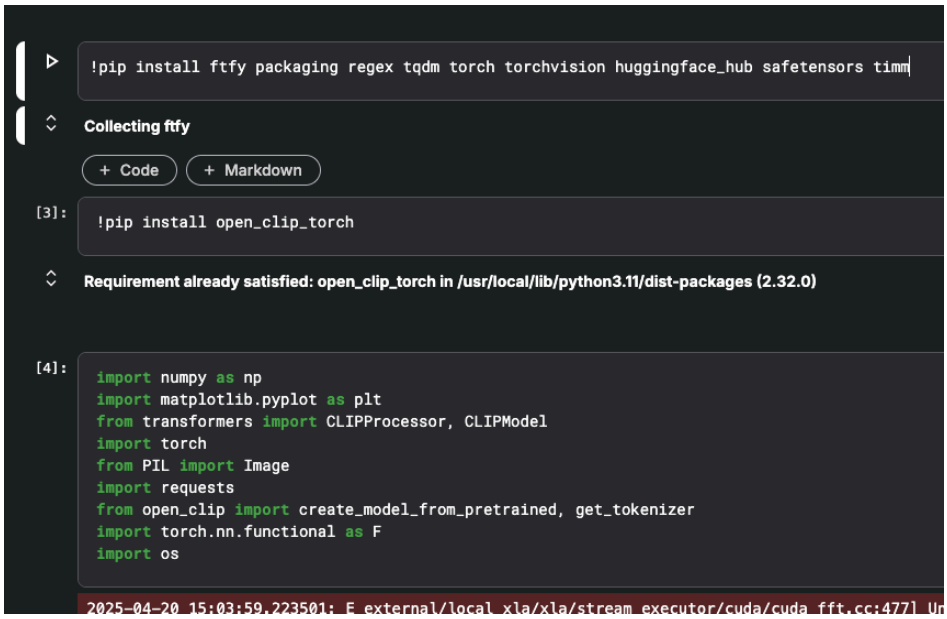
Q1.1 Manually installed the required dependencies, loaded the model using transformers.CLIPModel and the processor using transformers.CLIPProcessor

