CV Assignment 3

Vikranth Udandarao
2022570

Coding

1. 1. 2.

```python
from transformers import CLIPProcessor, CLIPModel
from PIL import Image
import torch

# Load model and processor
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
```

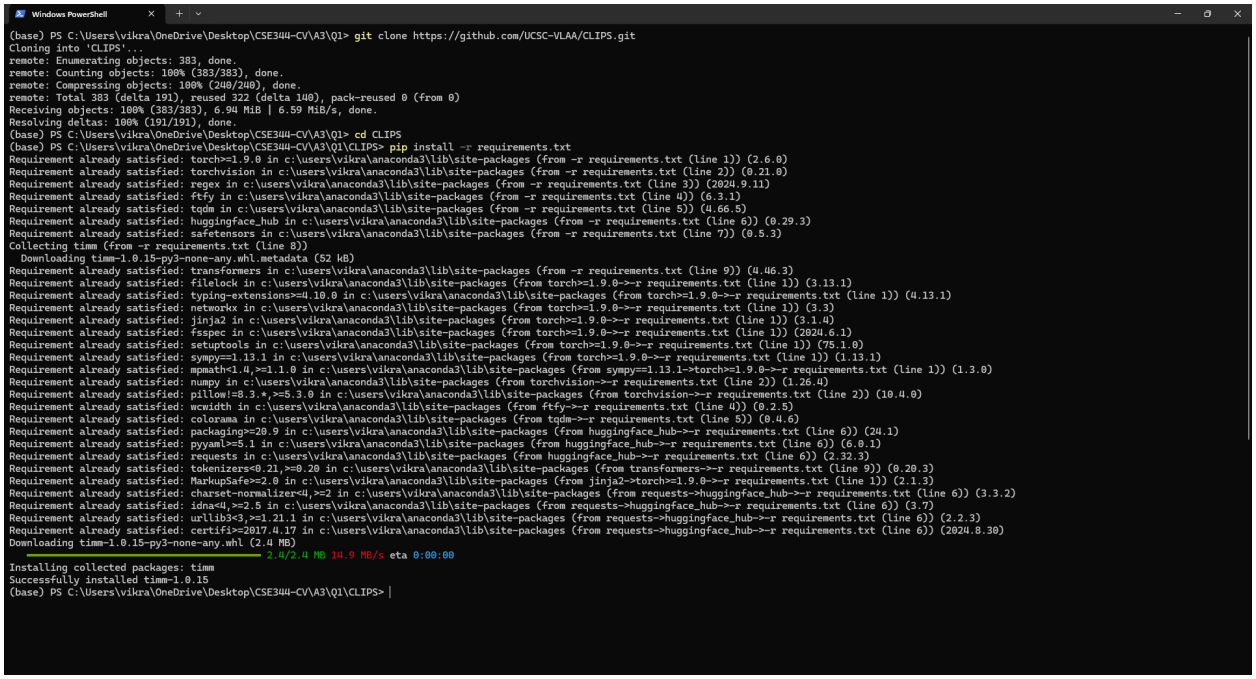
3. For this Image:

I used these text descriptions:

Caption	Reason
1. a man holding a giant dog	Baseline short caption; top match for both models
2. a man in a white shirt and gray pants holding a massive dog in his arms	Best long match; shows CLIPS precision
3. a man in casual clothing holding a very large brown dog on a leash	Similar but slightly wrong — tests tolerance
4. a dog sitting on a couch	Partial object match; triggers false positives in CLIP
5. a woman playing with her cat	Semantic distractor (wrong subject and object)
6. a professionally dressed man in an office petting a golden retriever	Same structure, wrong setting
7. a bookshelf full of books	Completely irrelevant (semantic zero shot)
8. a blurry photo of a man trying to restrain a dog during a dog show	Similar objects but vague quality — tests robustness
9. a drawing of a man and a dog in a surreal fantasy landscape	Visual domain mismatch (cartoon vs real)
10. a man feeding his pet dog while sitting on a wooden floor near a fireplace	Near-match with scene setting and action shift

S.No.	Text Description	CLIP Score
1	a man holding a giant dog	0.0761
2	a man in a white shirt and gray pants holding a massive dog in his arms	0.9207
3	a man in casual clothing holding a very large brown dog on a leash	0.0022
4	a dog sitting on a couch	0.0002
5	a woman playing with her cat	0.0002
6	a professionally dressed man in an office petting a golden retriever	0.0003
7	a bookshelf full of books	0.0000
8	a blurry photo of a man trying to restrain a dog during a dog show	0.0000
9	a drawing of a man and a dog in a surreal fantasy landscape	0.0001
10	a man feeding his pet dog while sitting on a wooden floor near a fireplace	0.0003

4.



5.

```python
import torch
import torch.nn.functional as F
from PIL import Image
from open_clip import create_model_from_pretrained, get_tokenizer

# Load model and tokenizer
model, preprocess = create_model_from_pretrained('hf-hub:UCSC-VLAA/ViT-L-14-CLIPS-Recap-DataComp-1B')
tokenizer = get_tokenizer('hf-hub:UCSC-VLAA/ViT-L-14-CLIPS-Recap-DataComp-1B')
```

6.

S.No.	Text Description	CLIPS Score
1	a man holding a giant dog	0.1529
2	a man in a white shirt and gray pants holding a massive dog in his arms	0.8466
3	a man in casual clothing holding a very large brown dog on a leash	0.0004
4	a dog sitting on a couch	0.0000
5	a woman playing with her cat	0.0000
6	a professionally dressed man in an office petting a golden retriever	0.0000
7	a bookshelf full of books	0.0000
8	a blurry photo of a man trying to restrain a dog during a dog show	0.0000
9	a drawing of a man and a dog in a surreal fantasy landscape	0.0000
10	a man feeding his pet dog while sitting on a wooden floor near a fireplace	0.0000

7.

