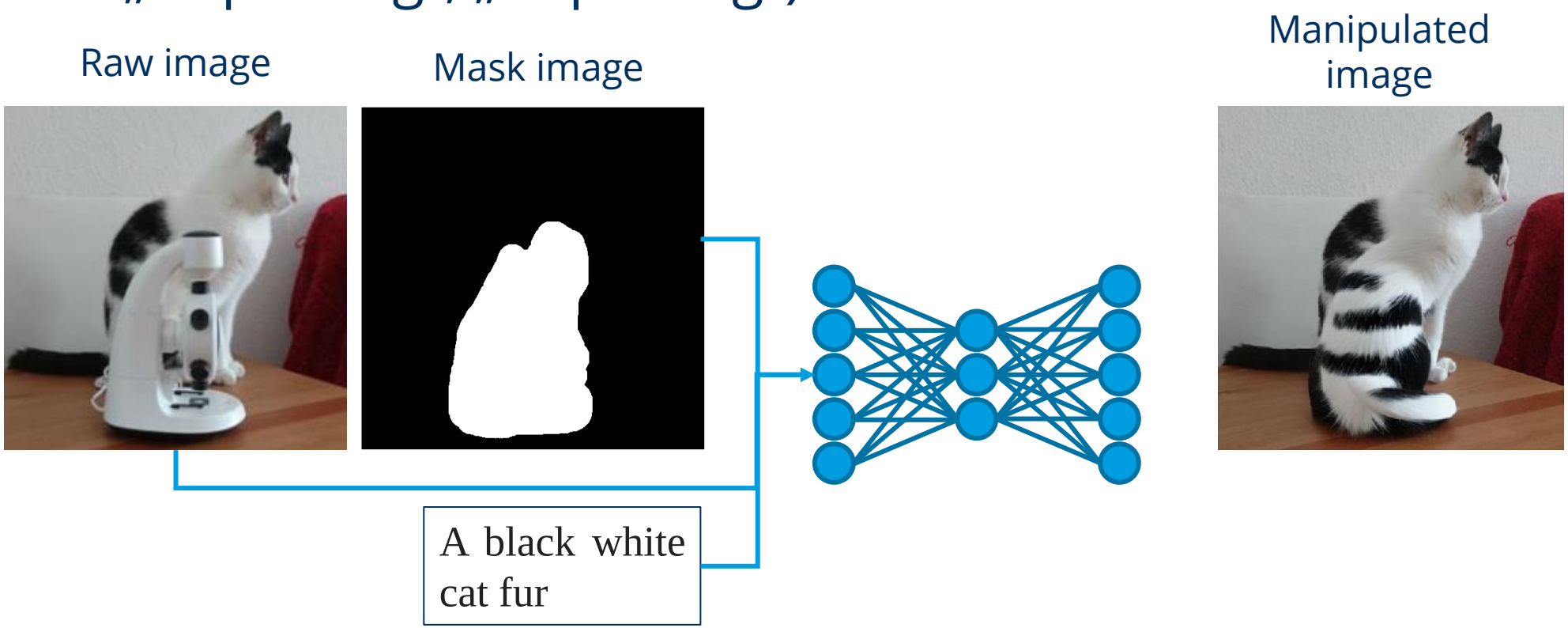



Dall-E 2

# Inpainting

Replacing regions in images

(also „Gap-filling“, „Replacing“)

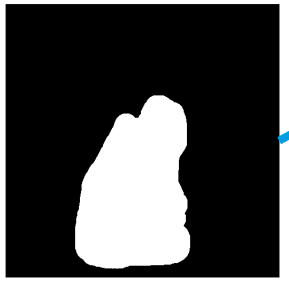


# Inpainting in Python: Huggingface

```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



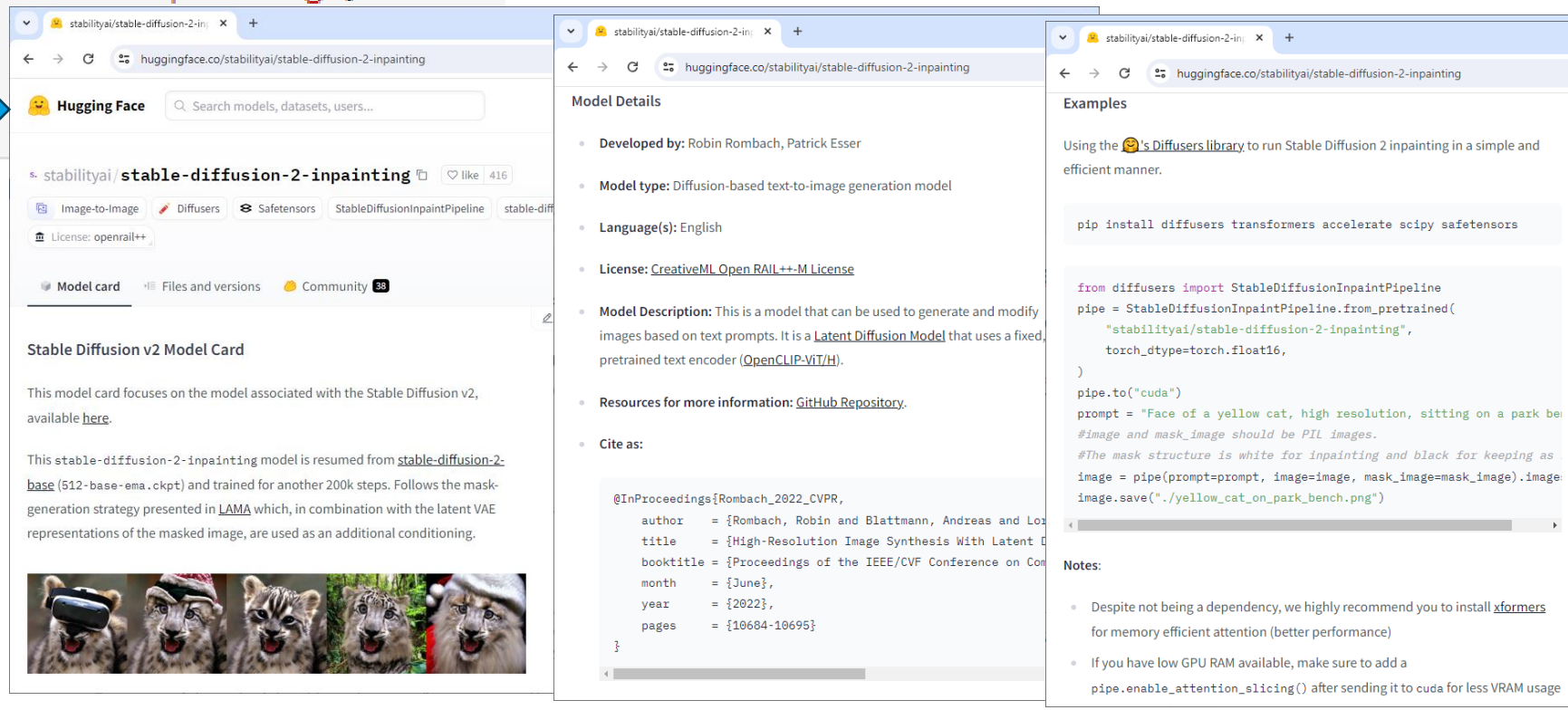
```python
prompt = "A black white cat fur"
image = pipe(prompt=prompt,
             image=init_image,
             mask_image=mask_image,
             num_inference_steps=50,
             width=512,
             height=512,
             num_images_per_prompt=1,
            ).images[0]
```

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.

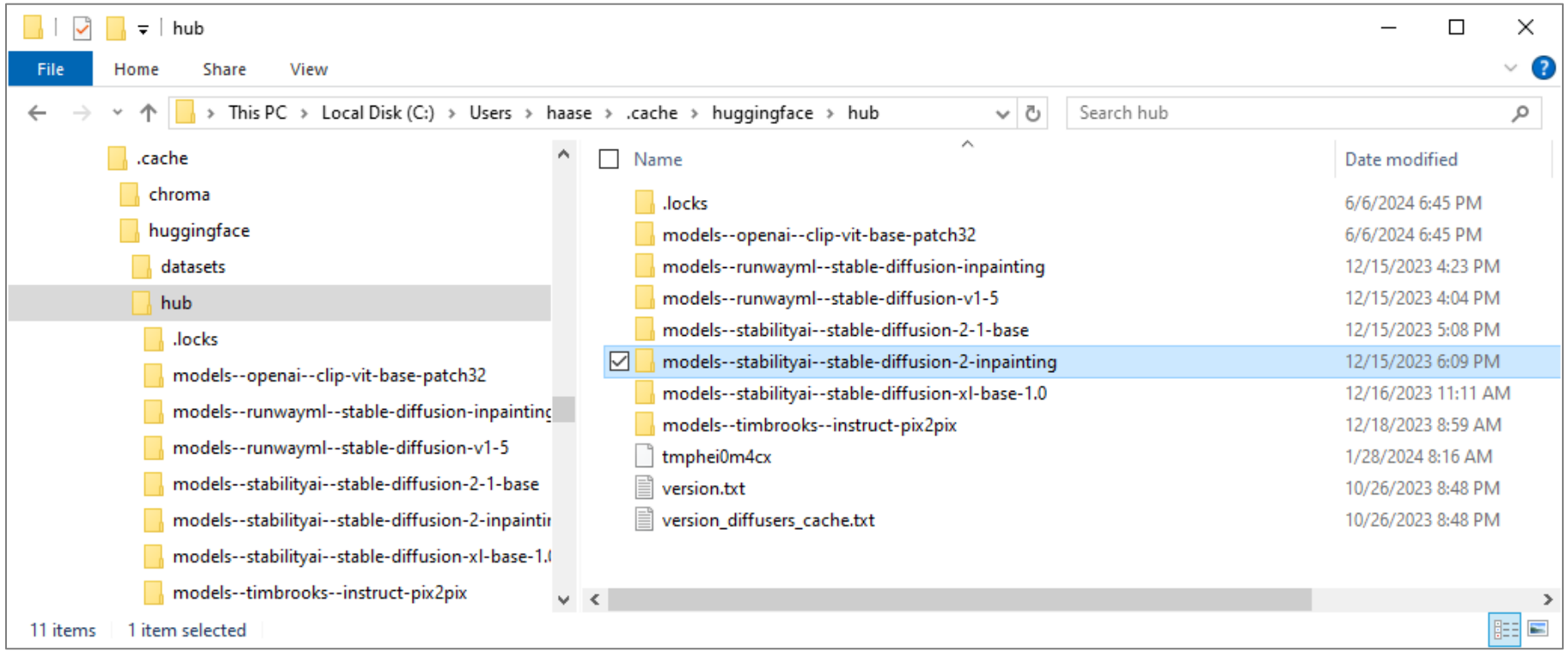
```
pipe = StableDiffusionInpaintPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-inpainting",
    torch_dtype=torch.float16
)
pipe = pipe.to("cuda")
```



# 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 