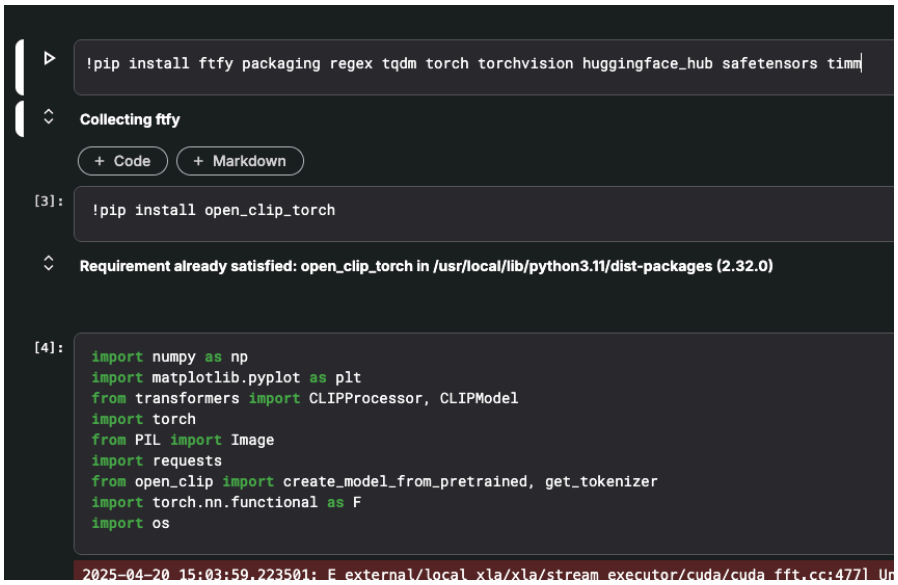


Q1.2 Loaded the CLIP weights as present in the transformers library

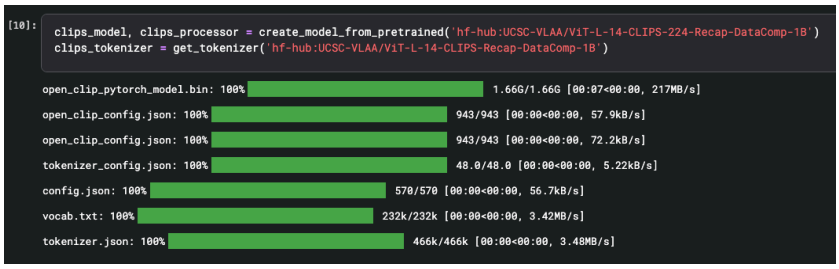
Q1.3 Chose these sample captions and generated the similarity scores

```
Similarity scores for each text description:
a person walking with a dog: 27.0259
a man and his pet: 29.0035
a human and canine companion: 27.7823
a person holding a dog lika a baby: 27.1600
a man playing with his dog: 29.0925
a person and dog resting: 26.1550
a human holding a pet: 29.5404
a man and his loyal companion: 27.6204
a person with their furry friend: 28.8493
a human and dog enjoying home time: 29.2507
```

Q1.4 Download the dependencies and used the open_clip library for the model and tokenizer



Q1.5 Loaded the weights as per the github

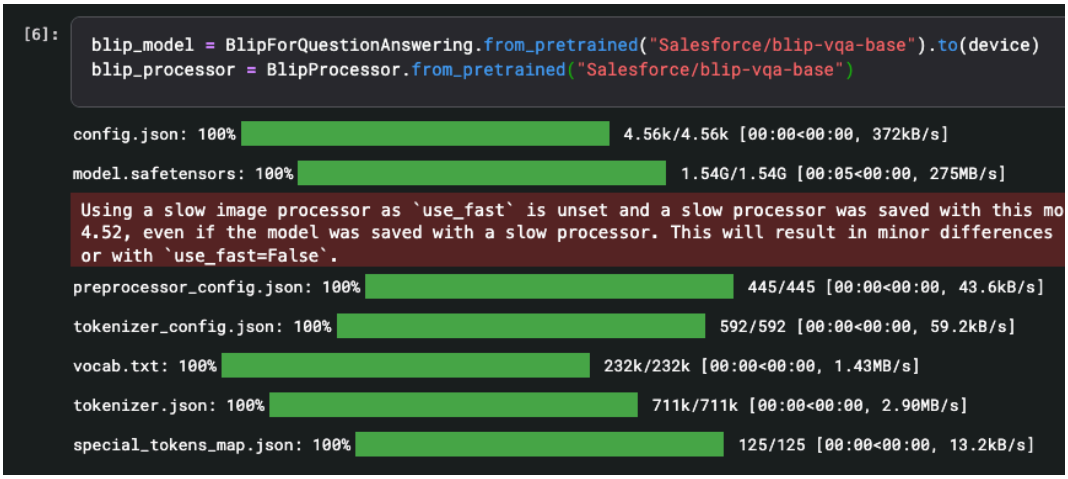


Q1.6 Calculated the similarity scores using the CLIPS model

```
CLIPS Similarity scores for each text:
a person walking with a dog: 8.3078
a man and his pet: 15.4639
a human and canine companion: 13.9483
a person holding a dog lika a baby: 11.1285
a man playing with his dog: 13.9156
a person and dog resting: 7.2605
a human holding a pet: 13.1432
a man and his loyal companion: 13.7897
a person with their furry friend: 12.7554
a human and dog enjoying home time: 12.4032
```

Q1.7 The scores generated by the CLIP (ViT-B/32) have very low variance, ranging from 26.1550 to 29.5404. It gives a higher score to a semantically more accurate caption, such as “a human holding a pet”. On the other hand, the CLIPS model (ViT-L/14) shows quite a high variance, ranging from 7.2605 to 15.4639, punishing captions which are quite incorrect with regards to the image, such as “a person and dog resting”, when the human is standing in the image, and gives a high score to simple but correct prompts such as “a man and his pet”.

