# CV HW3: Vision Language Models

Shobhit Raj (2022482)

## Question 1: CLIP

This task involves exploring **CLIP (Contrastive Language-Image Pretraining)** and **CLIPS (recent enhancement of CLIP)** models by computing similarity scores between a given image and ten textual descriptions, and compare their performance.

### Part 1. Installation of CLIP Dependencies

I installed required libraries (`ftfy`, `regex`, `tqdm`) and the **official OpenAI CLIP package** using the GitHub repository. This ensured access to the `clip` module and its pretrained models. A screenshot of the code of installing dependencies is shown below.

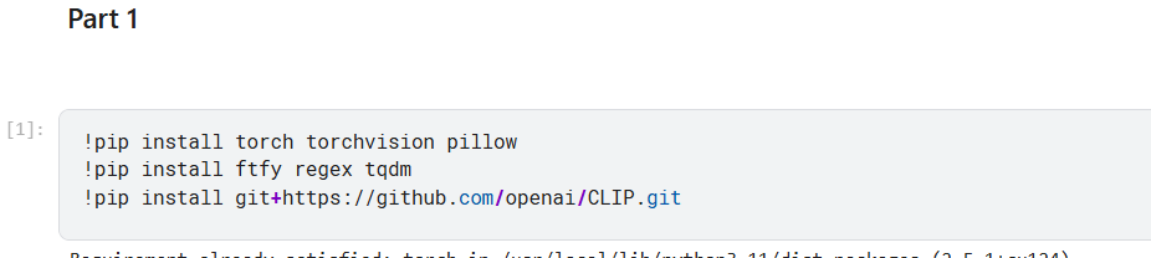



Figure 1: Installing dependencies for CLIP model

### Part 2. Loading Pretrained CLIP Model

The CLIP model `ViT-B/32` was loaded along with its image preprocessing pipeline. The screenshot of code of loading the model is shown below.


Figure 2: Loading the CLIP model

### Part 3. Computing Similarity Scores using CLIP

A sample image of a man holding a dog was processed. Ten textual descriptions were encoded using CLIP's tokenizer and passed through the model. Cosine similarities were calculated between the image and each text embedding. The results obtained are given below.

- A man holding a large black dog – 0.3005
- A dog being held by its owner – 0.2868
- A bookshelf filled with collection of encyclopedias, including World Book (A-Z) – 0.1656
- **A man posing with his dog at home – 0.3351**
- A living room with wooden floors, bookshelves and a man holding a dog – 0.2677
- A dog with floppy ears being carried – 0.2878
- A man in formal attire holding a large dog – 0.3014
- A room with wooden floors, bookshelves, and a closed door – 0.1871
- A man in a white shirt and gray pants holding a large dog – 0.3141
- A man with neatly groomed beard and short dark hair standing in a room – 0.2329

The highest scores were observed for relevant descriptions like:

**A man posing with his dog at home** → 0.3351
**A man in a white shirt and gray pants holding a large dog** → 0.3141

### Part 4. Installation of CLIPS Dependencies

I installed the `open_clip_torch package` to access the CLIPS model. A screenshot of the code of installing dependencies is shown below.


Figure 3: Installing dependencies for CLIPS model

## Part 5. Loading Pretrained CLIPS Model

The **CLIPS-Large-14-224 model** was successfully loaded using `open_clip` with its corresponding tokenizer and preprocessing function.

