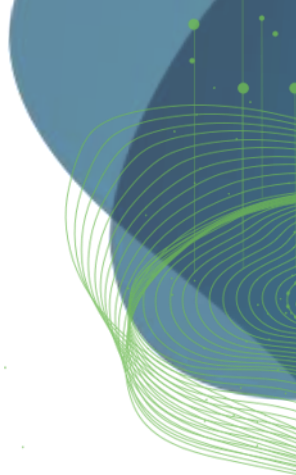


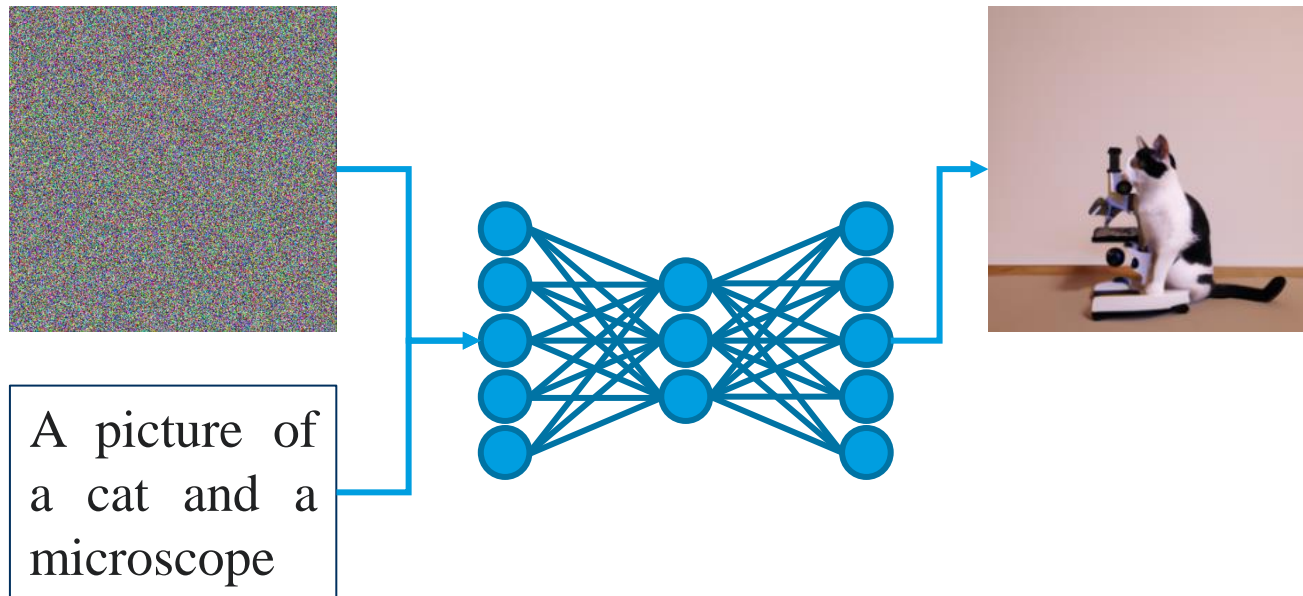
# Image Generation and Vision Language Models