Q2.1 Manually installed the dependencies, used transformers.BlipProcessor and transformers.BlipForQuestionAnswering, and imported the pretrained weights of the base model



Q2.2 Generated answer using the appropriate model and processor objects for the question

```
[8]: inputs_3 = blip_processor(image_2, question, return_tensors="pt").to(device)

with torch.no_grad():
    output_3 = blip_model.generate(**inputs_3)

answer_3 = blip_processor.decode(output_3[0], skip_special_tokens=True)
print("Question:", question)
print("Reply:", answer_3)

Question: Where is the dog present in the image?
Reply: in man ' s arms
```

Q2.3 Did the same for the other question to be asked

```
[9]: question_2 = "Where is the man present in the image?"
inputs_3_2 = blip_processor(image_2, question_2, return_tensors="pt").to(device)

with torch.no_grad():
    output_3_2 = blip_model.generate(**inputs_3_2)

answer_3_2 = blip_processor.decode(output_3_2[0], skip_special_tokens=True)
print("Question:", question_2)
print("Reply:", answer_3_2)

Question: Where is the man present in the image?
Reply: living room
```

Q2.4 The answers of the above questions were accurate and straightforward.

Q3.1 Used the transformers.BlipForConditionalGeneration library and loaded the weights for the base model

```
[8]: blipc_model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base").to(device)
blipc_processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")

config.json: 100% 4.56k/4.56k [00:00<00:00, 453kB/s]
pytorch_model.bin: 100% 990M/990M [00:11<00:00, 86.0MB/s]
model.safetensors: 100% 990M/990M [00:11<00:00, 82.8MB/s]
preprocessor_config.json: 100% 287/287 [00:00<00:00, 21.9kB/s]
tokenizer_config.json: 100% 506/506 [00:00<00:00, 36.1kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 13.0MB/s]
tokenizer.json: 100% 711k/711k [00:00<00:00, 26.6MB/s]
special_tokens_map.json: 100% 125/125 [00:00<00:00, 10.8kB/s]
```

Q3.2 Generated Caption for each image using the model

Caption: a small dog running across a green field

ILSVRC2012_test_00000004.jpg

Caption: a man riding a bike down a wet street

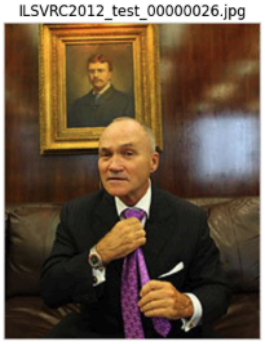
ILSVRC2012_test_00000023.jpg

Caption: a small dog standing on a stone ledge

ILSVRC2012_test_00000022.jpg

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

ILSVRC2012_test_00000026.jpg



Caption: a family sitting in a pool with a towel

ILSVRC2012_test_00000018.jpg

Caption: a small bird sitting on a plant

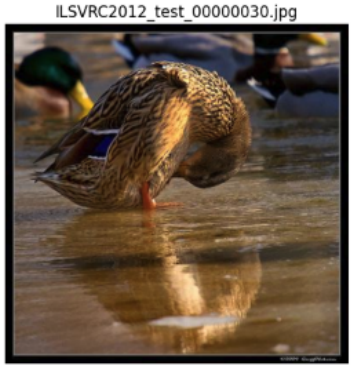
ILSVRC2012_test_00000019.jpg

Caption: a small dog walking on a green carpet

ILSVRC2012_test_00000003.jpg

Caption: a duck drinking water from a pond

ILSVRC2012_test_00000030.jpg



Q3.3 Used CLIP to evaluate the semantic accuracy of the Generated Caption, Similarity scores are as follows:

Caption: a small dog running across a green field
CLIP Eval Score: 32.702602386474 61

ILSVRC2012_test_00000004.jpg

Caption: a man riding a bike down a wet street
CLIP Eval Score: 30.83452606201172