1. Primary Match Detection

Both models successfully identified the most accurate caption —
"a man in a white shirt and gray pants holding a massive dog in his arms" —
as the best match. CLIP assigned it a high score of **0.9207**, while CLIPS was similarly confident at **0.8466**.

This shows that both models are capable of grounding visual content in textual form when the match is precise and descriptive.

2. Score Distribution

- **CLIP** spreads some probability mass across other captions that are *semantically close* but not exact matches. For instance:
    - *"a man holding a giant dog"*: **0.0761**
    - *"a man in casual clothing holding a very large brown dog on a leash"*: **0.0022**
- **CLIPS**, in contrast, is **much sharper and more selective**. It gives **0.1529** to *"a man holding a giant dog"* and **near-zero to all others*.
  This behavior reflects its training with **synthetic captions**, which enforces **tight alignment** between images and captions, minimizing tolerance for semantic drift.

3. Handling of Distractors

Captions that are clearly unrelated — such as *"a bookshelf full of books"* or *"a woman playing with her cat"* — were assigned **scores close to zero** by both models, indicating robust filtering of irrelevant descriptions.

However, CLIP **still gives tiny non-zero values** (e.g., 0.0002–0.0003), while CLIPS confidently assigns **absolute zero** to such distractors, showing stricter confidence boundaries.

4. Sensitivity to Caption Specificity

- CLIPS clearly prefers **longer, precise captions** that match multiple visual attributes (clothing, pose, object scale, etc.).
- CLIP can handle both short and long captions, but tends to **favor more detailed ones** for higher scores.

CLIPS demonstrates **tighter visual-semantic alignment** and **higher precision**, making it ideal for applications that require unambiguous matching (e.g., visual question answering, caption evaluation).
CLIP, on the other hand, offers **greater generalization** and may be better suited for **retrieval**, **semantic search**, or tasks where multiple captions could validly describe the same image.

2. 1.

```python
from transformers import BlipProcessor, BlipForQuestionAnswering
from PIL import Image

# Load model and processor
processor = BlipProcessor.from_pretrained("Salesforce/blip-vqa-base")
model = BlipForQuestionAnswering.from_pretrained("Salesforce/blip-vqa-base")
```

2. 3. 4. Sample Image is this:

Question 1: “Where is the dog present in the image?”

Answer: *“in man’s arms”*

The model correctly identifies the **spatial relationship** between the dog and the man.

It demonstrates fine-grained **understanding of objects and their interaction**, which shows that BLIP can reason about **contextual positioning**.
Highly accurate and context-aware answer.

Question 2: “Where is the man present in the image?”

Answer: *“living room”*

The model accurately infers the **scene type** from background cues like bookshelves, wooden flooring, and indoor furniture.

This reflects BLIP's ability to **capture the overall environment** rather than just objects. *Correct high-level scene understanding.*

3. 1.

```python
from PIL import Image
from transformers import BlipProcessor, BlipForConditionalGeneration
import torch

# Load model and processor
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
```

2.

Image Name	BLIP Caption
1.jpg	a dog is walking on a green carpet
2.jpg	a small dog running across a green field
3.jpg	the girls in the pool
4.jpg	a bird perched on a plant
5.jpg	a small dog standing on a stone ledge next to a pool
6.jpg	a man riding a bike down the street

7.jpg	a brown butterfly sitting on a green plant
8.jpg	a man in a suit and tie sitting on a couch
9.jpg	a duck drinking water from a pond
10.jpg	a coffee machine with a cup of coffee

3. **CLIP Scores:**

Image	BLIP Caption	CLIP Similarity Score
1.jpg	a dog is walking on a green carpet	0.3008
2.jpg	a small dog running across a green field	0.3295
3.jpg	the girls in the pool	0.2696
4.jpg	a bird perched on a plant	0.2800
5.jpg	a small dog standing on a stone ledge next to a pool	0.3225
6.jpg	a man riding a bike down the street	0.2648

7.jpg	a brown butterfly sitting on a green plant	0.2965
8.jpg	a man in a suit and tie sitting on a couch	0.2876
9.jpg	a duck drinking water from a pond	0.3037
10.jpg	a coffee machine with a cup of coffee	0.2816

4. **CLIP Scores:**

Image	BLIP Caption	CLIPS Similarity Score
1.jpg	a dog is walking on a green carpet	0.1849
2.jpg	a small dog running across a green field	0.2014
3.jpg	the girls in the pool	0.1510
4.jpg	a bird perched on a plant	0.1666
5.jpg	a small dog standing on a stone ledge next to a pool	0.1887

6.jpg	a man riding a bike down the street	0.1526
7.jpg	a brown butterfly sitting on a green plant	0.1739
8.jpg	a man in a suit and tie sitting on a couch	0.1298
9.jpg	a duck drinking water from a pond	0.1705
10.jpg	a coffee machine with a cup of coffee	0.1330

5.

1. Cosine Similarity (CLIP)

- **Quick and widely used** baseline metric.
- Doesn't require reference captions.
- High similarity means the caption is semantically aligned to the image.
- **Limitation**: Can be fooled by plausible-sounding but incorrect captions.

Example Use Case: Use it for: Simple alignment testing when reference captions aren't available.

