WED-PM-248

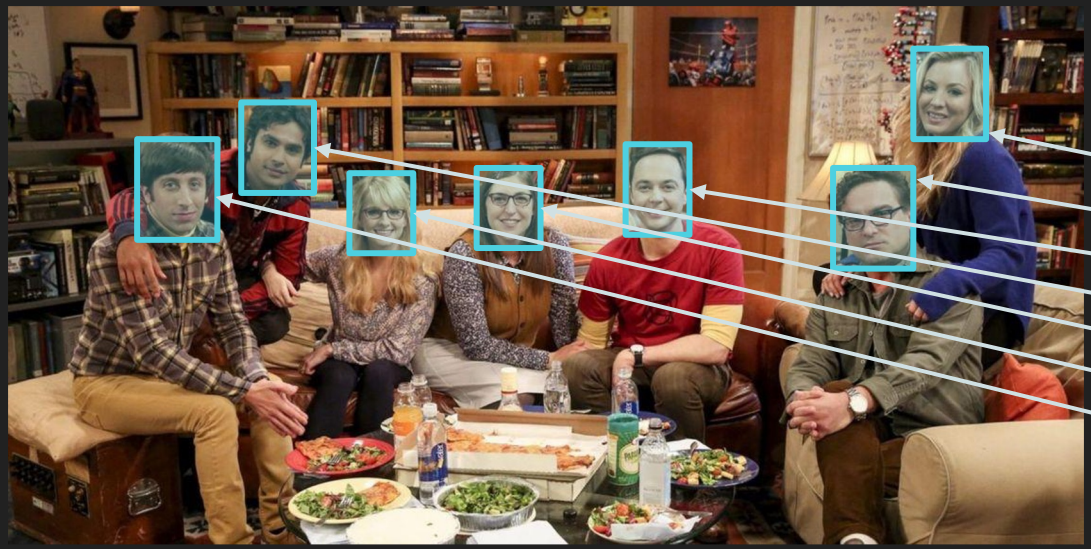
Visual Programming

Solving computer vision tasks by generating and executing code

Tanmay Gupta

Aniruddha Kembhavi

Tag the characters on the TV show Big Bang Theory



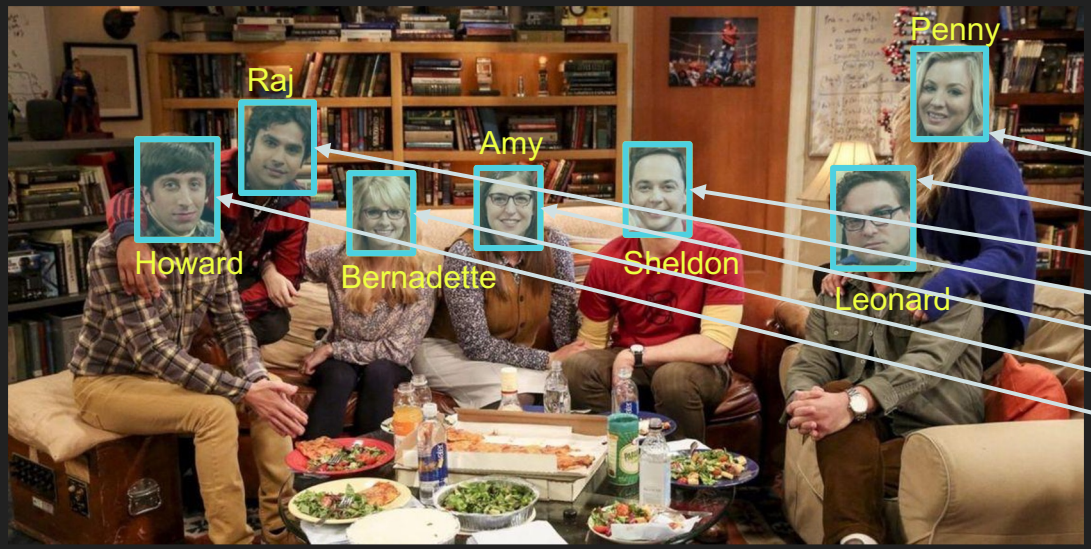
1. detect faces

2. query knowledge

Penny
Leonard
Sheldon
Raj
Amy
Bernadette
Howard

3. matches faces and names

Tag the characters on the TV show Big Bang Theory



1. detect faces

2. query knowledge

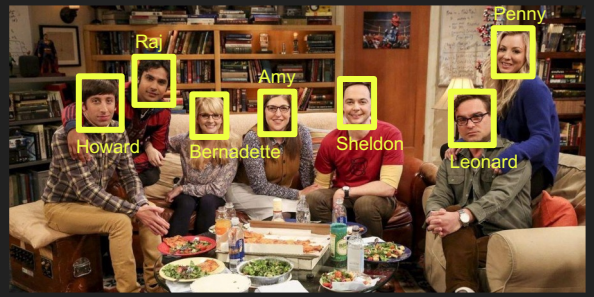
Penny
Leonard
Sheldon