```python
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

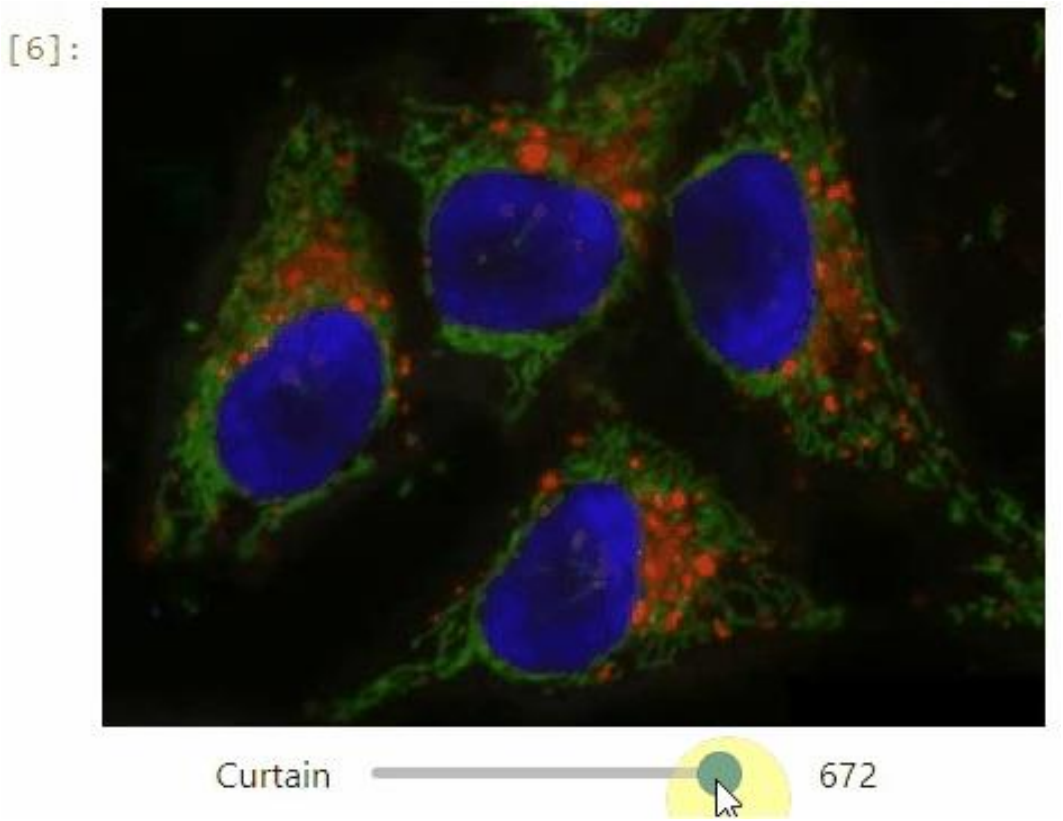
Size must match

Supported: 256, 512, 1024 pixels

Result: List of URL(s)