Robert Haase



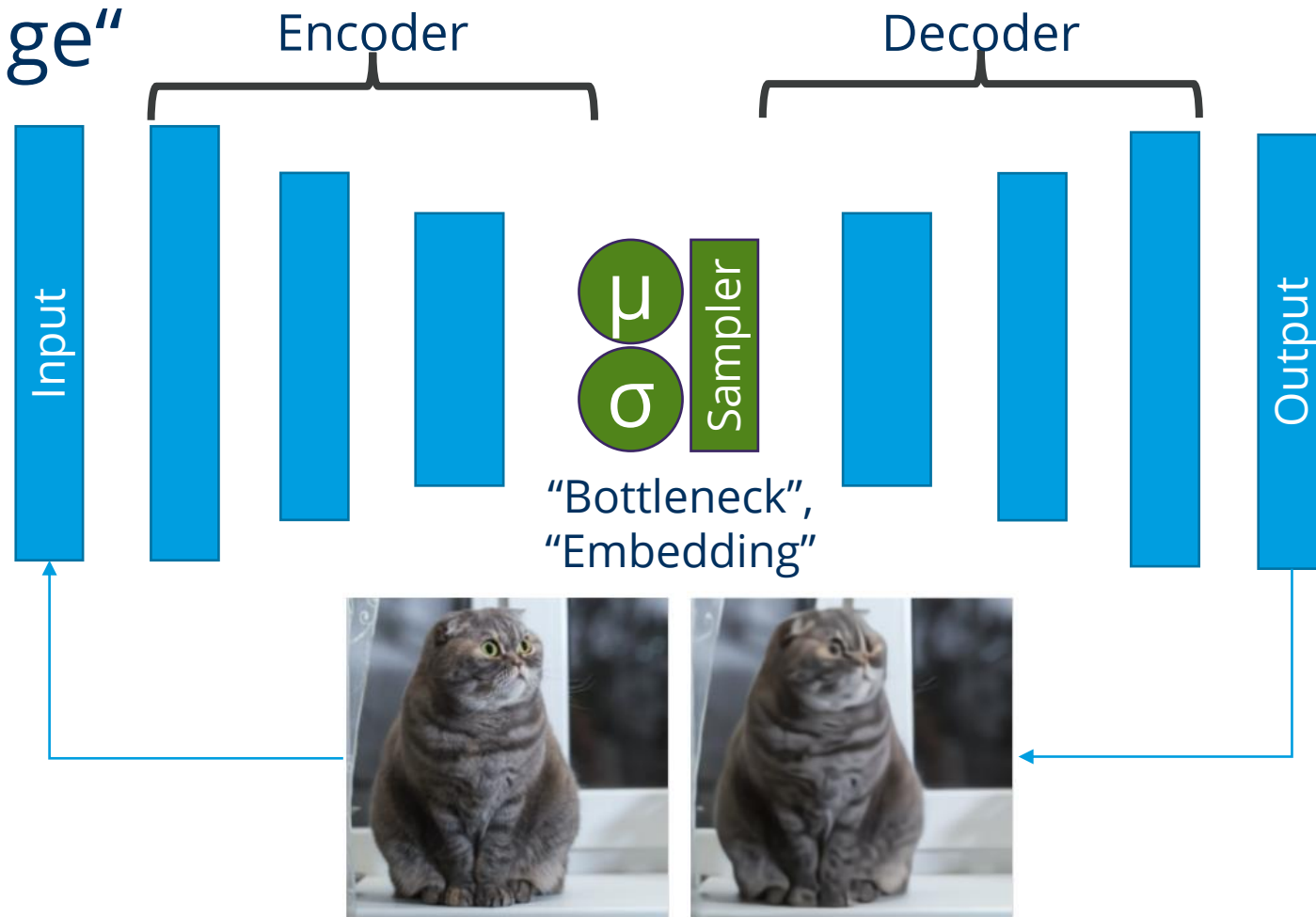
# Image Generation

„text-to-image“

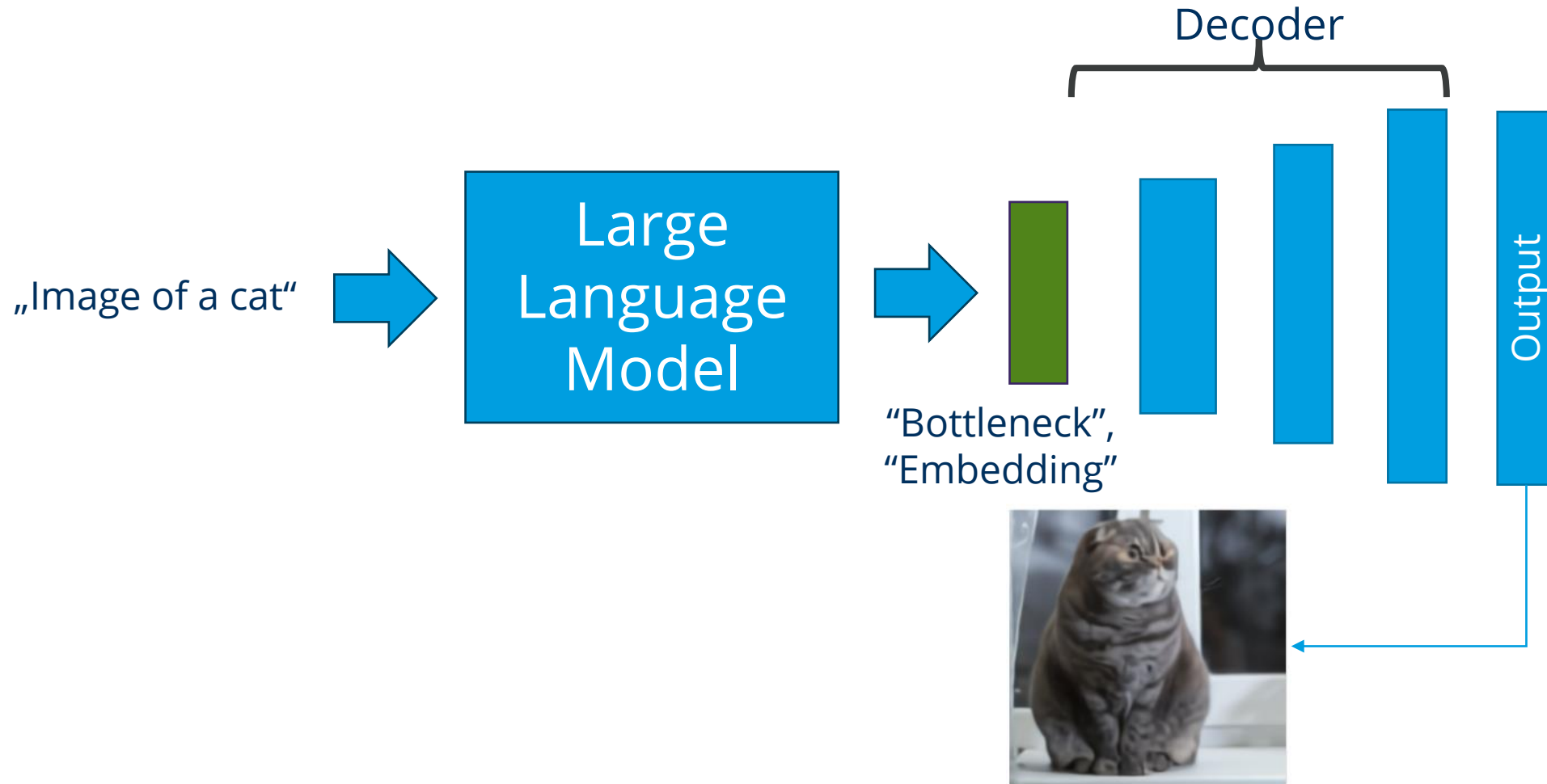


# Variational Auto-Encoder

„image-to-image“

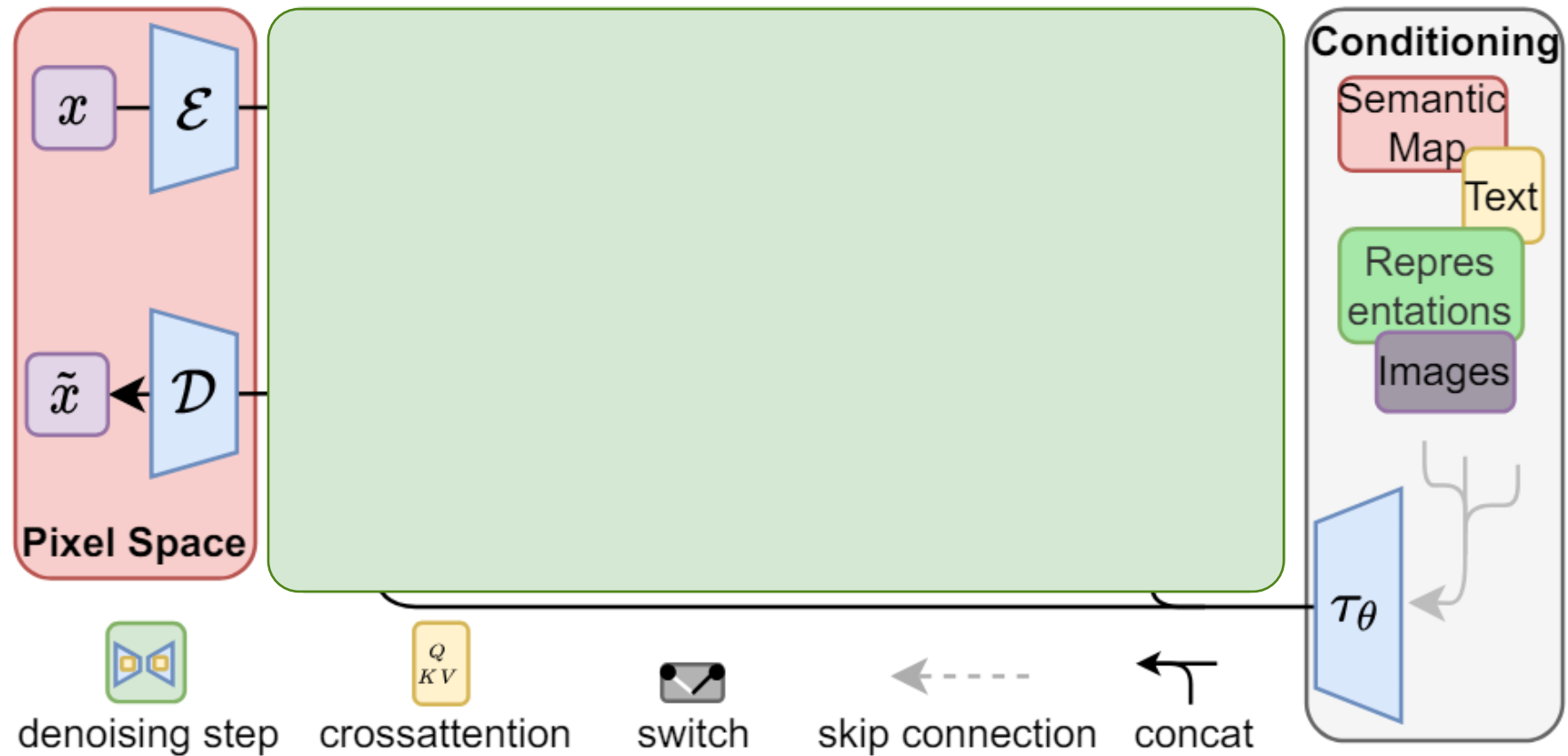


# Image Generation



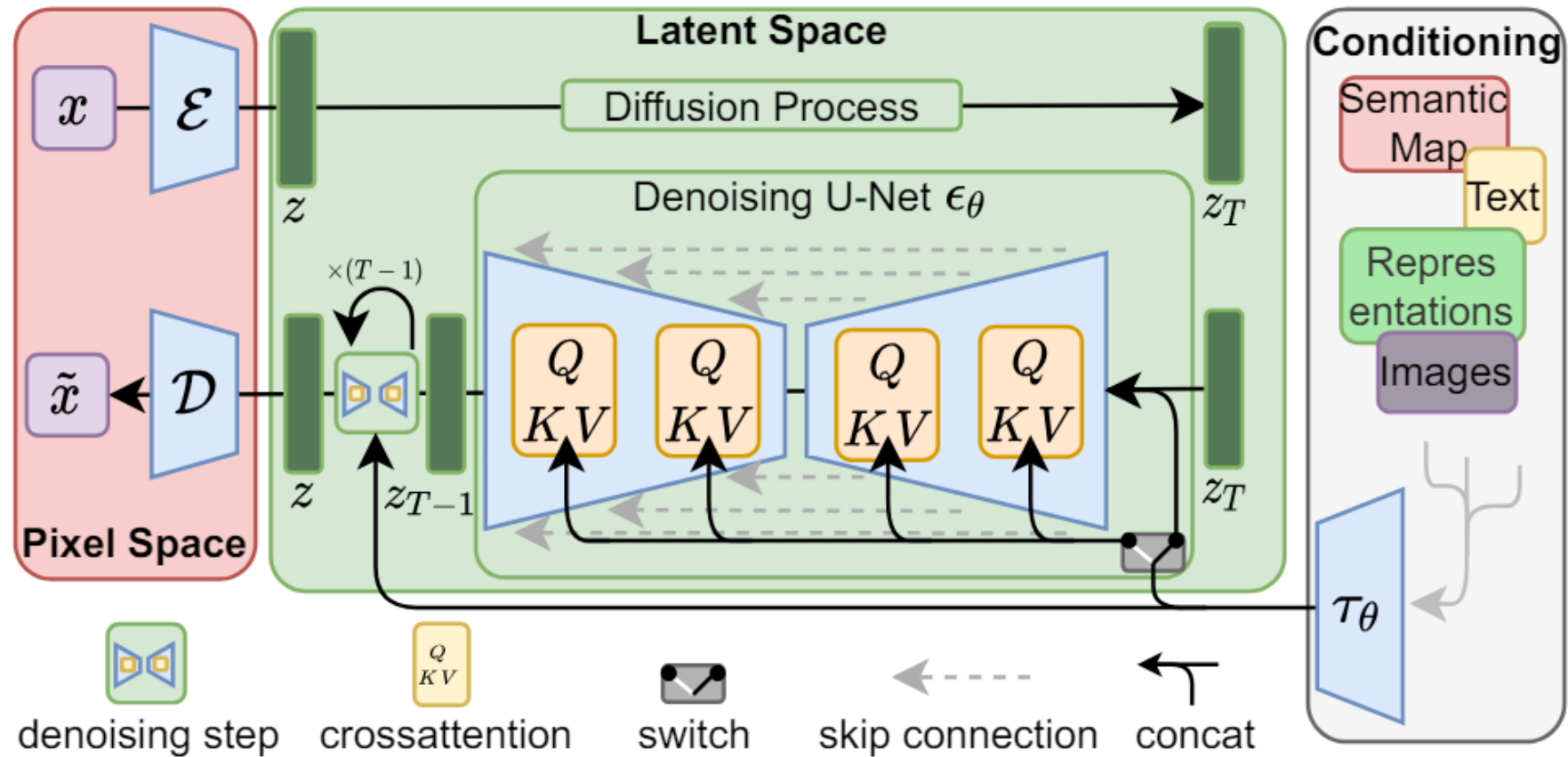
# Stable Diffusion

Diffusion: reverse denoising



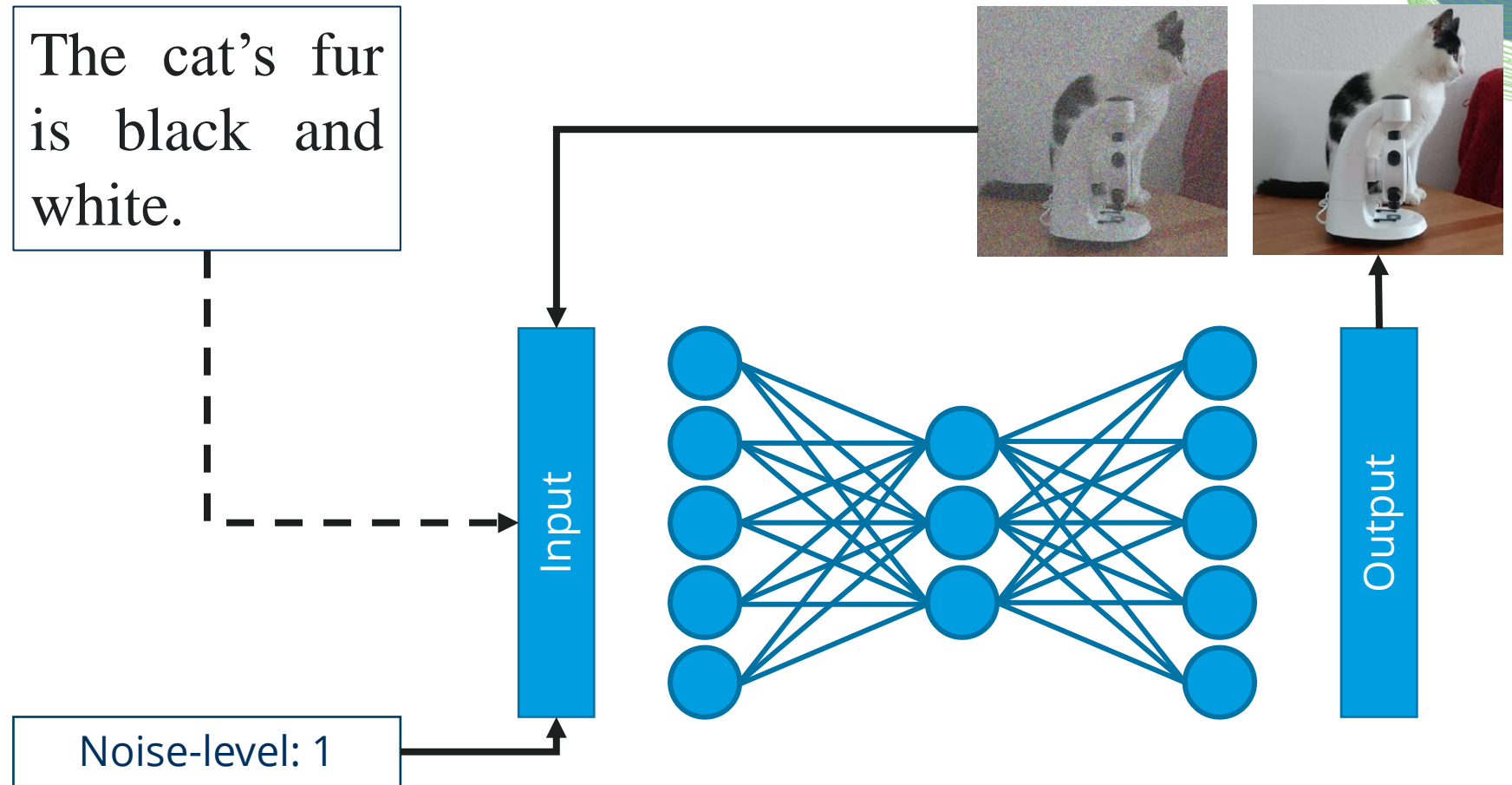
# Stable Diffusion

Diffusion: reverse denoising



# How does it work?

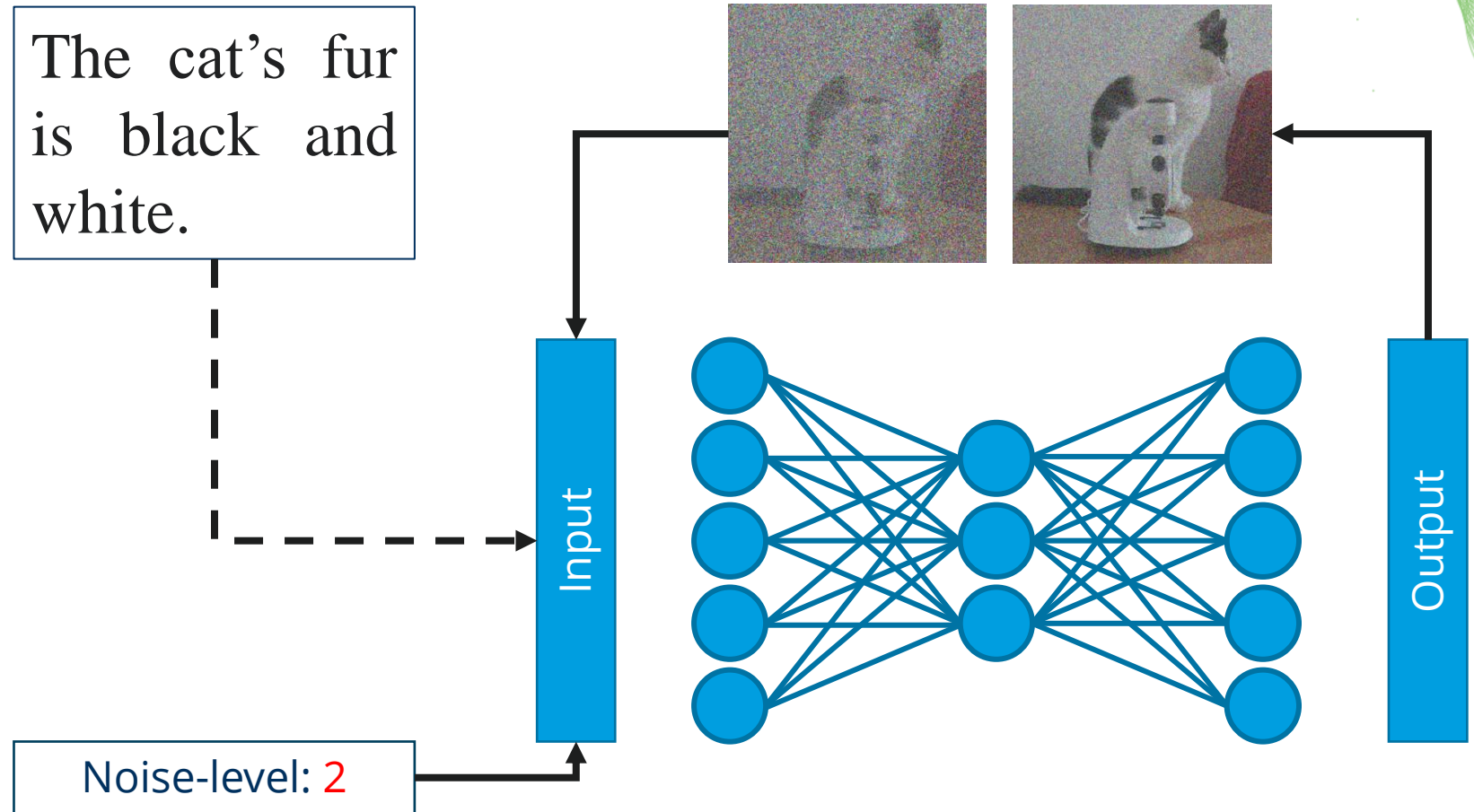
Train a U-Net on  
data: image +  
noisy image +  
description +  
noise-level