Raj
Amy
Bernadette
Howard

3. matches faces and names

4. display labeled bboxes

Tag the characters on the TV show
Big Bang Theory

end-to-end
model



Face detection

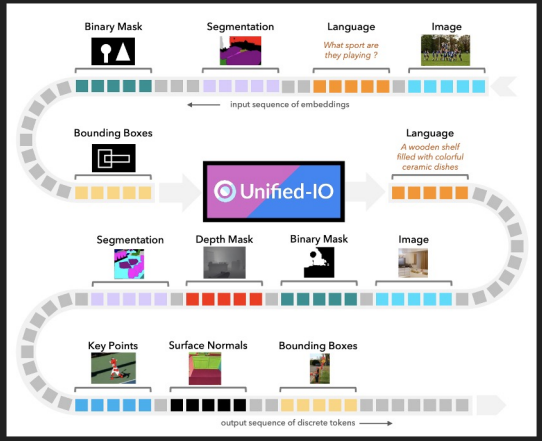
Knowledge Retrieval

Image-Text Matching

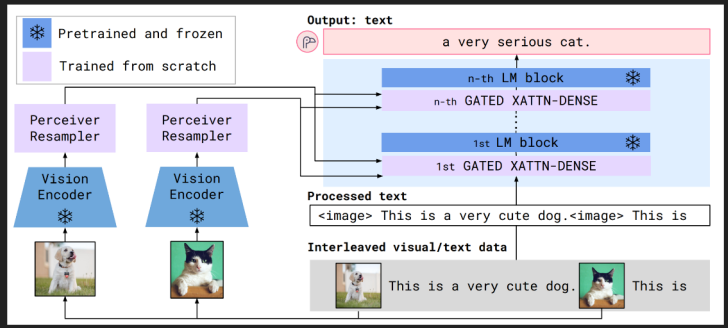
Image Processing & Plotting

Training Funnel

end-to-end model



Unified-IO



Flamingo

- Face detection
- Emoji Generation
- Knowledge Retrieval
- Segmentation
- Image-Text Matching
- Text-to-Image Generation
- Image Processing & Plotting

