ScaDS.AI
DRESDEN LEIPZIG

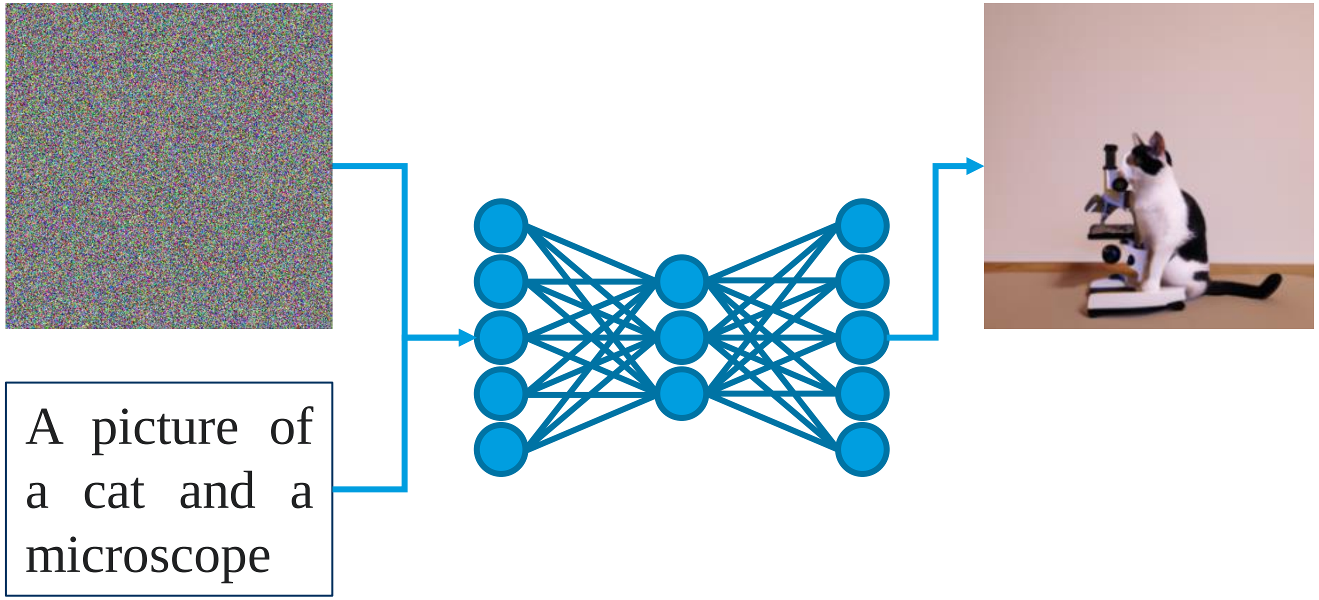
CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Image Generation and Vision Language Models

Robert Haase

Image Generation

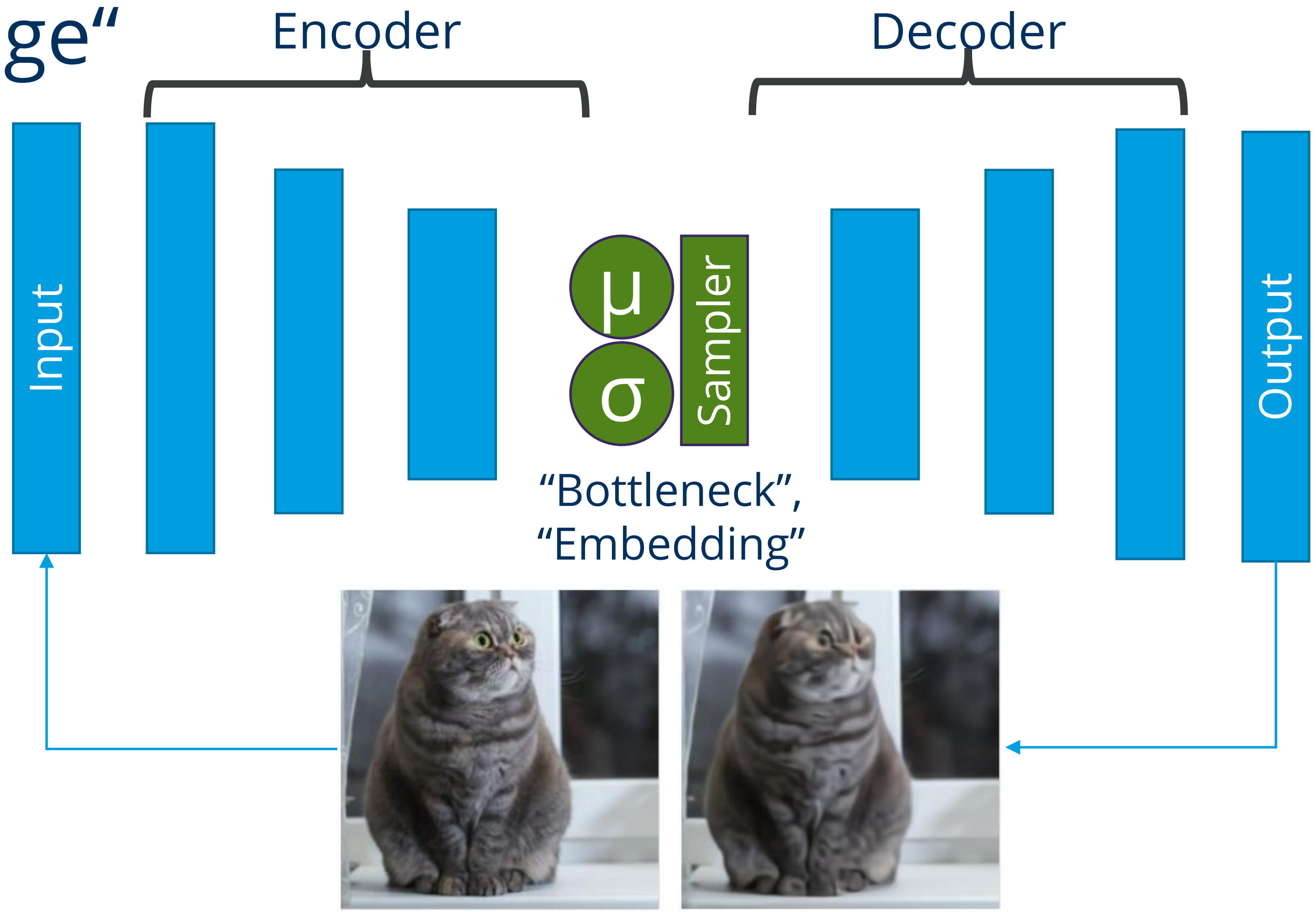
„text-to-image“



A picture of a cat and a microscope

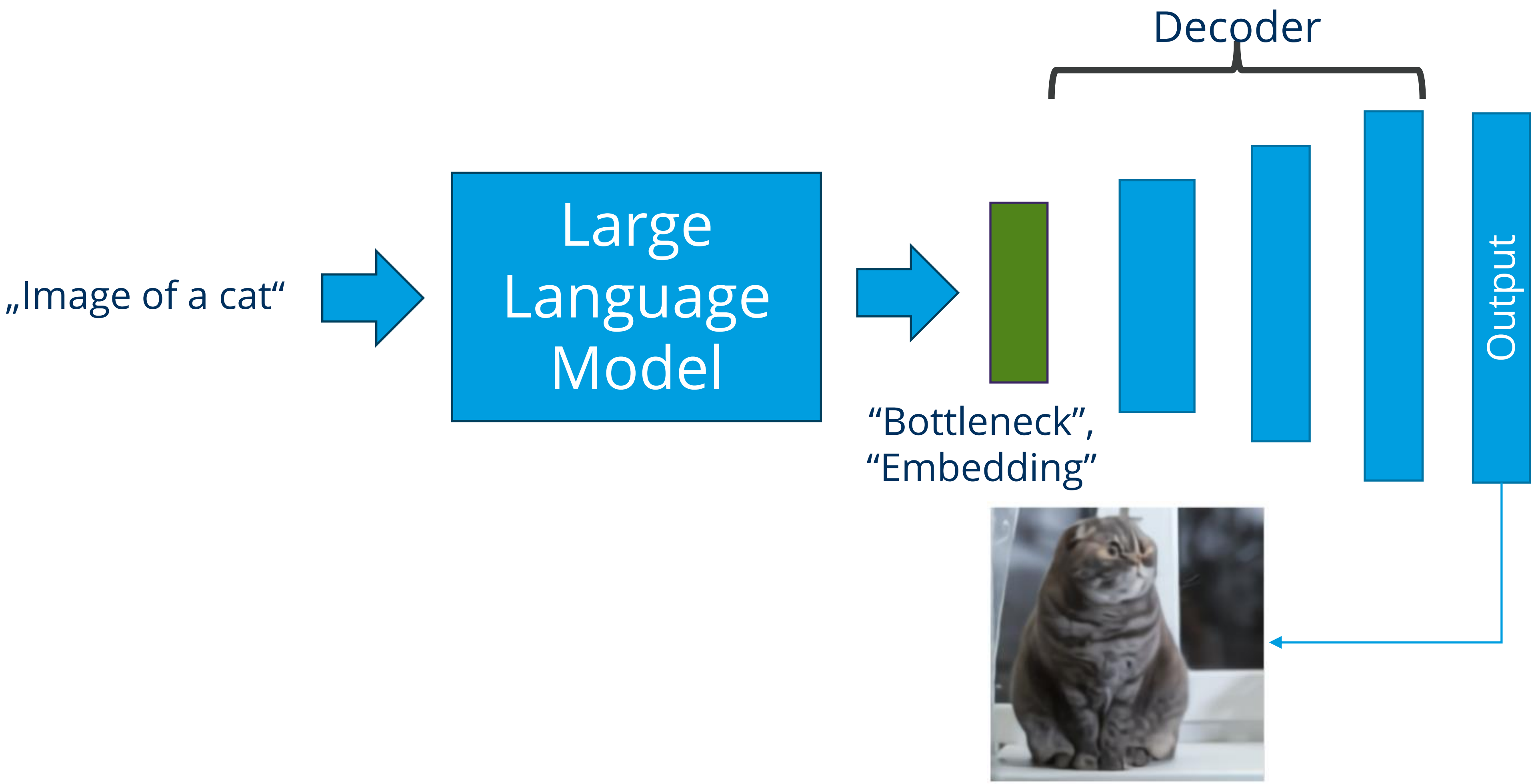
Variational Auto-Encoder

„image-to-image“



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

Image Generation



„Image of a cat“

Large Language Model

“Bottleneck”, “Embedding”