# How does it work?

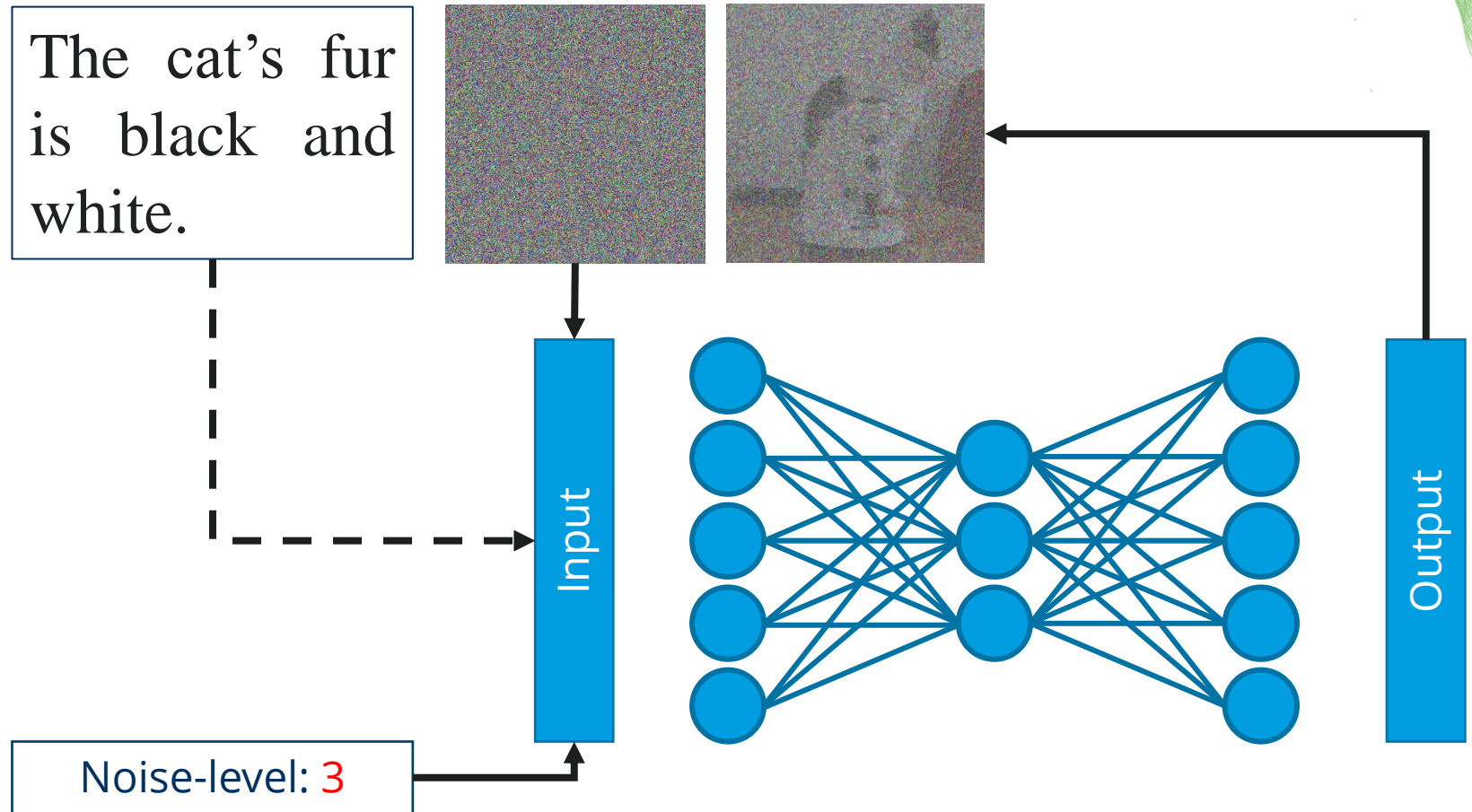
Train a U-Net on  
data: image +  
noisy image +  
description +  
noise-level





# How does it work?

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

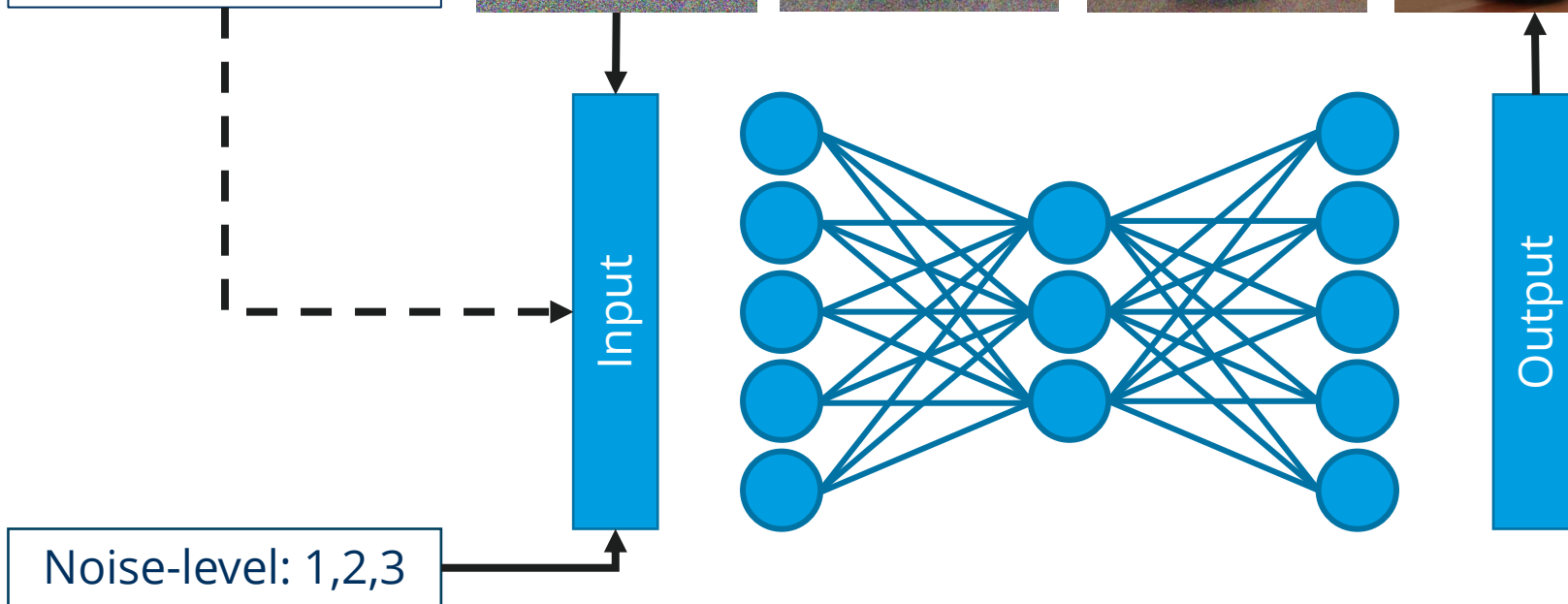
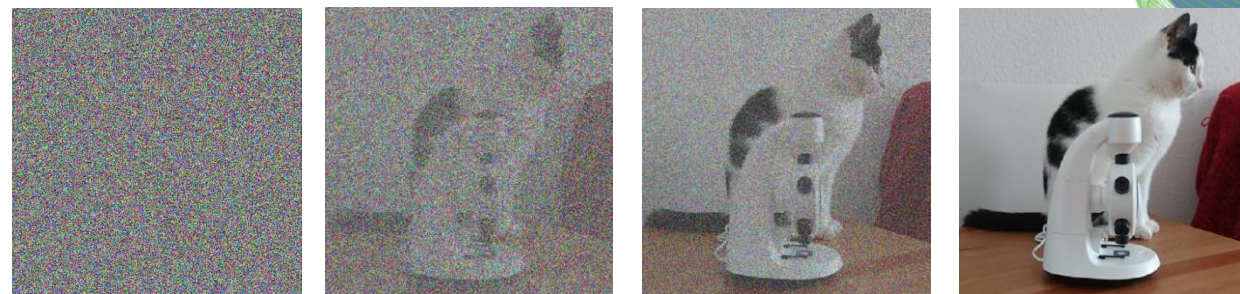


# How does it work?

Prediction is  
iterative denoising  
of:

Pure noise +  
text prompt

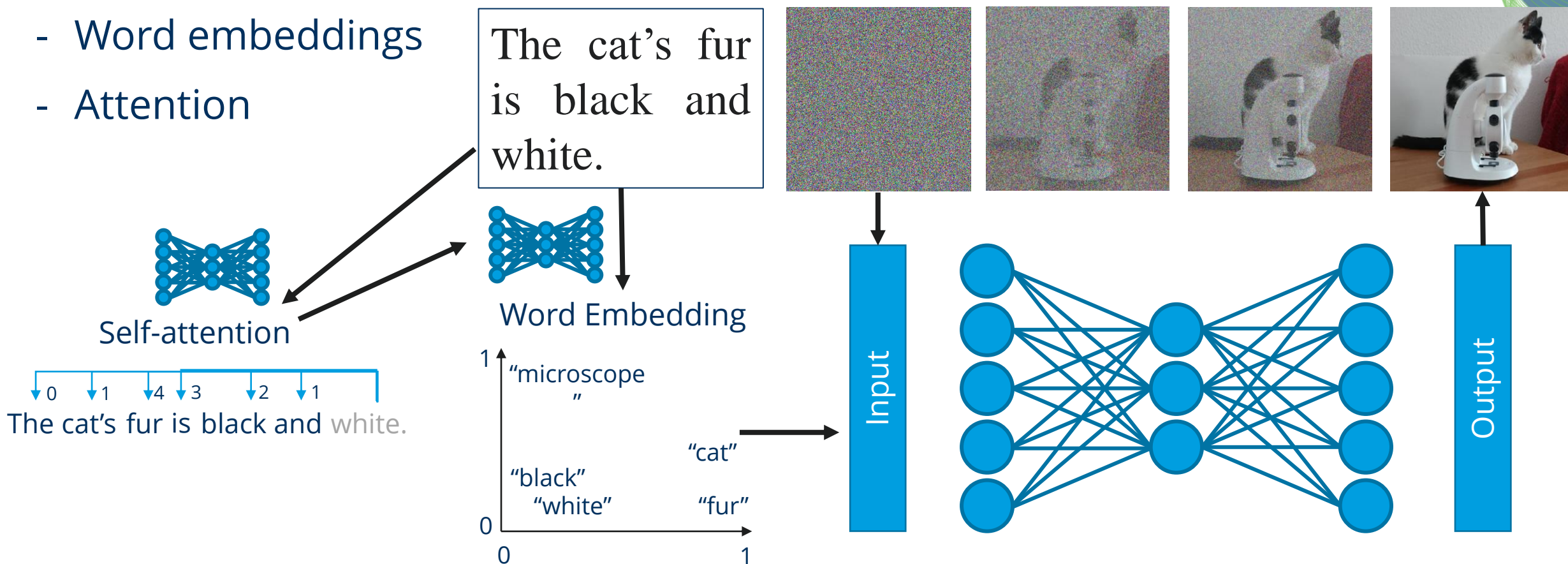
The cat's fur  
is black and  
white.