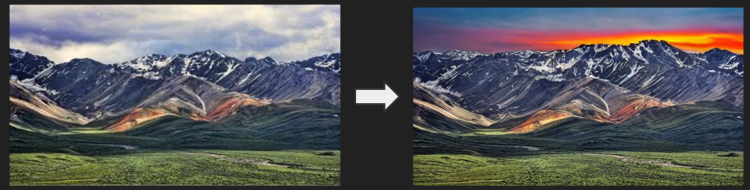
Training Funnel

end-to-end model

Tag the characters on the TV show Big Bang Theory

Hide Chandler and Joey with :ps

Replace the sky with sunrise behind mountains

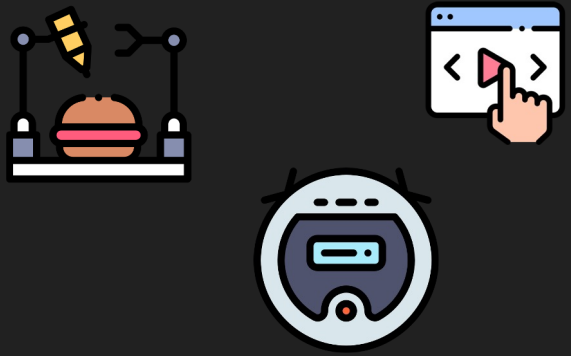


Face detection
Emoji Generation
Knowledge Retrieval
Segmentation
Image-Text Matching
Text-to-Image Generation
Image Processing & Plotting

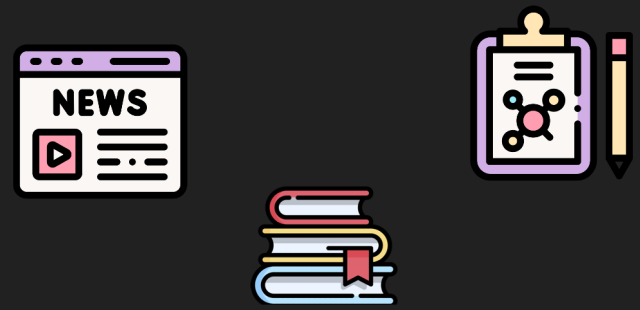
Training Funnel

end-to-end model

Complex multi-step visual tasks



Keep up with new knowledge and skills



Tag the characters on the TV show Big Bang Theory

An implementation for each step

tag_characters.py

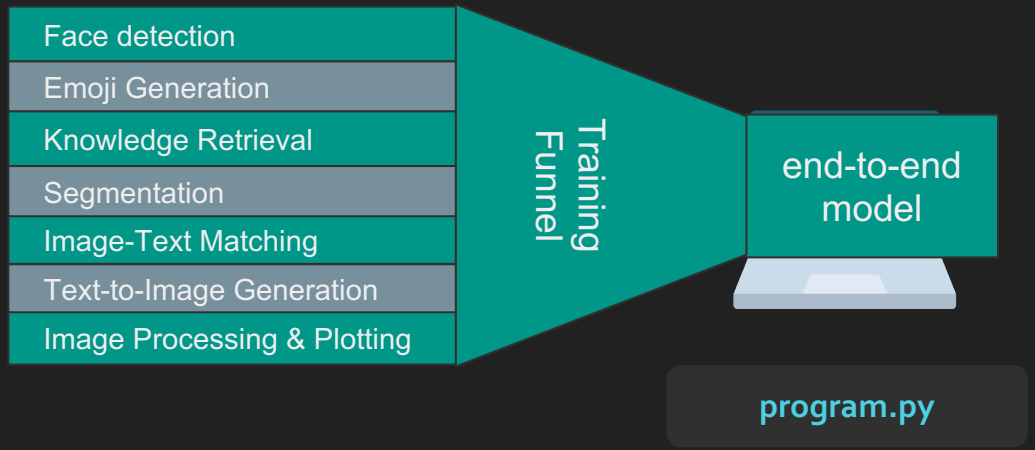
```python
from PIL import Image

from tagging_module import (
  detect_faces,
  get_character_names,
  match_faces_and_names,
  visualize
)
```

The program that calls the functions in sequence

```python
def tag_characters(input_image, tv_show_name):

  face_bboxes = detect_faces(input_image)

  # query an appropriate knowledge base or use GPT4
  list_of_characters = get_character_names(tv_show_name)

  # match detected faces to names using CLIP
  labeled_face_bboxes = match_faces_and_names(input_image,
                                              face_bboxes,
                                              list_of_characters)


  # create a new image with the bboxes and labels visualized
  output_image = visualize(input_image,labeled_face_bboxes)

  return output_image
```