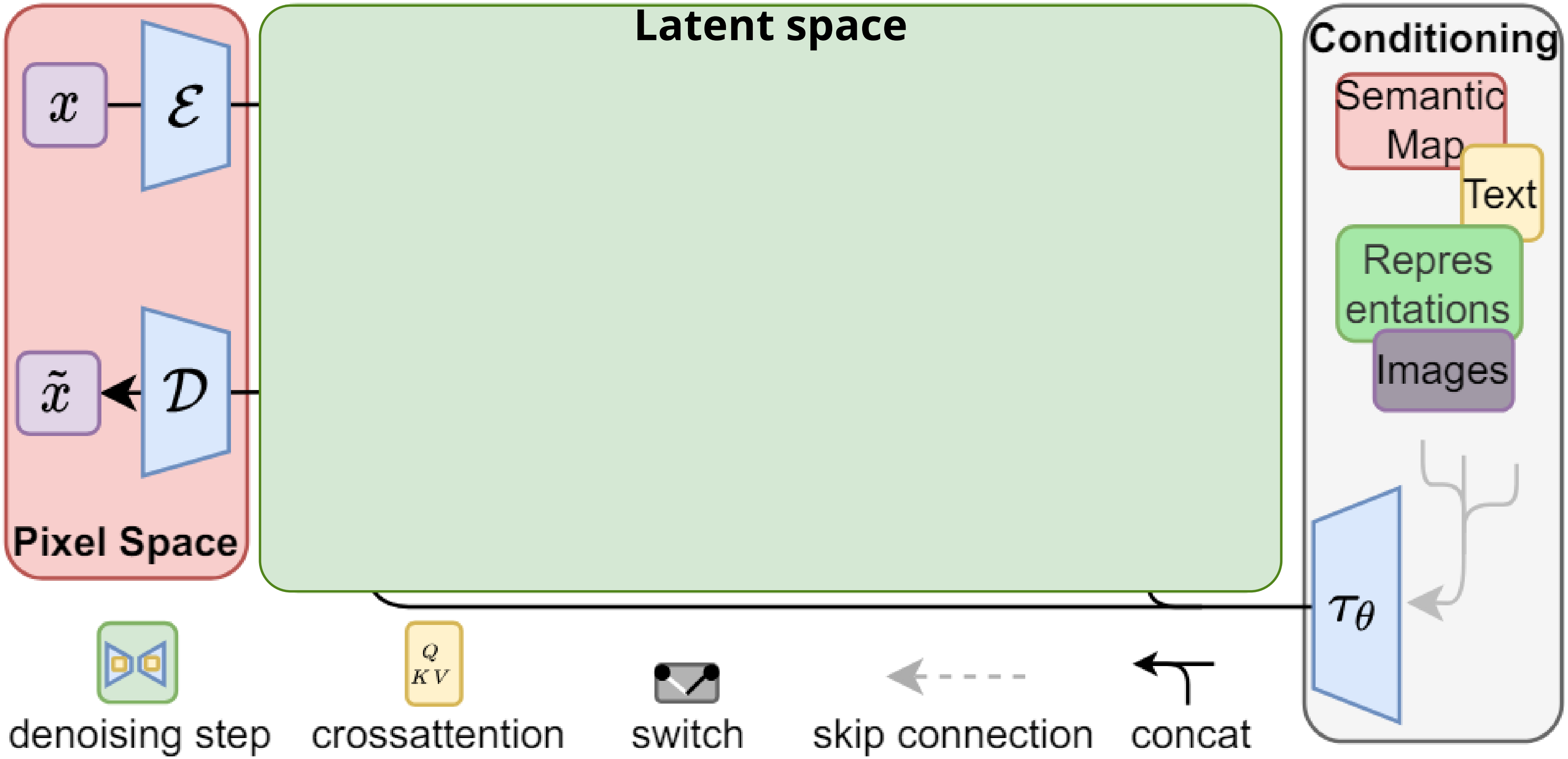
Decoder

Output

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

Stable Diffusion

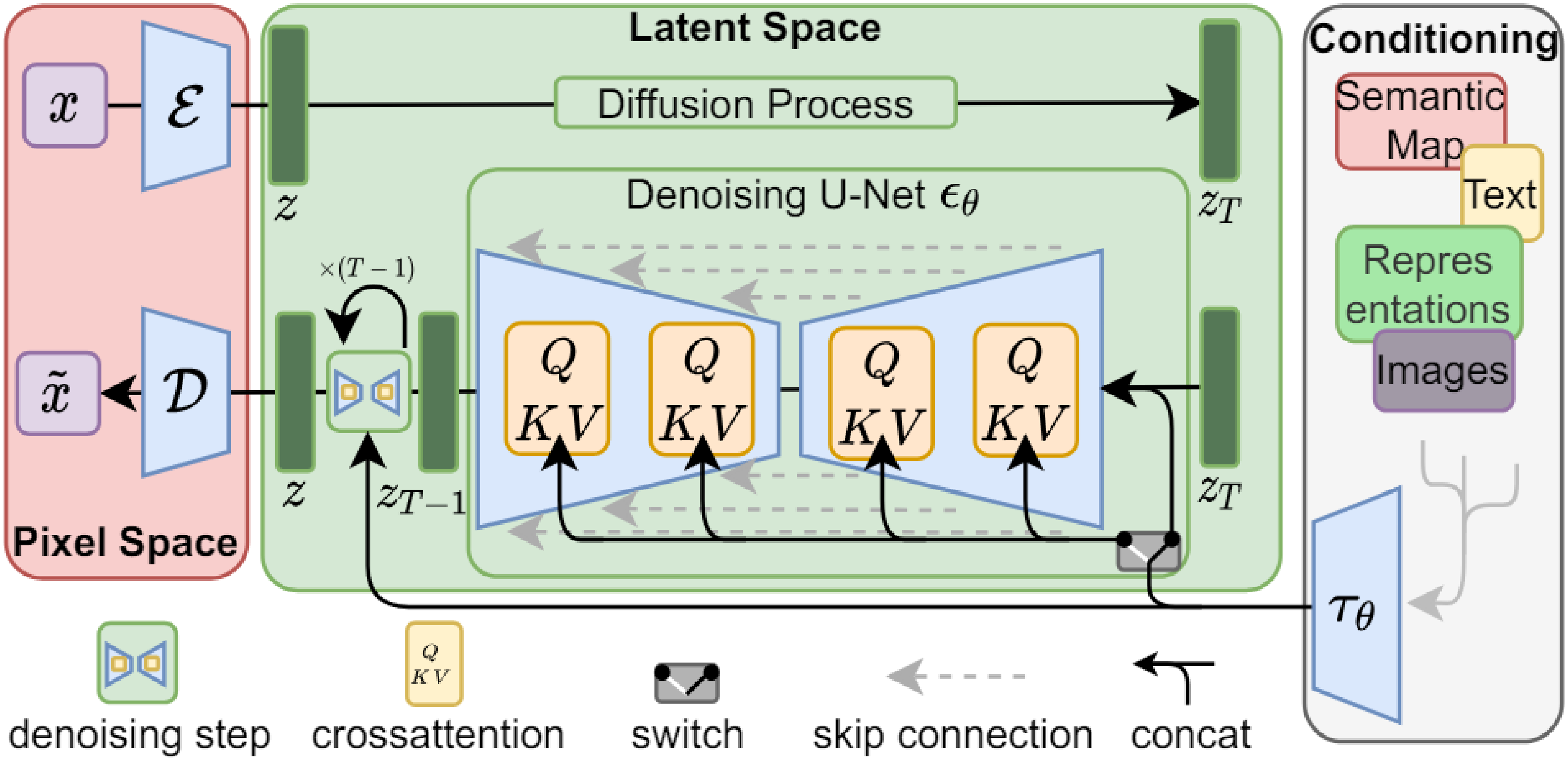
Diffusion: reverse denoising



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

Stable Diffusion

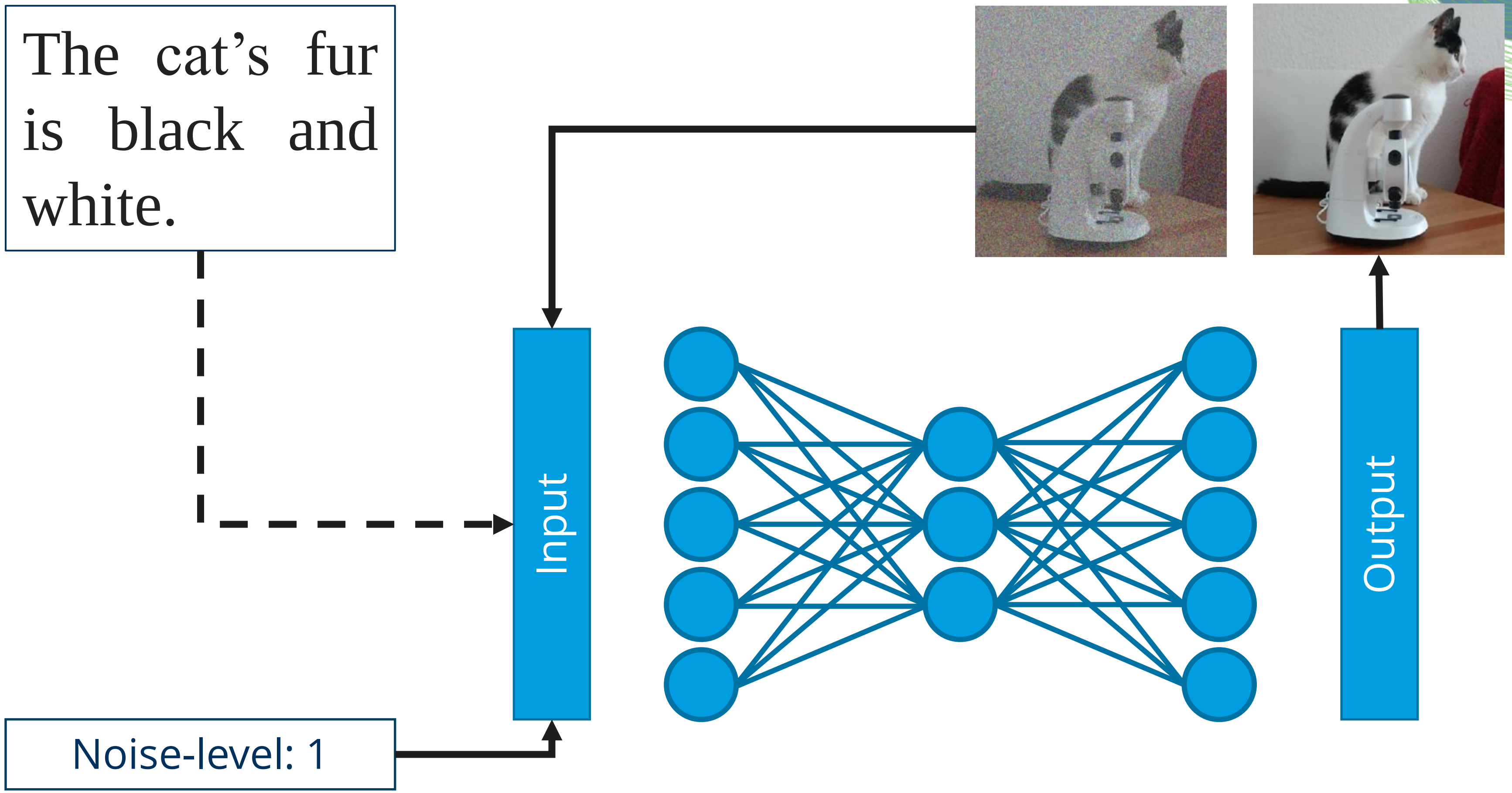
Diffusion: reverse denoising



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

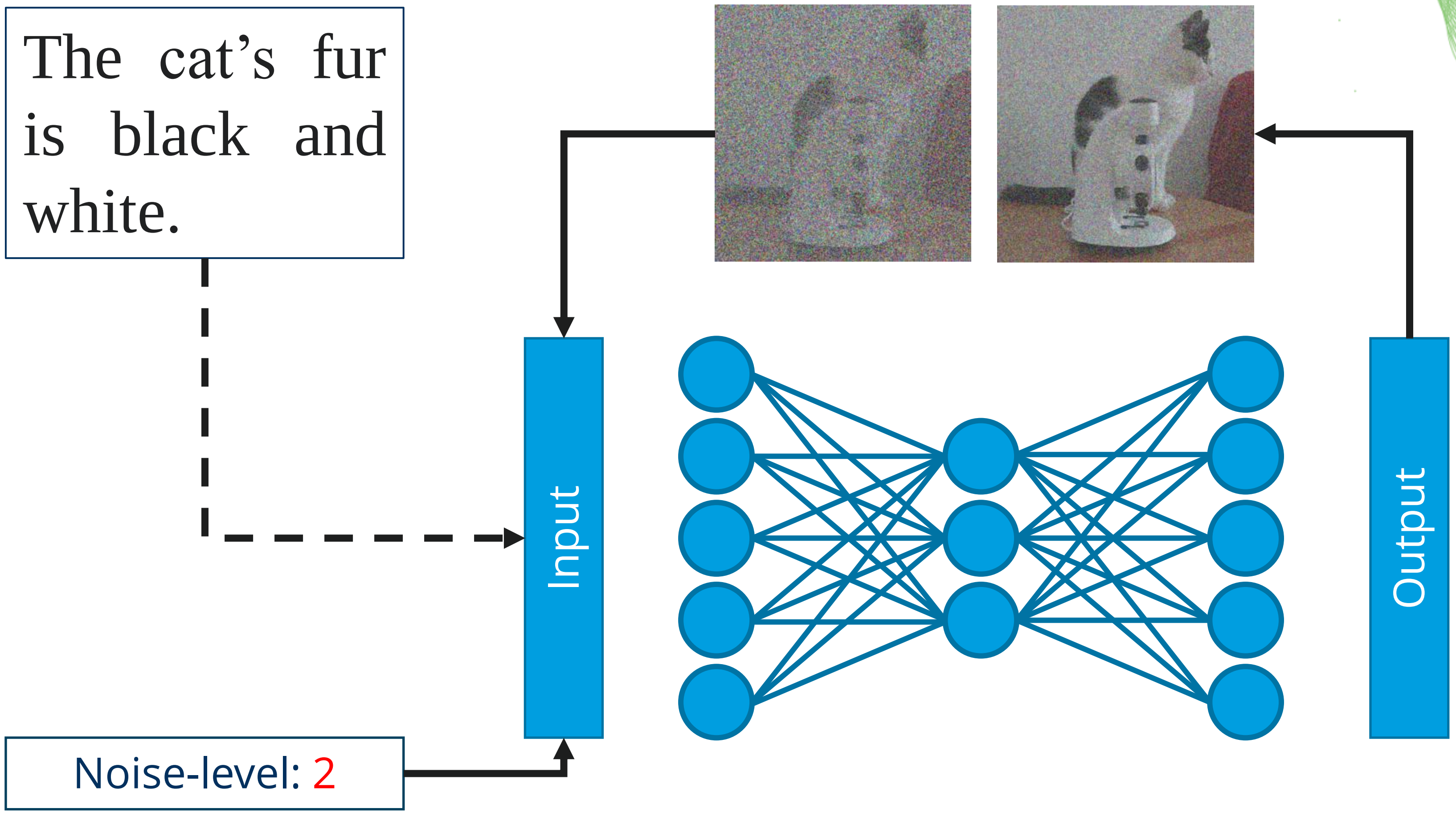


The cat’s fur is black and white.

Noise-level: 1

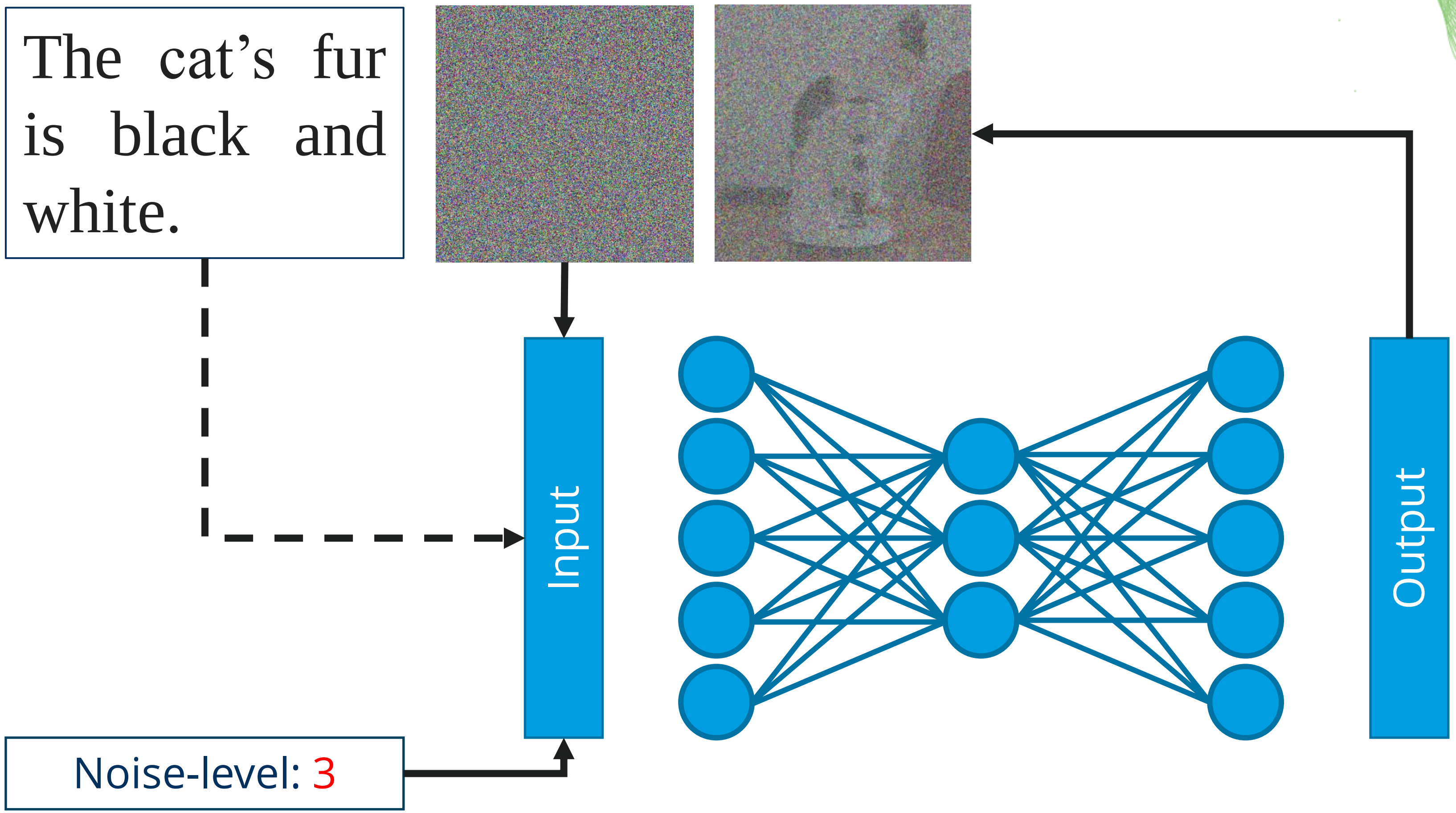
How does it work?

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



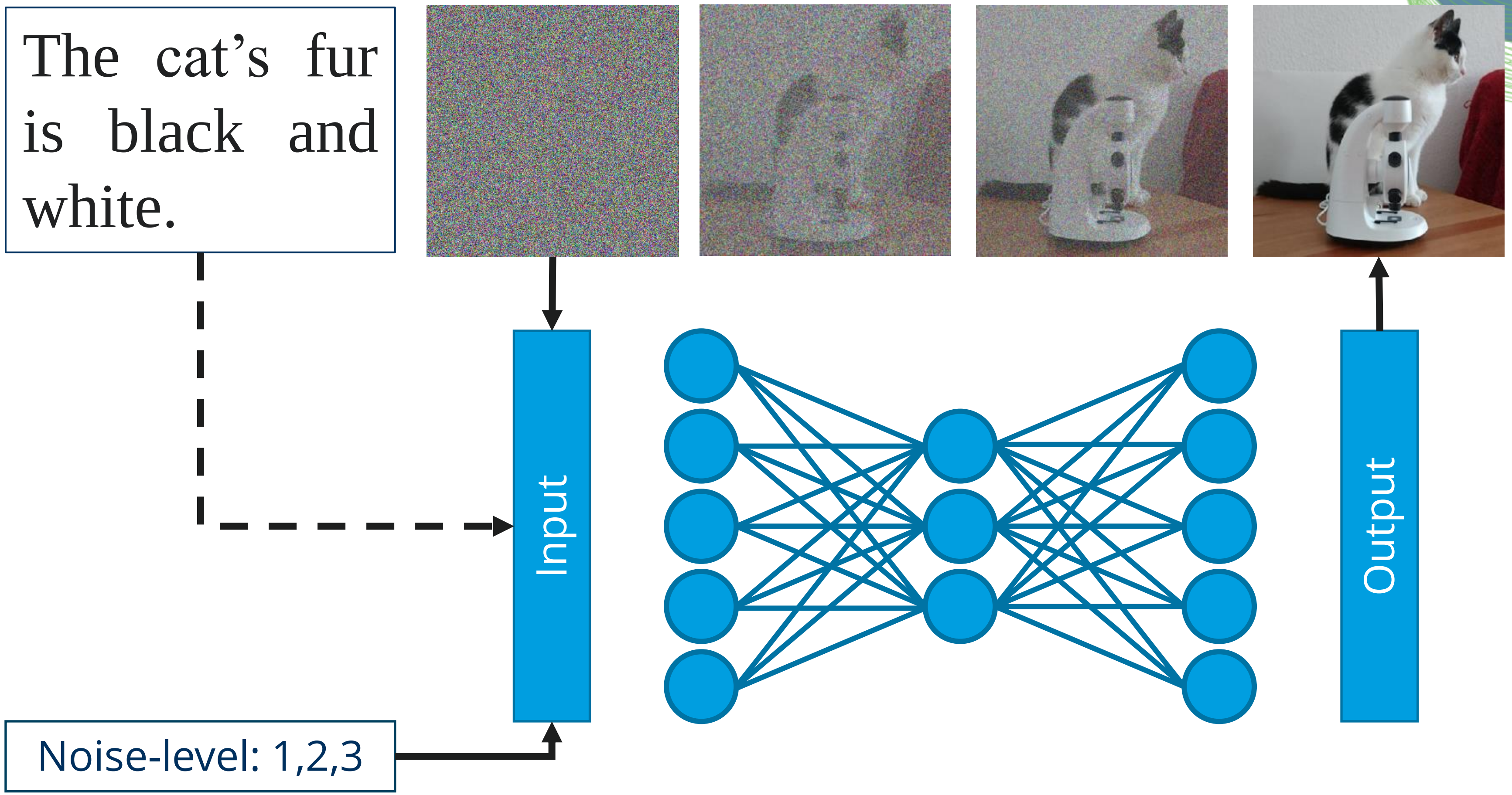
The cat's fur is black and white.

Noise-level: 3

How does it work?

Prediction is iterative denoising of:

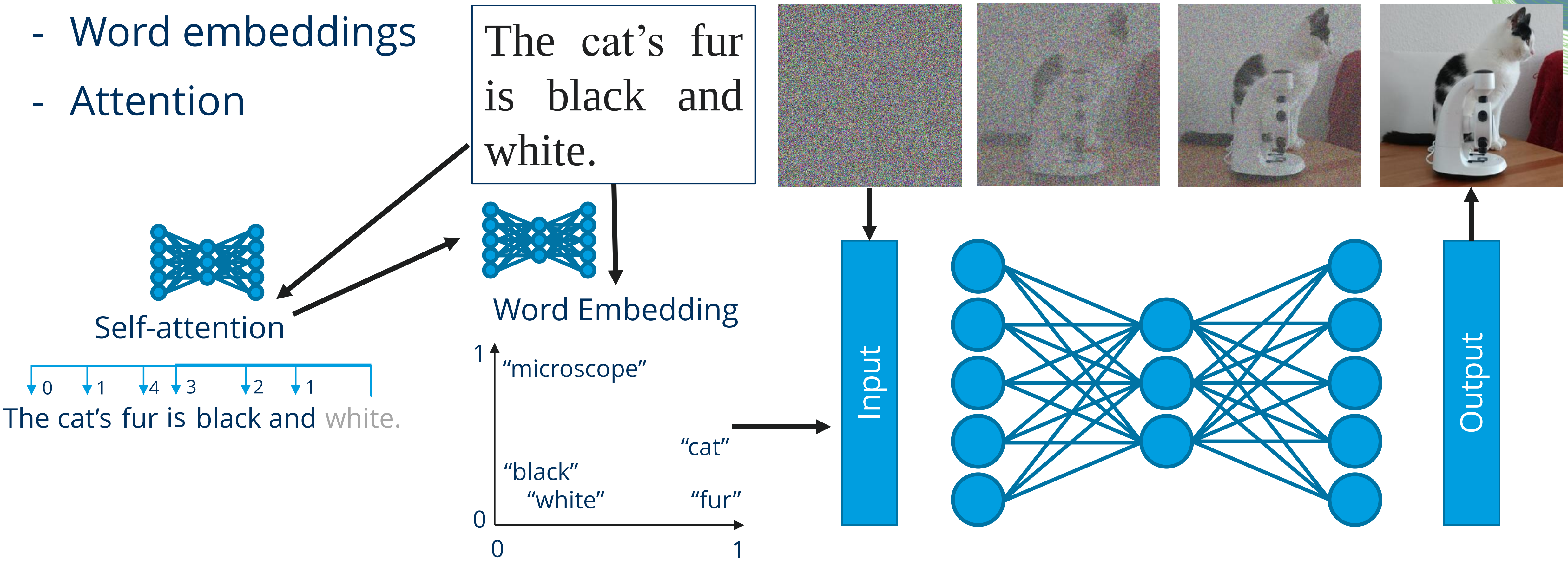
Pure noise + text prompt



How does it work?