# New technologies bring new risks...

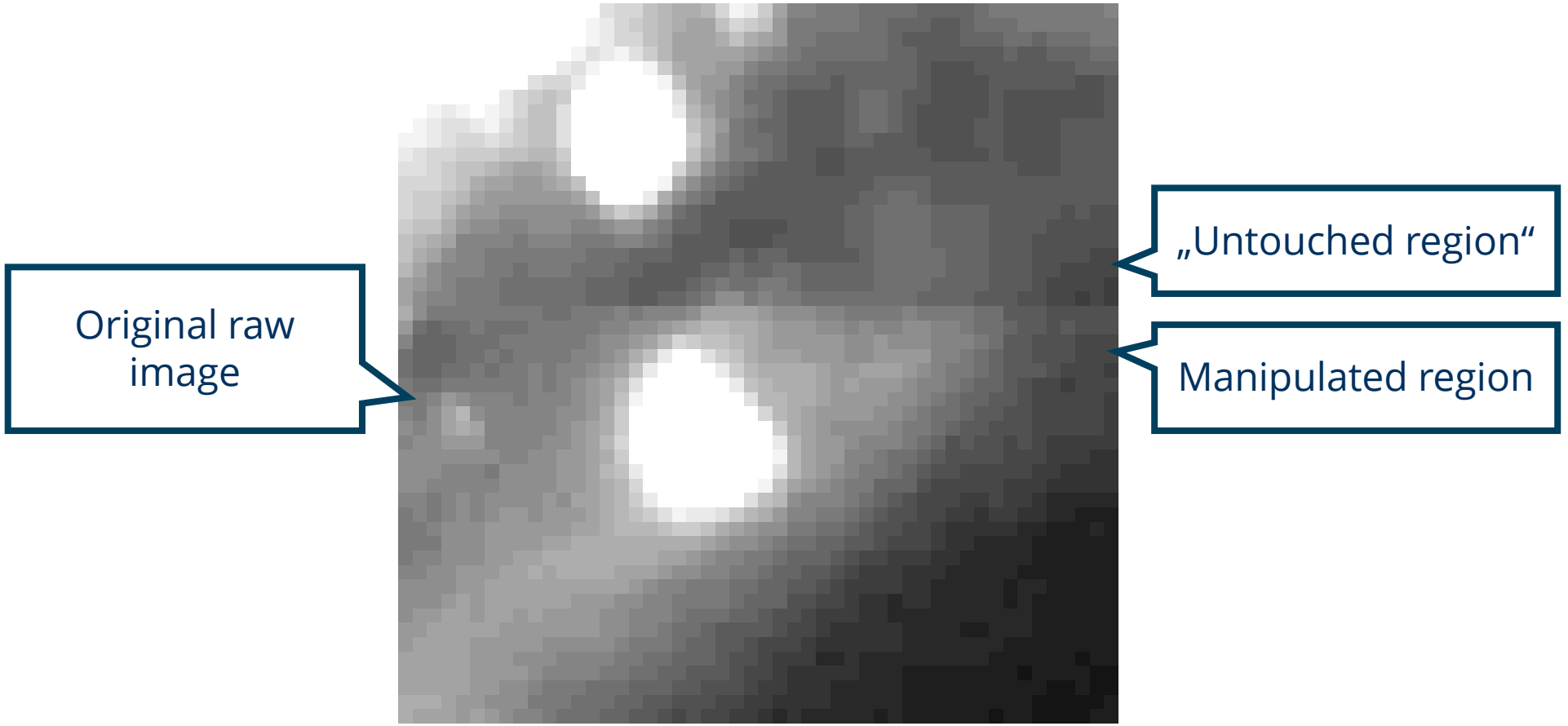
If you can generate images,
you can also generate parts of images....




Interesting challenges for our community ahead

# Image manipulation detection

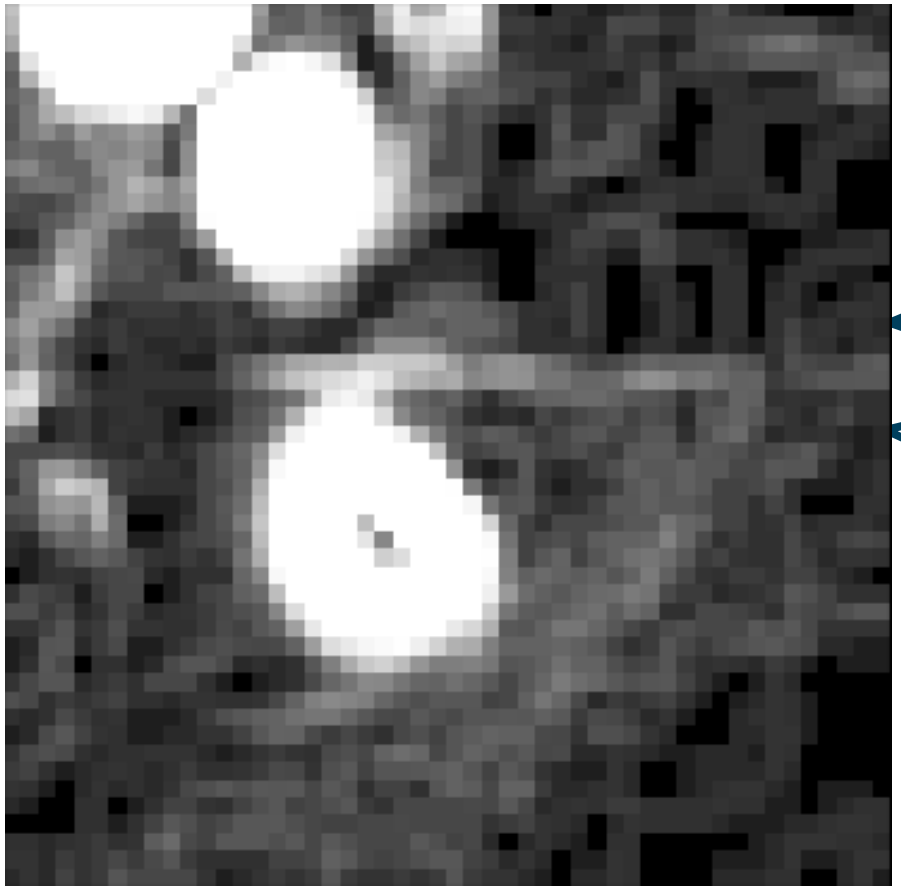
The noise pattern differs between raw and processed images…