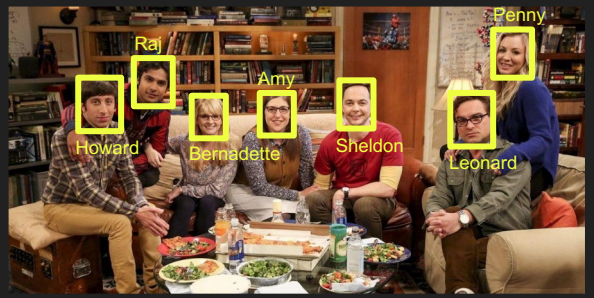
Execute the program on the given input

```python
if __name__=='__main__':
  in_img = Image.open('big_bang_theory_image.png')
  out_img = tag_characters(in_img, 'The Big Bang Theory')
  out_img.show()
```


Face detection
Emoji Generation
Knowledge Retrieval
Segmentation
Image-Text Matching
Text-to-Image Generation
Image Processing & Plotting
Training Funnel
end-to-end model
program.py

- Face detection
- Emoji Generation
- Knowledge Retrieval
- Segmentation
- Image-Text Matching
- Text-to-Image Generation
- Image Processing & Plotting

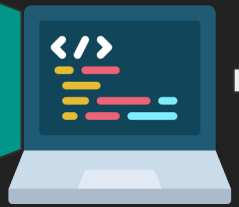
Invoke



tag_characters.py

- Face detection
- Emoji Generation
- Knowledge Retrieval
- Segmentation
- Image-Text Matching
- Text-to-Image Generation
- Image Processing & Plotting

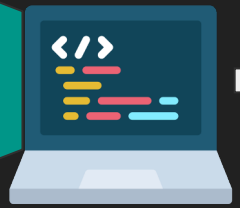
Invoke

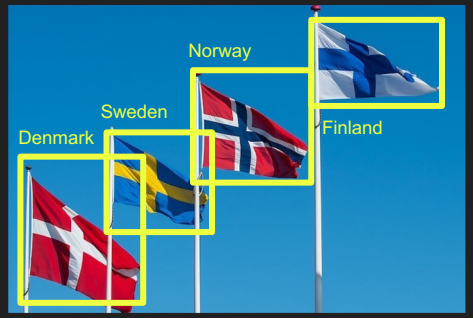


tag_female_characters.py

- Face detection
- Emoji Generation
- Knowledge Retrieval
- Segmentation
- Image-Text Matching
- Text-to-Image Generation
- Image Processing & Plotting

Invoke

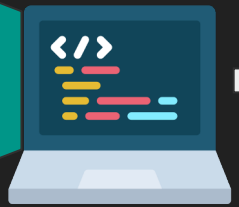


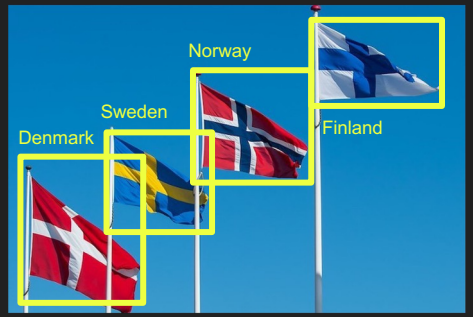


tag_scandinavian_flags.py

- Face detection
- Emoji Generation
- Knowledge Retrieval
- Segmentation
- Image-Text Matching
- Text-to-Image Generation
- Image Processing & Plotting