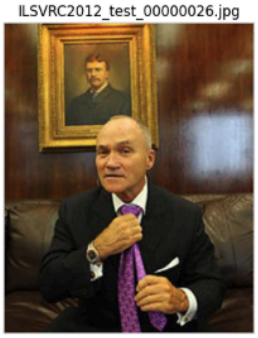
ILSVRC2012_test_00000023.jpg

Caption: a small dog standing on a stone ledge
CLIP Eval Score: 31.03472137451172

ILSVRC2012_test_00000022.jpg

Caption: a man in a suit and tie sitting on a couch
CLIP Eval Score: 28.895353317260742

ILSVRC2012_test_00000026.jpg



Caption: a family sitting in a pool with a towel
CLIP Eval Score: 31.336458206176758

ILSVRC2012_test_00000018.jpg

Caption: a small bird sitting on a plant
CLIP Eval Score: 28.93827247619629

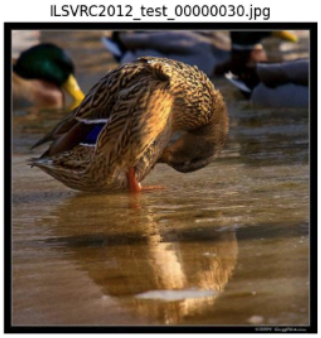
ILSVRC2012_test_00000019.jpg

Caption: a small dog walking on a green carpet
CLIP Eval Score: 31.566001892089844

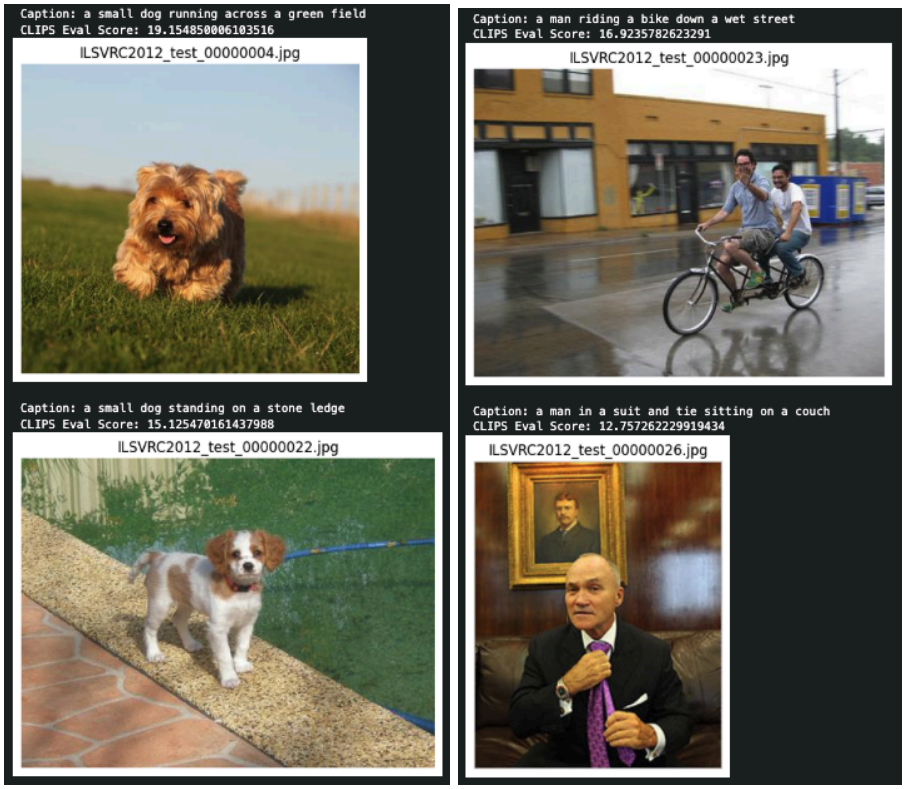
ILSVRC2012_test_00000003.jpg

Caption: a duck drinking water from a pond
CLIP Eval Score: 30.52522087097168

ILSVRC2012_test_00000030.jpg



Q3.4 Did the same using the CLIPS model



Caption: a family sitting in a pool with a towel
CLIPS Eval Score: 14.84744644165039