# Image manipulation detection

## e.g. by studying noise-patterns

Local standard deviation filter



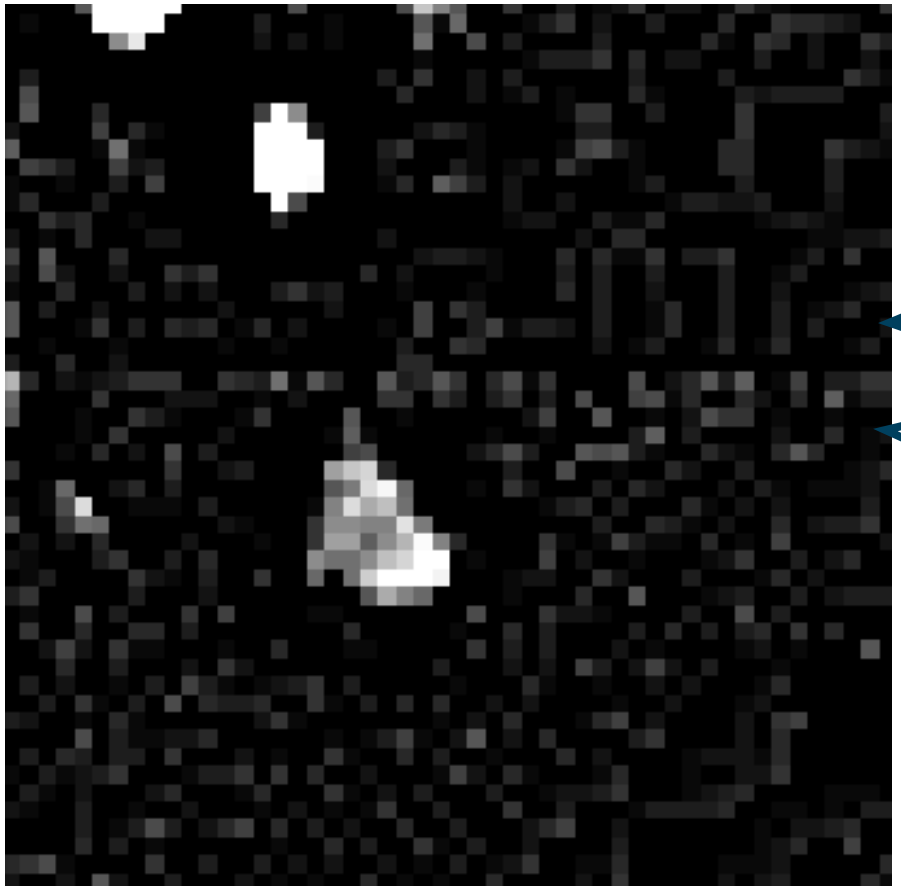
„Untouched region“

Manipulated region

# Image manipulation detection

e.g. by studying noise-patterns

Sobel filter

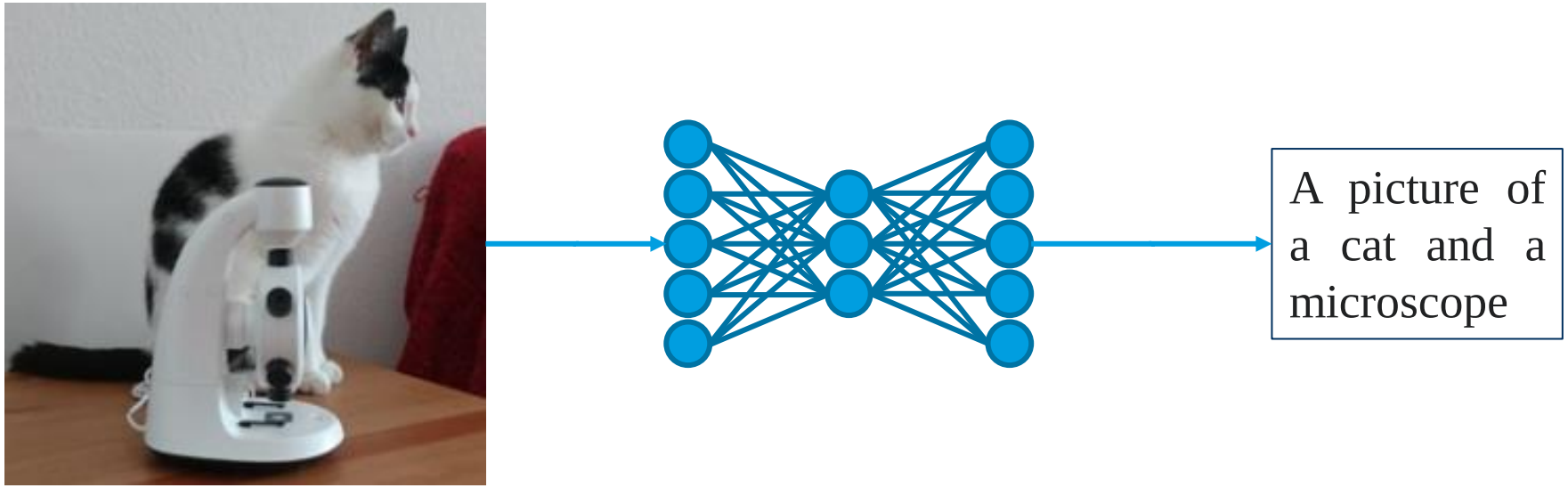


„Untouched region“

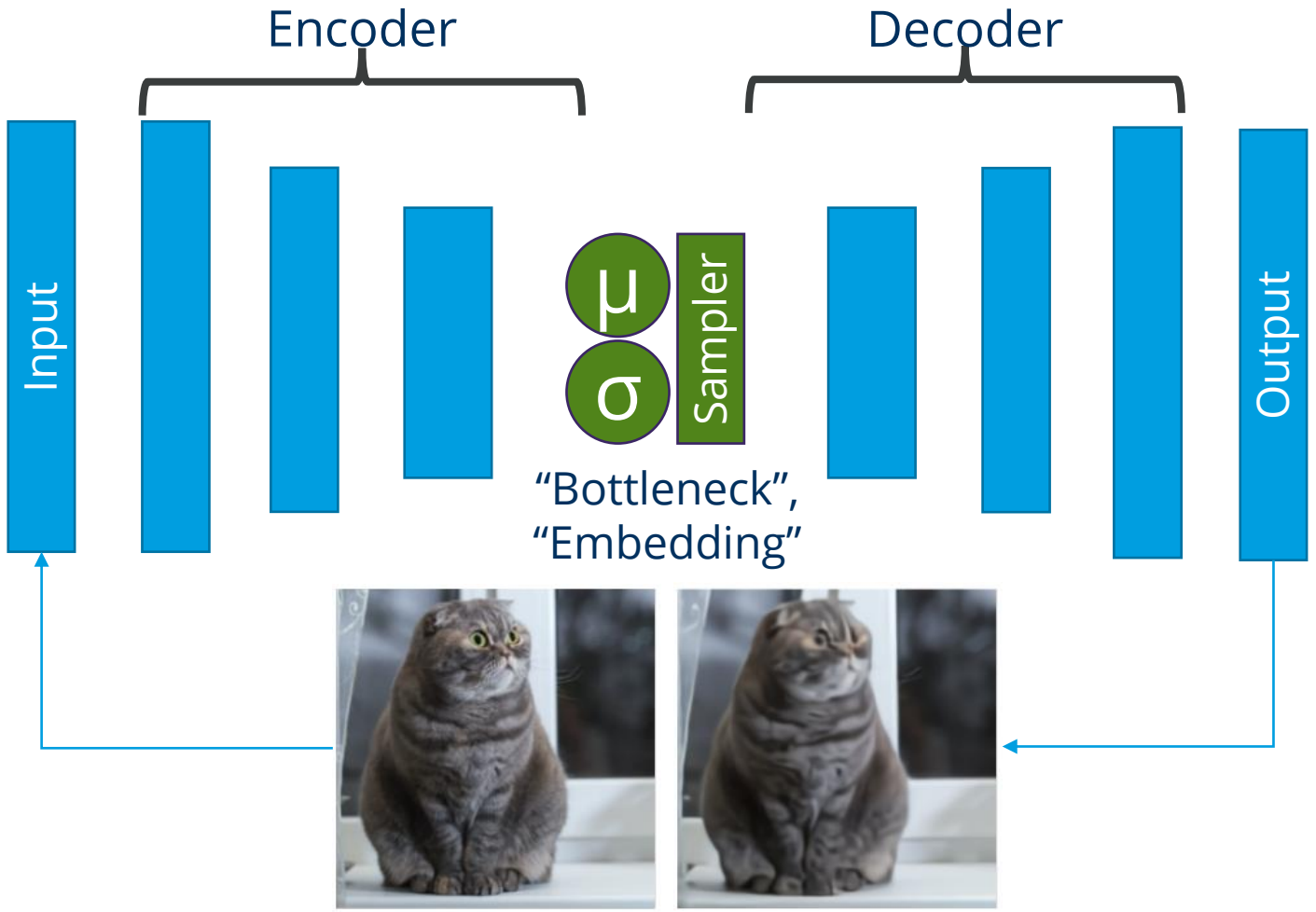
Manipulated region