Invoke

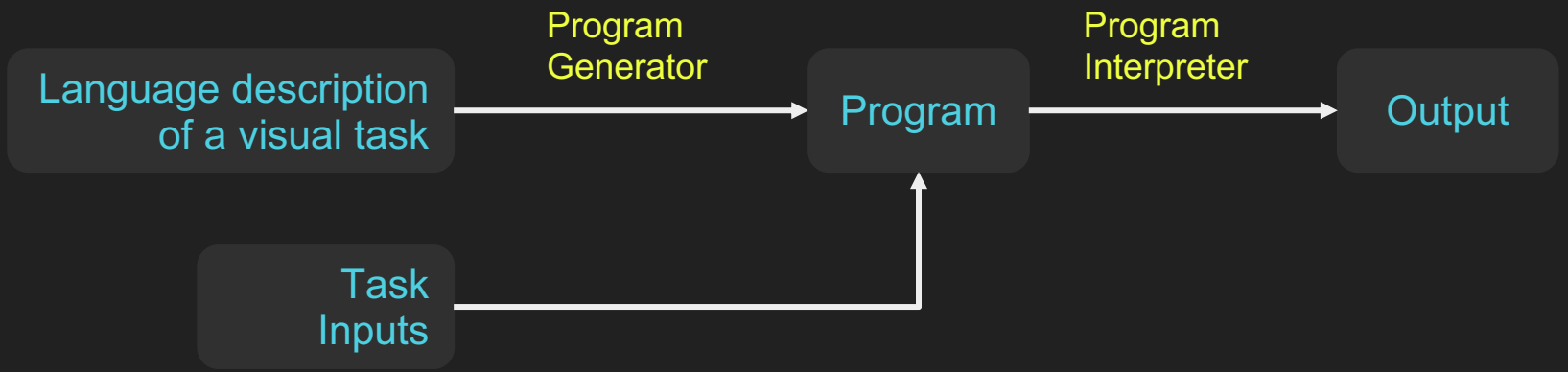




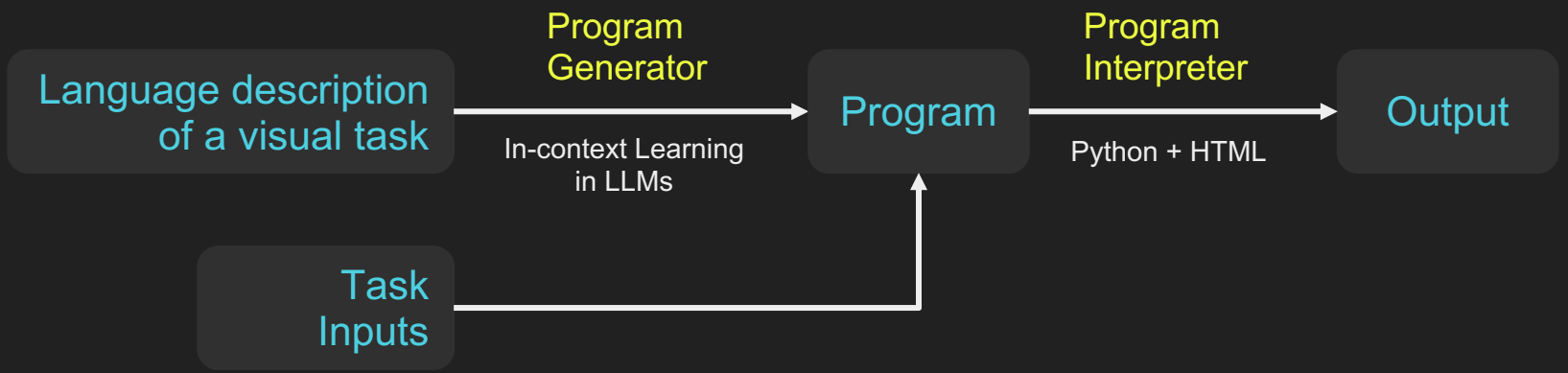
tag_scandinavian_flags.py

Tag these flags of Scandinavian countries

Visual Programming



*Vis*ual *Prog*ramming



VisProg Modules

Image Understanding	**Loc** OWL-ViT	**FaceDet** DSFD (pypi)	**Seg** MaskFormer	**Select** CLIP-ViT	**Classify** CLIP-ViT	**Vqa** ViLT
Image Manipulation	**Replace** Stable Diffusion	**ColorPop** PIL.convert() cv2.grabCut()	**BgBlur** PIL.GaussianBlur() cv2.grabCut()	**Tag** PIL.rectangle() PIL.text()	**Emoji** Augly (pypi)	
	Crop PIL.crop()	**CropLeft** PIL.crop()	**CropRight** PIL.crop()	**CropAbove** PIL.crop()	**CropBelow** PIL.crop()	
Knowledge Retrieval	**List** GPT3	Arithmetic & Logical	**Eval** eval()	**Count** len()	**Result** dict()	