ILSVRC2012_test_00000018.jpg

Caption: a small bird sitting on a plant
CLIPS Eval Score: 16.976831436157227

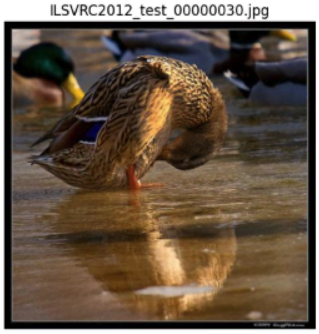
ILSVRC2012_test_00000019.jpg

Caption: a small dog walking on a green carpet
CLIPS Eval Score: 18.1807804107666

ILSVRC2012_test_00000003.jpg

Caption: a duck drinking water from a pond
CLIPS Eval Score: 16.51715850830078

ILSVRC2012_test_00000030.jpg



Caption: a coffee machine with two cups on it
CLIPS Eval Score: 15.60645866394043

ILSVRC2012_test_00000034.jpg

Caption: a brown butterfly sitting on a green plant
CLIPS Eval Score: 16.74422836303711

ILSVRC2012_test_00000025.jpg

Q3.5 The direct **CLIP/CLIPS scores** are cosine similarity values between the textual and image embeddings. They are great if one wants an overall score of the semantic meaning between the text and the image. The **softmaxed version** is great when we want to compare multiple text descriptions of the image. It should be used to get the best description out of many. Another metric is the **CLIP-Score** which is the normalized version of the above scores and is great to compare the similarity across different pairs of text and images and different datasets

Q4.1 Cloned the github and installed the dependencies. Largely used the github (demo_inference.py file)

[15]:

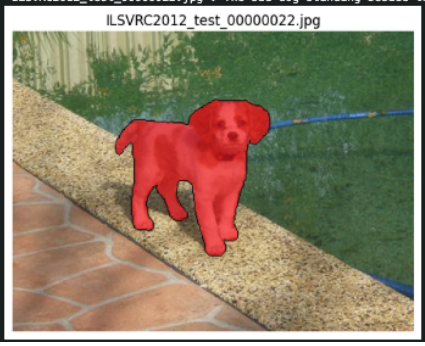
```
from bert.modeling_bert import BertModel
from lib import segmentation

weights = '/kaggle/input/refcoco.pth'

class args:
    swin_type = 'base'
    window12 = True
    mha = ''
    fusion_drop = 0.0


single_model = segmentation.__dict__['lavt'](pretrained='', args=args)
single_model.to(device)
model_class = BertModel
single_bert_model = model_class.from_pretrained('bert-base-uncased')
single_bert_model.pooler = None

checkpoint = torch.load(weights, map_location='cpu')
single_bert_model.load_state_dict(checkpoint['bert_model'])
single_model.load_state_dict(checkpoint['model'])
model = single_model.to(device)
bert_model = single_bert_model.to(device)
```

```
/usr/local/lib/python3.11/dist-packages/timm/models/layers/__init__.py:48: FutureWarni
```