Reminder:

- Word embeddings
- Attention



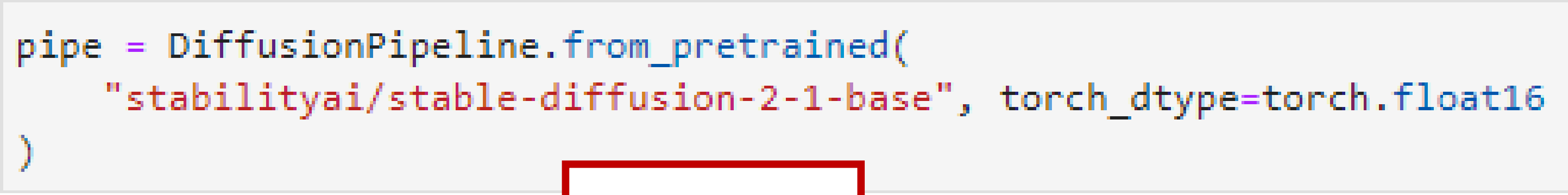
The cat's fur is black and white.

Self-attention

Word Embedding

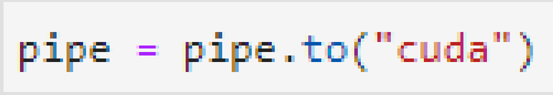
Image generation in Python: Huggingface

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

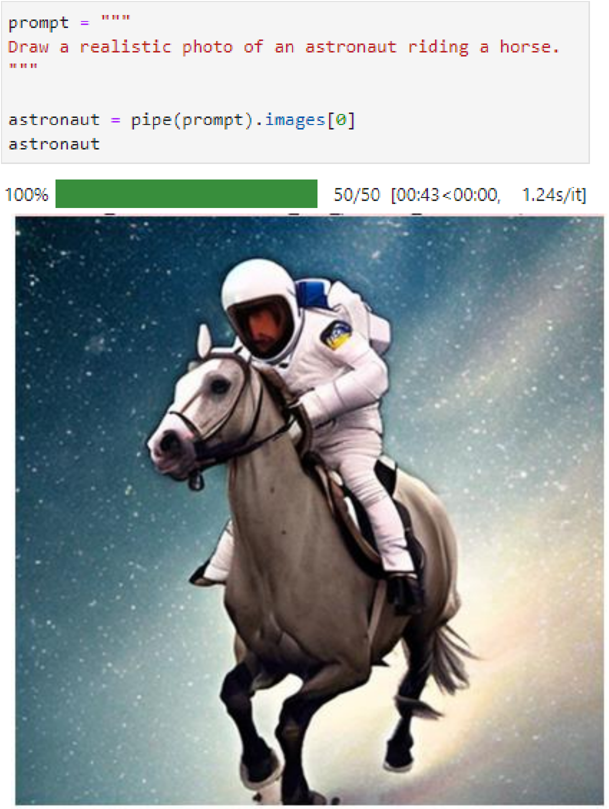


```
pipe = pipe.to("cuda")
```

Needs
Nvidia GPU

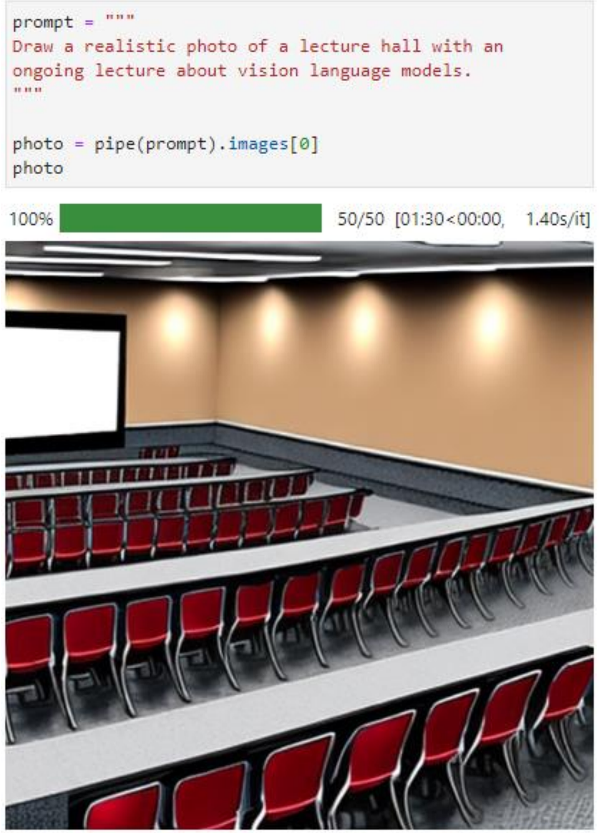
Image generation in Python: Huggingface

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



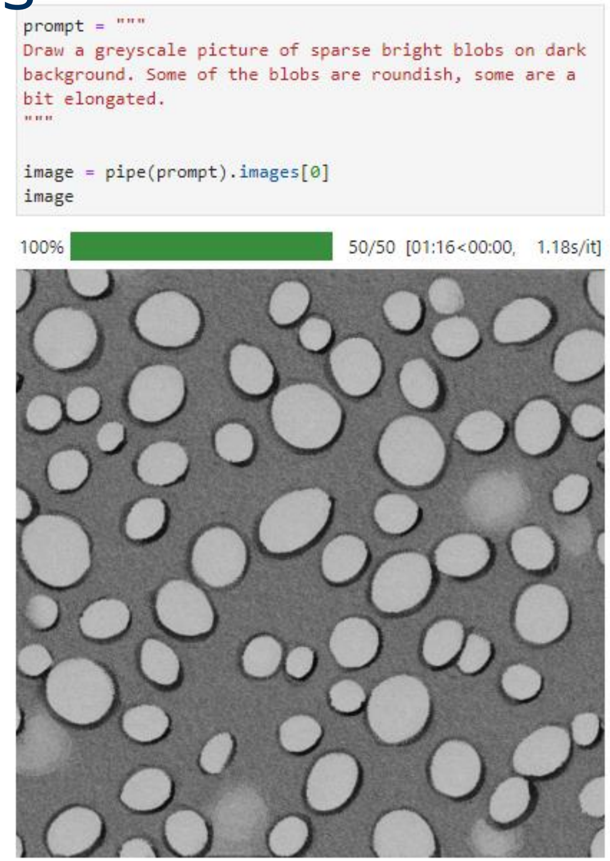
```
prompt = """
Draw a realistic photo of an astronaut riding a horse.
"""

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

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



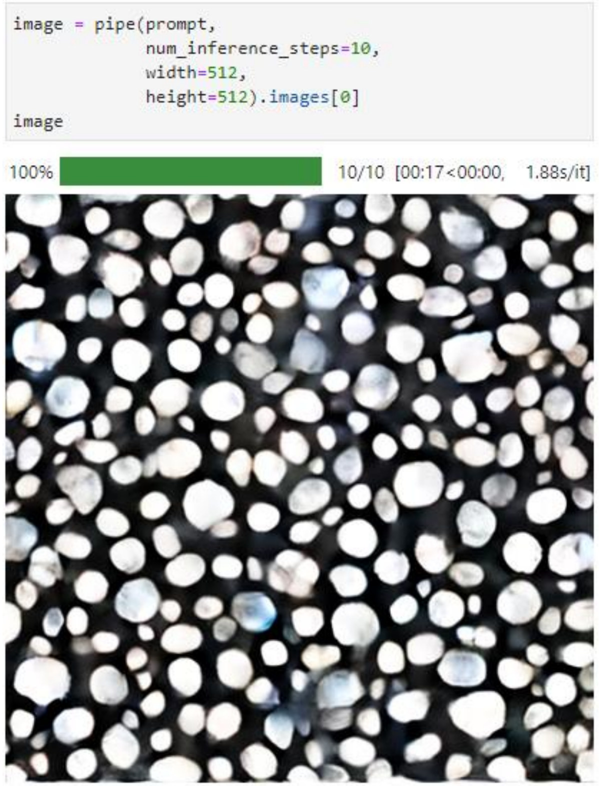
```
prompt = """
Draw a realistic photo of a lecture hall with an
ongoing lecture about vision language models.
"""

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]
```

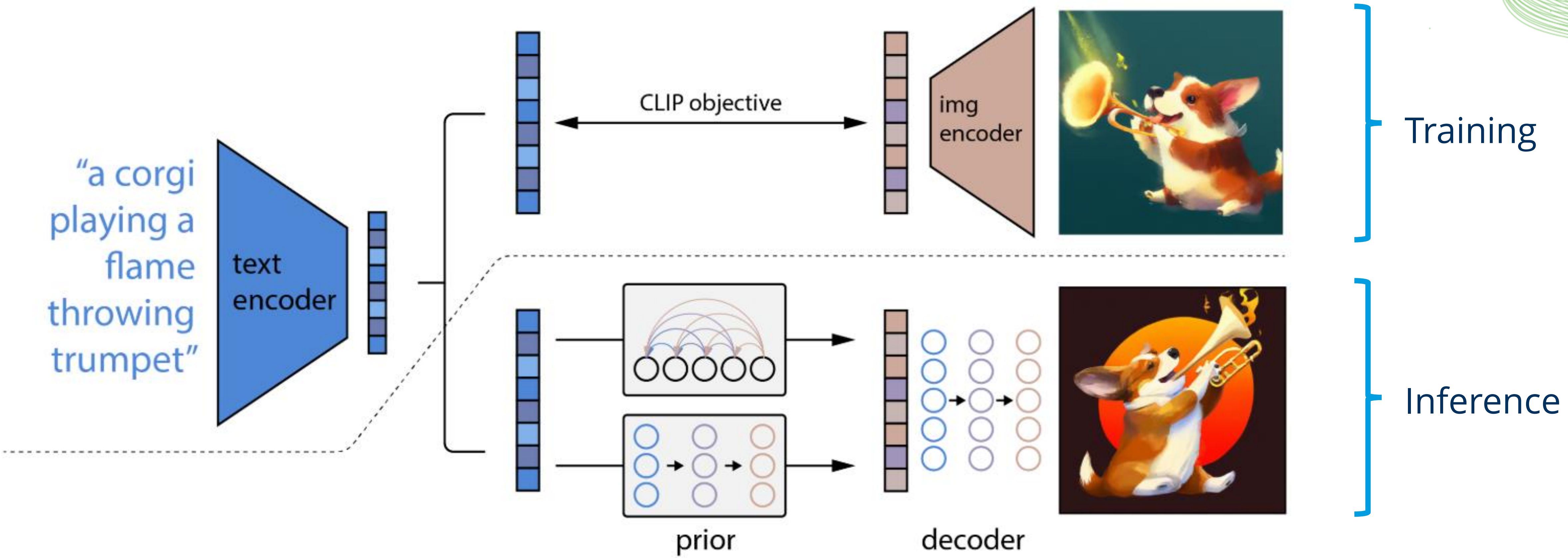


```
image = pipe(prompt,
             num_inference_steps=10,
             width=512,
             height=512).images[0]
image
```

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

Dall-E

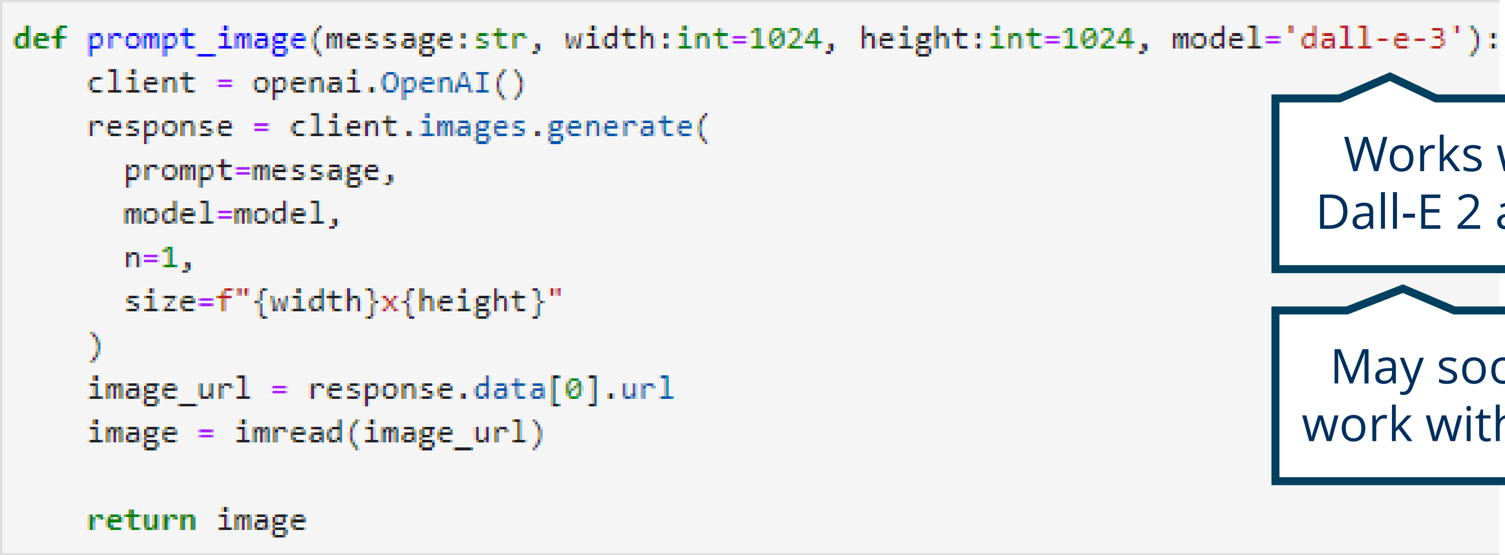
OpenAI‘s model for image geneation based on diffusion models + CLIP transformer



Source: Cropped from Ramesh et al. 2022, licensed CC-BY 4.0 https://arxiv.org/abs/2204.06125

Image generation in Python: Dall-E

No need for a GPU, but costs 💸 💸 💸



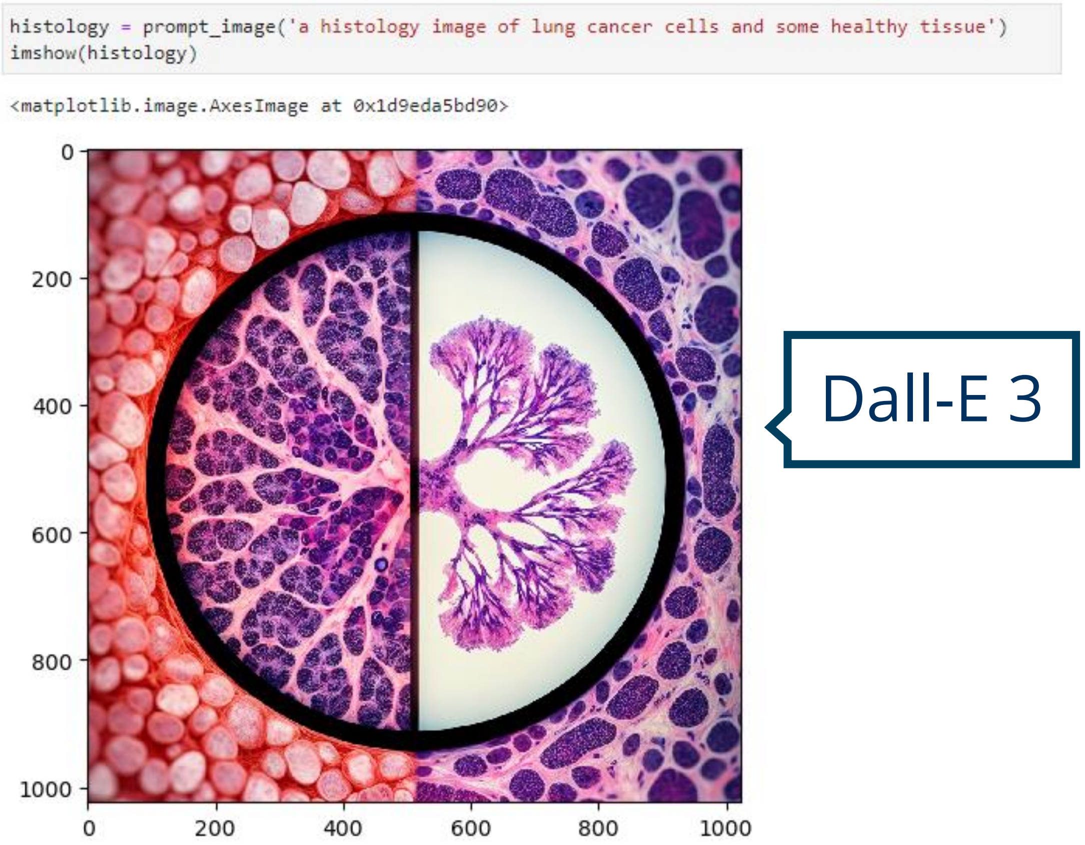
```
def prompt_image(message:str, width:int=1024, height:int=1024, model='dall-e-3'):
    client = openai.OpenAI()
    response = client.images.generate(
     prompt=message,
     model=model,
     n=1,
     size=f"{width}x{height}"
    )
    image_url = response.data[0].url
    image = imread(image_url)