2. CLIPS Score (CLIPS-C/S/R)

- Measures how well the **caption captures concepts, scene, and relationships**.
- Designed for **fine-grained visual grounding** evaluation.

- Low scores may indicate hallucinations or missing relationships.

Example Use Case: Use it when you want to go beyond "is this caption close?" to "does it say the right thing about the scene?"

3. BLEU / ROUGE

- Lexical overlap metrics.
- Require ground-truth captions.
- Can be misleading for valid but paraphrased captions.

Example Use Case: Use these when comparing with human-written ground truths in datasets like MSCOCO or Flickr30K.

4. BERTScore

- Captures **semantic similarity** using contextual embeddings.
- Better at matching **paraphrases** than BLEU or ROUGE.

Example Use Case: Use when evaluating **free-form captions** or testing **semantic variation** robustness.

5. SPICE

- Converts caption into a scene graph and compares to reference.
- Best for evaluating **complex relationships** in scene understanding.

Example Use Case: Use when relationships matter: "boy holding a bat" vs. "bat holding a boy."

6. CIDEr

- Scores based on **consensus** of multiple human-written references.
- Favors **relevant and rare content**.

Example Use Case: Use in benchmark settings (e.g., MSCOCO leaderboard).

7. CLIPScore

- Recently introduced; relies on **CLIP similarity scaled for caption quality**.
- Correlates well with human judgments in some settings.
- Doesn't need references.

Example Use Case: Use for zero-shot, reference-free caption quality evaluation.

4. 1.

```
import torch
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import torchvision.transforms as T
from bert_.modeling_bert import BertModel
from bert_.tokenization_bert import BertTokenizer
from lib import segmentation
```

✓ 2.7s Python

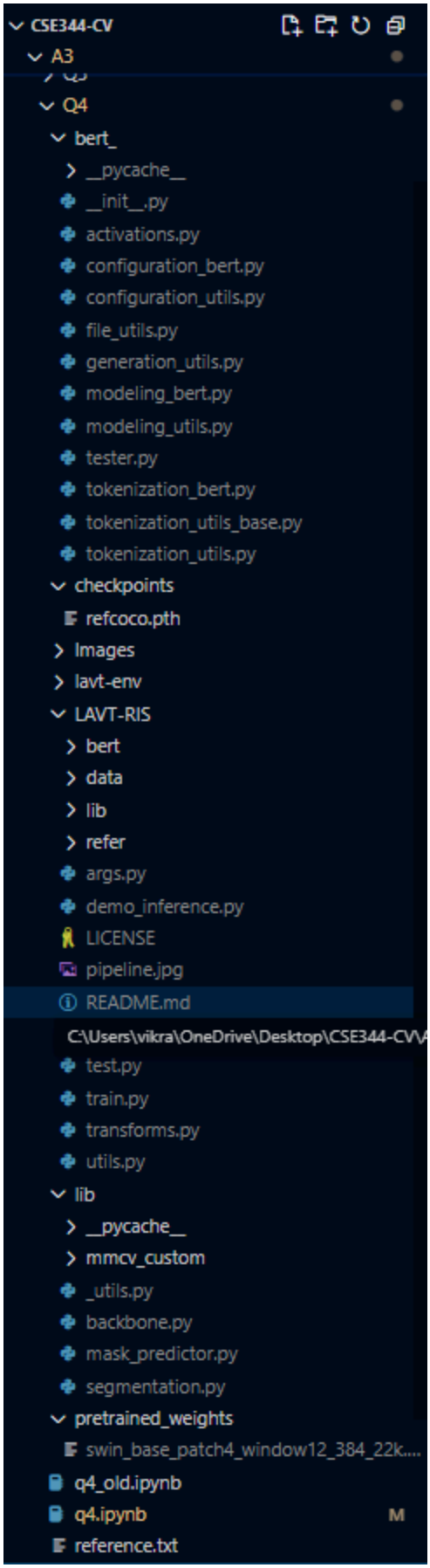
```
c:\Users\vikra\OneDrive\Desktop\CSE344-CV\.venv\Lib\site-packages\timm\models\layers\__init__.py:48: FutureWarning: Importing from timm.models.layers is deprecated, please import via timm.layers
  warnings.warn(f"Importing from {__name__} is deprecated, please import via timm.layers", FutureWarning)
```

```
def initialize_model(weights_path):
    args_ = Args()  # Create an instance of Args
    model = segmentation.lavt(pretrained='', args=args_)
    bert_model = BertModel.from_pretrained('bert-base-uncased')
    bert_model.pooler = None

    checkpoint = torch.load(weights_path, map_location='cpu', weights_only=False)
    bert_model.load_state_dict(checkpoint['bert_model'])
    model.load_state_dict(checkpoint['model'])

    return model.to(device), bert_model.to(device)
```

```
model, bert_model = initialize_model('checkpoints/refcoco.pth')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

✓ 8.9s