Q4.2 Performed segmentation as per the given prompts

ILSVRC2012_test_00000004.jpg : The smiling dog in the grass

ILSVRC2012_test_00000022.jpg : The sad dog standing beside the pool

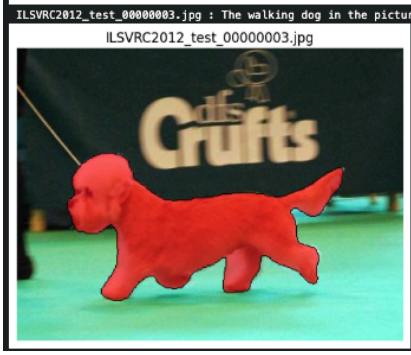


ILSVRC2012_test_00000023.jpg : The guy in white shirt on the bicycle

ILSVRC2012_test_00000026.jpg : The mang wearing a suite and tie

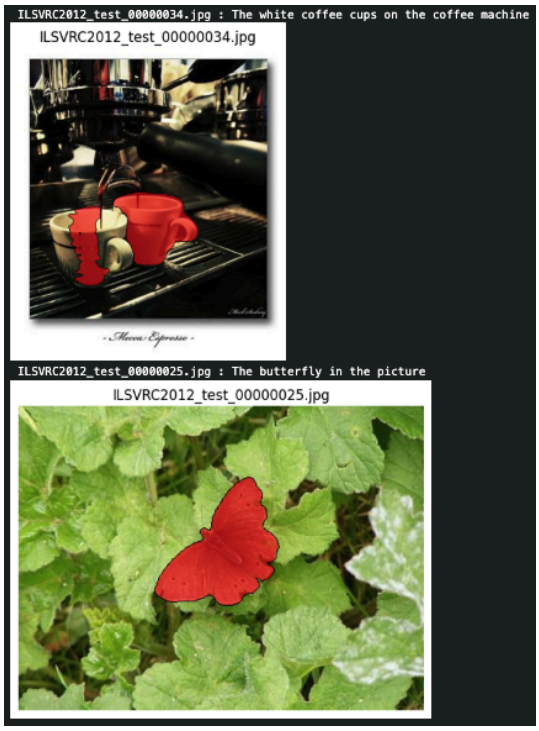
ILSVRC2012_test_00000018.jpg : The boy on left smiling and holding icecream