    return image
```

Works with Dall-E 2 and 3

May soon also work with gpt-4o

Image generation in Python: Dall-E

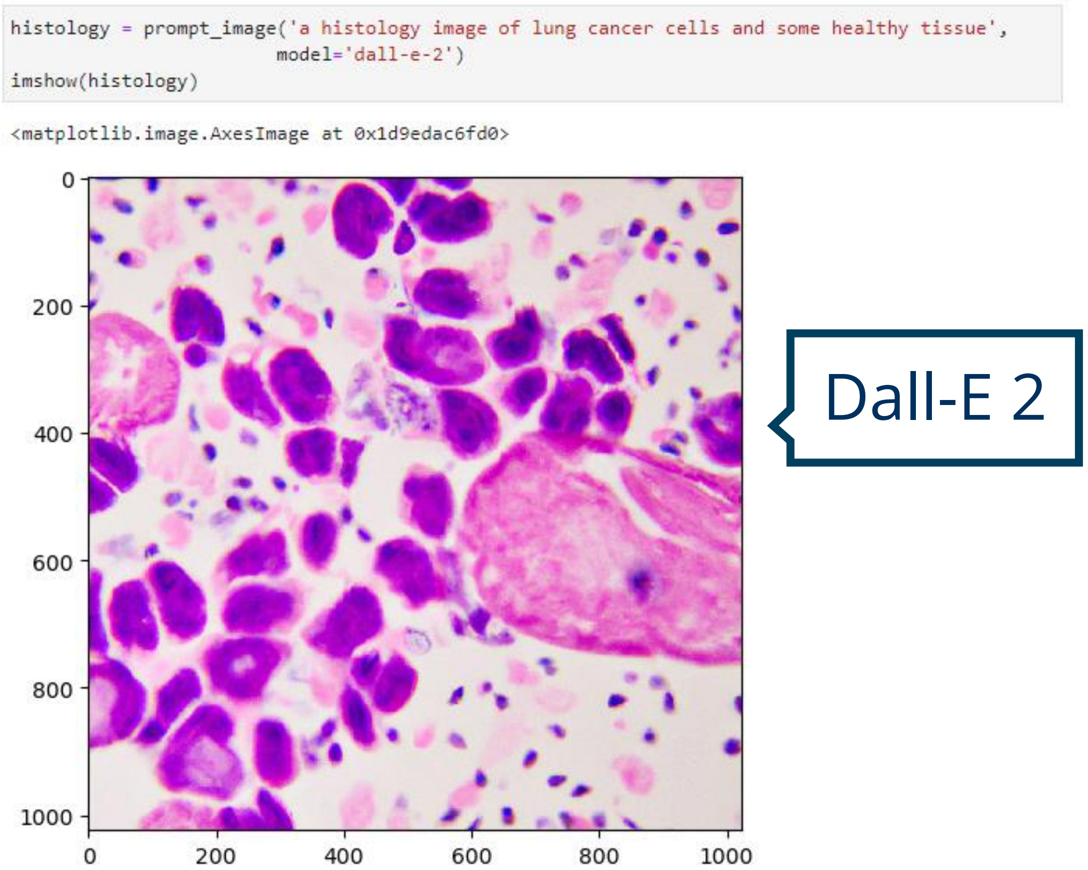
Is Dall-E 2 more capable of creating realistic microscopy images than Dall-E 3?



```
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)
```

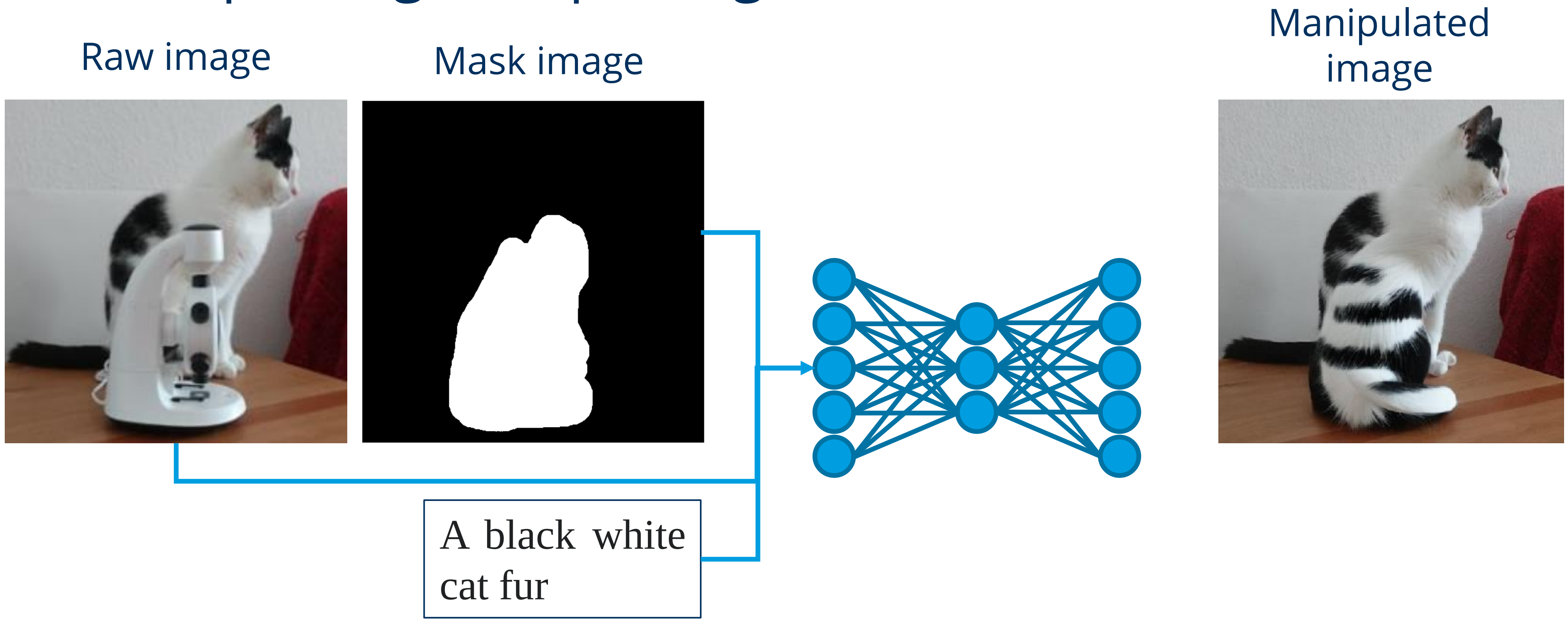
```
<matplotlib.image.AxesImage at 0x1d9edac6fd0>
```

Dall-E 2

Inpainting

Replacing regions in images

(also „Gap-filling“, „Replacing“)



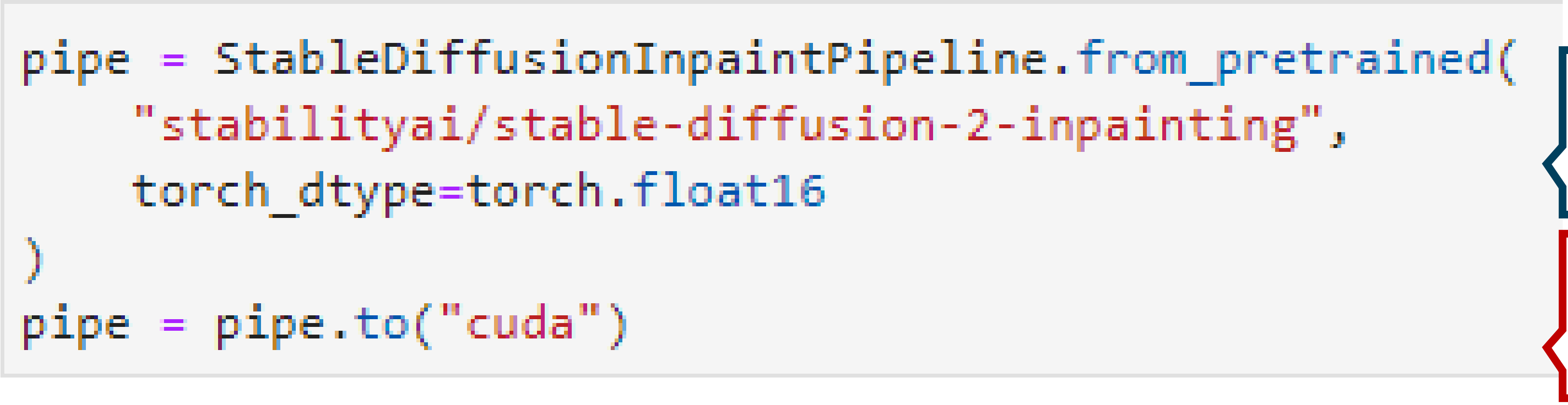
Raw image

Mask image

Manipulated image

A black white cat fur

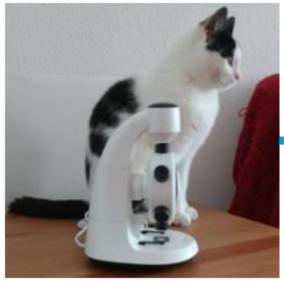
Inpainting in Python: Huggingface

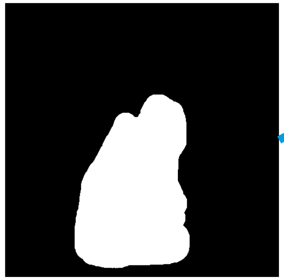


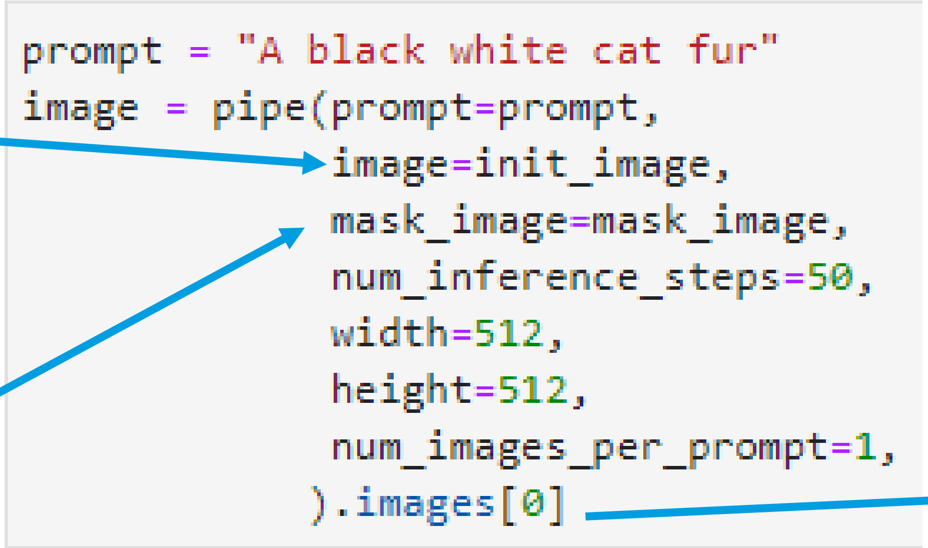
```
pipe = StableDiffusionInpaintPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-inpainting",
    torch_dtype=torch.float16
)
pipe = pipe.to("cuda")
```

Downloads 4.8 GB

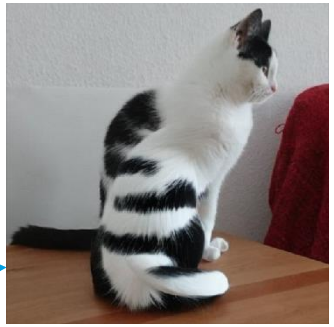
Needs Nvidia GPU







```
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]
```



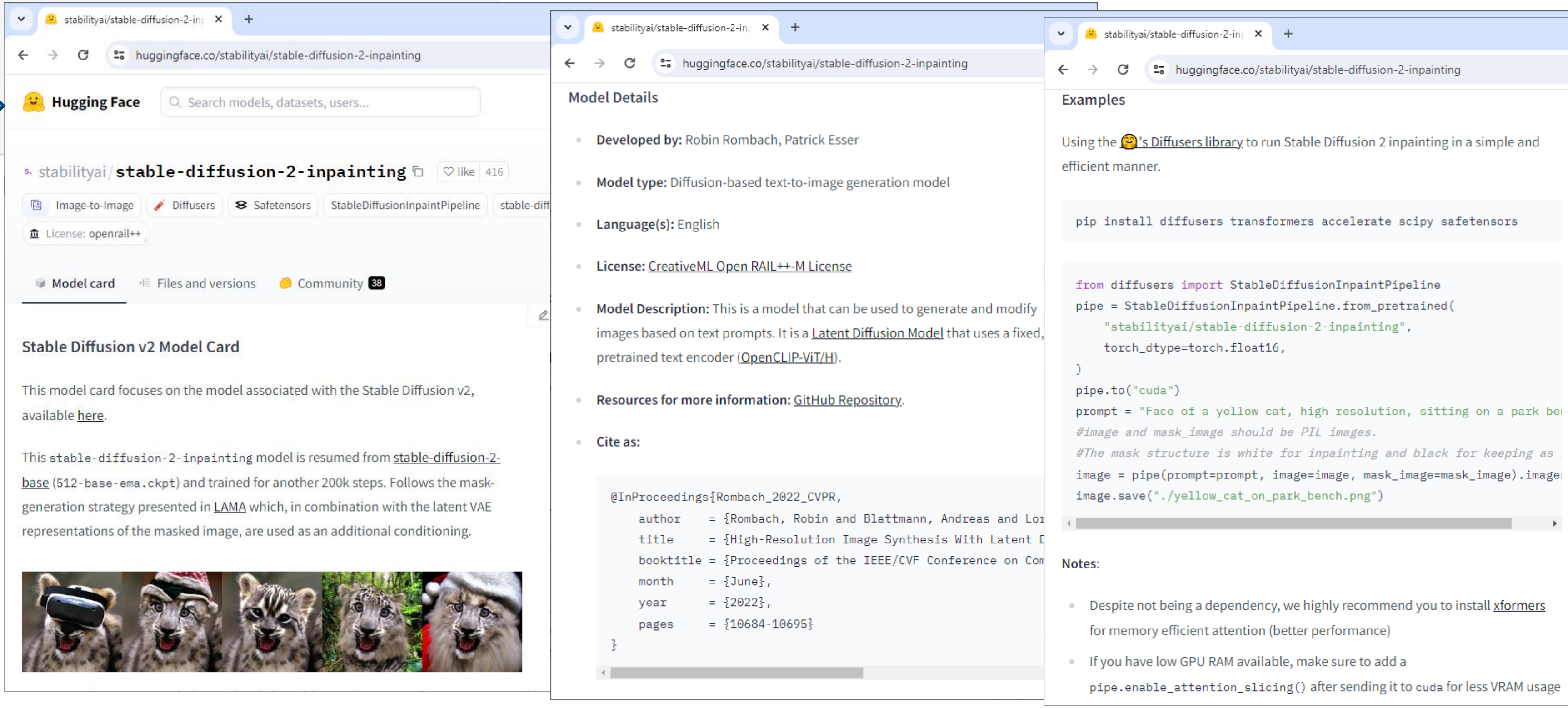
Robert Haase
@haesleinhuepf
BIDS Lecture 12/14
June 18th 2024

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")
```



stabilityai/stable-diffusion-2-inp
huggingface.co/stabilityai/stable-diffusion-2-inpainting
Hugging Face
Search models, datasets, users...
stabilityai / stable-diffusion-2-inpainting
like 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 Lo
    title     = {High-Resolution Image Synthesis With Latent D
    booktitle = {Proceedings of the IEEE/CVF Conference on Co
    month     = {June},
    year      = {2022},
    pages     = {10684-10695}
}
```

Examples

Using the 🤗's 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 be
#image and mask_image should be PIL images.
#The mask structure is white for inpainting and black for keeping as
image = pipe(prompt=prompt, image=image, mask_image=mask_image).image
image.save("./yellow_cat_on_park_bench.png")
```

Notes:

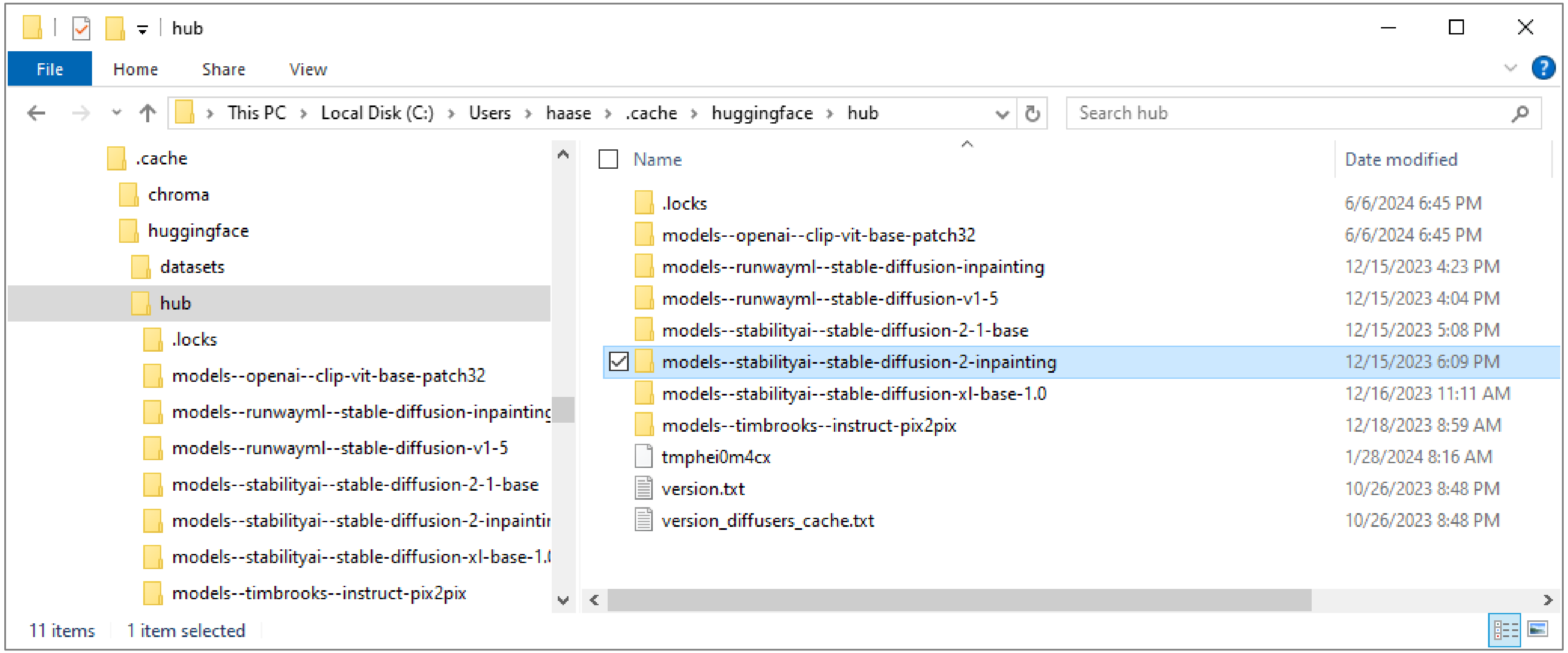
- Despite not being a dependency, we highly recommend you to install xformers for memory efficient attention (better performance)
- If you have low GPU RAM available, make sure to add a `pipe.enable_attention_slicing()` after sending it to cuda for less VRAM usage

https://huggingface.co/stabilityai/stable-diffusion-2-inpainting

Inpainting in Python: Huggingface

You find the downloaded models cached in your home directory

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



This PC › Local Disk (C:) › Users › haase › .cache › huggingface › hub

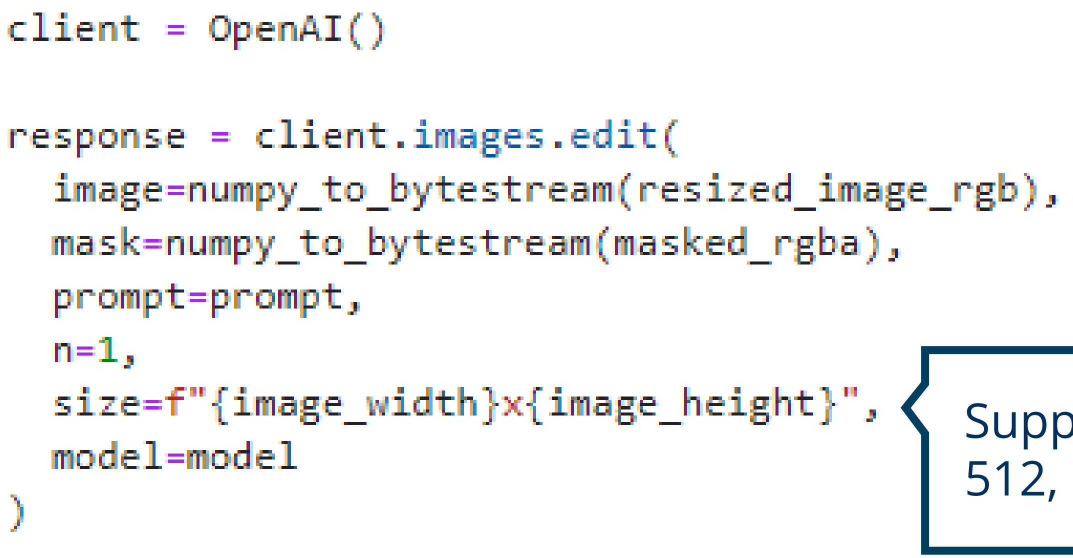
- .cache
  - chroma
  - huggingface
    - datasets
    - hub
      - .locks
      - models--openai--clip-vit-base-patch32
      - models--runwayml--stable-diffusion-inpainting
      - models--runwayml--stable-diffusion-v1-5
      - models--stabilityai--stable-diffusion-2-1-base
      - models--stabilityai--stable-diffusion-2-inpaintir
      - models--stabilityai--stable-diffusion-xl-base-1.
      - models--timbrooks--instruct-pix2pix

	Name	Date modified
□	.locks	6/6/2024 6:45 PM
	models--openai--clip-vit-base-patch32	6/6/2024 6:45 PM
	models--runwayml--stable-diffusion-inpainting	12/15/2023 4:23 PM
	models--runwayml--stable-diffusion-v1-5	12/15/2023 4:04 PM
	models--stabilityai--stable-diffusion-2-1-base	12/15/2023 5:08 PM
☑	models--stabilityai--stable-diffusion-2-inpainting	12/15/2023 6:09 PM
	models--stabilityai--stable-diffusion-xl-base-1.0	12/16/2023 11:11 AM
	models--timbrooks--instruct-pix2pix	12/18/2023 8:59 AM
	tmphei0m4cx	1/28/2024 8:16 AM
	version.txt	10/26/2023 8:48 PM
	version_diffusers_cache.txt	10/26/2023 8:48 PM

11 items 1 item selected

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

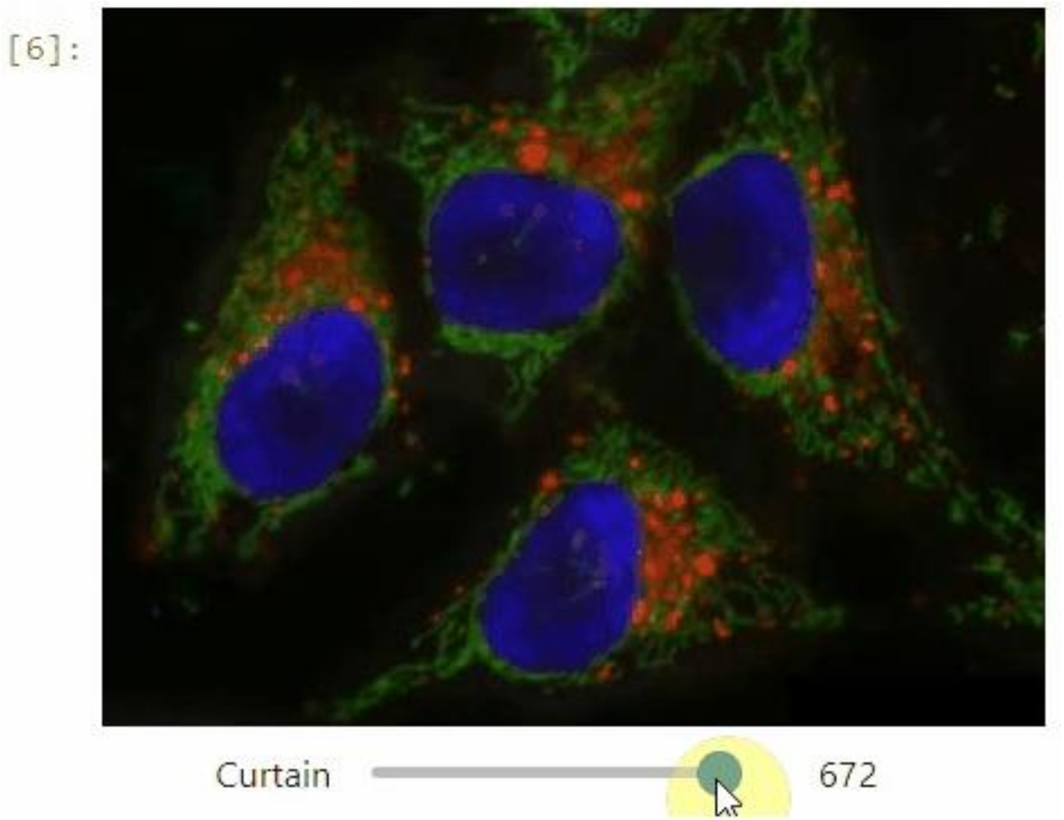
Size must match

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….