- neural models
- image processing libraries
- any program

Replace the desert with lush green grass

```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

Replace the desert with lush green grass

IMAGE

```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

Replace the desert with lush green grass

```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

Replace the desert with lush green grass

```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

Replace the desert with lush green grass

```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

Replace the desert with lush green grass

```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

Replace the desert with lush green grass

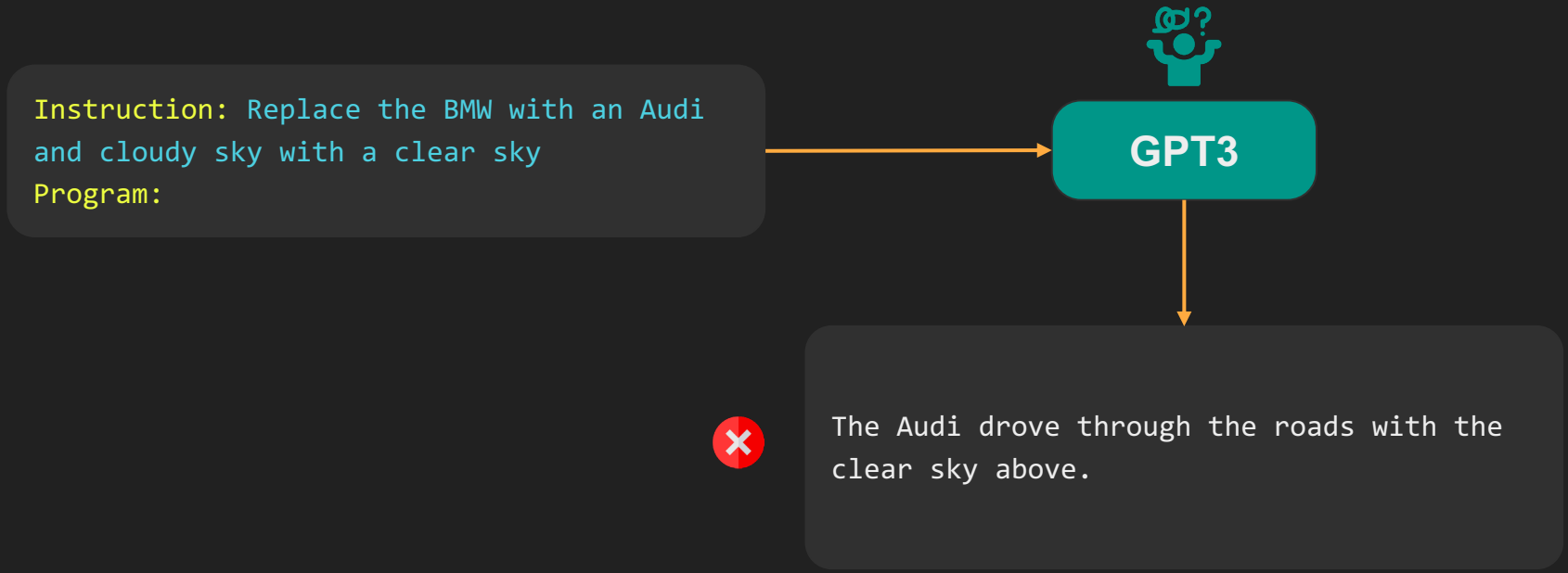
```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

Replace the desert with lush green grass

FINAL_RESULT