# 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
```

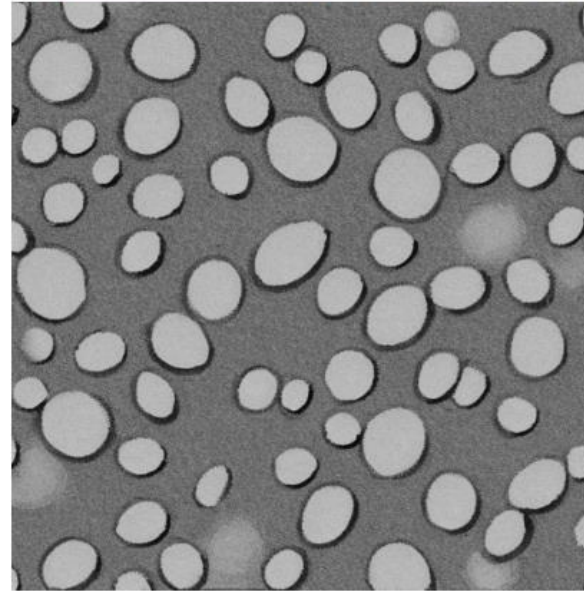
100% 50/50 [01:30<00:00, 1.40s/it]



```
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]



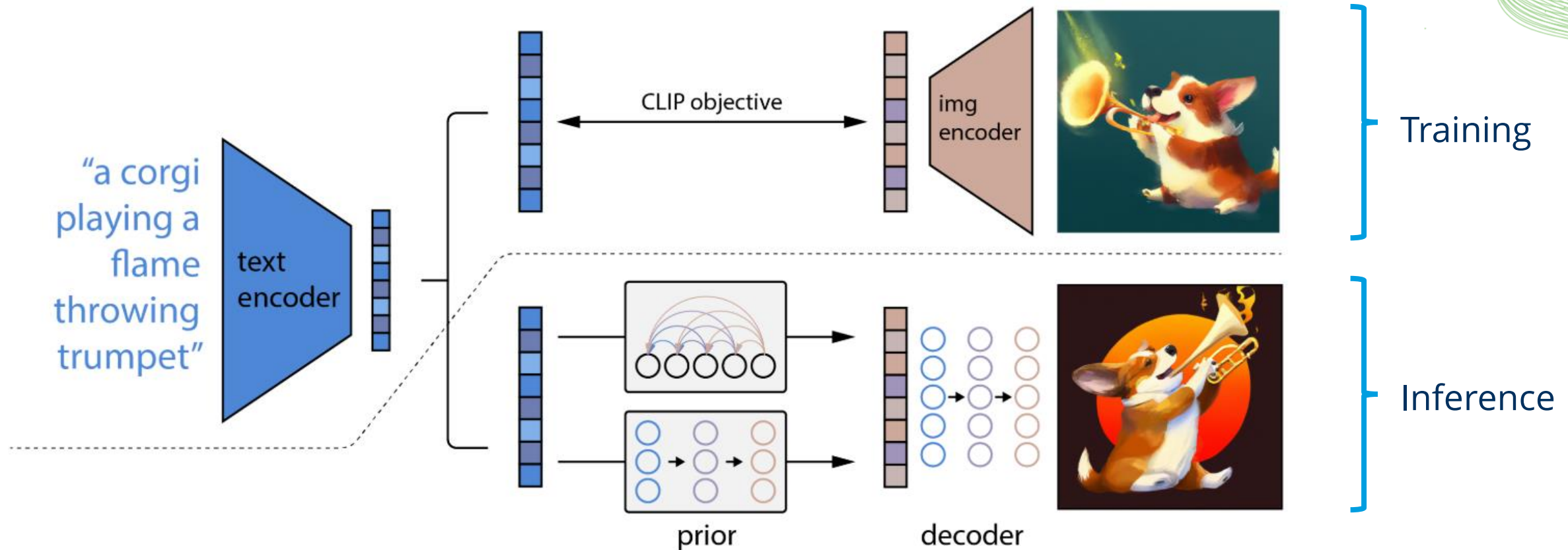
```
image = pipe(prompt,  
    num_inference_steps=10,  
    width=512,  
    height=512).images[0]  
image
```

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



# Dall-E

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





# 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

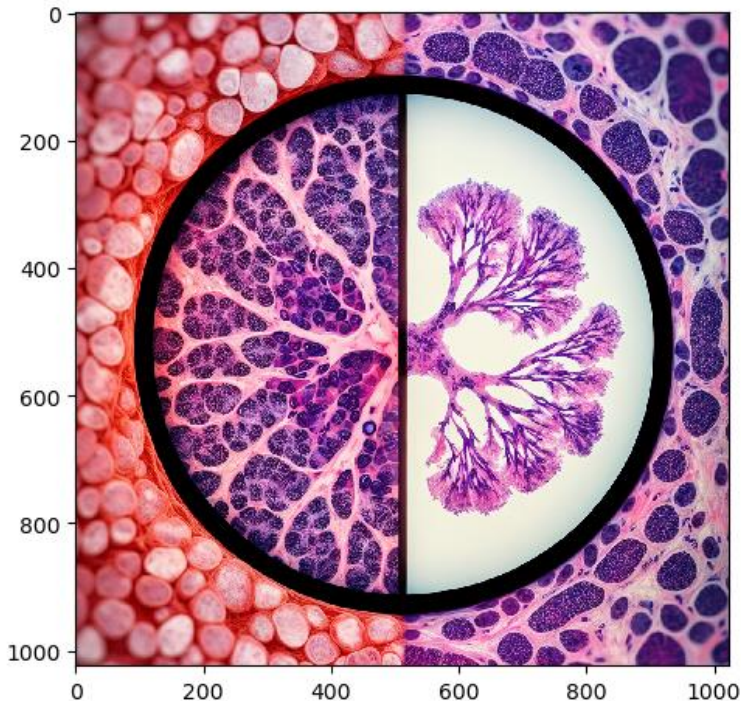
May soon also  
work with gpt-4o

# Image generation 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)
```

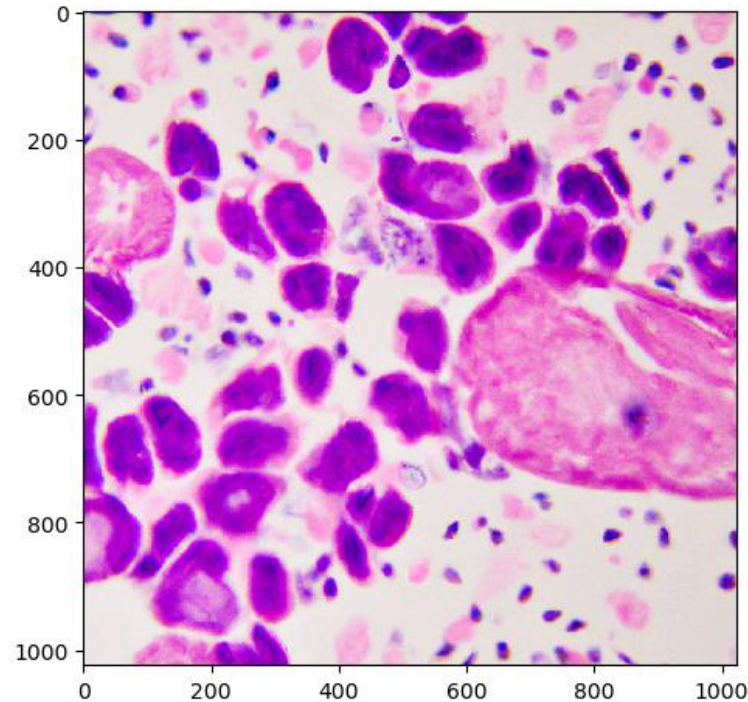
<matplotlib.image.AxesImage at 0x1d9eda5bd90>



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“)

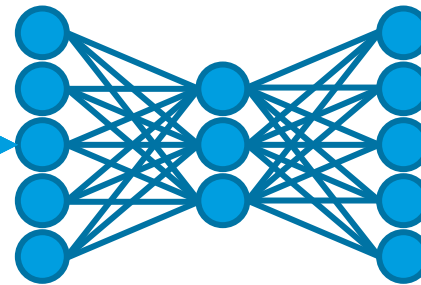
Raw image



Mask image



A black white  
cat fur



Manipulated  
image

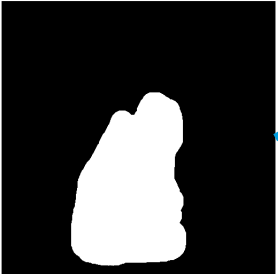


# Inpainting in Python: Huggingface

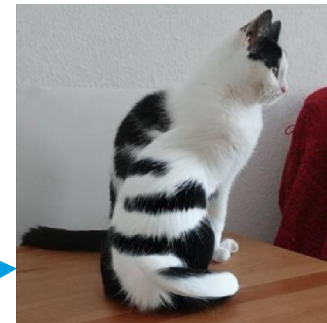
```
pipe = StableDiffusionInpaintPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-inpainting",
    torch_dtype=torch.float16
)
pipe = pipe.to("cuda")
```

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]
```