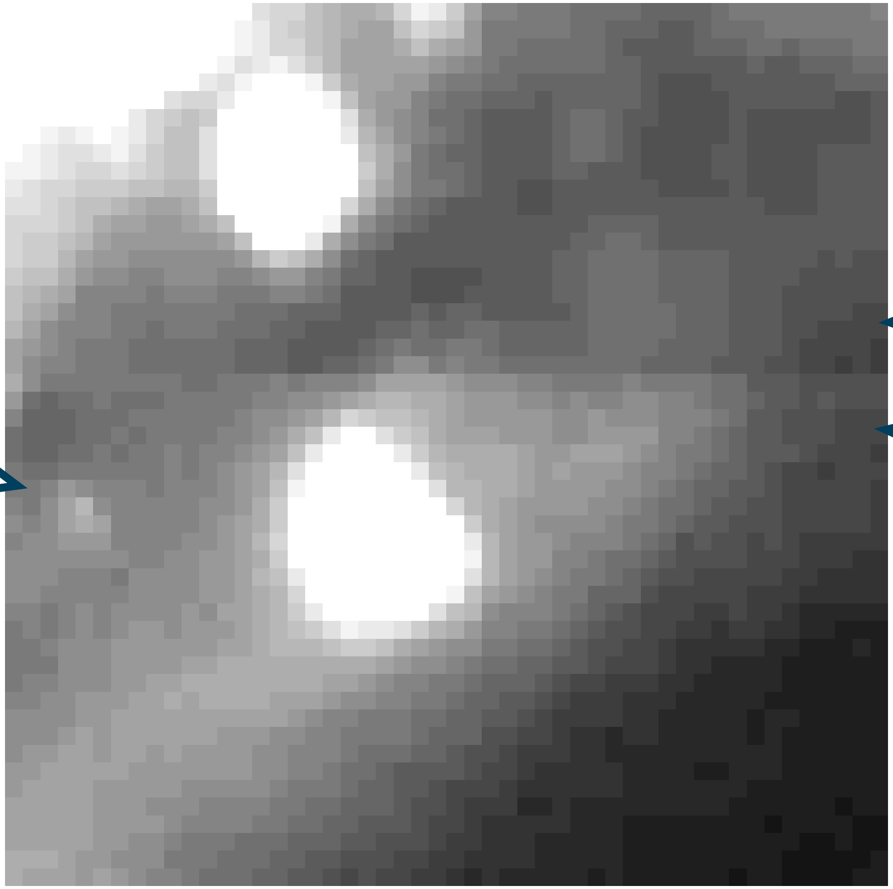


Interesting challenges for our community ahead

https://github.com/haesleinhuepf/darth-d/blob/main/demo/demo_replacing.ipynb

Image manipulation detection

The noise pattern differs between raw and processed images…



Original raw image

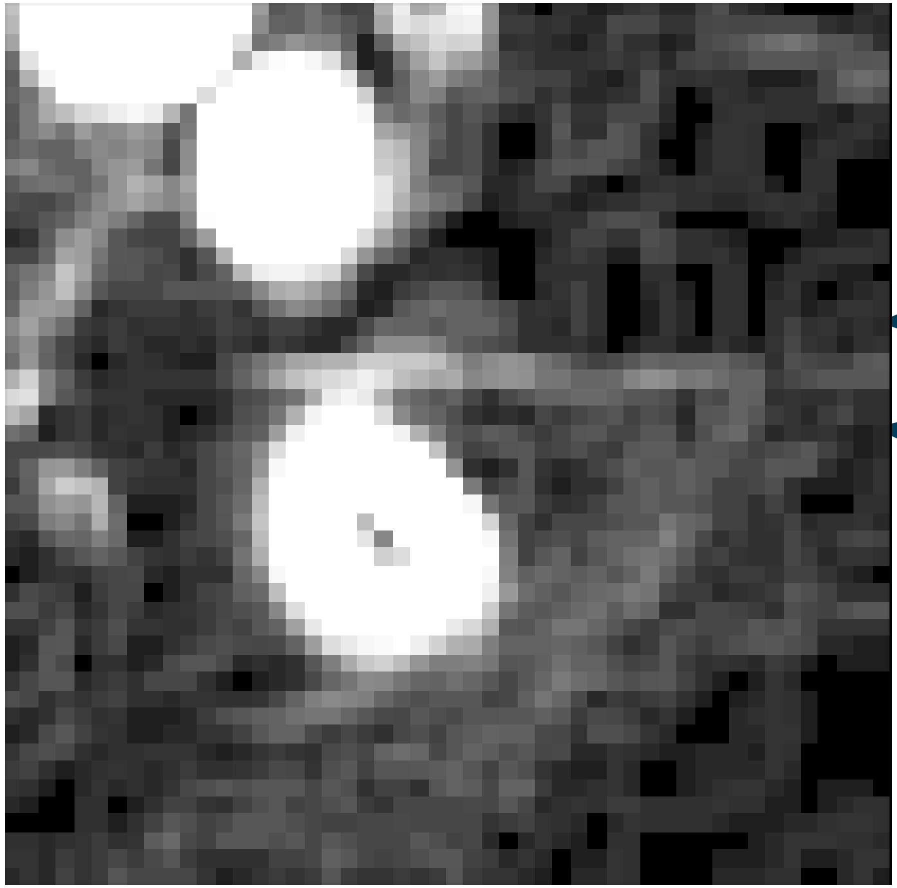
„Untouched region“

Manipulated region

Image manipulation detection

e.g. by studying noise-patterns

Local standard deviation filter



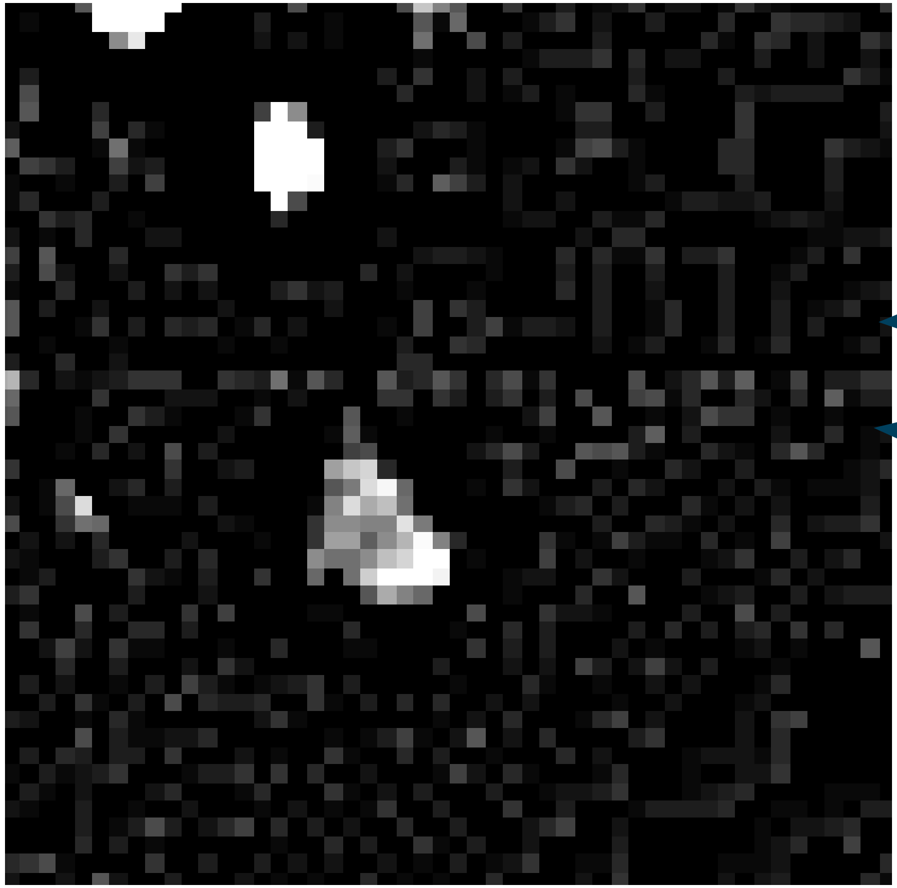
„Untouched region“

Manipulated region

Image manipulation detection

e.g. by studying noise-patterns

Sobel filter

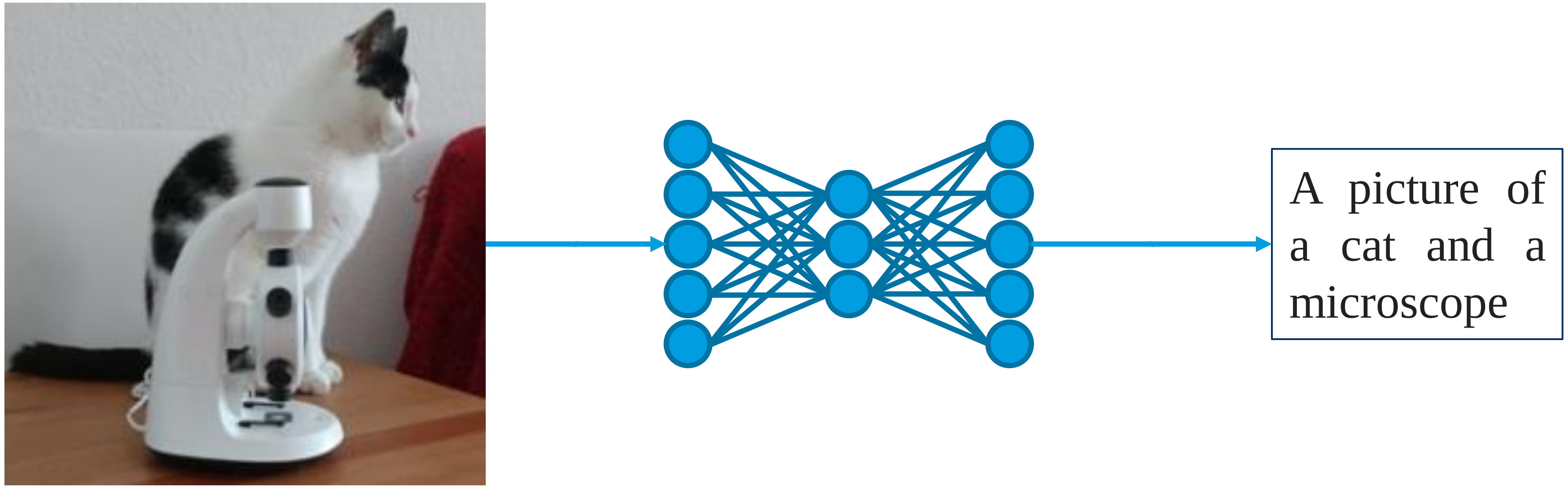


„Untouched region“

Manipulated region

Vision Language Models

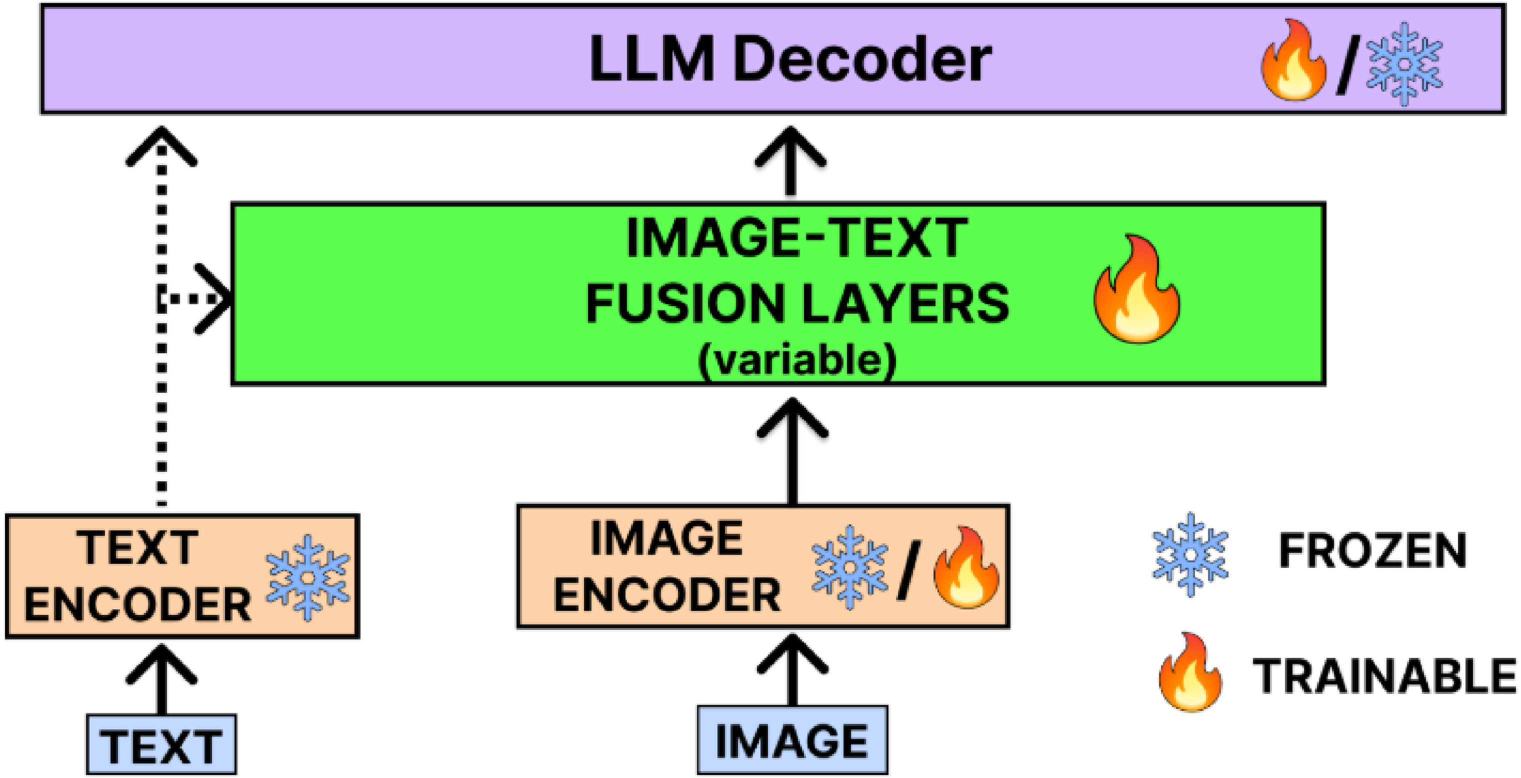
- Classifying images 🥱
- Describing images



A picture of a cat and a microscope

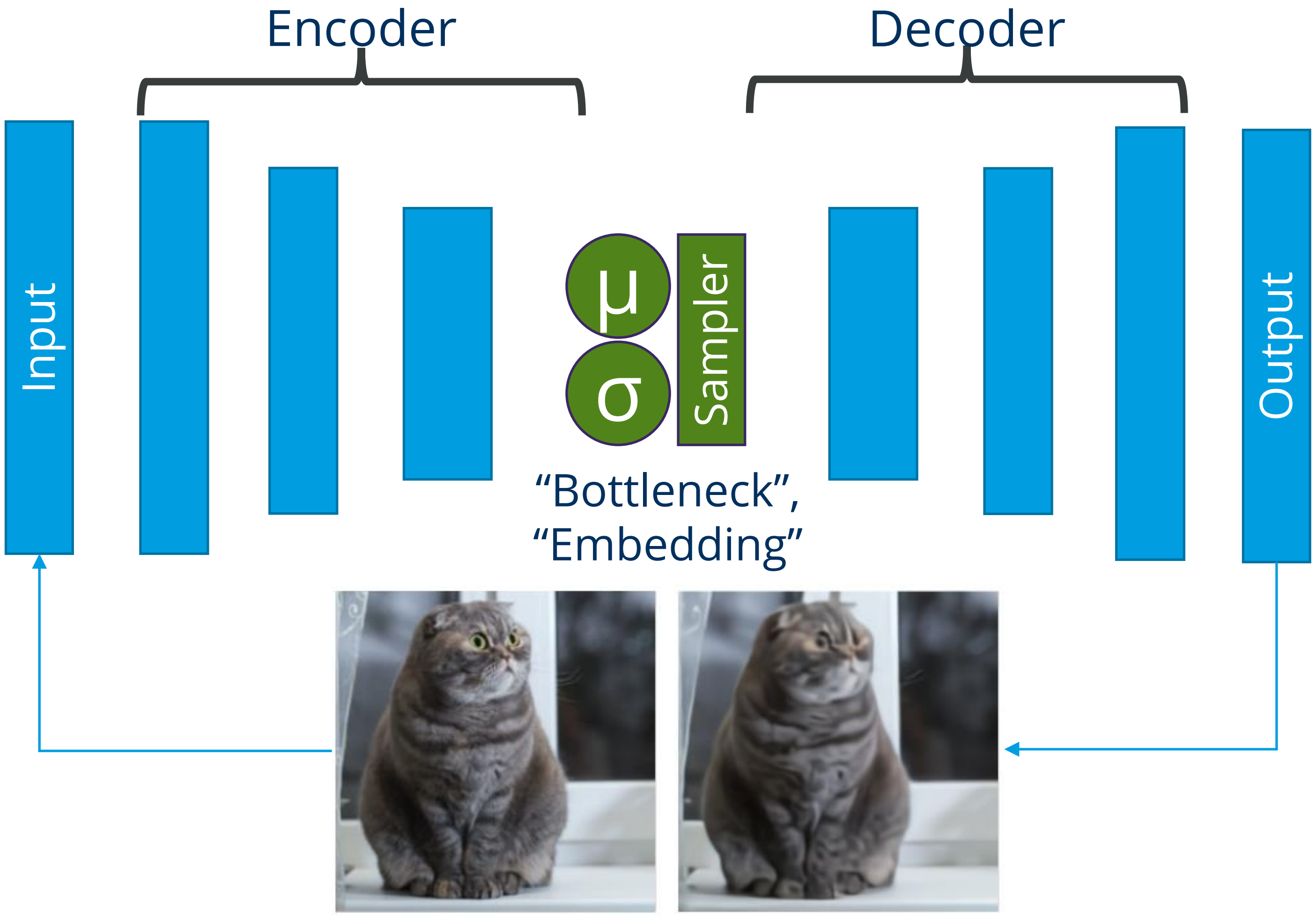
Vision Language Models (VLMs)

Goal: Describe images



Source: Gosh et al 2024, licensed CC-BY 4.0
https://arxiv.org/abs/2404.07214v1

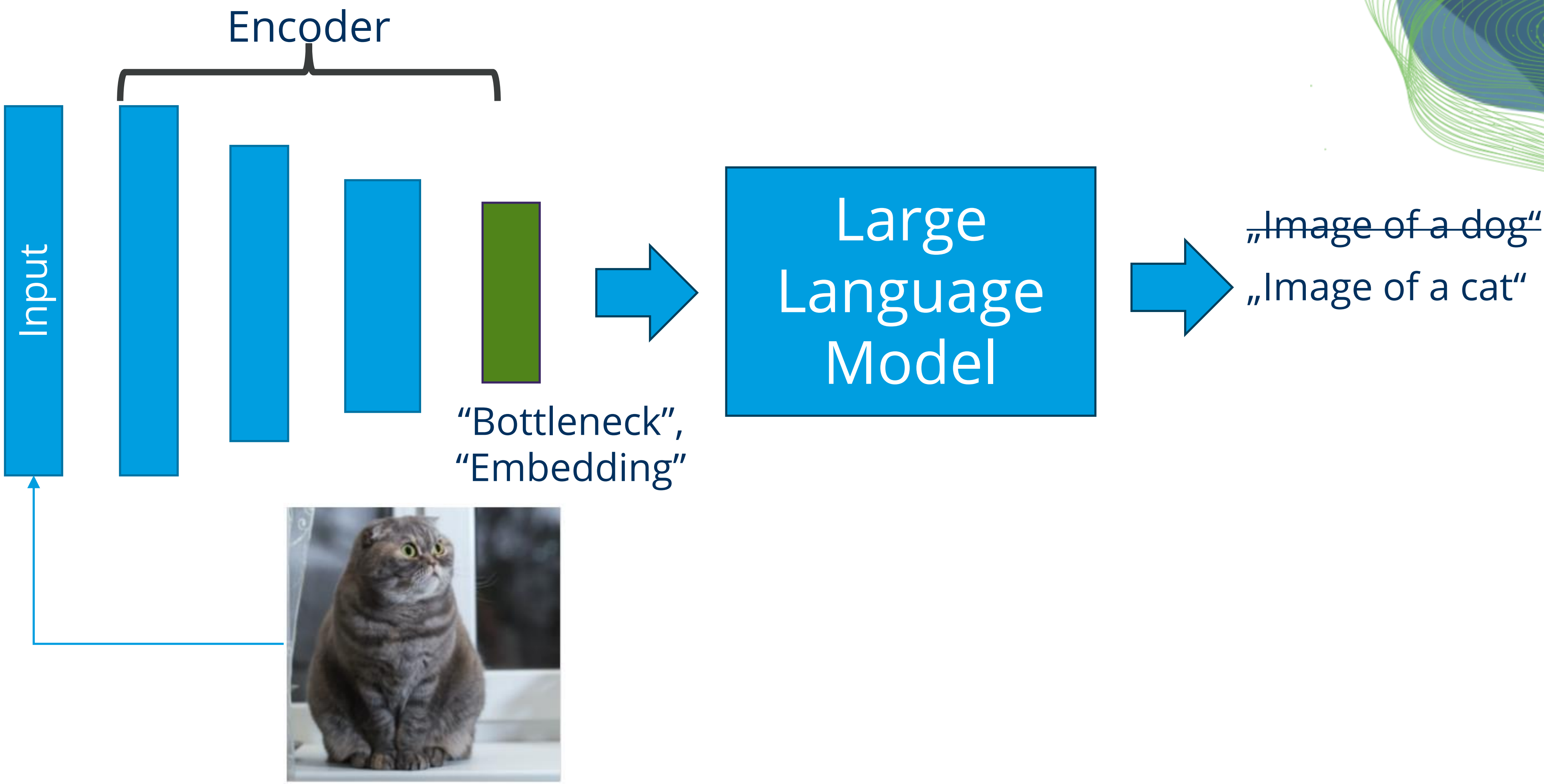
Variational Auto-Encoder



"Bottleneck",
"Embedding"

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

Image classification



Encoder

Input

“Bottleneck”, “Embedding”

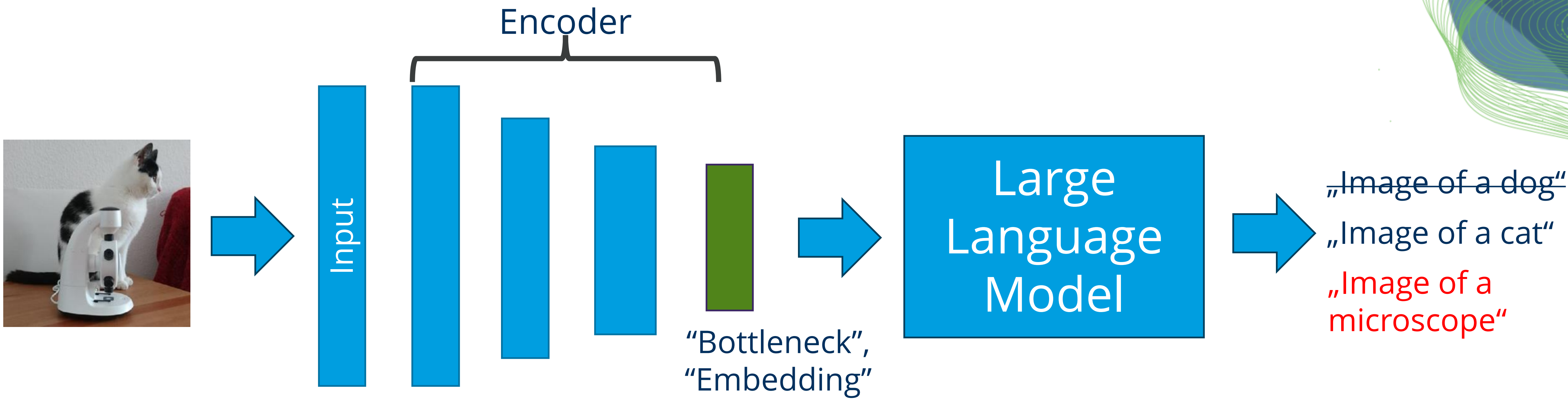
Large Language Model

~~„Image of a dog“~~

„Image of a cat“

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

Image classification -> image describing



Encoder

Input

“Bottleneck”,
“Embedding”

Large Language Model

~~„Image of a dog“~~

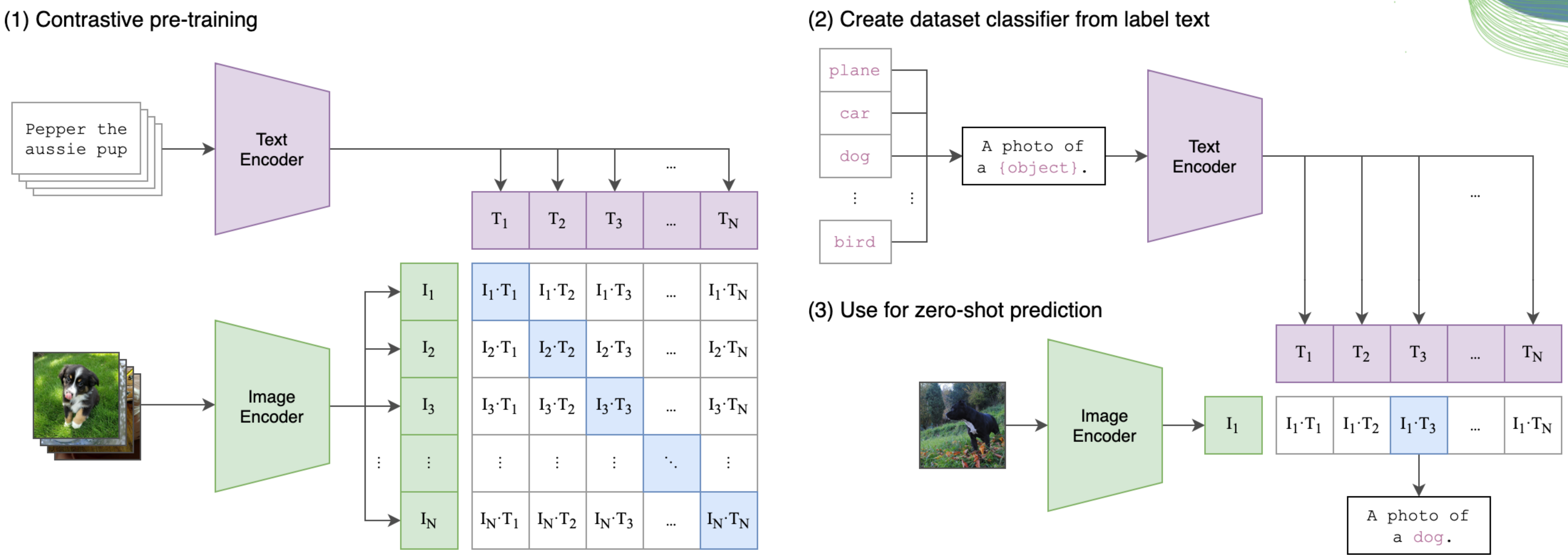
„Image of a cat“

„Image of a microscope“

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

Contrastive Language-Image Pre-Training

„CLIP“ Transformers


(1) Contrastive pre-training

(2) Create dataset classifier from label text

(3) Use for zero-shot prediction

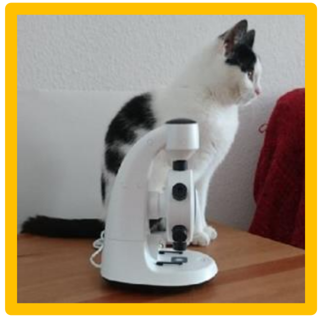
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
plane
car
dog
⋮
bird
A photo of a {object}.
Text Encoder
T_1 T_2 T_3 ... T_N
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