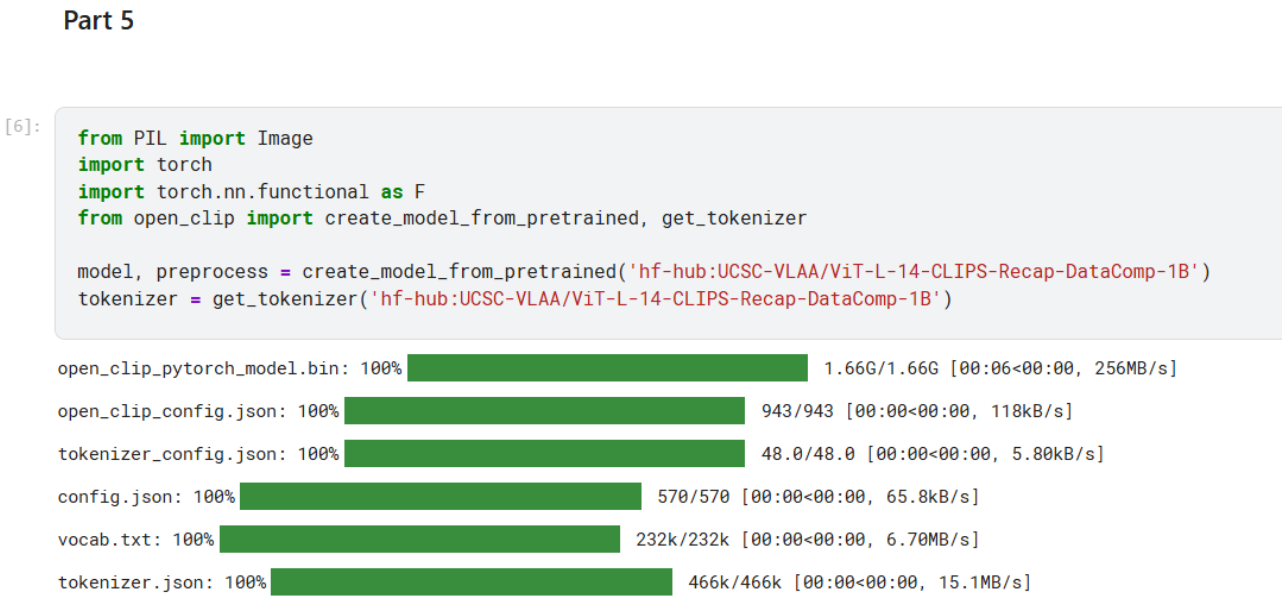



Figure 4: Loading the CLIPS model

## Part 6. Computing Similarity Scores using CLIPS

Using the same image and textual descriptions, similarity scores were computed using CLIPS. The results obtained are given below.

- A man holding a large black dog – 0.1773
- A dog being held by its owner – 0.1561
- A bookshelf filled with collection of encyclopedias, including World Book (A-Z) – 0.0137
- A man posing with his dog at home – 0.1913
- A living room with wooden floors, bookshelves and a man holding a dog – 0.1613
- A dog with floppy ears being carried – 0.1300
- **A man in formal attire holding a large dog – 0.1924**
- A room with wooden floors, bookshelves, and a closed door – 0.0366
- **A man in a white shirt and gray pants holding a large dog – 0.2103**
- A man with neatly groomed beard and short dark hair standing in a room – 0.1018

Although the absolute scores were generally lower, the most relevant captions still had relatively higher similarity:

**A man in a white shirt and gray pants holding a large dog** → 0.2103
**A man in formal attire holding a large dog** → 0.1924

**Part 7. Comparative Analysis of CLIP vs CLIPS**

Despite CLIPS being a newer model, in this particular example CLIP produced higher and more distinct similarity scores, making it easier to identify the most relevant captions. CLIP showed a clear separation between relevant and irrelevant captions (ranging from 0.1656 to 0.3351), while CLIPS had lower overall scores and smaller margins between good and bad matches. This could be due to CLIPS being trained on a broader range of synthetic data, making it more general but less confident in specific, context-limited scenarios as CLIPS shows potential for filtering irrelevant captions, such as "A bookshelf filled with encyclopedias", receive much lower scores (0.0137) in CLIPS. CLIP's embeddings appear more confident and discriminative for this specific image.

## References


- **CLIP Model Usage Guide (Official Model Card):** [OpenAI GitHub - CLIP]
- **Official CLIP Interactive Notebook:** [Interacting with CLIP Notebook]
- **CLIPS Model Usage Guide (Model Card):** [UCSC-VLAA GitHub - CLIPS]

## Question 2: Visual question answering

This task involves exploring **Visual Question Answering (VQA)** using the **BLIP model (Bootstrapping Language-Image Pre-training)**. I used the `Salesforce/blip-vqa-capfilt-large` checkpoint from Hugging Face to perform VQA on the sample image.

### Part 1. Setup and Model Loading

I installed required dependencies and load the pre-trained BLIP model and its associated processor using the `transformers` library. The model used is **Salesforce/blip-vqa-capfilt-large**, which is specifically fine-tuned for visual question answering. The screenshot of code of installing dependencies & loading the model is shown below.