```
Window size 12!
Randomly initialize Multi-modal Swin Transformer weights.
Randomly initialize Multi-modal Swin Transformer weights.
```


CSE344-CV
A3
Q4
bert_
__pycache__
__init__.py
activations.py
configuration_bert.py
configuration_utils.py
file_utils.py
generation_utils.py
modeling_bert.py
modeling_utils.py
tester.py
tokenization_bert.py
tokenization_utils_base.py
tokenization_utils.py
checkpoints
refcoco.pth
Images
lavt-env
LAVT-RIS
bert
data
lib
refer
args.py
demo_inference.py
LICENSE
pipeline.jpg
README.md
C:\Users\vikra\OneDrive\Desktop\CSE344-CV\A
test.py
train.py
transforms.py
utils.py
lib
__pycache__
mmcv_custom
_utils.py
backbone.py
mask_predictor.py
segmentation.py
pretrained_weights
swin_base_patch4_window12_384_22k....
q4_old.ipynb
q4.ipynb
M
reference.txt

I downloaded the weights and checkpoints from the README of https://github.com/yz93/LAVT-RIS and also cloned the repo to local for inference and running.

2. I did the normal segmentation and then again put overlay on top of the original image as given in the collab file in classroom comments.

Original Image

Success Segmentation
"The walking dog in the picture"

Success Segmentation Overlay
"The walking dog in the picture"

Original Image

Success Segmentation
"The smiling dog in the grass"

Success Segmentation Overlay
"The smiling dog in the grass"

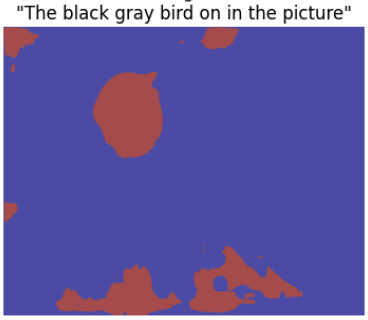
Original Image

Success Segmentation
"The boy on left smiling and holding icecream"

Success Segmentation Overlay
"The boy on left smiling and holding icecream"

Original Image

Success Segmentation
"The black gray bird on in the picture"



Success Segmentation Overlay
"The black gray bird on in the picture"

Original Image

Success Segmentation
"The sad dog standing beside the pool"

Success Segmentation Overlay
"The sad dog standing beside the pool"

Original Image

Success Segmentation
"The guy in white shirt on the bicycle"

Success Segmentation Overlay
"The guy in white shirt on the bicycle"

Original Image

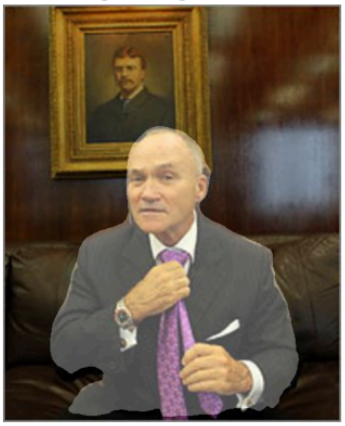
Success Segmentation
"The butterfly in the picture"