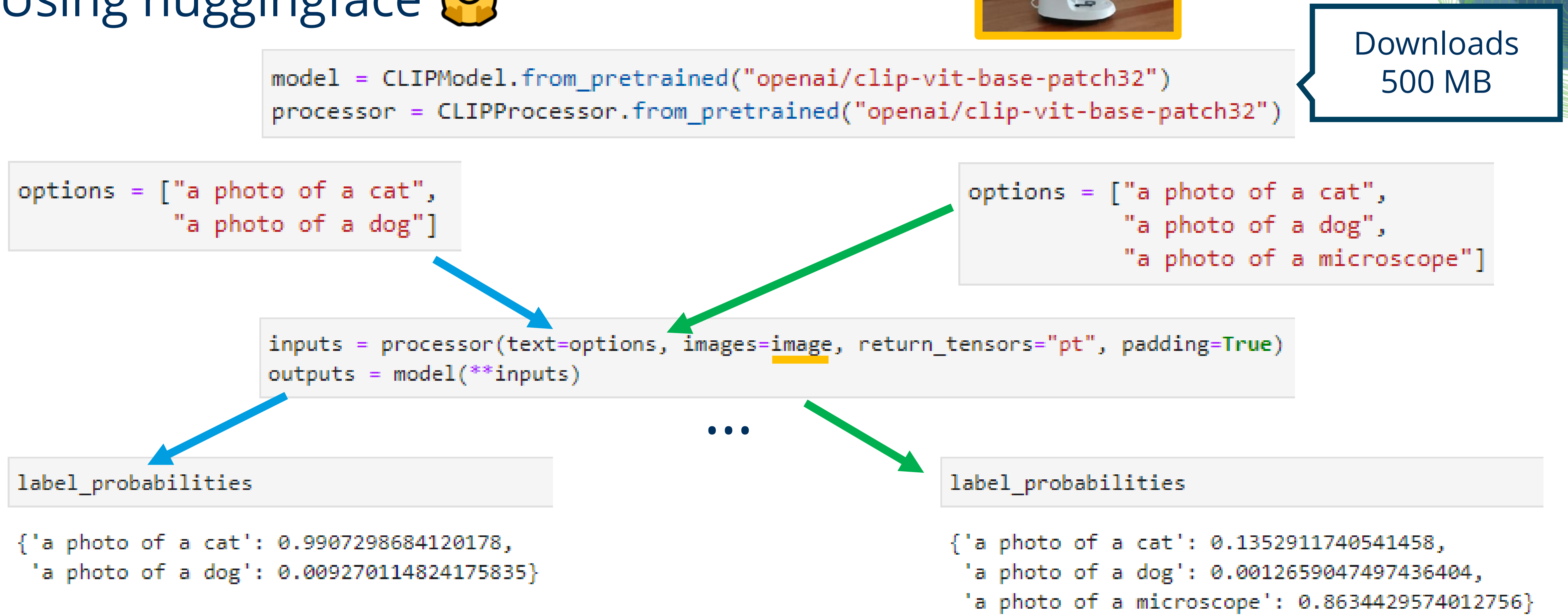
	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$	...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$	...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	⋱	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

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}
```

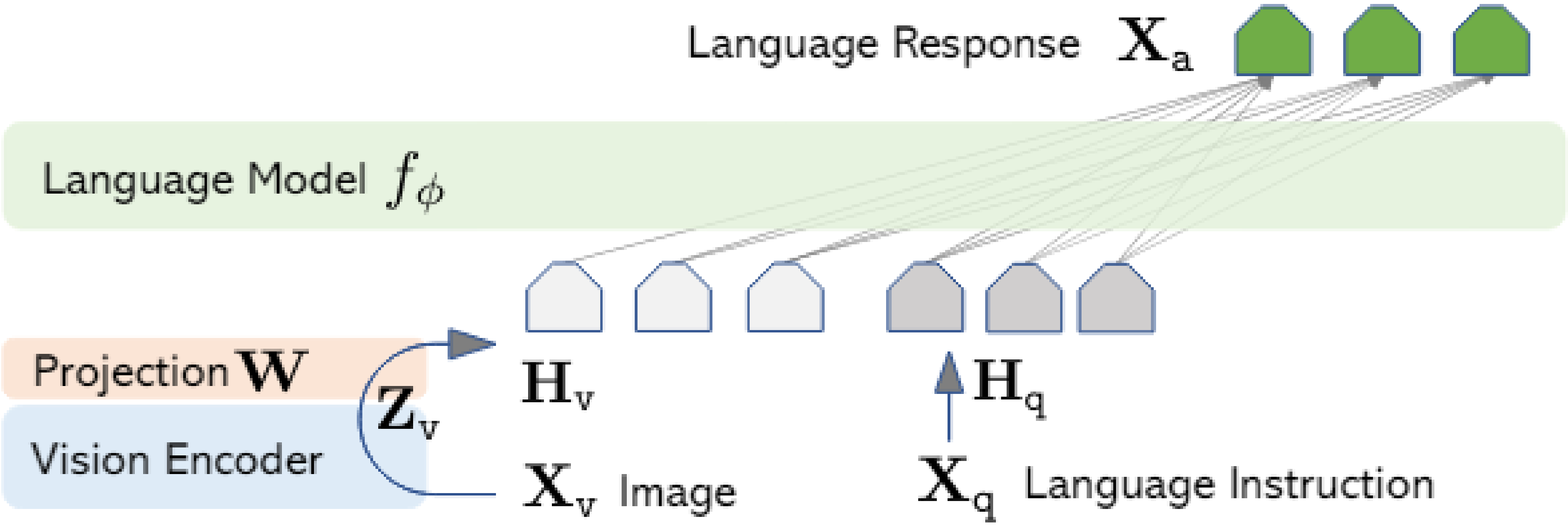
```
label_probabilities
```

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

Code example adapted from:
https://huggingface.co/docs/transformers/en/model_doc/clip

LLAVA

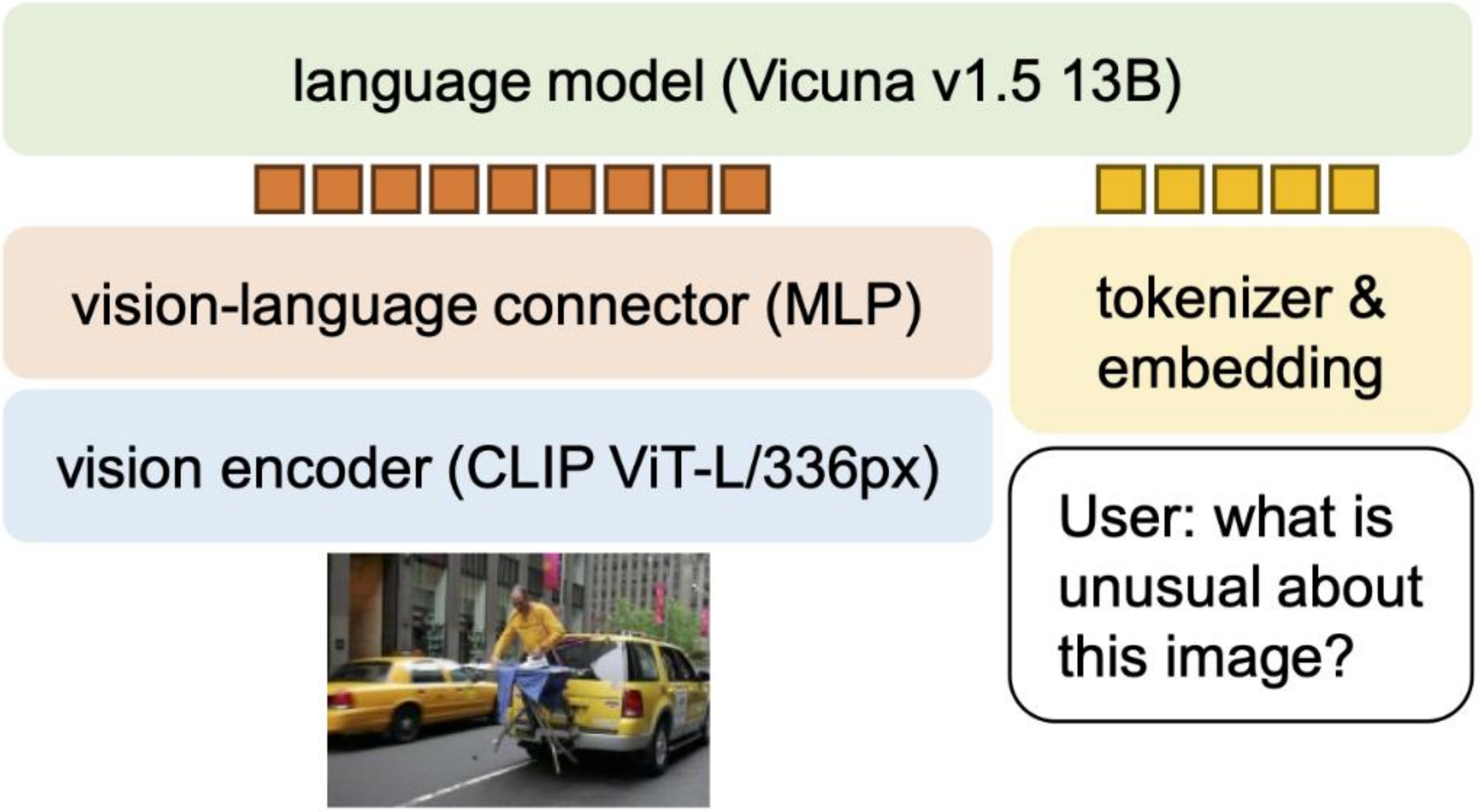
Large Language and Vision Assistant



Source: Hui et al 2023, licensed CC-BY 4.0
https://arxiv.org/abs/2304.08485

LLAVA 1.5

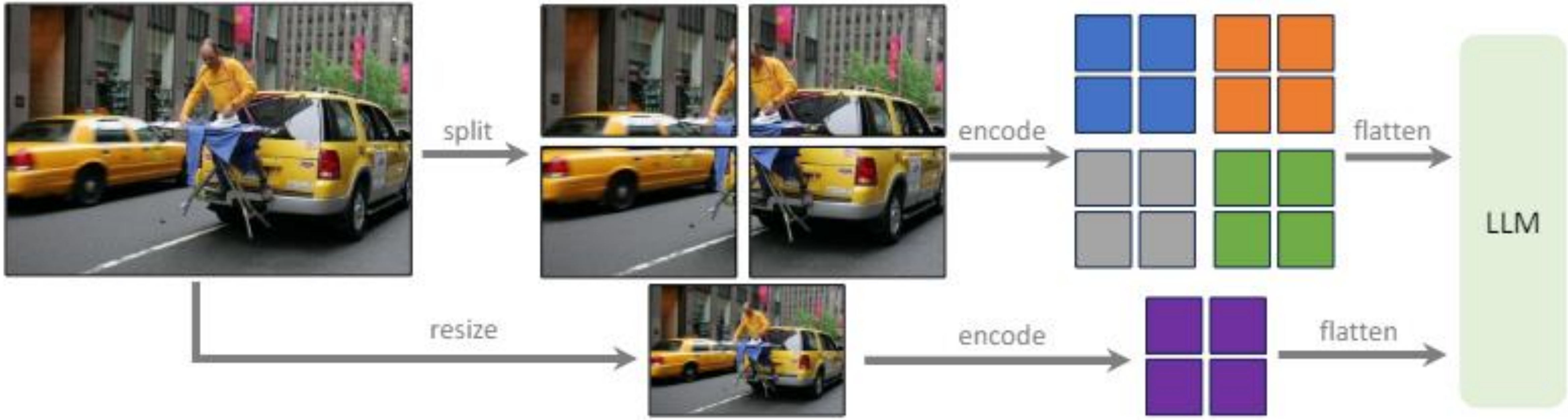
Combining LLAVA with CLIP


language model (Vicuna v1.5 13B)
vision-language connector (MLP)
tokenizer & embedding
vision encoder (CLIP ViT-L/336px)
User: what is unusual about this image?

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

LLAVA 1.5 HD

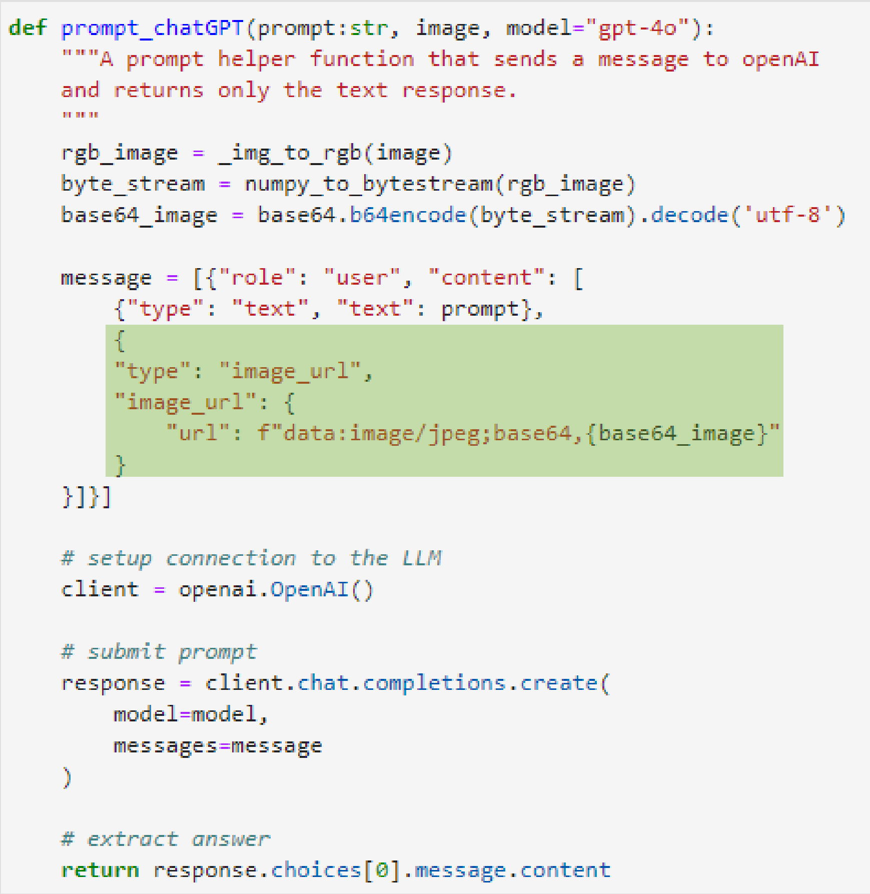
Giving the model multiple perspectives on the same scene



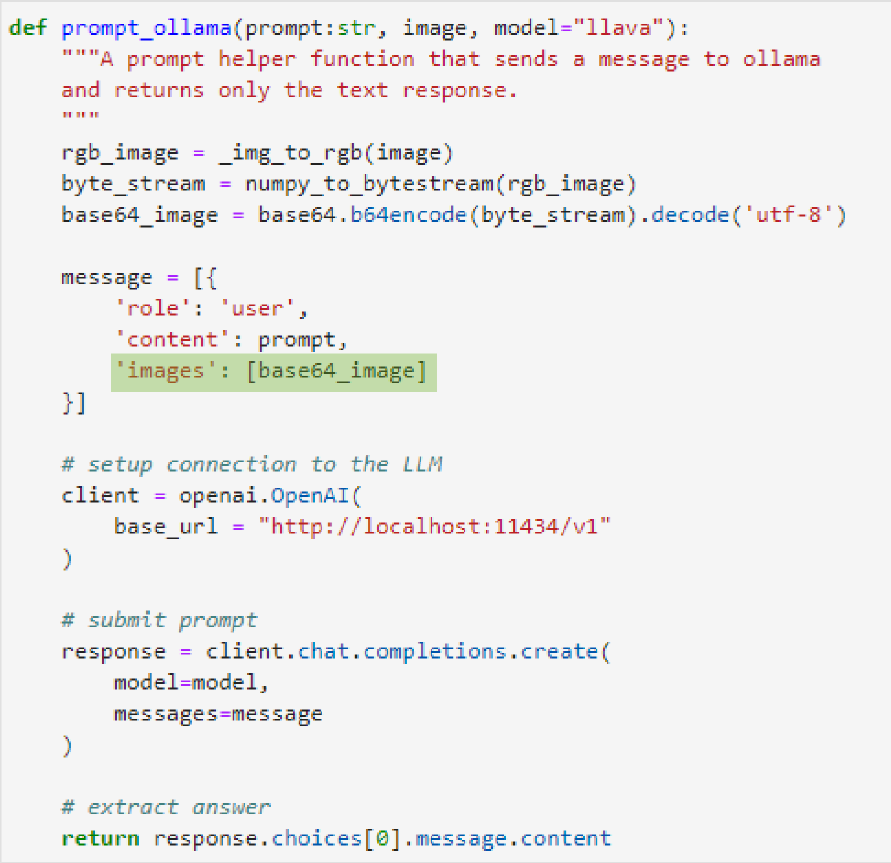
Source: Hui et al 2024, licensed CC-BY 4.0
https://arxiv.org/abs/2310.03744

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"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
```



```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
```