# Vision Language Models

- Classifying images 🤧
- Describing images



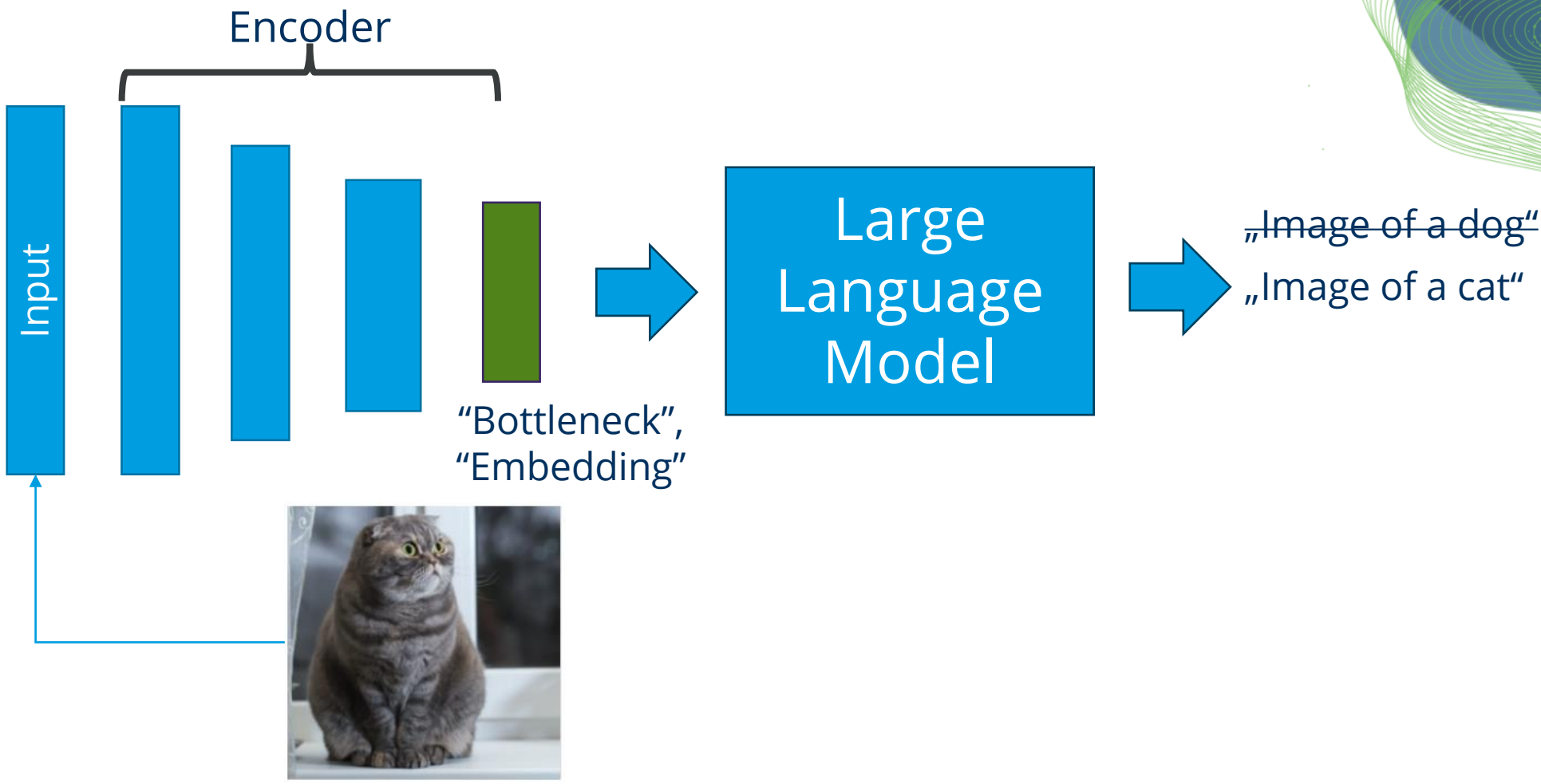
# Variational Auto-Encoder





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

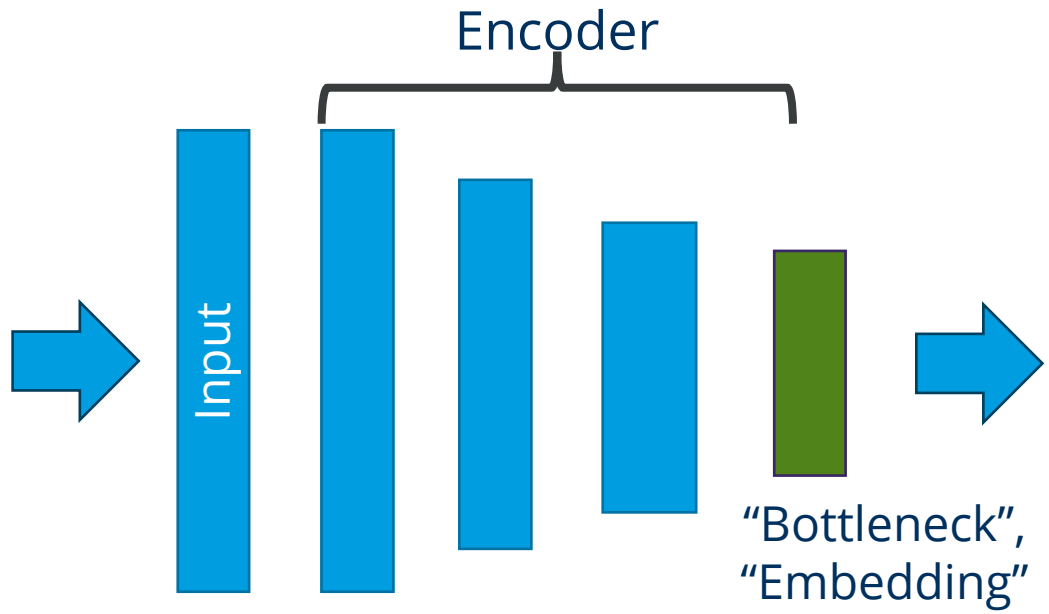
# Image classification





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

# Image classification -> image describing




Large Language Model