Part 1

[1]:
```
!pip install transformers torch torchvision pillow -q
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.4 MB/s eta 0:00:000:00:0100:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.1 MB/s eta 0:00:000:00:0100:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 2.6 MB/s eta 0:00:000:00:0100:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 5.5 MB/s eta 0:00:000:00:0100:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 13.0 MB/s eta 0:00:00:00:0100:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 7.8 MB/s eta 0:00:000:00:0100:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 64.4 MB/s eta 0:00:00:00:0100:01
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. Thi
pylibcugraph-cu12 24.12.0 requires pylibraft-cu12==24.12.*, but you have pylibraft-cu12 25.2.0 which is incomp
pylibcugraph-cu12 24.12.0 requires rmm-cu12==24.12.*, but you have rmm-cu12 25.2.0 which is incompatible.
```

[2]:
```
import torch
from PIL import Image
from transformers import BlipProcessor, BlipForQuestionAnswering

device = "cuda" if torch.cuda.is_available() else "cpu"

processor = BlipProcessor.from_pretrained("Salesforce/blip-vqa-capfilt-large")
model = BlipForQuestionAnswering.from_pretrained("Salesforce/blip-vqa-capfilt-large").to(device)
```

Figure 5: Installing dependencies & loading BLIP model

### Part 2. Answering Question - "Where is the dog present in the image?"

I provided the image and question to the model, which returns the answer:

**Answer:** *in man's arms*

This accurately describes the position of the dog in the image, as visually inferred.

### Part 3. Answering Question - "Where is the man present in the image?"

Similarly, I asked the model about the man's location. The answer is:

**Answer:** *living room*

This is also correct, as the man is visibly standing in what appears to be a living room.

**Part 4. Analysis**

The generated answers are accurate, concise, and contextually appropriate. The BLIP model effectively identifies key spatial and semantic relationships in the image:

- The dog's location is described with respect to the man ("in man's arms"), showing relational understanding.
- The man's environment is recognized correctly as a "living room", showing scene-level comprehension.

While the answers are short, they are highly relevant and semantically meaningful, indicating strong performance on this task.

## References


- **BLIP VQA Model Usage Guide (Official Model Card):** [Salesforce Huggingface - BLIP VQA]

# Question 3: BLIP vs CLIP

This task involves comprehensive comparison between **CLIP & CLIPS** used to evaluating image captions generated by **BLIP image captioning** model.

## Part 1. Setup and Model Loading

I installed required dependencies and load the pre-trained BLIP model and its associated processor using the `transformers` library. I used the BLIP model **(Salesforce/blip-image-captioning-large)** which is specifically fine-tuned for image captioning. The screenshot of code of installing dependencies & loading the model is shown below.

```
[1]: !pip install transformers torch torchvision pillow -q
     !pip install ftfy regex tqdm -q
     !pip install git+https://github.com/openai/CLIP.git -q
     !pip install open_clip_torch -q

     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.6 MB/s eta 0:00:000:00:0100:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.6 MB/s eta 0:00:000:00:0100:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 8.1 MB/s eta 0:00:000:00:0100:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 16.5 MB/s eta 0:00:00:00:0100:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 13.6 MB/s eta 0:00:00:00:0100:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 8.3 MB/s eta 0:00:000:00:0100:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 84.5 MB/s eta 0:00:00:00:0100:01
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour
conflicts.
pylibcugraph-cu12 24.12.0 requires pylibraft-cu12==24.12.*, but you have pylibraft-cu12 25.2.0 which is incompatible.
pylibcugraph-cu12 24.12.0 requires rmm-cu12==24.12.*, but you have rmm-cu12 25.2.0 which is incompatible.
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.8/44.8 kB 263.3 kB/s eta 0:00:000:00:01
  Preparing metadata (setup.py) ... done
  Building wheel for clip (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 21.4 MB/s eta 0:00:0000:0100:01

  + Code   + Markdown

[2]: import torch
     import torch.nn.functional as F
     from PIL import Image
     from transformers import BlipProcessor, BlipForConditionalGeneration
     import os
     import matplotlib.pyplot as plt
     import clip
     from open_clip import create_model_from_pretrained, get_tokenizer

     device = "cuda" if torch.cuda.is_available() else "cpu"

     processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
     model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large").to(device)
```

Figure 6: Installing dependencies & loading BLIP model

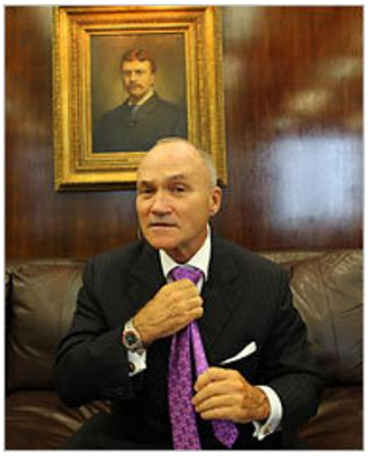
## Part 2. BLIP for Captioning

I used the BLIP model **(Salesforce/blip-image-captioning-large)** to generate captions for 10 diverse images. The captions generated are shown below.