ILSVRC2012_test_00000003.jpg : The walking dog in the picture

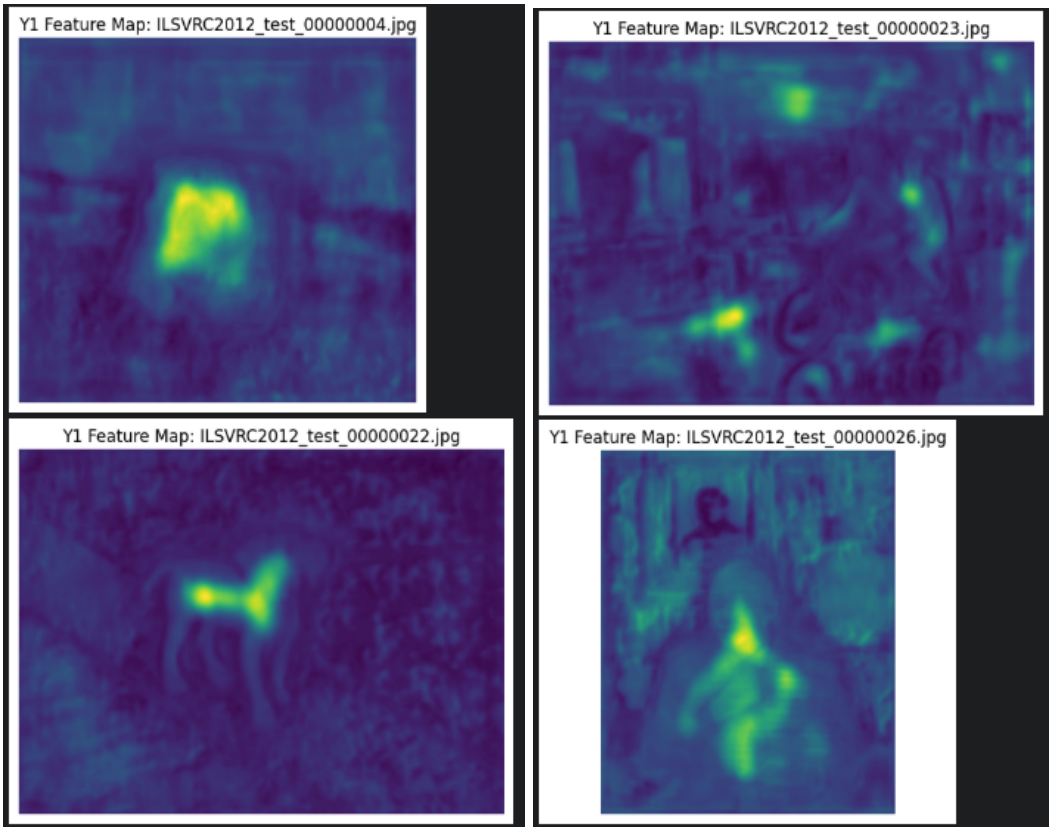


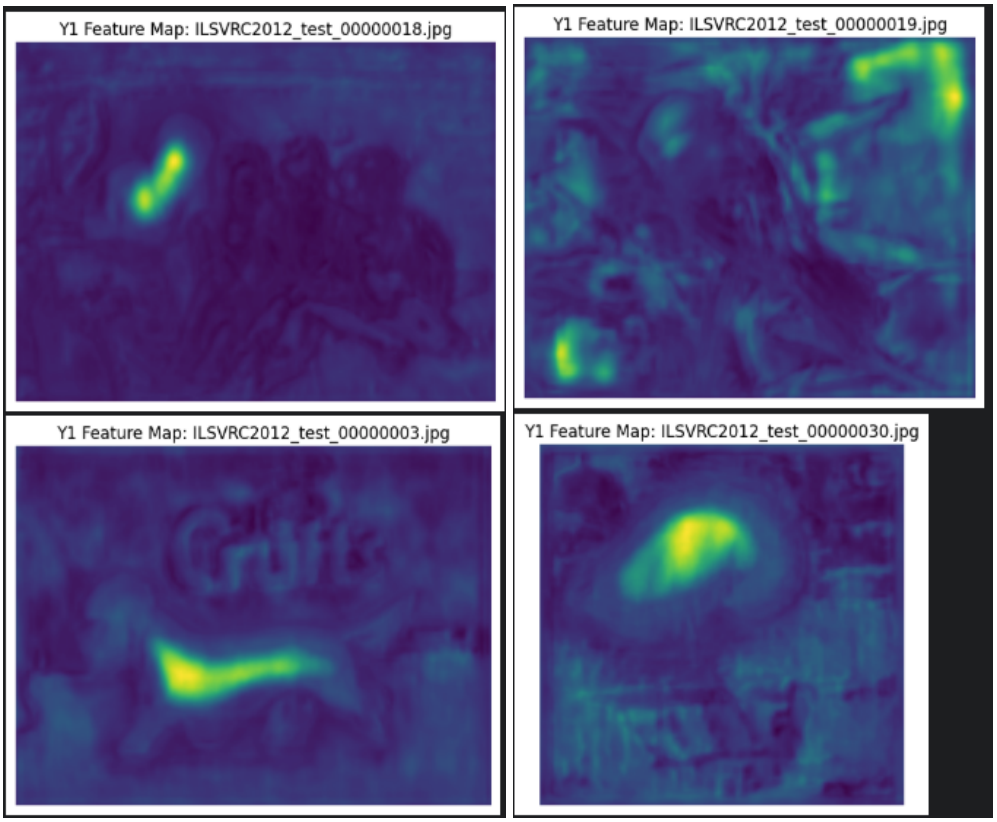
ILSVRC2012_test_00000019.jpg : The black gray bird on in the picture

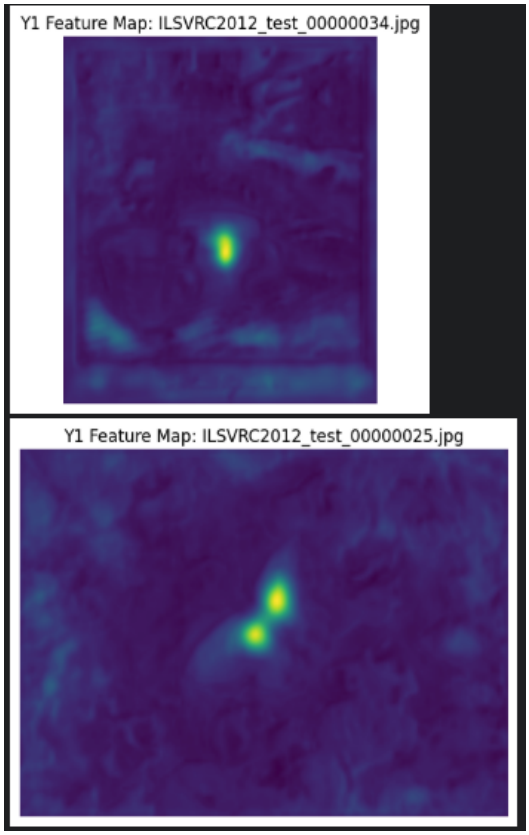
ILSVRC2012_test_00000030.jpg : The duck in the picture