Success Segmentation Overlay
"The butterfly in the picture"

Original Image

Success Segmentation
"The mang wearing a suite and tie"

Success Segmentation Overlay
"The mang wearing a suite and tie"



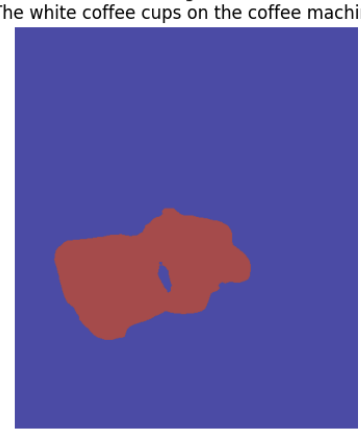
Original Image

Success Segmentation
"The duck in the picture"

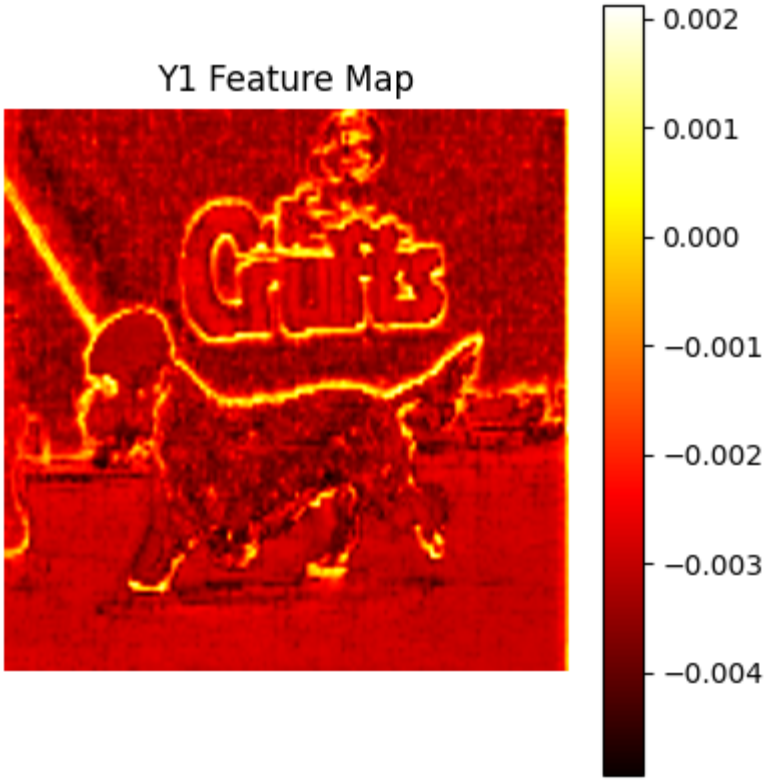
Success Segmentation Overlay
"The duck in the picture"

Original Image

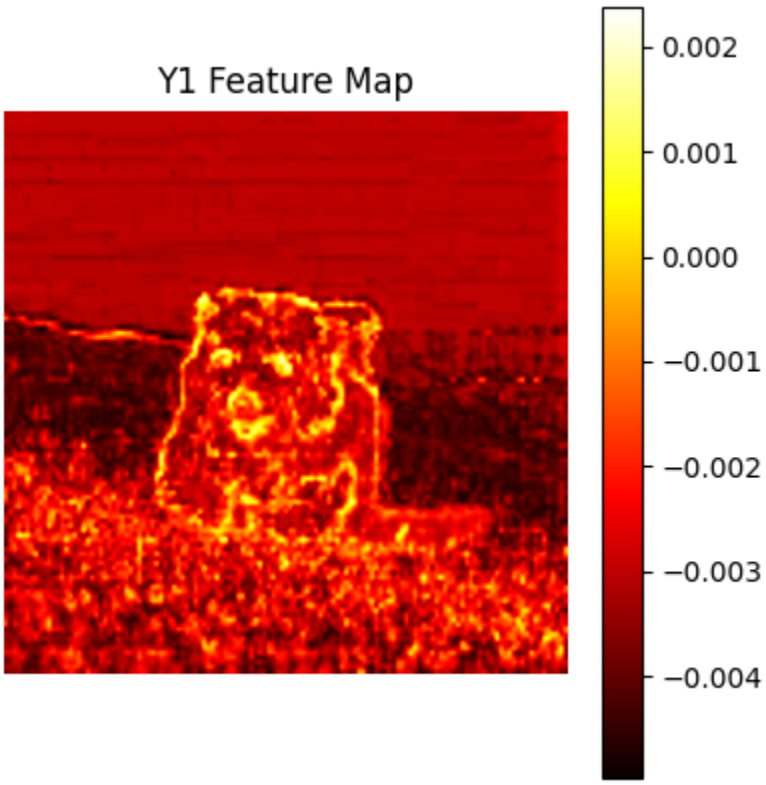
Success Segmentation
"The white coffee cups on the coffee machine"



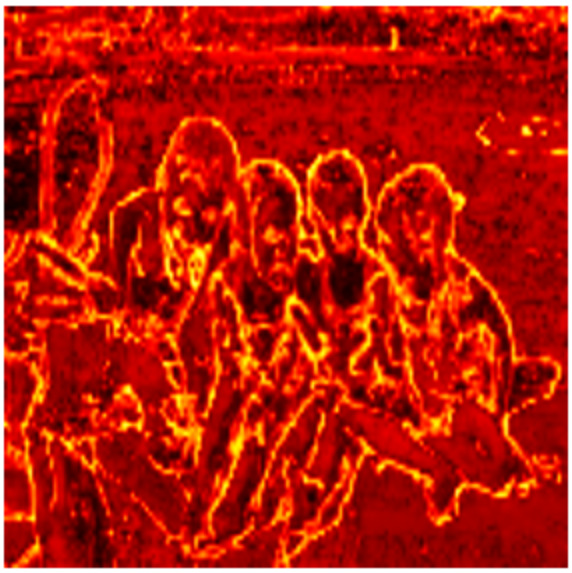
Success Segmentation Overlay
"The white coffee cups on the coffee machine"

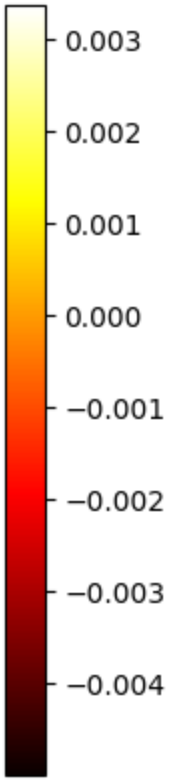

Y1 Feature Map
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004

3.

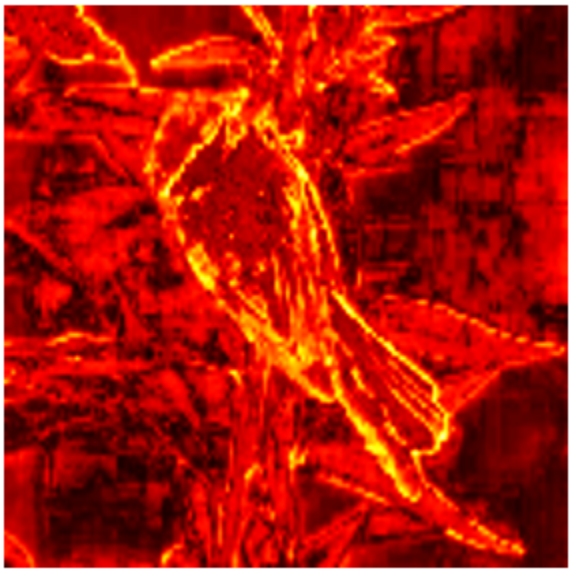

Y1 Feature Map
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004

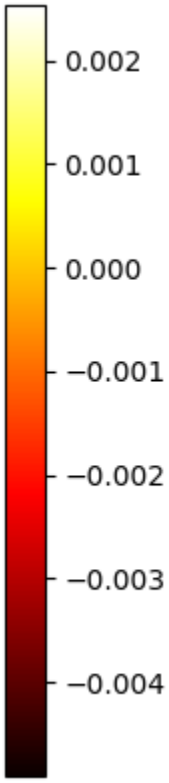
Y1 Feature Map



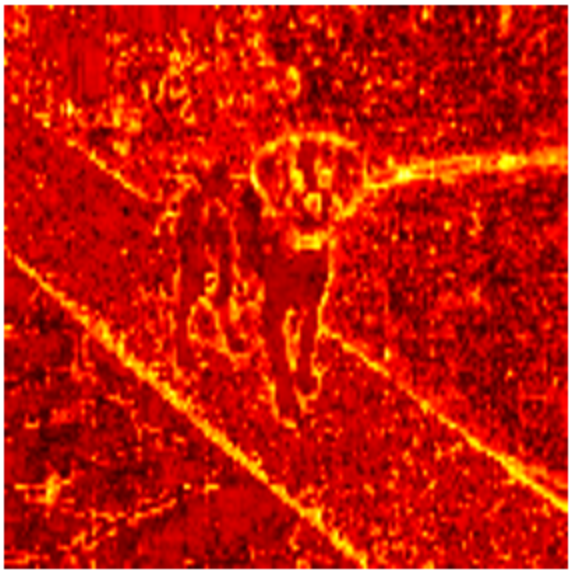


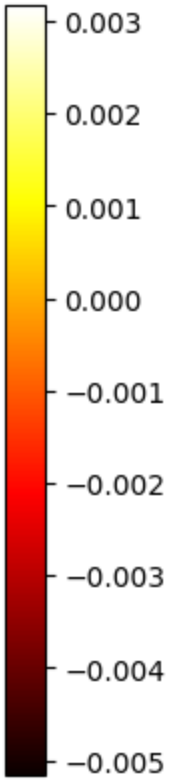
Y1 Feature Map