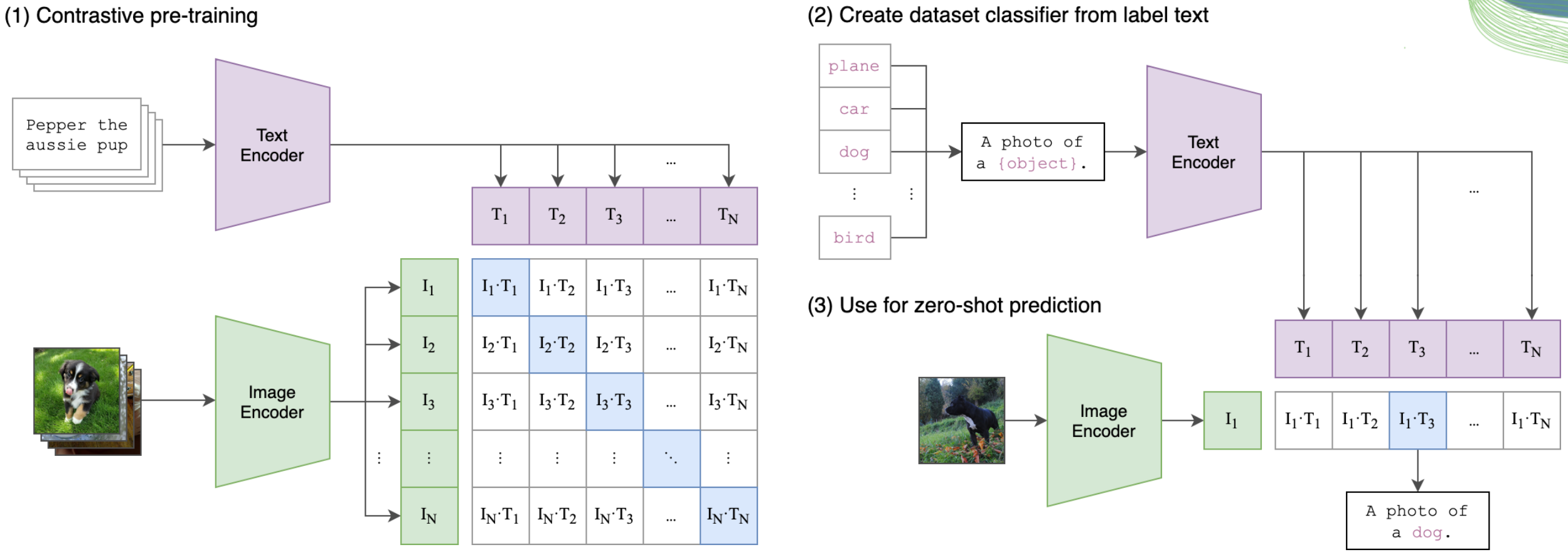
~~„Image of a dog“~~

„Image of a cat“

„Image of a microscope“

# 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
```

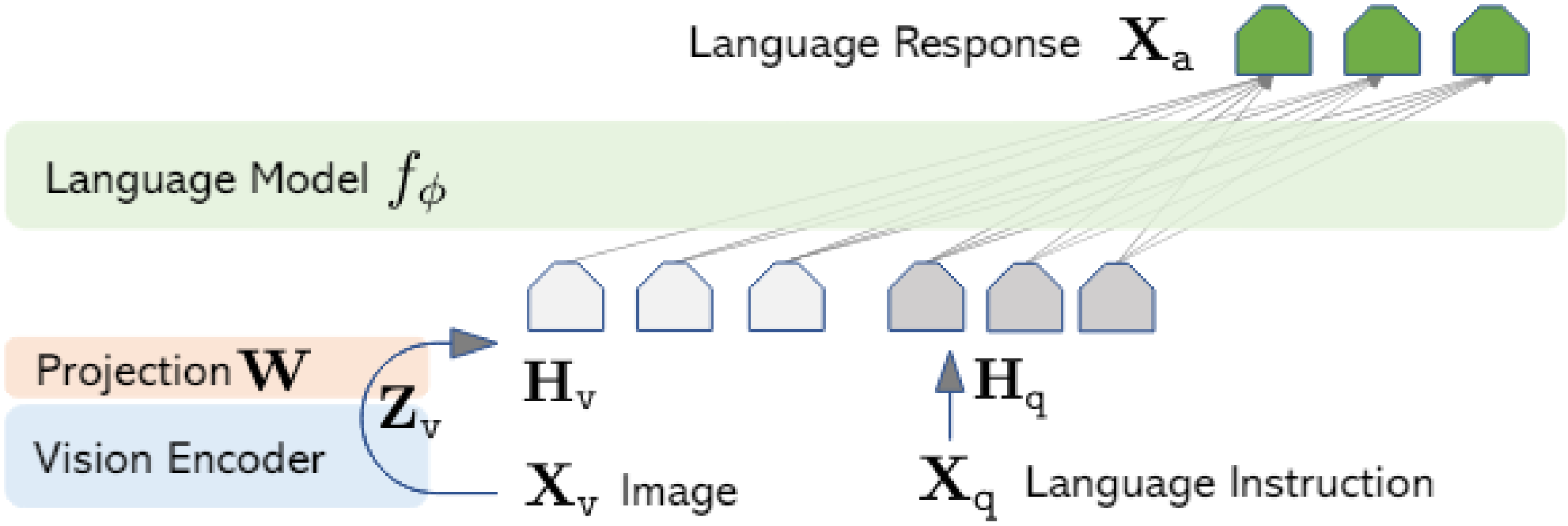
```
{'a photo of a cat': 0.9907298684120178,
 'a photo of a dog': 0.009270114824175835}
```

```
label_probabilities
```

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