Caption: arafed dog running in a field of grass with a sky background

Caption: there is a small dog standing on a ledge near a pool

Caption: they are riding on a bike in the rain in the street



Caption: arafed man in a suit and tie sitting on a couch

Caption: there are four children sitting on a towel by a pool


Caption: araffe dog running on a leash at a dog show

Caption: there is a bird that is sitting on a plant with green leaves

Caption: ducks are standing in the water and one is drinking


Caption: two cups of coffee being poured into a coffee machine

Caption: there is a brown butterfly sitting on a leaf in the grass

The captions describe the main subjects, context, and some background information reasonable well. However, I noticed some decoding issues like the word "arafed" or "araffe", likely stemming from either noise in the pretraining data or token decoding artifacts.

### Part 3. CLIP Similarity Evaluation

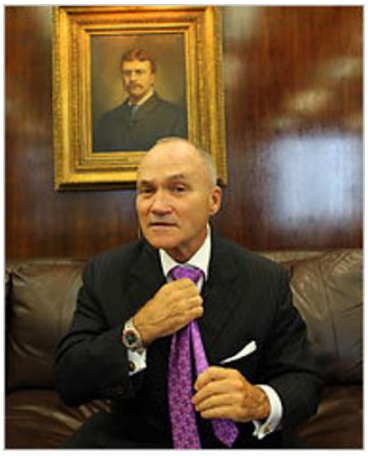
I used **OpenAI CLIP (ViT-B/32)** to evaluate how semantically aligned the BLIP-generated captions are with the images. The resultant cosine similarities ranged from $\tilde{0}.27$ to 0.34. The results are shown below.

Higher scores were found for images with clearly described scenes (Eg., "children sitting by a pool", 0.3367) Lower scores occurred for more abstract or cluttered images (Eg., bird on a plant, 0.2658)

**Caption:** arafed dog running in a field of grass with a sky background
**CLIP Score:** 0.3078

**Caption:** there is a small dog standing on a ledge near a pool
**CLIP Score:** 0.3024

**Caption:** they are riding on a bike in the rain in the street
**CLIP Score:** 0.3242



**Caption:** arafed man in a suit and tie sitting on a couch
**CLIP Score:** 0.2868

**Caption:** there are four children sitting on a towel by a pool
**CLIP Score:** 0.3367


**Caption:** araffe dog running on a leash at a dog show
**CLIP Score:** 0.3321

**Caption:** there is a bird that is sitting on a plant with green leaves
**CLIP Score:** 0.2658

**Caption:** ducks are standing in the water and one is drinking
**CLIP Score:** 0.3054


**Caption:** two cups of coffee being poured into a coffee machine
**CLIP Score:** 0.3021

**Caption:** there is a brown butterfly sitting on a leaf in the grass
**CLIP Score:** 0.2904

## Part 4. CLIPS Similarity Evaluation

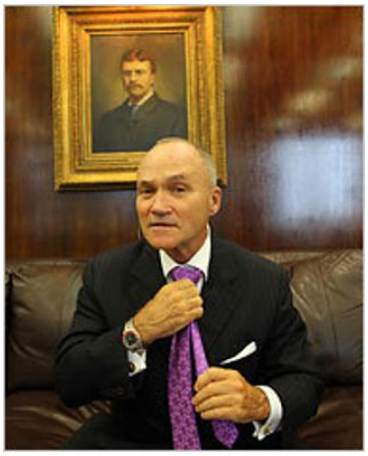
I used **CLIPS-Large-14-224** model to evaluate how semantically aligned the BLIP-generated captions are with the images. The resultant cosine similarities ranged lower from $\tilde{0}.09$ to 0.21. The results are shown below. CLIPS is more sensitive and nuanced, reflected in the lower raw scores but better separation. Despite being lower in magnitude, the relative ranking of caption quality aligns well with CLIP.