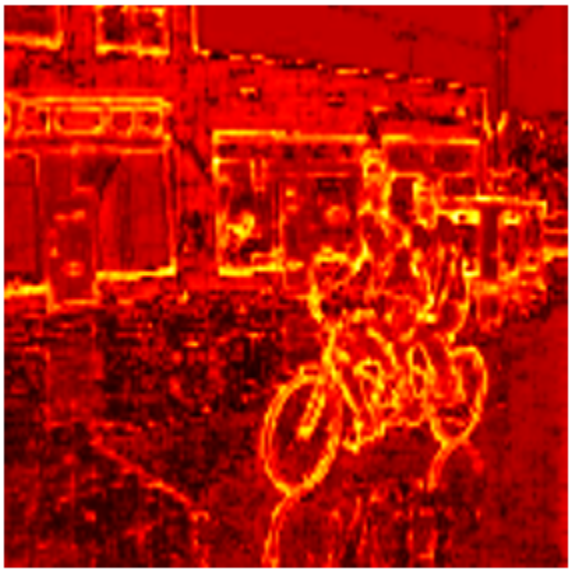


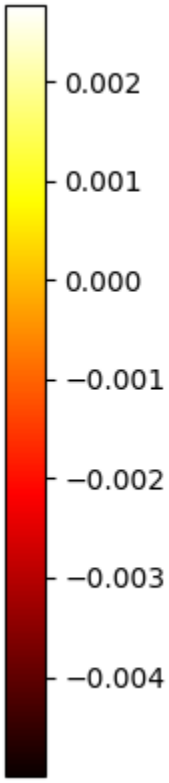
Y1 Feature Map

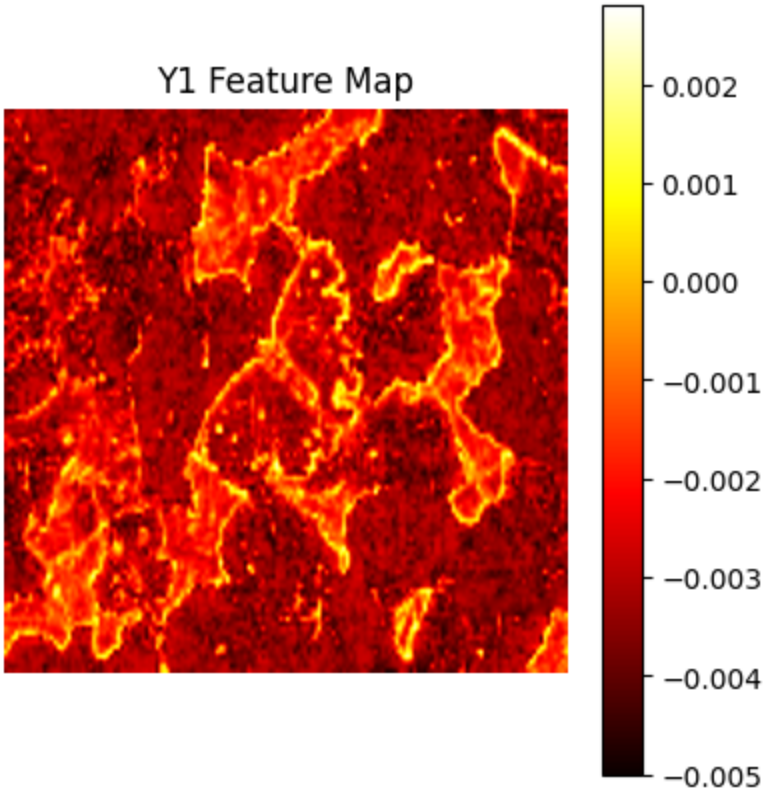


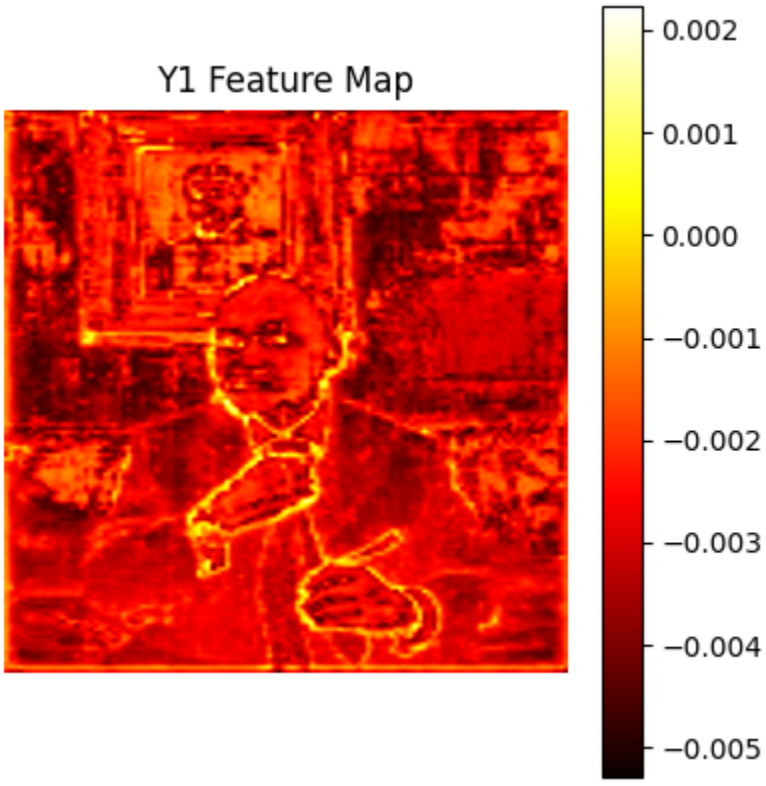


Y1 Feature Map

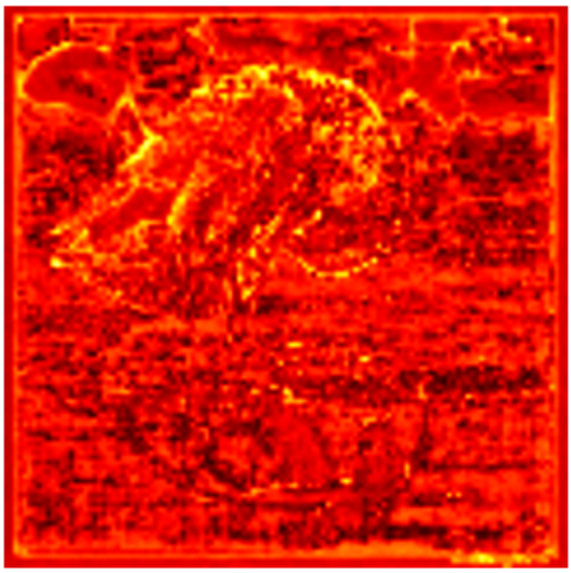


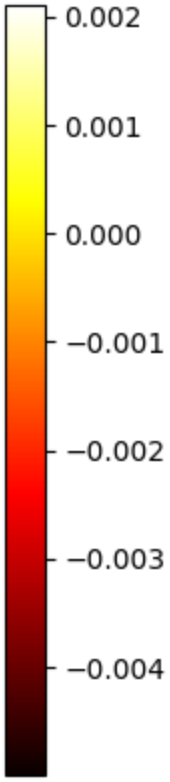



Y1 Feature Map
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
−0.005

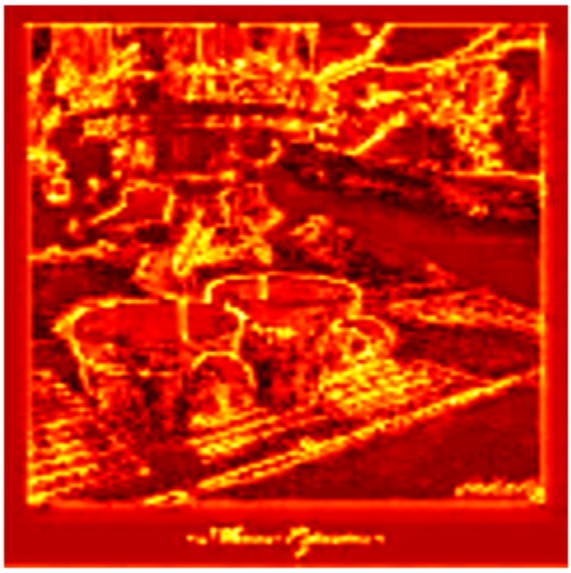

Y1 Feature Map
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
−0.005

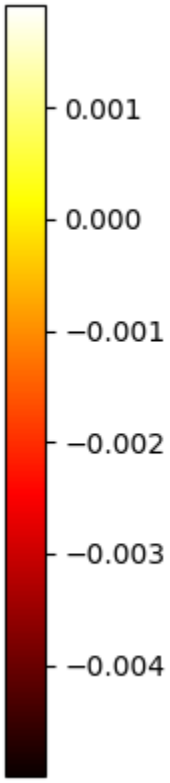
Y1 Feature Map





Y1 Feature Map





4. I used the below as my reference text.

ILSVRC2012_test_00000003.jpg : "A well-groomed dog walks proudly at a dog show."

ILSVRC2012_test_00000004.jpg : "A fluffy dog runs through the grass on a sunny day."

ILSVRC2012_test_00000018.jpg : "Four kids sit by the pool enjoying popsicles together."

ILSVRC2012_test_00000019.jpg : "A small bird with black and white feathers sits on a plant."

ILSVRC2012_test_00000022.jpg : "A puppy stands near a pool, looking at the camera."