```
OBJ0 = Segment(image=IMAGE)
OBJ1 = Select(image=IMAGE, object=OBJ0, query='desert')
IMAGE0 = Replace(image=IMAGE, object=OBJ1, prompt='lust green grass')
FINAL_RESULT = Result(var=IMAGE0)
```

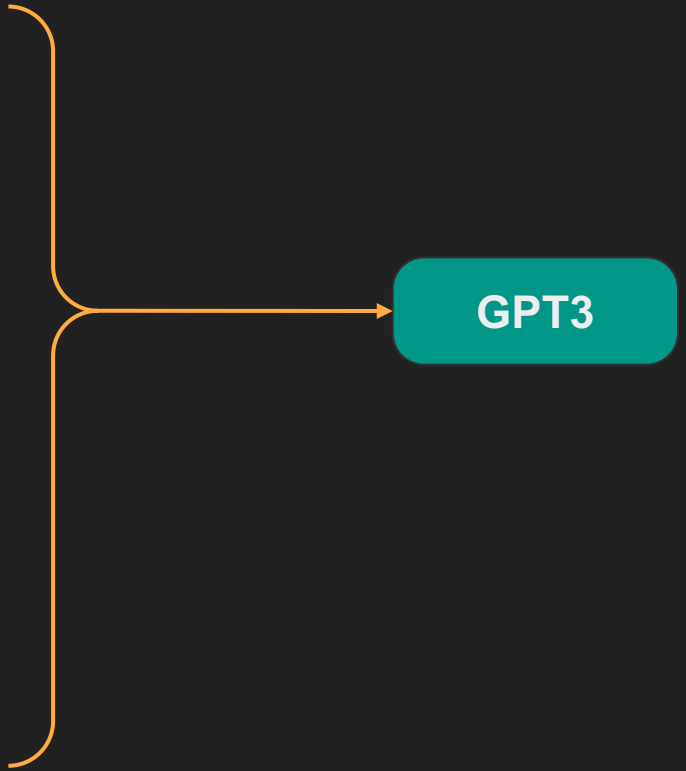
Program Generator



Program Generator

Information about the task and tools available for the task

```
Instruction: Replace the BMW with an Audi and cloudy sky with a clear sky
Program:
```



Program Generator

```
Instruction: Hide the face of Nicole Kidman with :p
Program:
OBJ0=Facedet(image=IMAGE)
OBJ1=Select(image=IMAGE, object=OBJ0, query='Nicole Kidman')
IMAGE0=Emoji(image=IMAGE, object=OBJ1, emoji='face_with_tongue')
RESULT=IMAGE0

Instruction: Create a color pop of the white Audi
Program:
OBJ0=Seg(image=IMAGE)
OBJ1=Select(image=IMAGE, object=OBJ0, query='white Audi')
IMAGE0=ColorPop(image=IMAGE, object=OBJ1)
RESULT=IMAGE0

Instruction: Replace the red car with a blue car
Program:
OBJ0=Seg(image=IMAGE)
OBJ1=Select(image=IMAGE, object=OBJ0, query='red car')
IMAGE0=Replace(image=IMAGE, object=OBJ1, prompt='blue car')
RESULT=IMAGE0
```

```
Instruction: Replace the BMW with an Audi and cloudy sky with a
clear sky
Program:
```

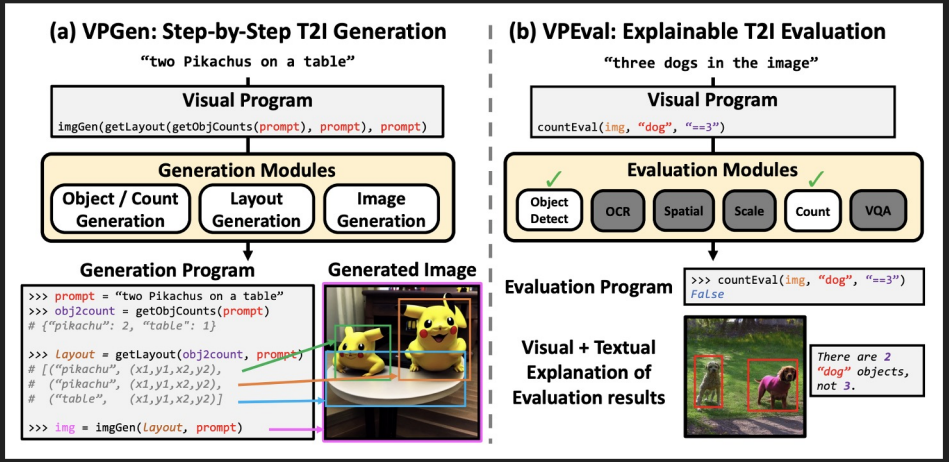