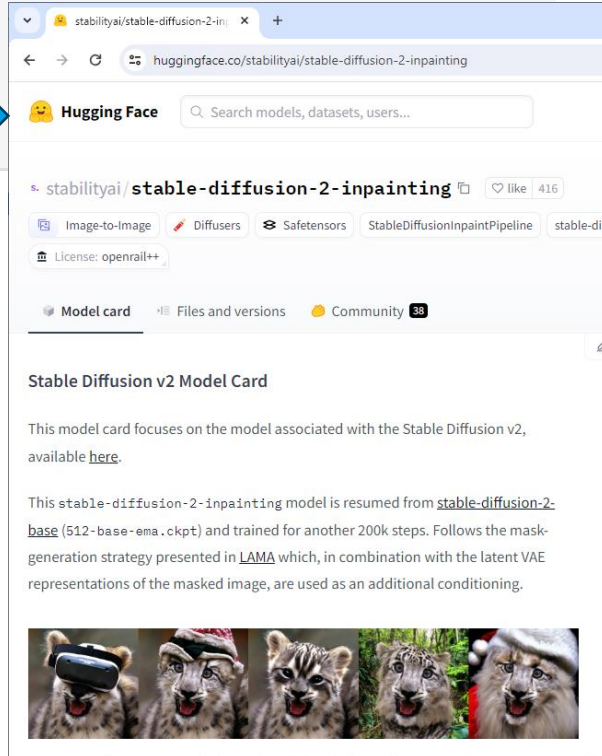




# Inpainting in Python: Huggingface

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")
```









**stabilityai/stable-diffusion-2-inpainting**  416

Image-to-Image  Diffusers  Safetensors  StableDiffusionInpaintPipeline

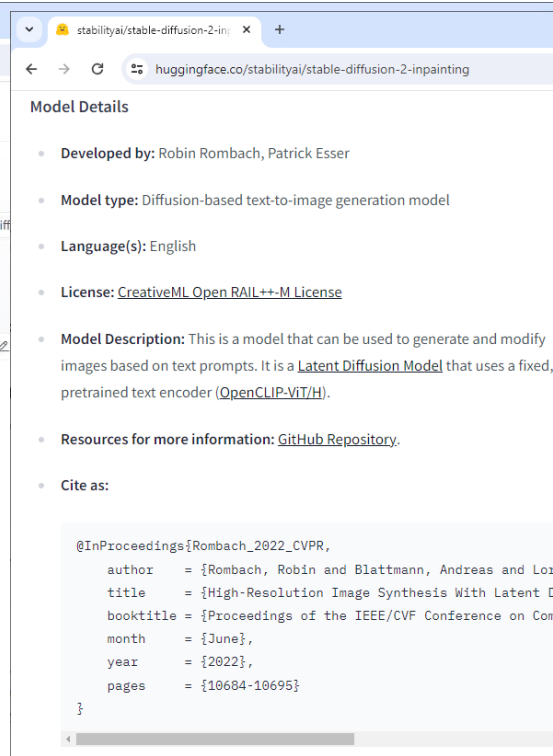

License: openrail++

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](#).

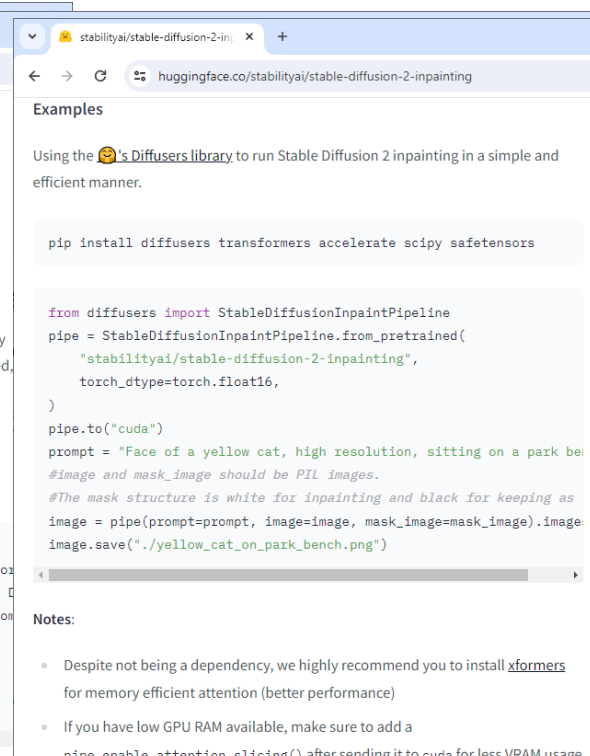
This stable-diffusion-2-inpainting model is resumed from [stable-diffusion-2-base](#) (512-base-ema.ckpt) and trained for another 200k steps. Follows the mask-generation strategy presented in [LAMA](#) which, in combination with the latent VAE representations of the masked image, are used as an additional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

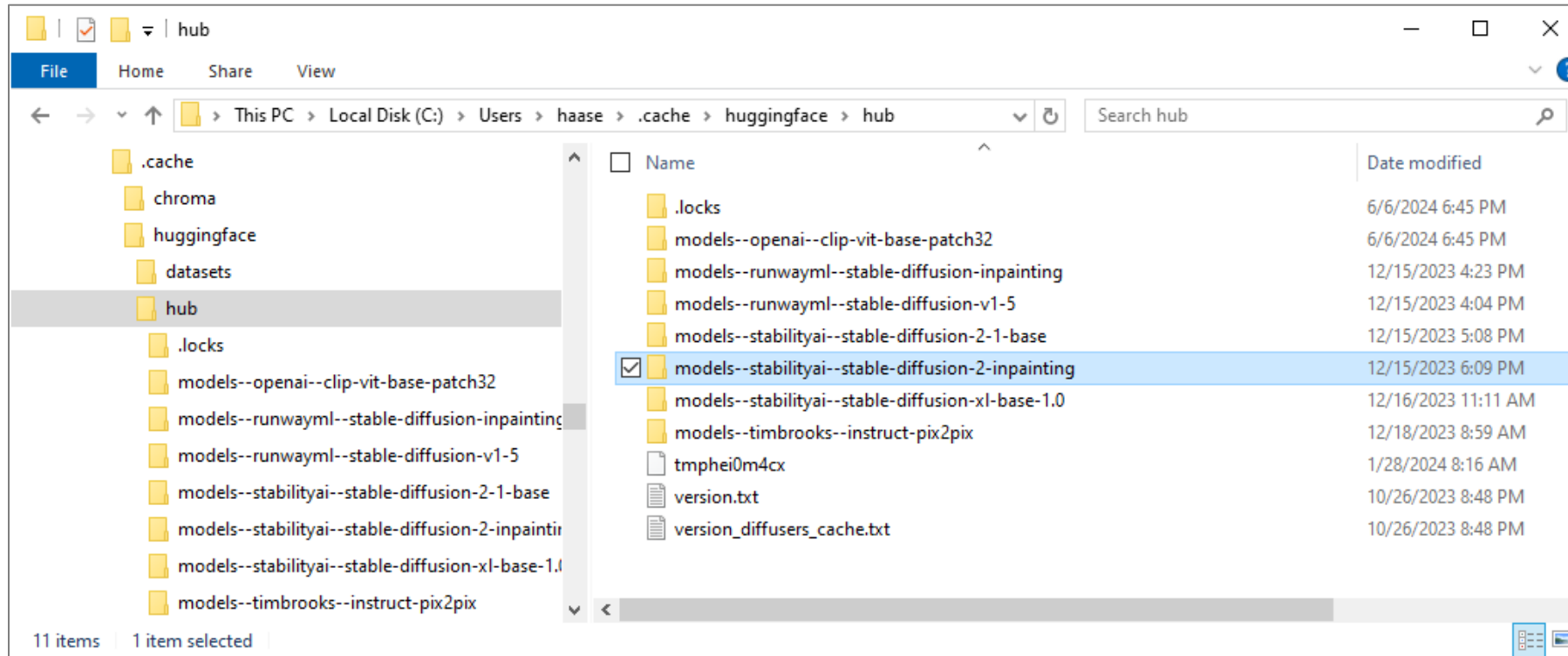
### Notes:

- 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



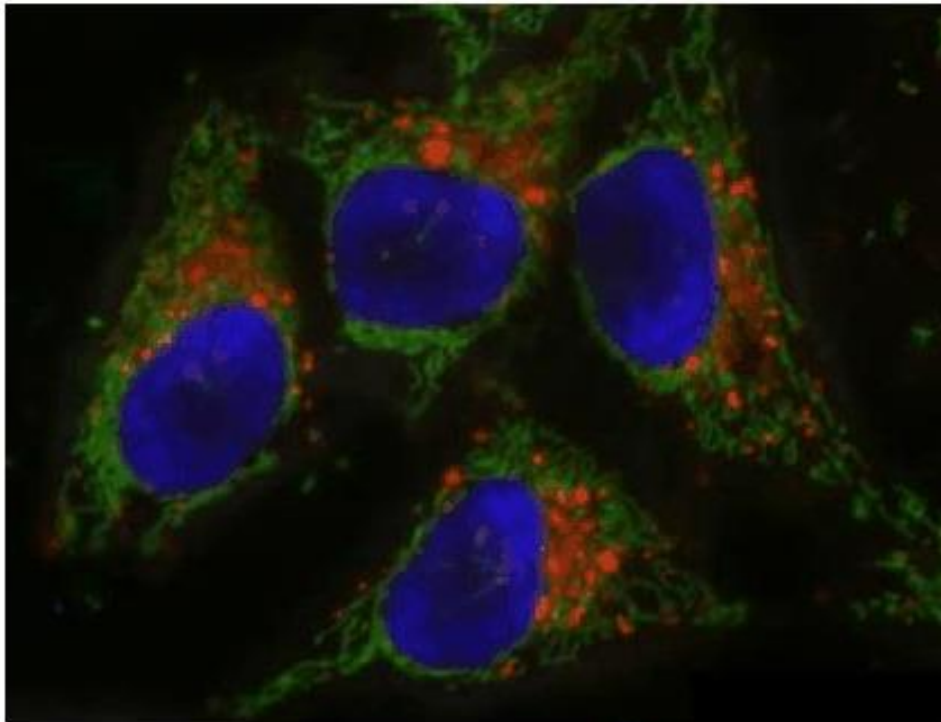
Result: List of URL(s)