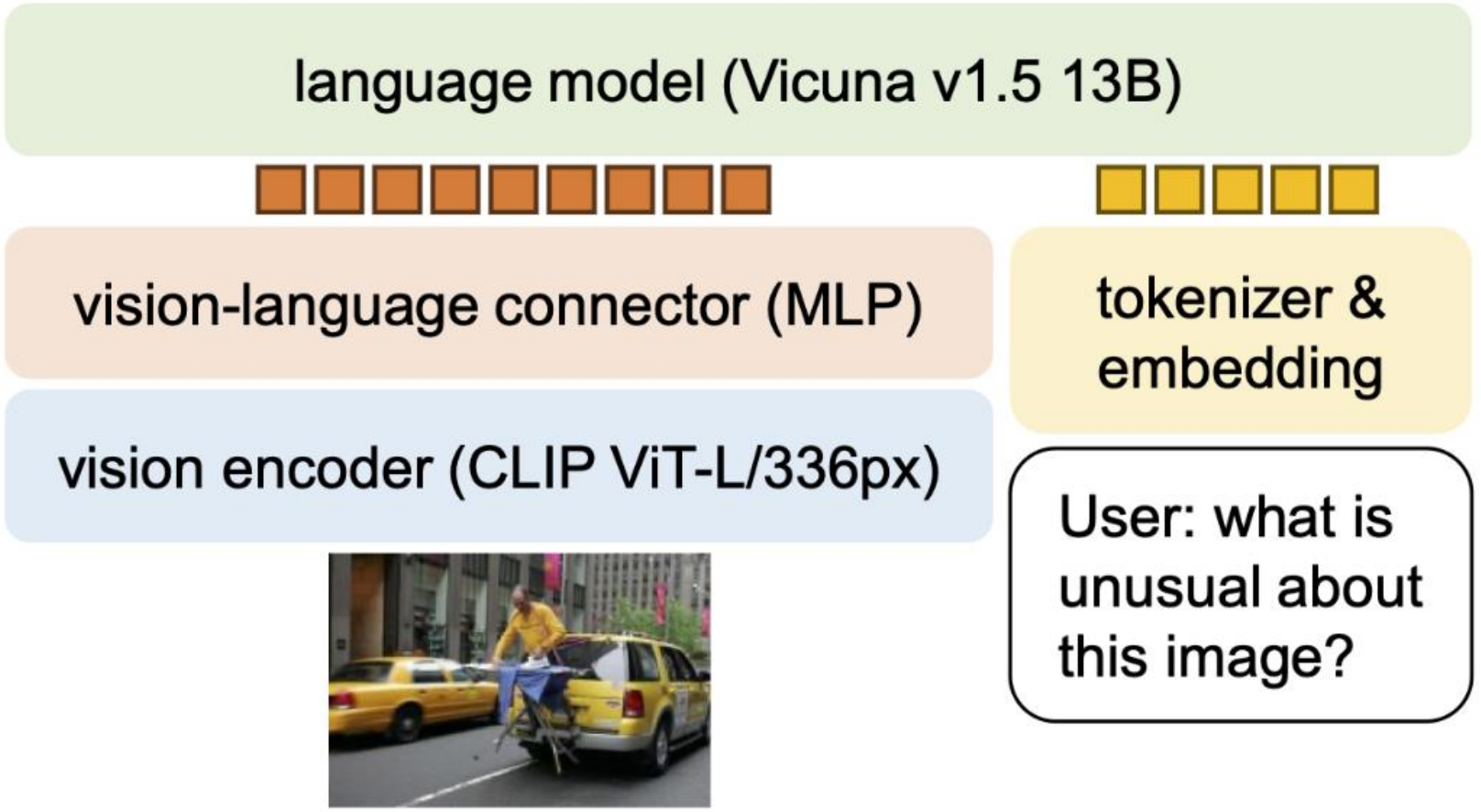
# LLAVA

## Large Language and Vision Assistant



# LLAVA 1.5

## Combining LLAVA with CLIP

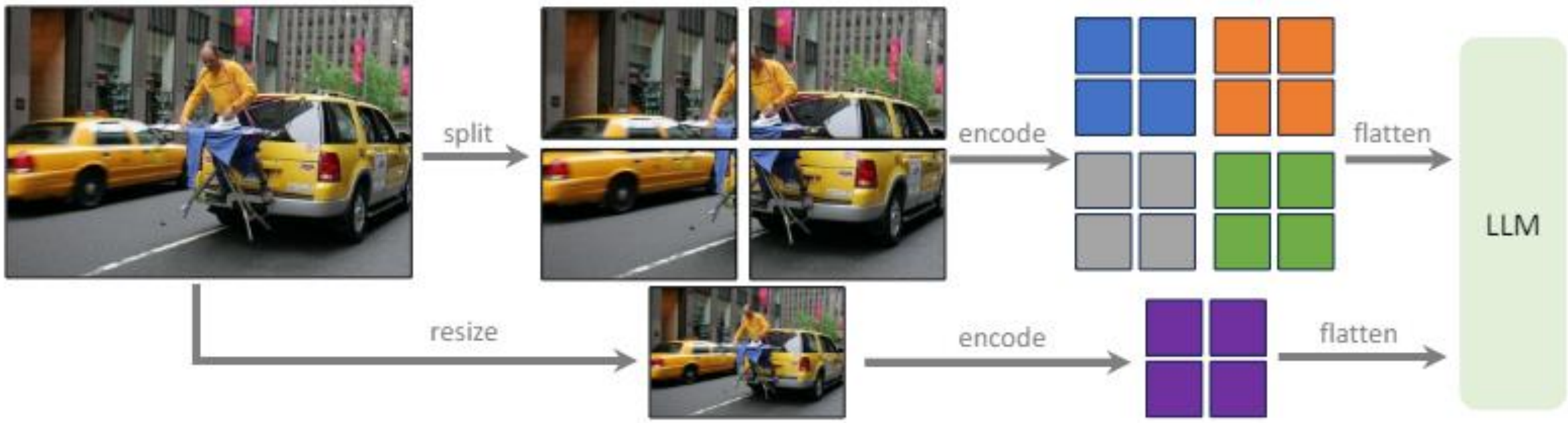





Source: Hui et al 2024, licensed CC-BY 4.0
https://arxiv.org/abs/2310.03744

# LLAVA 1.5 HD

Giving the model multiple perspectives on the same scene



# Accessing VLMs using Python

## API not standardized (yet)