"four children sitting on a towel by a pool" scored **highest** on both CLIP (0.3367) and CLIPS (0.2089)
"bird on a plant" scored **lowest** on both

**Caption:** arafed dog running in a field of grass with a sky background
**CLIPS Score:** 0.1627

**Caption:** there is a small dog standing on a ledge near a pool
**CLIPS Score:** 0.1485

**Caption:** they are riding on a bike in the rain in the street
**CLIPS Score:** 0.1798



**Caption:** arafed man in a suit and tie sitting on a couch
**CLIPS Score:** 0.1295

**Caption:** there are four children sitting on a towel by a pool
**CLIPS Score:** 0.2089


**Caption:** araffe dog running on a leash at a dog show
**CLIPS Score:** 0.1776

**Caption:** there is a bird that is sitting on a plant with green leaves
**CLIPS Score:** 0.0926

**Caption:** ducks are standing in the water and one is drinking
**CLIPS Score:** 0.1705


**Caption:** two cups of coffee being poured into a coffee machine
**CLIPS Score:** 0.1521

**Caption:** there is a brown butterfly sitting on a leaf in the grass
**CLIPS Score:** 0.1237

**Part 5. Evaluation Metrics**

- **Cosine Similarity**: Measures angle between normalized image and text embeddings. Useful when focusing on semantic alignment between images and text, and ranking multiple captions for a single image or vice versa by relevance.
- **BLEU/ROUGE/METEOR**: Traditional text-overlap based metrics that compare generated captions to ground truth reference captions. Useful when comparing generated captions to human-written captions (ground truth).
- **CLIPScore**: Uses CLIP to assess semantic similarity between generated caption and image. Useful when human judgment is not feasible, better for open-ended generation tasks.
- **Human Evaluation**: It is the most important metric in generation tasks. Useful when automated metrics are insufficient for capturing qualitative aspects.

## References


- **BLIP Model Usage Guide (Official Model Card):** [Salesforce Huggingface - BLIP Image Captioning]
- **Official BLIP Demo Notebook:** [Salesforce GitHub - BLIP Demo Notebook]
- **Overview of Evaluation Metrics in Vision-Language Models:** [Medium Article - Performance Metrics in Evaluating Stable Diffusion Models]

## Question 4: Referring Image Segmentation (RIS)

This task involves **RIS models like LAVT** jointly reason over language and vision to handle contextual and relational cues

### Part 1. Setup and Model Loading

I cloned the **LAVT-RIS GitHub** and installed its Python dependencies (mmcv, mmsegmentation, timm, etc.) via `requirements.txt`. I downloaded the **RefCOCO pre-trained checkpoint (refcoco.pth)** and loaded it into the LAVT segmentation backbone and accompanying BERT language encoder. The screenshot of code of installing dependencies & loading the model is shown below.

```
!git clone https://github.com/yz93/LAVT-RIS.git
!pip install h5py
!pip install timm
!pip install mmcv==1.3.12
!pip install mmsegmentation==0.17.0
!pip install -r /kaggle/working/LAVT-RIS/requirements.txt
```

```
!gdown 13D-OeEOijV8KTC3BkFP-gOJymc6DLwVT
```

```
weights = "refcoco.pth"

import os
import sys
import torch
import numpy as np
from PIL import Image
import torchvision.transforms as T
import torch.nn.functional as F
import matplotlib.pyplot as plt

sys.path.append("/kaggle/working/LAVT-RIS")

from lib import segmentation
from bert.tokenization_bert import BertTokenizer
from bert.modeling_bert import BertModel
from scipy.ndimage.morphology import binary_dilation

device = "cuda:0" if torch.cuda.is_available() else "cpu"

class Args:
    swin_type="base"
    window12=True
    mha=""
    fusion_drop=0.0

model_segm = segmentation.__dict__['lavt'](pretrained="", args=Args())
bert_model = BertModel.from_pretrained("bert-base-uncased")
bert_model.pooler = None

ckpt = torch.load(weights, map_location="cpu")
bert_model.load_state_dict(ckpt['bert_model'])
model_segm.load_state_dict(ckpt['model'])
```

Figure 12: Installing dependencies & loading LAVT model

## Part 2. Segmentation with Provided References

For each sample image in `/samples/`, we read its paired reference text from `reference.txt`, tokenized it with BERT, and ran a forward pass through LAVT. The output masks were upsampled to the original image size and overlaid as translucent red regions, demonstrating accurate segmentation of the described object in most cases. The results are shown below.