```
OBJ0=Seg(image=IMAGE)
OBJ1=Select(image=IMAGE, object=OBJ0, query='BMW')
IMAGE0=Replace(image=IMAGE, object=OBJ1, prompt='Audi')
OBJ1=Seg(image=IMAGE0)
OBJ2=Select(image=IMAGE0, object=OBJ1, query='cloudy sky')
IMAGE1=Replace(image=IMAGE0, object=OBJ2, prompt='clear sky')
RESULT=IMAGE1
```

Compositional Question Answering

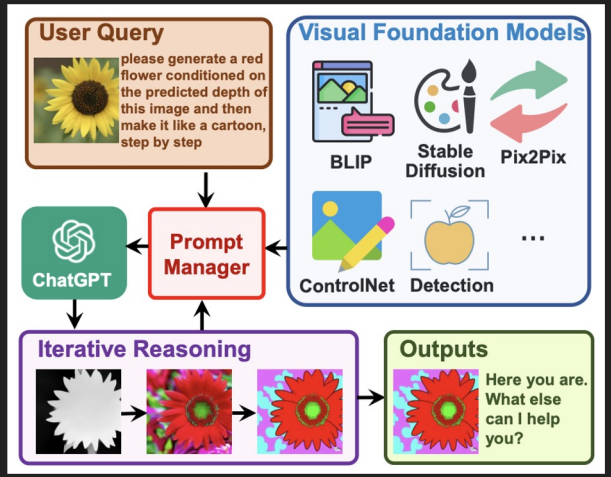
Are there both ties and glasses in the picture?

Prediction: no

	`IMAGE`
	`BOX0=Loc(image=IMAGE, object='ties')`
`1`	`ANSWER0=Count(bbox=BOX0)`
	`BOX1=Loc(image=IMAGE, object='glasses')`
`0`	`ANSWER1=Count(bbox=BOX1)`
`no`	`ANSWER2=Eval(expr="'yes' if {ANSWER0}>0 and {ANSWER1}>0 else 'no'")` `=Eval(expr="'yes' if 1>0 and 0>0 else 'no'")`

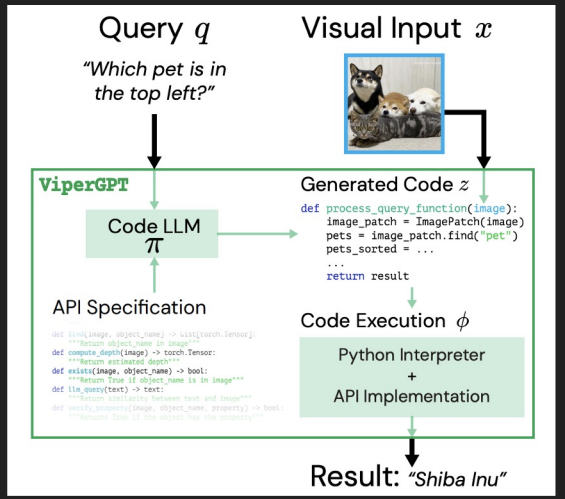


VP Gen & Eval
Cho et al. (May 2023)



Visual ChatGPT
Wu et al. (March 2023)

ViperGPT
Surís et al. (March 2023)



Thank You

https://prior.allenai.org/projects/visprog

https://github.com/allenai/visprog