# New technologies bring new risks...

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

Interesting  
challenges for our  
community ahead

[6]:

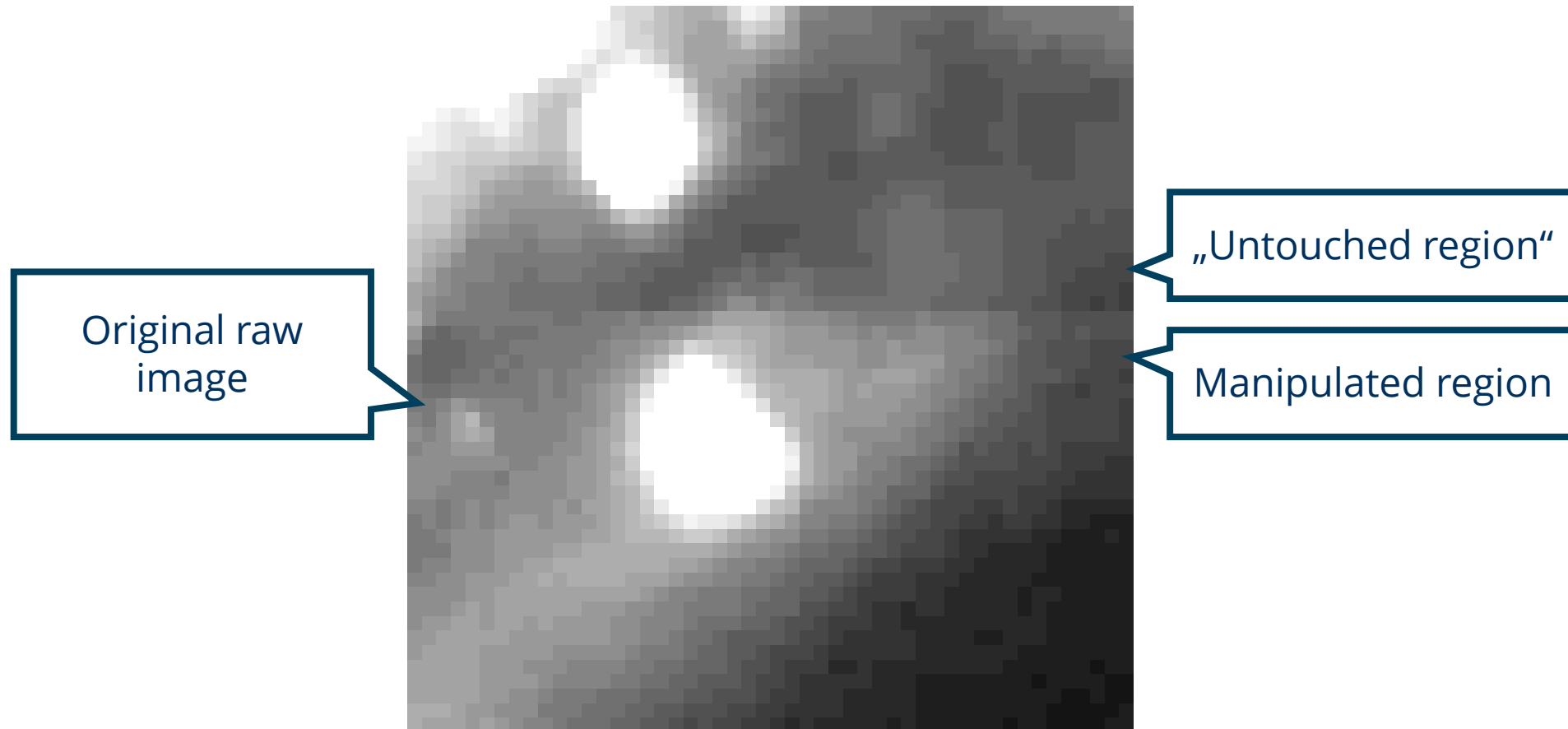


Curtain

672

# Image manipulation detection

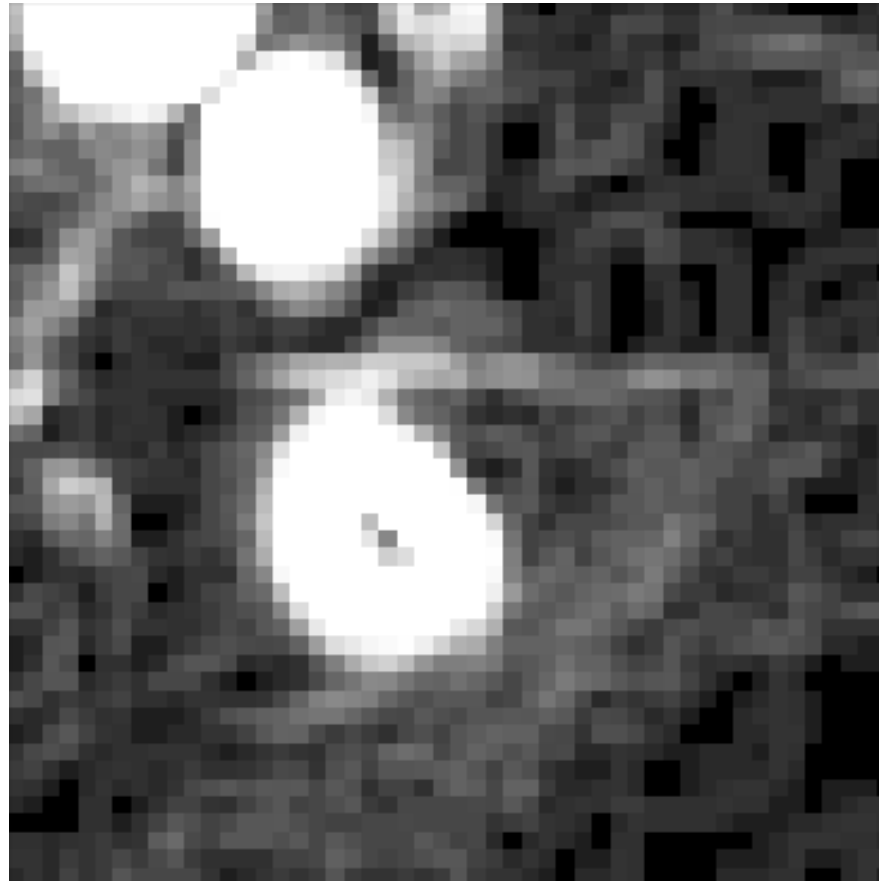
The noise pattern differs between raw and processed images...



# Image manipulation detection

e.g. by studying noise-patterns

Local standard  
deviation filter



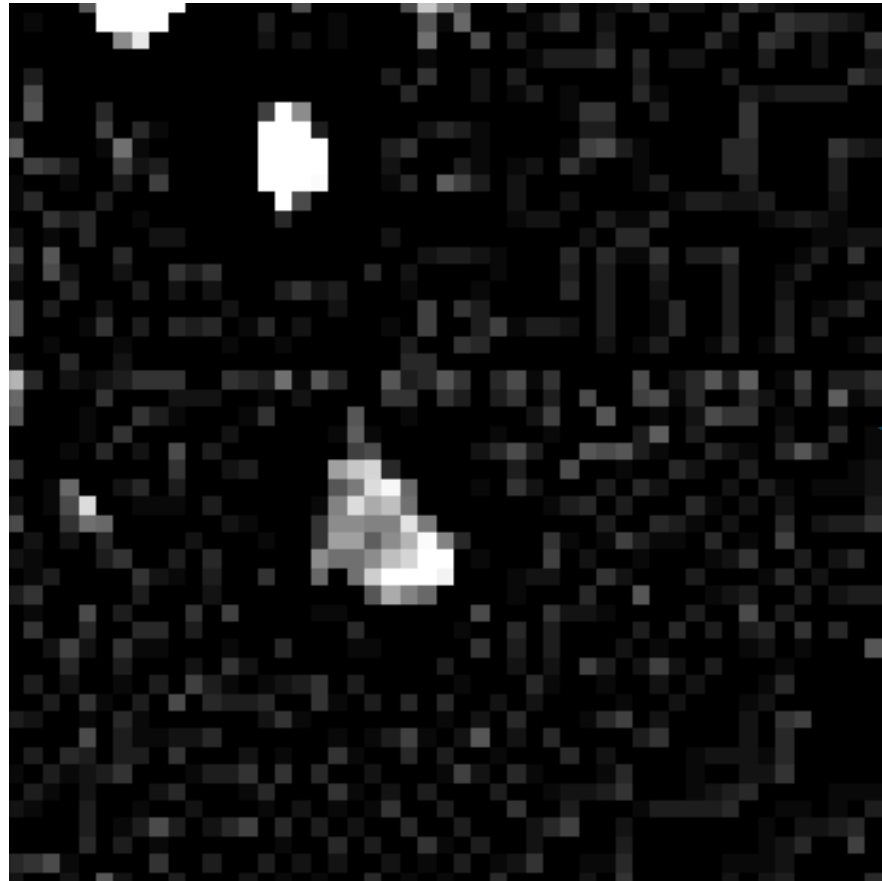
„Untouched region“

Manipulated region

# Image manipulation detection

e.g. by studying noise-patterns

Sobel filter

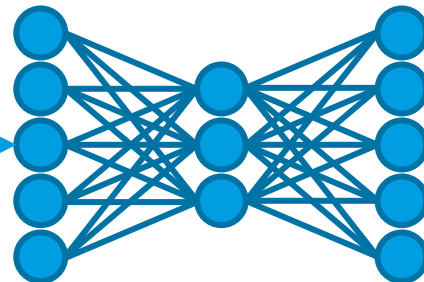


„Untouched region“

Manipulated region

# Vision Language Models

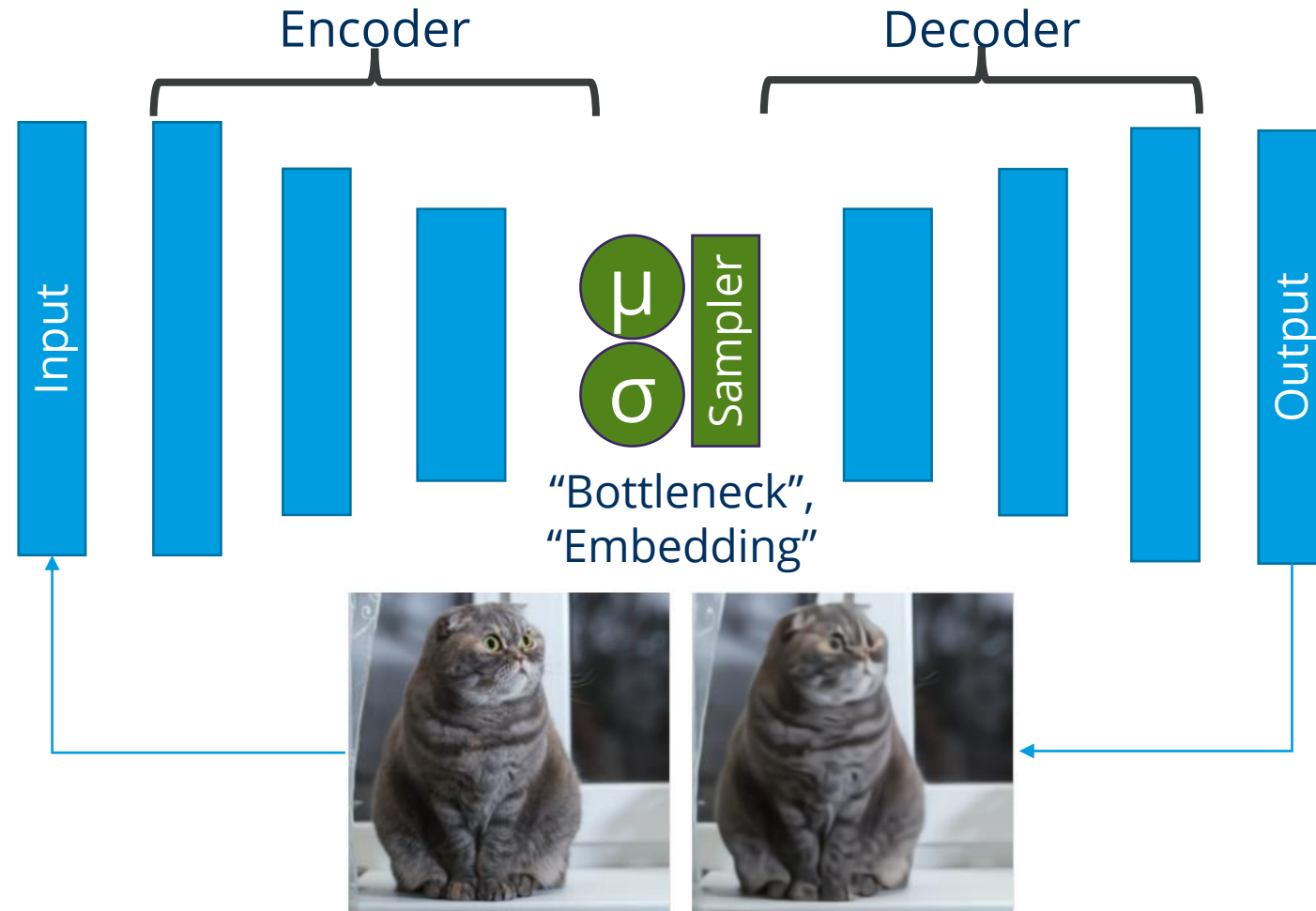
- Classifying images 🤔
- Describing images



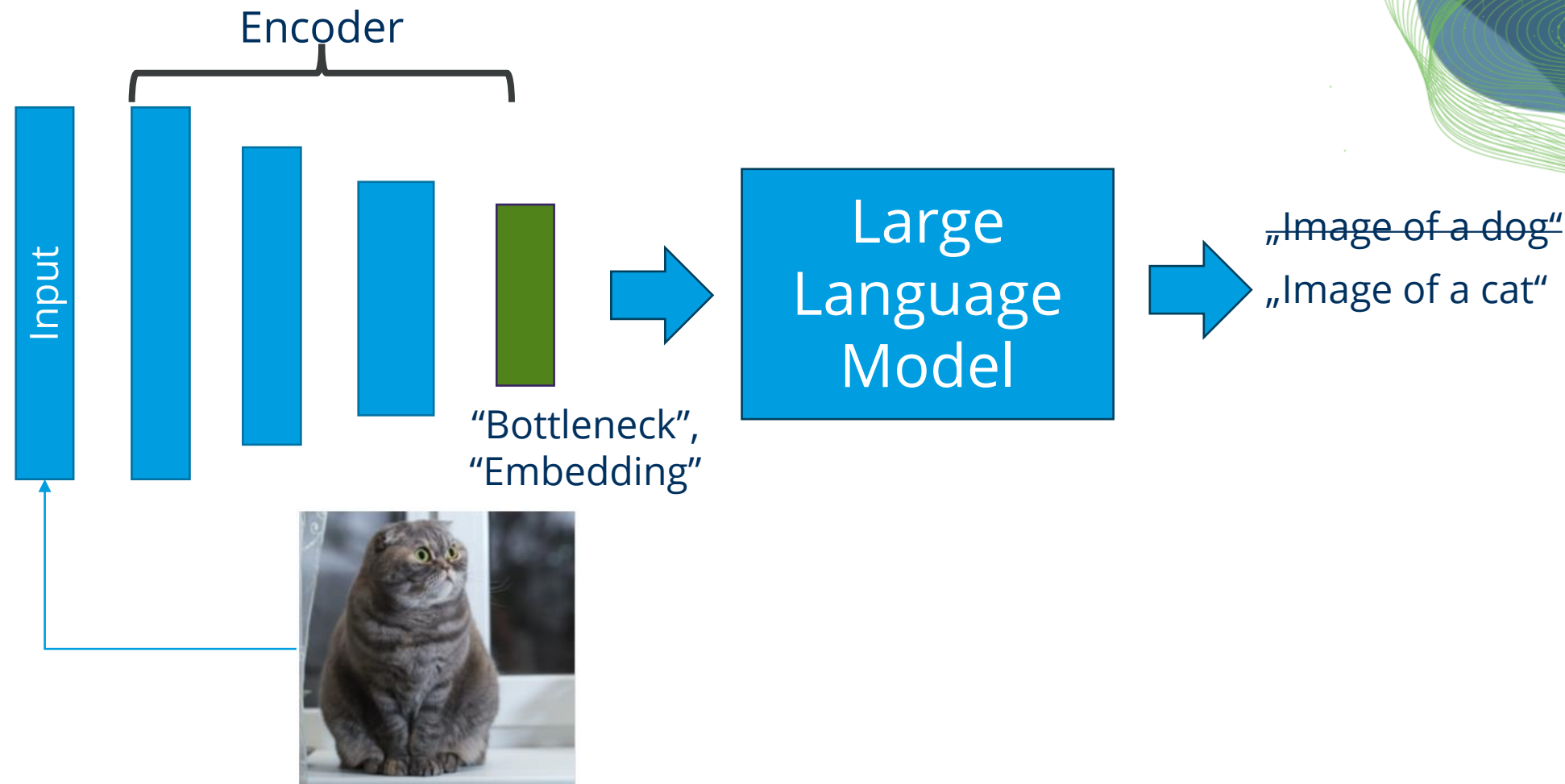
A picture of  
a cat and a  
microscope



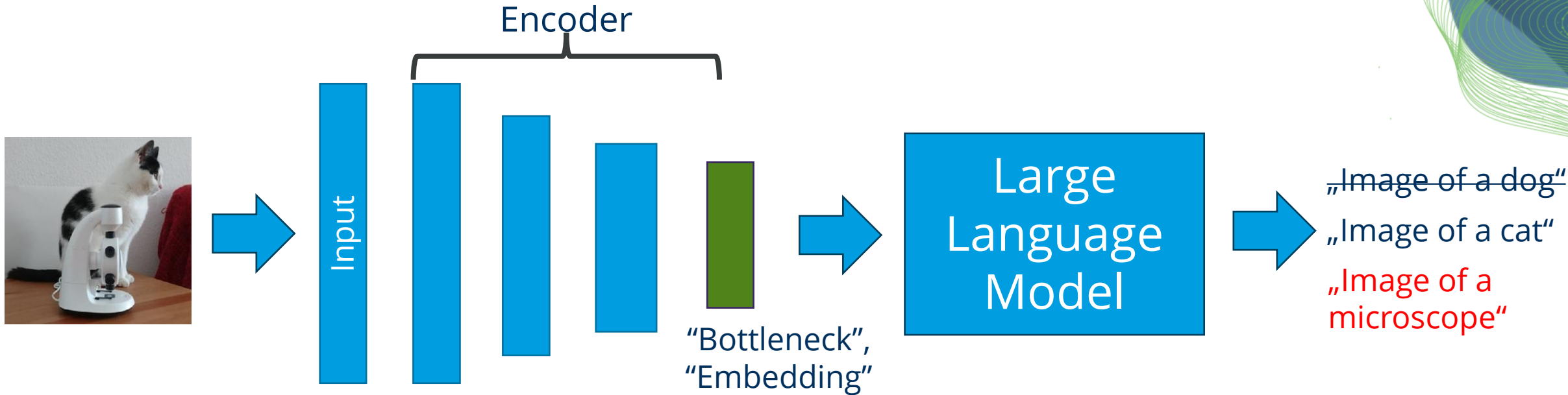
# Variational Auto-Encoder



# Image classification



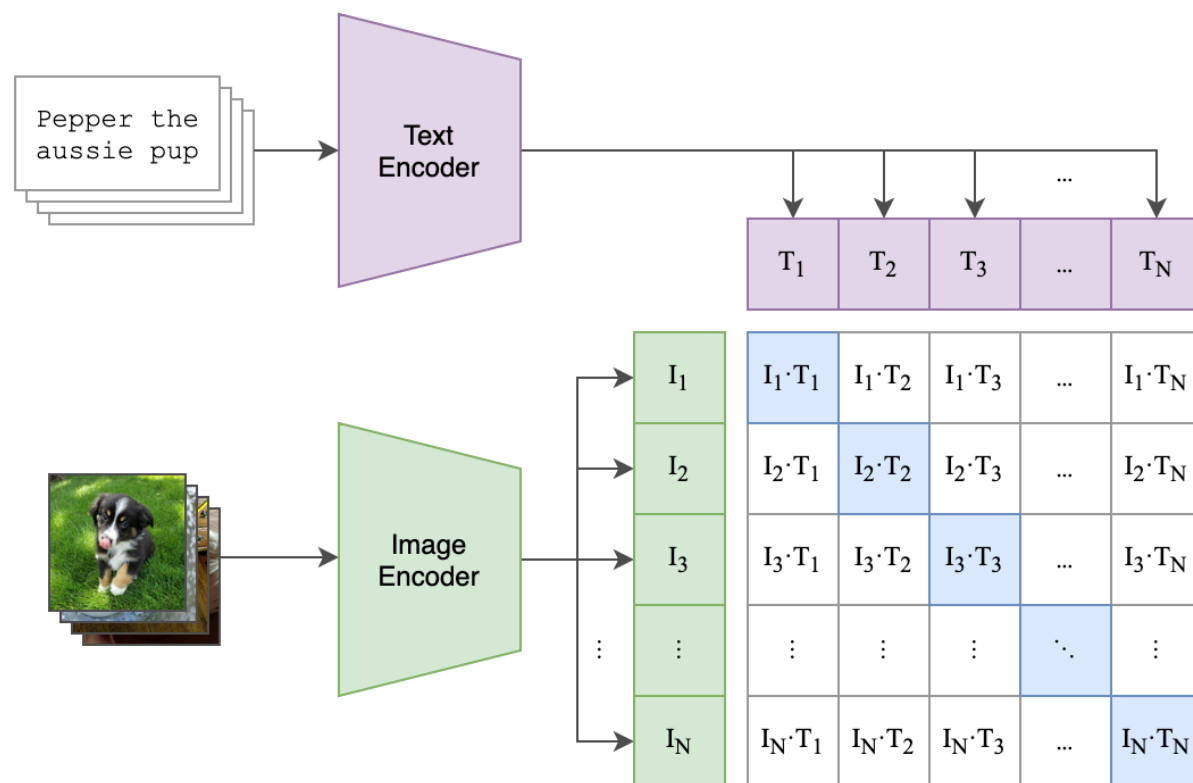