```python
def prompt_chatGPT(prompt:str, image, model="gpt-4o"):
    """A prompt helper function that sends a message to openAI
    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()

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

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

```python
def prompt_ollama(prompt:str, image, model="llava"):
    """A prompt helper function that sends a message to ollama
    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': prompt,
        'images': [base64_image]
    }]

    # setup connection to the LLM
    client = openai.OpenAI(
        base_url = "http://localhost:11434/v1"
    )

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

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

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

# Exercises

Robert Haase

Funded by

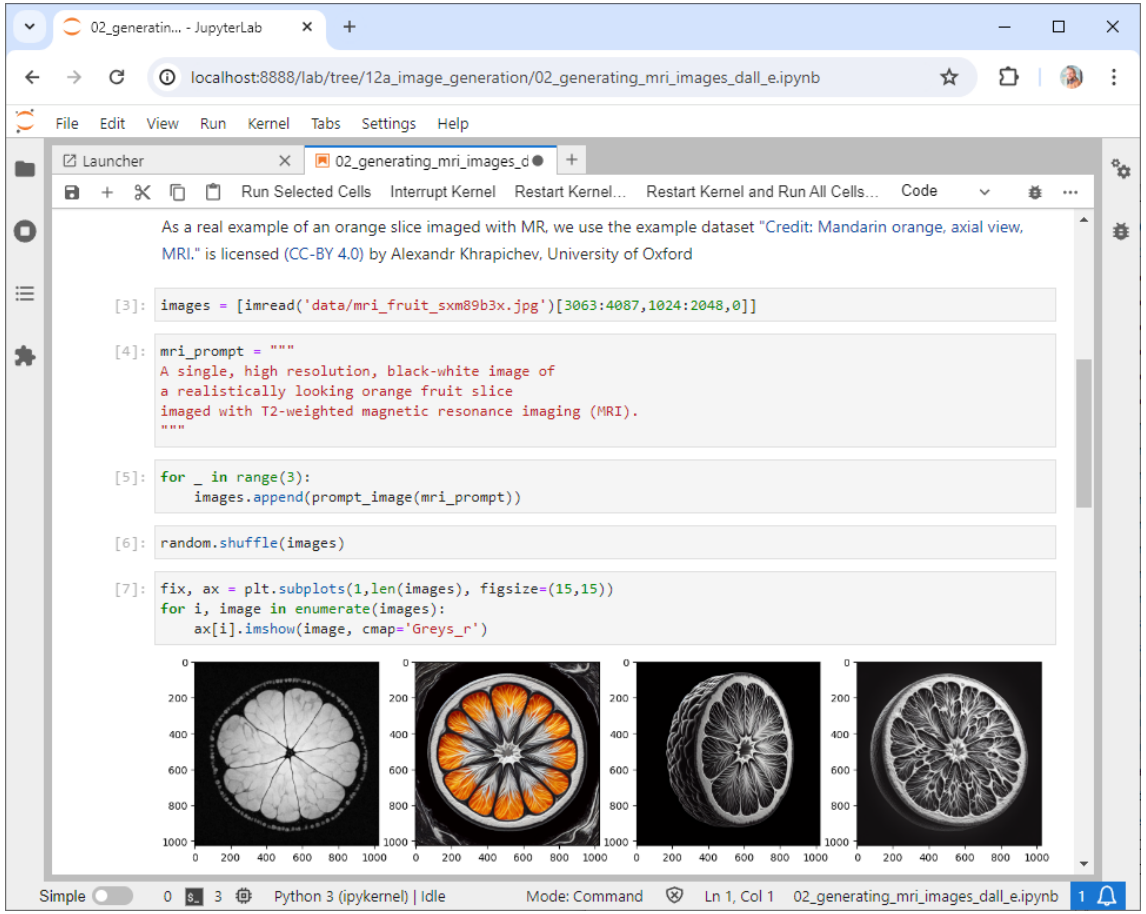
Bundesministerium für Bildung und Forschung

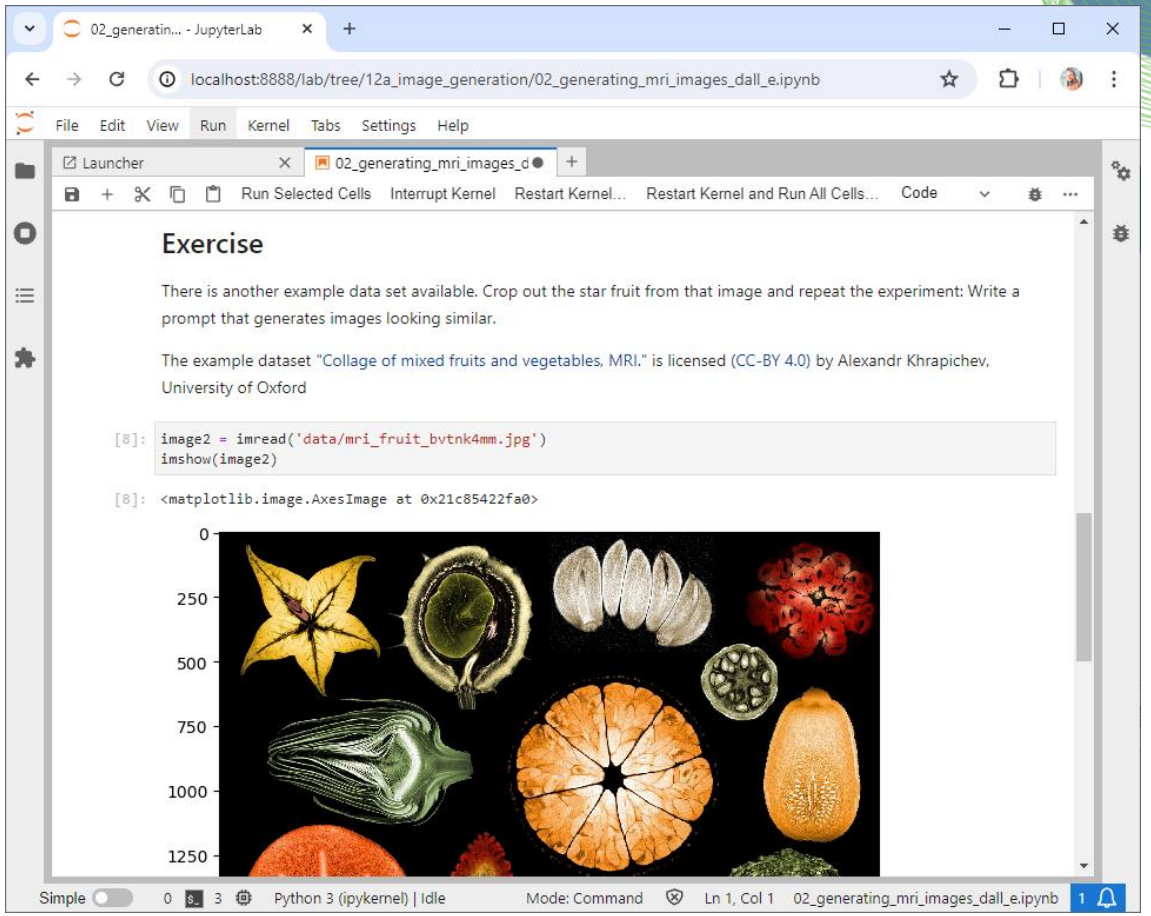
SACHSEN

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.

# Exercise: Image generation

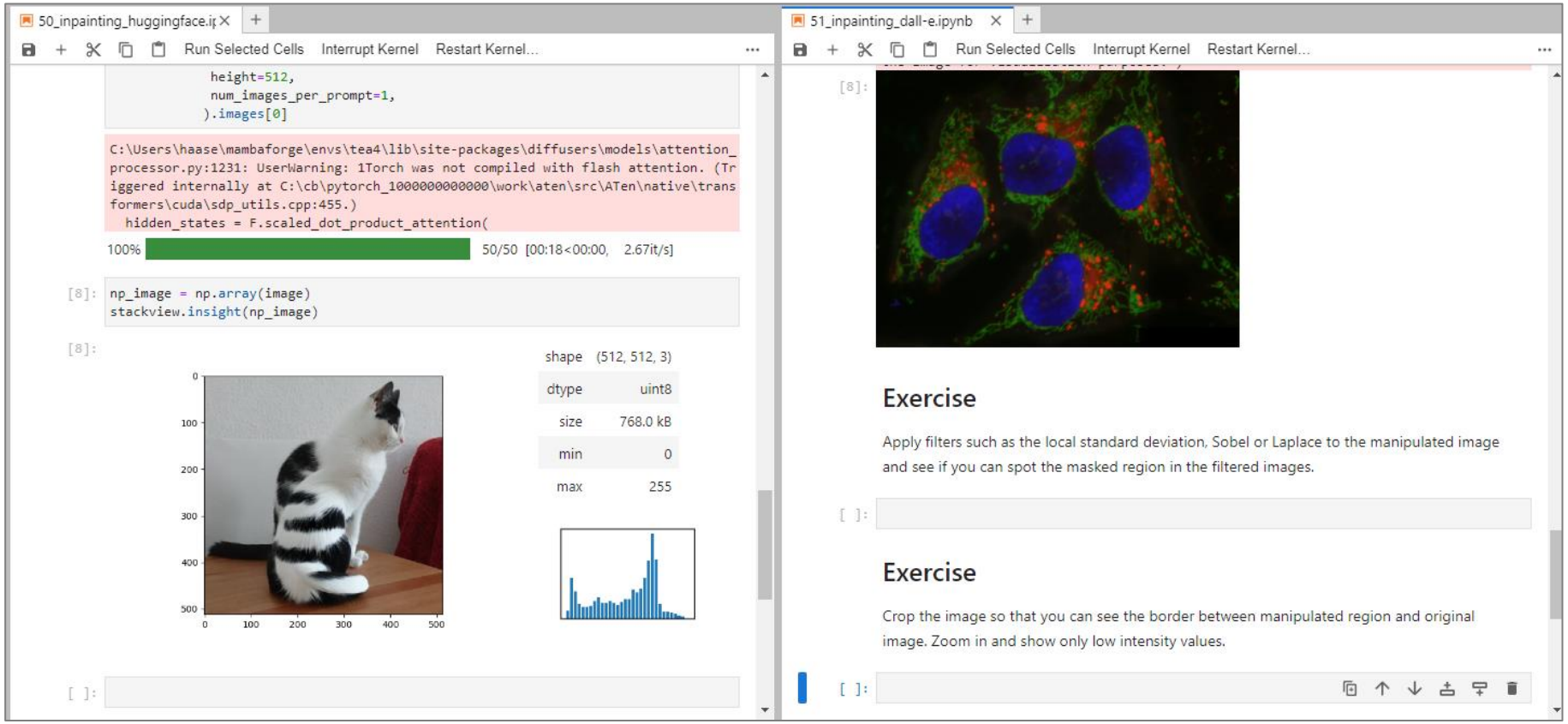
Try to identify and create realistically looking MRI images



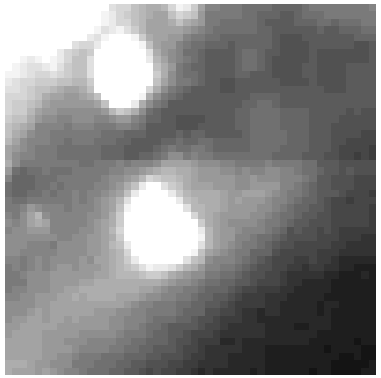



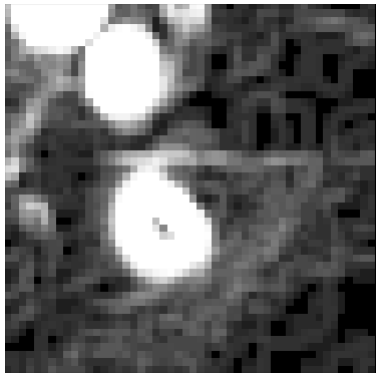
# Exercise: Image manipulation

Inspect the image carefully, try to find the border of the manipulated region




Hint:





# Exercise: Vision

As llava and gpt-4omni to describe an image *and* to produce Python code for analysing it.