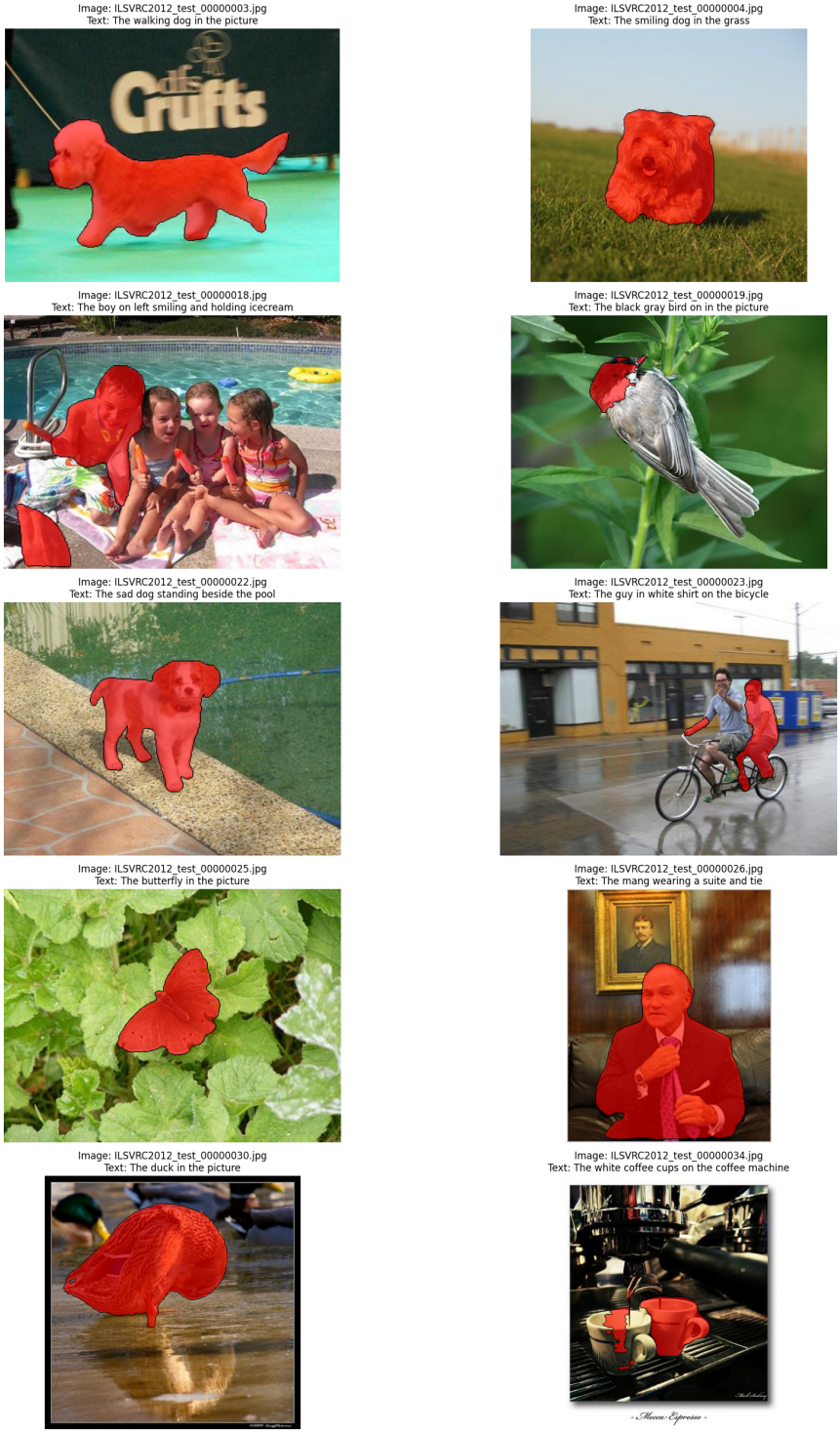



Figure 13: Segmentation with Provided References

## Part 3. Y1 Feature Map Visualization

Not attempted this part.

## Part 4. Failure Cases with Custom References

I supplied my own challenging or misleading descriptions in `my_reference.txt`. When re-running inference, the LAVT masks often missed or mis-localized the target, like incorrectly identifying the first man in the bicycle or not identifying the butterfly, these failure overlays alongside the custom prompts highlight the model's limitations on out-of-distribution or ambiguous queries. The results are shown below.