# Image classification -> image describing



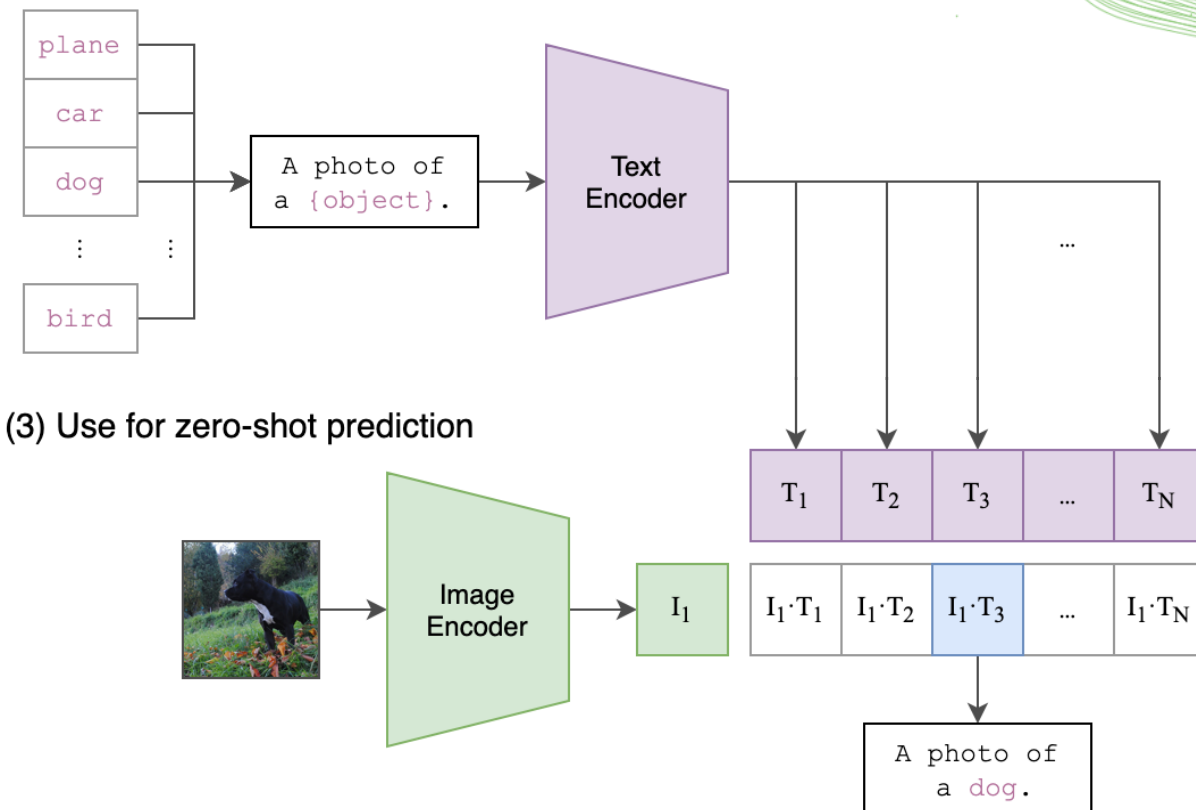
# Contrastive Language-Image Pre-Training

## „CLIP“ Transformers

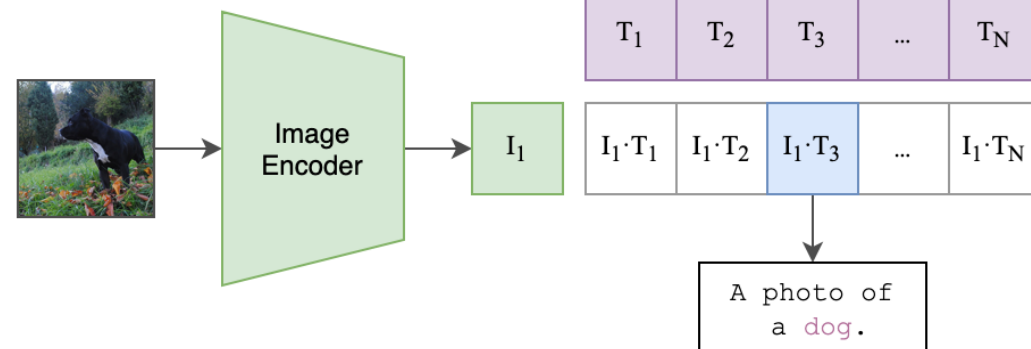
### (1) Contrastive pre-training



### (2) Create dataset classifier from label text

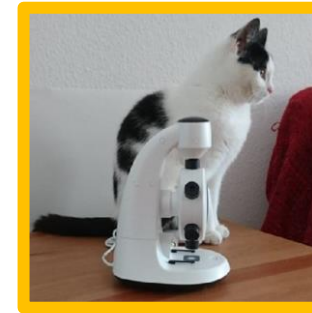


### (3) Use for zero-shot prediction



# 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

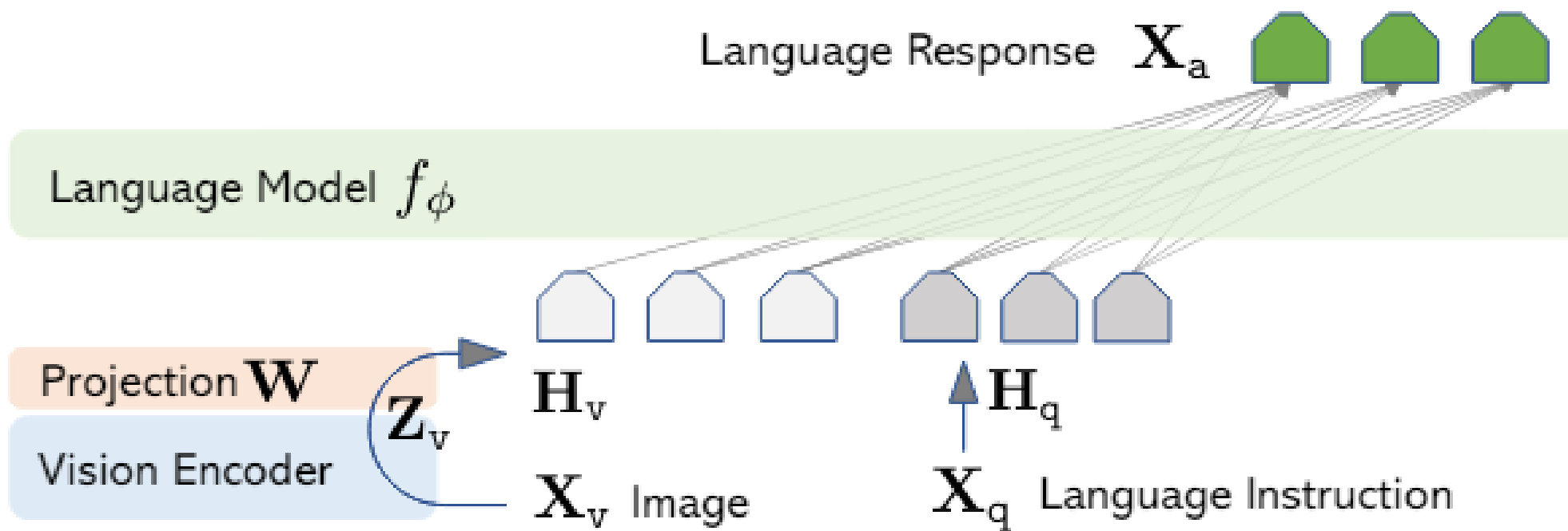
```
{'a photo of a cat': 0.9907298684120178,  
 'a photo of a dog': 0.009270114824175835}
```

label\_probabilities

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

# LLAVA

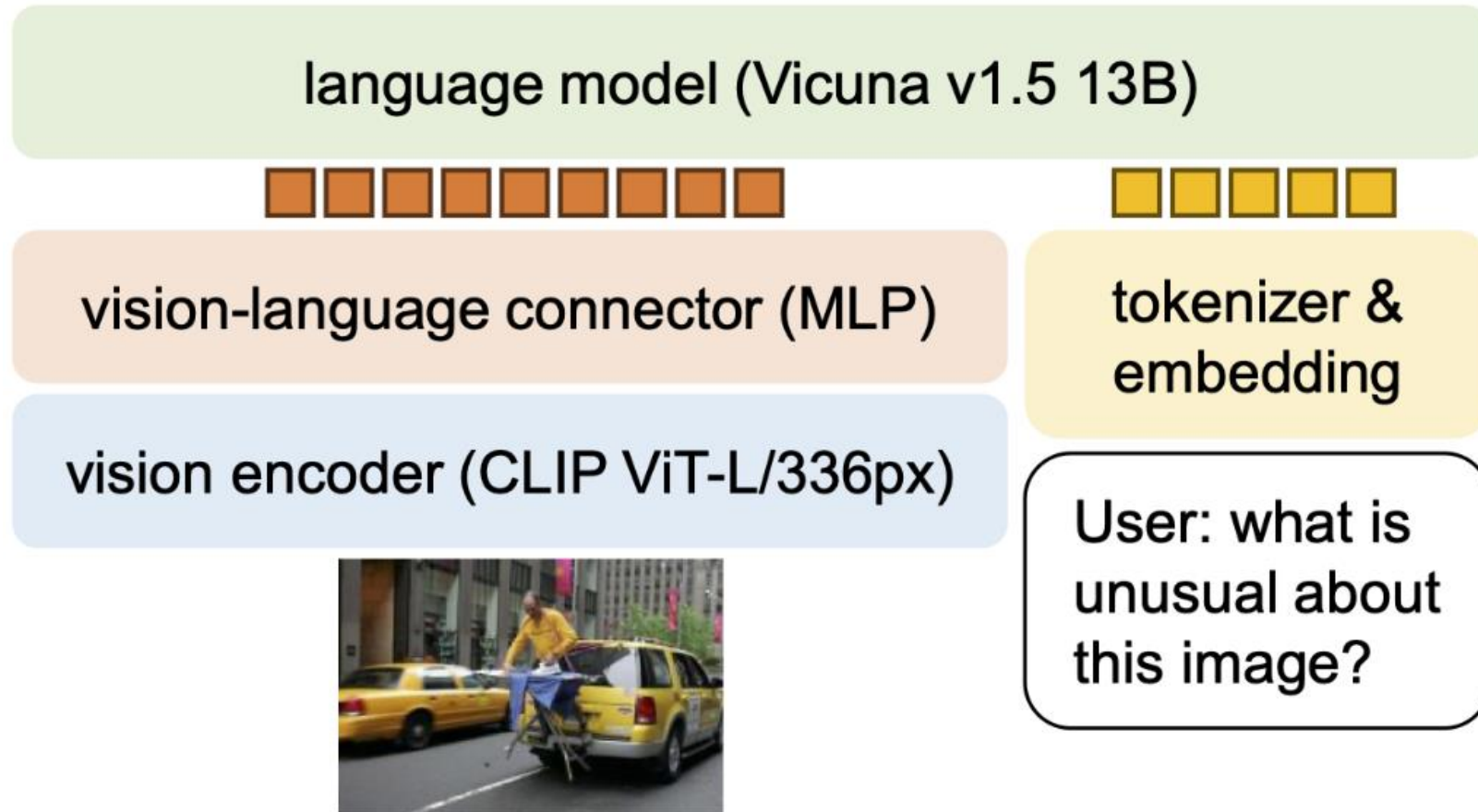
## Large Language and Vision Assistant





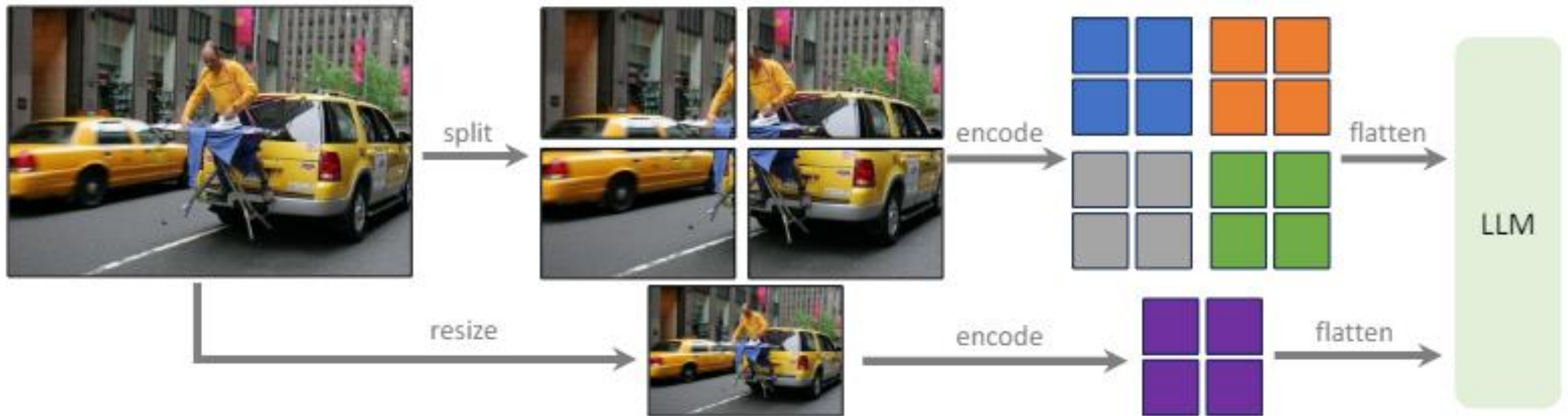
# LLAVA 1.5

## Combining LLAVA with CLIP



# LLAVA 1.5 HD

Giving the model multiple perspectives on the same scene



# Accessing VLMs using Python

API not standardized (yet)