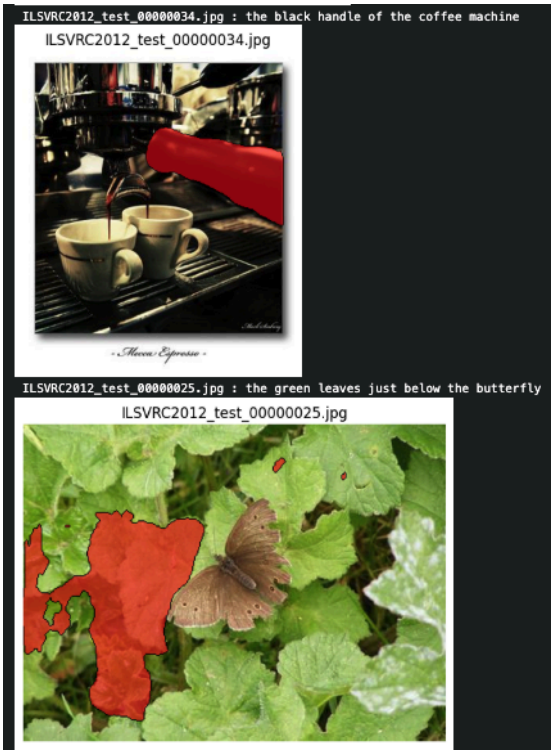
Q4.3 Plotted the Y1 feature map for each image, after saving the feature maps while performing segmentation




Y1 Feature Map: ILSVRC2012_test_00000018.jpg
Y1 Feature Map: ILSVRC2012_test_00000019.jpg
Y1 Feature Map: ILSVRC2012_test_00000003.jpg
Y1 Feature Map: ILSVRC2012_test_00000030.jpg


Y1 Feature Map: ILSVRC2012_test_00000034.jpg
Y1 Feature Map: ILSVRC2012_test_00000025.jpg

Q4.4 Created reference texts as similar in format to the original texts as possible, here are the results:



Q5.1 Cloned the github and installed the required dependencies, used eval code largely from the github

```
images_dir='/kaggle/input/cv-a3-all/Images-20250418T060633Z-001/Images'
output_dir='output_segmentations'
dinov2_weights='/kaggle/input/cv-a3-all/dinov2_vitl14_pretrain.pth'
sam_weights='/kaggle/input/cv-a3-all/sam_vit_h_4b8939.pth'

args = argparse.Namespace(
    images_dir=images_dir,
    output_dir=output_dir,
    dinov2_weights=dinov2_weights,
    sam_weights=sam_weights,

    dinov2_size='vit_large',
    sam_size='vit_h',
    num_centers=8,
    use_box=False,
    use_points_or_centers=False,
    sample_range="(4,6)",
    max_sample_iterations=30,
    alpha=1.0,
    beta=0.0,
    exp=0.0,
    emd_filter=0.0,
    purity_filter=0.0,
    coverage_filter=0.0,
    use_score_filter=False,
    deep_score_norm_filter=0.1,
    deep_score_filter=0.33,
    topk_scores_threshold=0.7,
    num_merging_mask=10,

    points_per_side=64,
    pred_iou_thresh=0.88,
    stability_score_thresh=0.95,
    sel_stability_score_thresh=0.0,
    iou_filter=0.0,
    box_nms_thresh=1.0,
    output_layer=3,
    dense_multimask_output=0,
    use_dense_mask=0,
    multimask_output=0
)

args.sample_range = eval(args.sample_range)

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

matcher = build_matcher_oss(args)
```

Q5.2 Used the matcher paradigm containing DinoV2 and SAM models