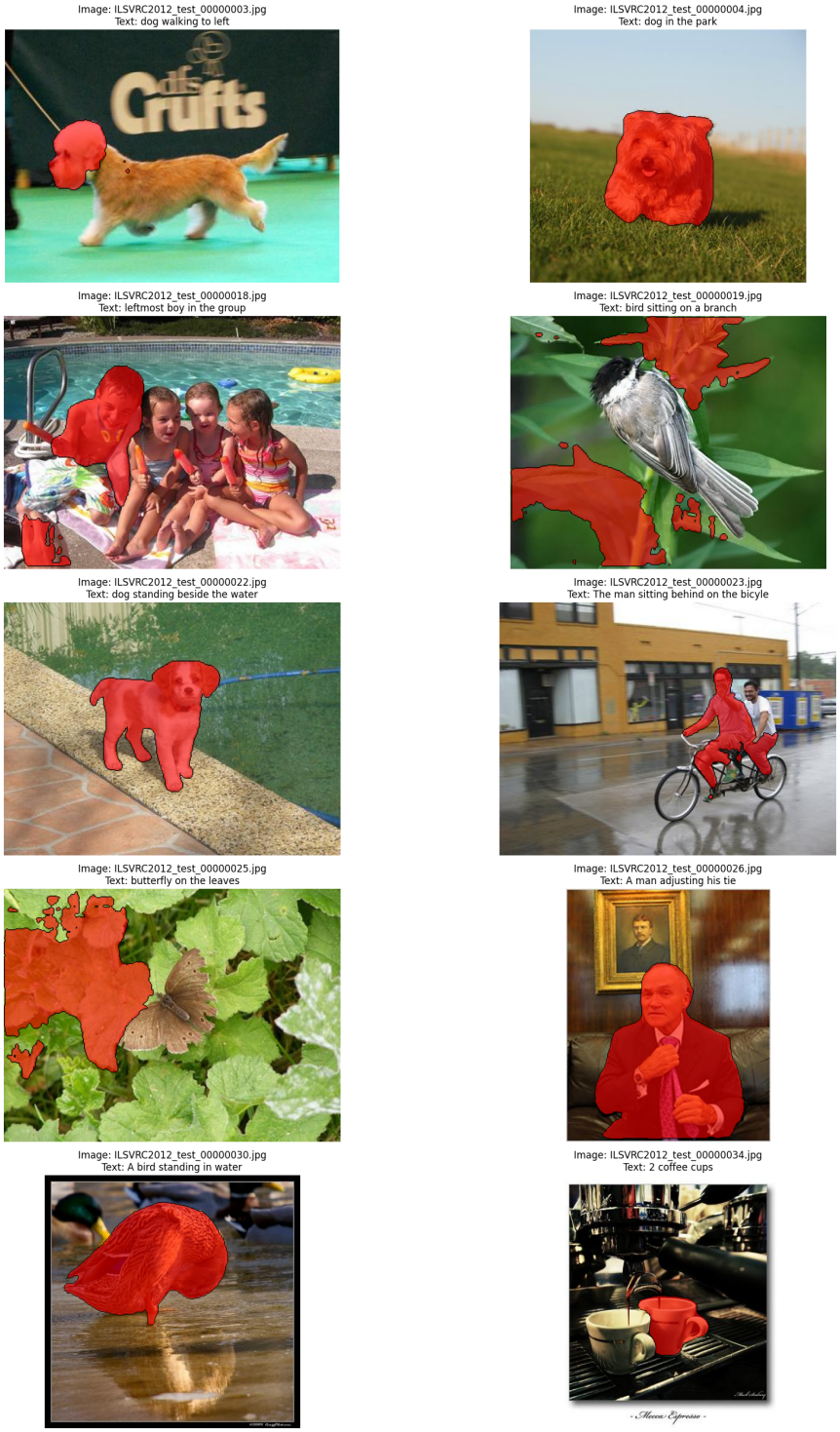



Figure 14: Failure Cases with Custom References

## References


- **Official LAVT Demo Script:** [LAVT-RIS GitHub – demo_inference.py]
- **TA's LAVT Colab Demo Notebook:** [Colab – LAVT RIS Demo by TA]
- **LAVT Paper (CVPR 2022):** [Yang et al. – LAVT: Language-Aware Vision Transformer for Referring Image Segmentation]


## Question 5: Image as reference

Not attempted this question.