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Exercises

Robert Haase

Funded by

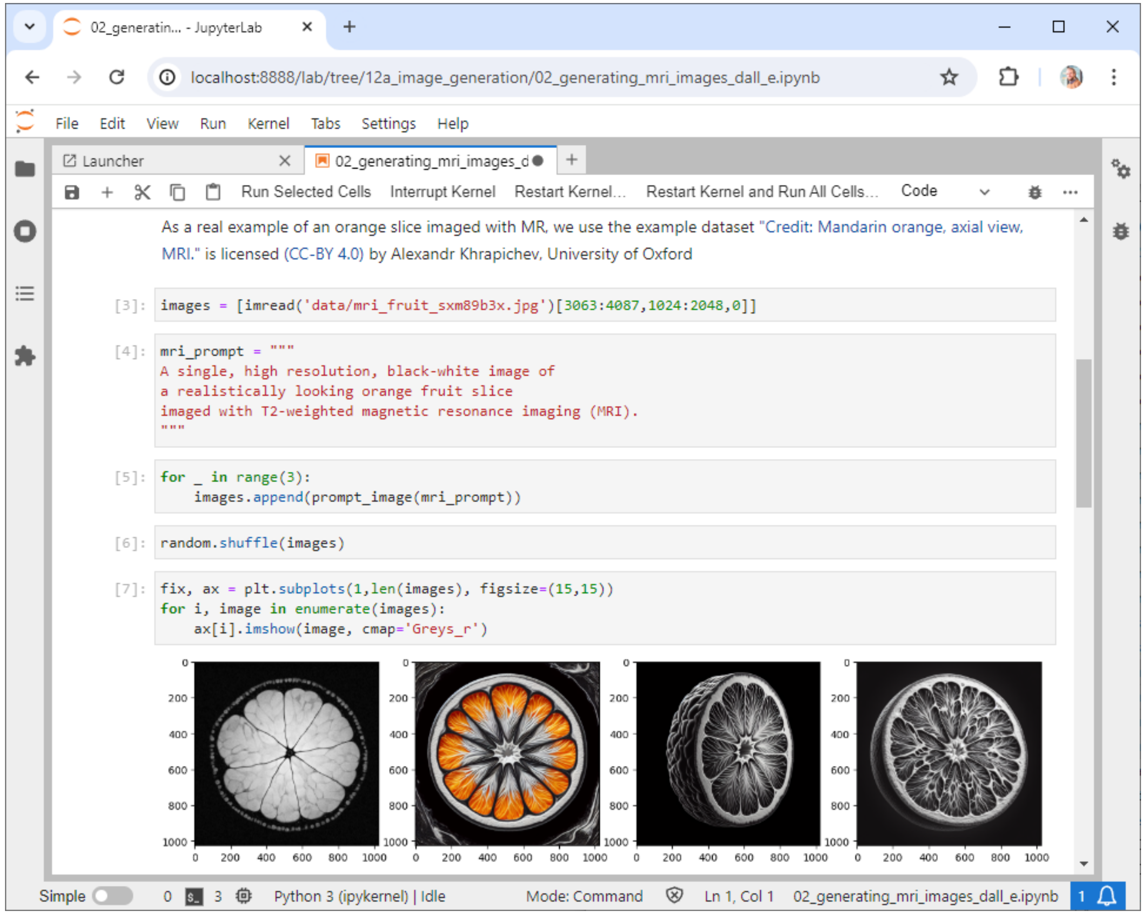
Bundesministerium für Bildung und Forschung

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


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
02_generating_mri_images_d
Run Selected Cells Interrupt Kernel Restart Kernel... Restart Kernel and Run All Cells... Code
0 200 400 600 800 1000
Simple 0 3 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 02_generating_mri_images_dall_e.ipynb 1

As a real example of an orange slice imaged with MR, we use the example dataset "Credit: Mandarin orange, axial view, MRI." is licensed (CC-BY 4.0) by Alexandr Khrapichev, University of Oxford

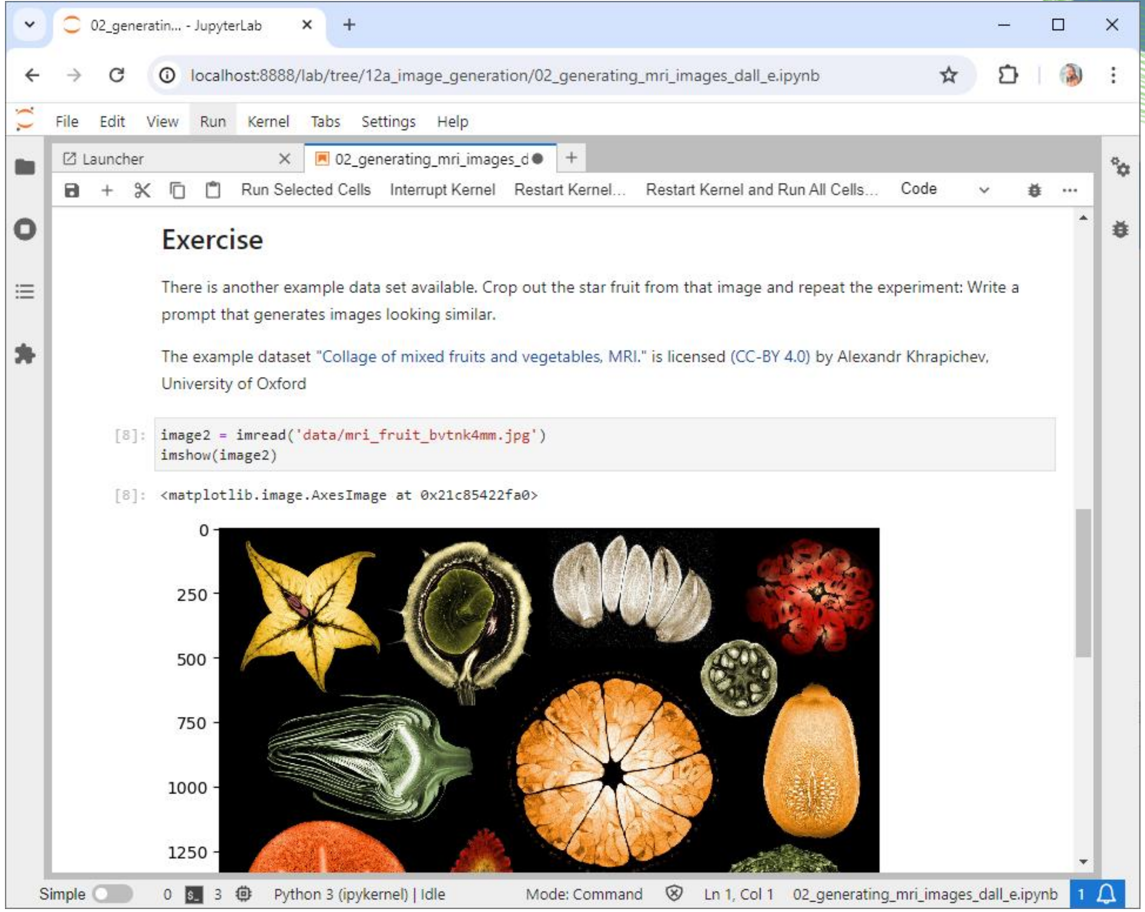
```
[3]: images = [imread('data/mri_fruit_sxm89b3x.jpg')[3063:4087,1024:2048,0]]
```

```
[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]: fix, 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
02_generating_mri_images_d
Run Selected Cells Interrupt Kernel Restart Kernel... Restart Kernel and Run All Cells... Code
0 250 500 750 1000 1250
Simple 0 3 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 02_generating_mri_images_dall_e.ipynb 1

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.

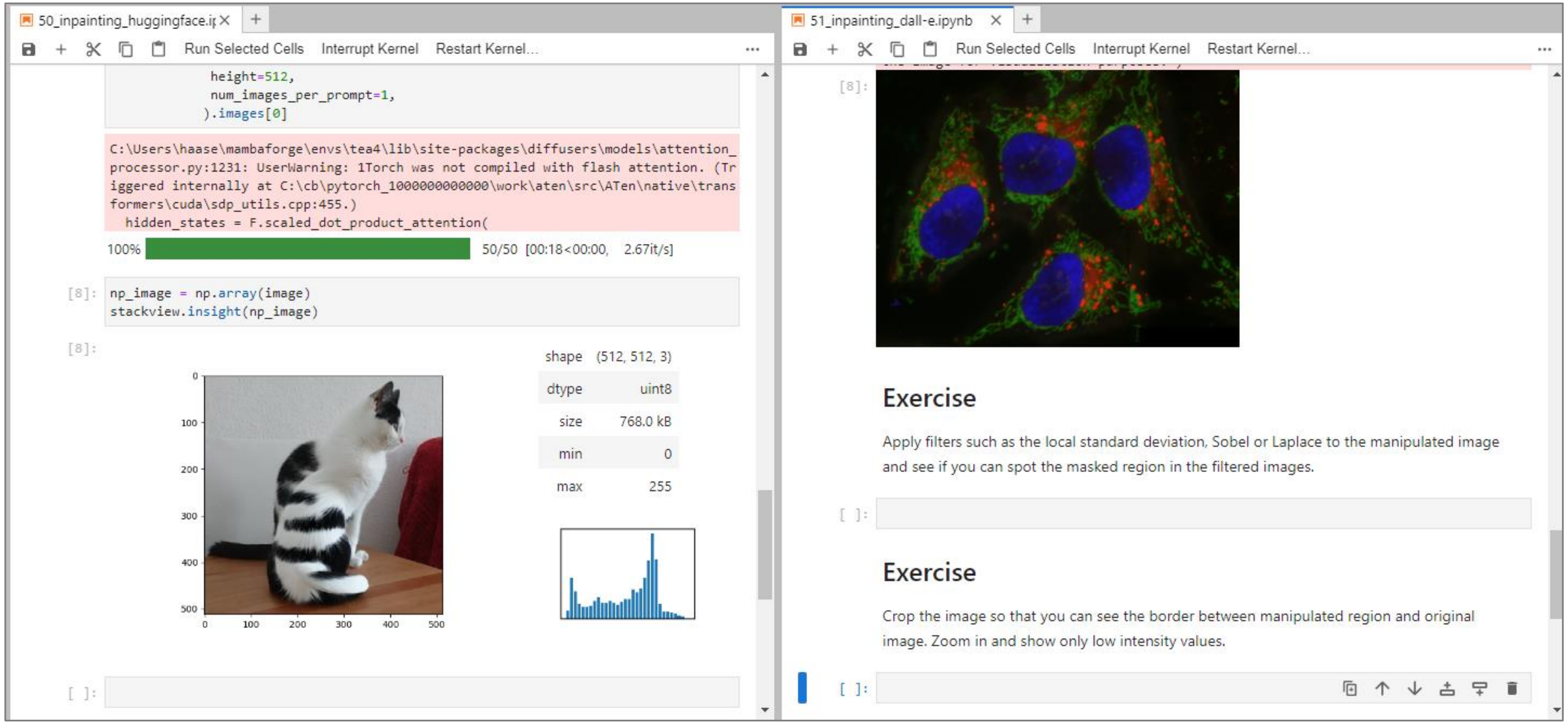
The example dataset "Collage of mixed fruits and vegetables, MRI." is licensed (CC-BY 4.0) by Alexandr Khrapichev, University of Oxford

```
[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



```
            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. (Tr
iggered internally at C:\cb\pytorch_1000000000000\work\aten\src\ATen\native\trans
formers\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

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.

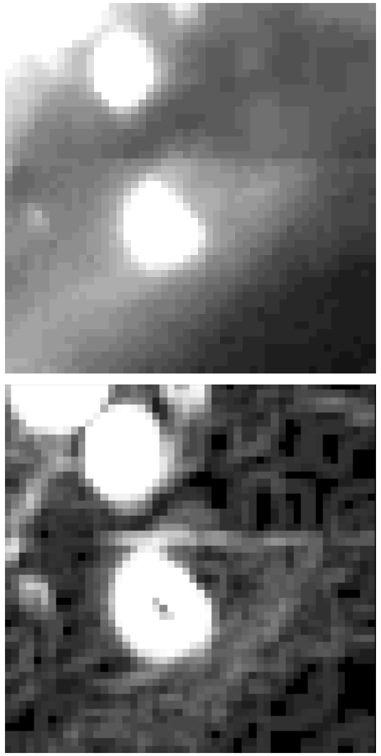
[ ]:

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.

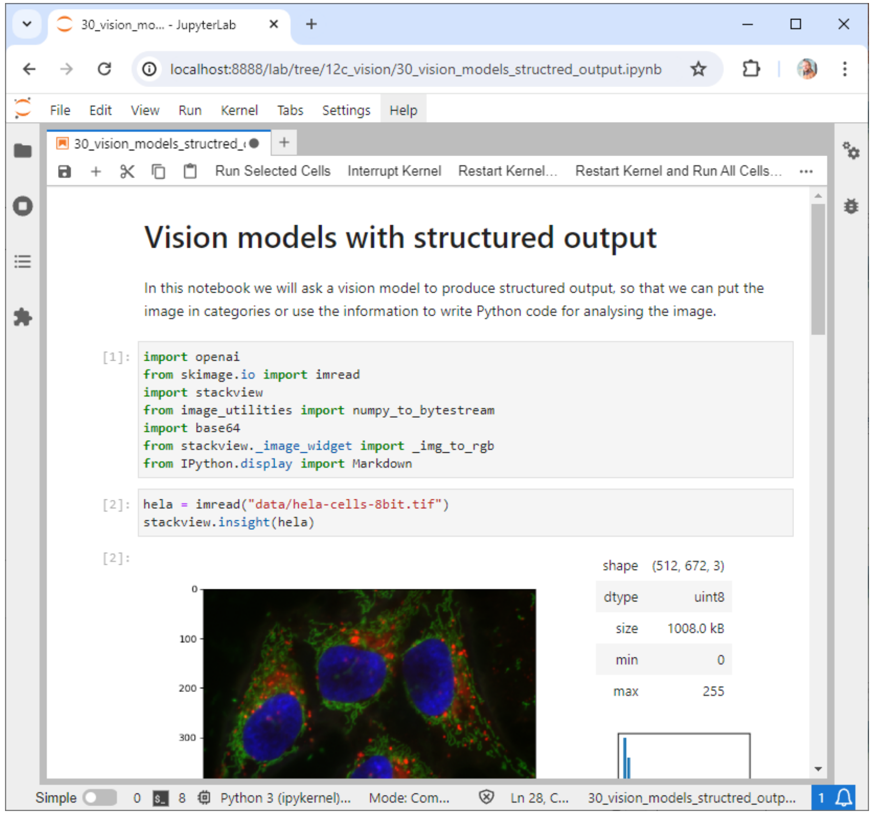
[ ]:

Hint:



Exercise: Vision

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



30_vision_mo... - JupyterLab
localhost:8888/lab/tree/12c_vision/30_vision_models_structred_output.ipynb
File
Edit
View
Run
Kernel
Tabs
Settings
Help
30_vision_models_structred_
Run Selected Cells
Interrupt Kernel
Restart Kernel...
Restart Kernel and Run All Cells...

Vision models with structured output

In this notebook we will ask a vision model to produce structured output, so that we can put the image in categories or use the information to write Python code for analysing the image.

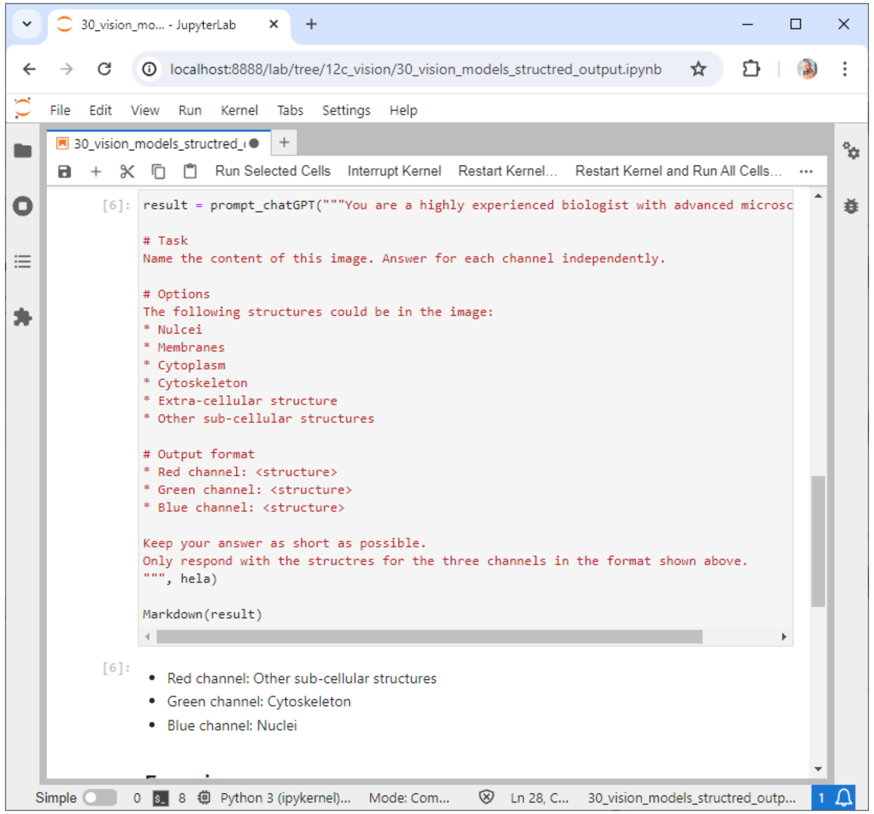
```
[1]: import openai
     from skimage.io import imread
     import stackview
     from image_utilities import numpy_to_bytestream
     import base64
     from stackview._image_widget import _img_to_rgb
     from IPython.display import Markdown
```

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

[2]:

shape	(512, 672, 3)
dtype	uint8
size	1008.0 kB
min	0
max	255

0
100
200
300
Simple
0
8
Python 3 (ipykernel)...
Mode: Com...
Ln 28, C...
30_vision_models_structred_outp...
1



30_vision_mo... - JupyterLab
localhost:8888/lab/tree/12c_vision/30_vision_models_structred_output.ipynb
File
Edit
View
Run
Kernel
Tabs
Settings
Help
30_vision_models_structred_
Run Selected Cells
Interrupt Kernel
Restart Kernel...
Restart Kernel and Run All Cells...

```
[6]: result = prompt_chatGPT("""You are a highly experienced biologist with advanced microsc

     # 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)
```

[6]:

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