```
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"data:image/jpeg;base64,{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
```

```
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
```

# Exercises

Robert Haase

Funded by



Bundesministerium  
für Bildung  
und Forschung

SACHSEN



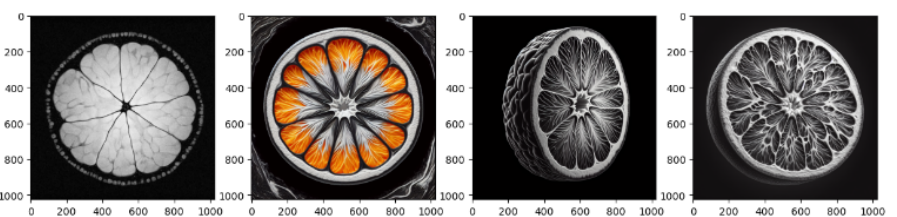
Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.




# Exercise: Image generation

Try to identify and create realistically looking MRI images

```
02_generatin... - JupyterLab
localhost:8888/lab/tree/12a_image_generation/02_generating_mri_images_dall_e.ipynb
File Edit View Run Kernel Tabs Settings Help
Launcher
Run Selected Cells Interrupt Kernel Restart Kernel... Restart Kernel and Run All Cells... Code
[3]: images = [imread('data/mri_fruit_sxm89b3x.jpg')[3063:4087,1024:2048,0]]
[4]: mri_prompt = """
A single, high resolution, black-white image of
a realistically looking orange fruit slice
imaged with T2-weighted magnetic resonance imaging (MRI).
"""
[5]: for _ in range(3):
images.append(prompt_image(mri_prompt))
[6]: random.shuffle(images)
[7]: fig, ax = plt.subplots(1,len(images), figsize=(15,15))
for i, image in enumerate(images):
ax[i].imshow(image, cmap='Greys_r')
```



```
02_generatin... - JupyterLab
localhost:8888/lab/tree/12a_image_generation/02_generating_mri_images_dall_e.ipynb
File Edit View Run Kernel Tabs Settings Help
Launcher
Exercise
There is another example data set available. Crop out the star fruit from that image and repeat the experiment: Write a prompt that generates images looking similar.
The example dataset "Collage of mixed fruits and vegetables, MRI." is licensed (CC-BY 4.0) by Alexandr Khrapichev, University of Oxford
[8]: image2 = imread('data/mri_fruit_bvtnk4mm.jpg')
imshow(image2)
[8]: <matplotlib.image.AxesImage at 0x21c85422fa0>
```



# Exercise: Image manipulation

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

Hint:

50\_inpainting\_huggingface.ipynb

Run Selected Cells Interrupt Kernel Restart Kernel...

```
height=512,
num_images_per_prompt=1,
).images[0]
```

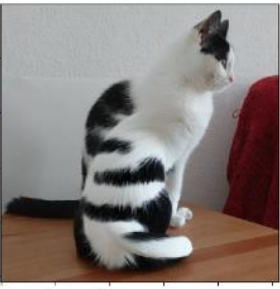
C:\Users\haase\mambaforge\envs\tea4\lib\site-packages\diffusers\models\attention\_processor.py:1231: UserWarning: 1Torch was not compiled with flash attention. (Triggered internally at C:\cb\pytorch\_100000000000\work\aten\src\ATen\native\transformers\cuda\sdp\_utils.cpp:455.)

```
hidden_states = F.scaled_dot_product_attention(
```

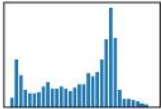
100% 50/50 [00:18<00:00, 2.67it/s]

[8]:  
np\_image = np.array(image)  
stackview.insight(np\_image)

[8]:



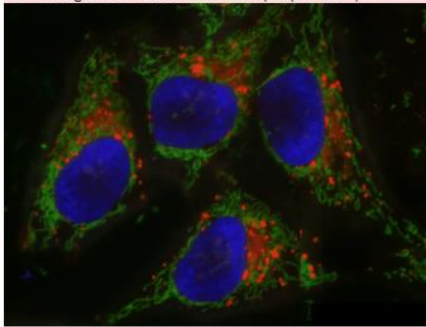
shape (512, 512, 3)  
dtype uint8  
size 768.0 kB  
min 0  
max 255



[ ]:

51\_inpainting\_dall-e.ipynb

Run Selected Cells Interrupt Kernel Restart Kernel...

[8]:  


Exercise

Apply filters such as the local standard deviation, Sobel or Laplace to the manipulated image and see if you can spot the masked region in the filtered images.

[ ]:

Exercise

Crop the image so that you can see the border between manipulated region and original image. Zoom in and show only low intensity values.

[ ]:

ScaDS.AI  
DRESDEN LEIPZIG

Robert Haase  
@haesleinhuepf  
BIDS Lecture 12/14  
June 18th 2024

Slide 44

TECHNISCHE  
UNIVERSITÄT  
DRESDEN

UNIVERSITÄT  
LEIPZIG



# Exercise: Vision

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