ILSVRC2012_test_00000023.jpg : "Two men ride a tandem bike, one smiling and waving."

ILSVRC2012_test_00000025.jpg : "A brown butterfly rests on green leaves."

ILSVRC2012_test_00000026.jpg : "A framed portrait of a man in a suit hangs on a wooden wall."

ILSVRC2012_test_00000030.jpg : "A duck stands on ice, bending its head to clean its feathers."

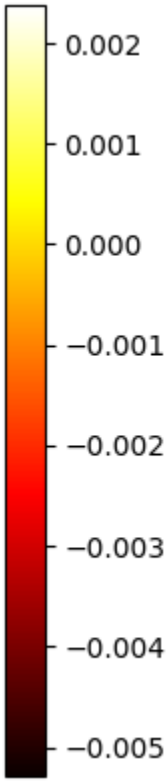
ILSVRC2012_test_00000034.jpg : "Espresso pours into two cups from a coffee machine."

Original Image

Success Segmentation
"A well-groomed dog walks proudly at a dog show."

Success Segmentation Overlay
"A well-groomed dog walks proudly at a dog show."

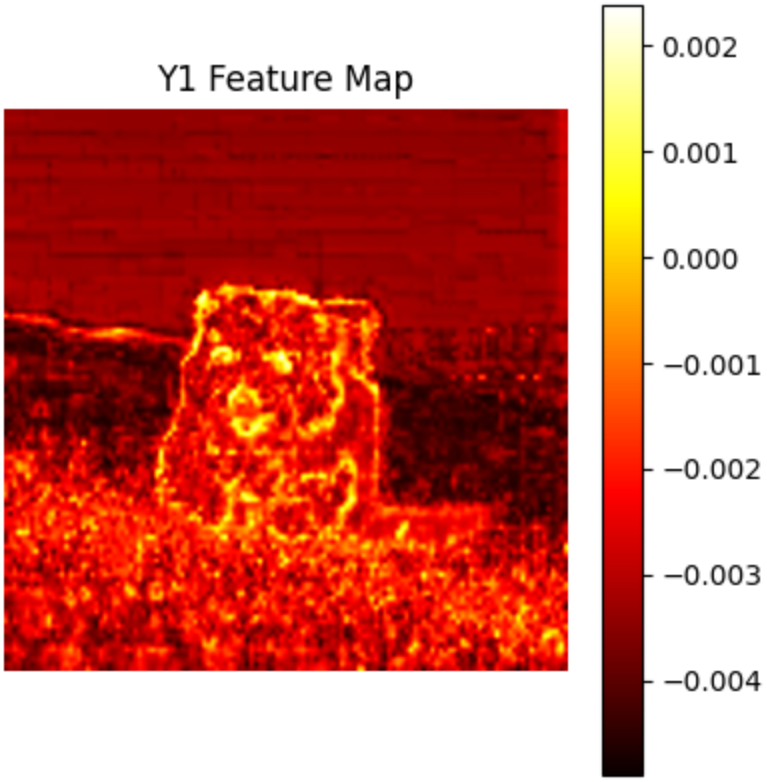
Y1 Feature Map



Original Image

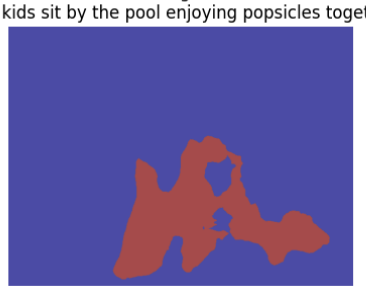
Success Segmentation
"A fluffy dog runs through the grass on a sunny day."

Success Segmentation Overlay
"A fluffy dog runs through the grass on a sunny day."

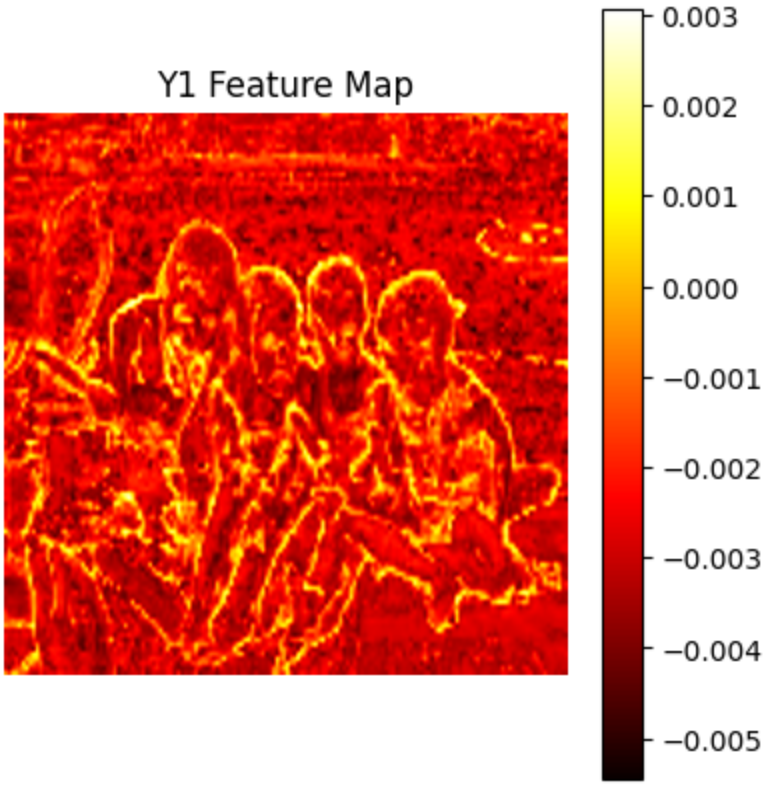


Original Image

Success Segmentation
"Four kids sit by the pool enjoying popsicles together."



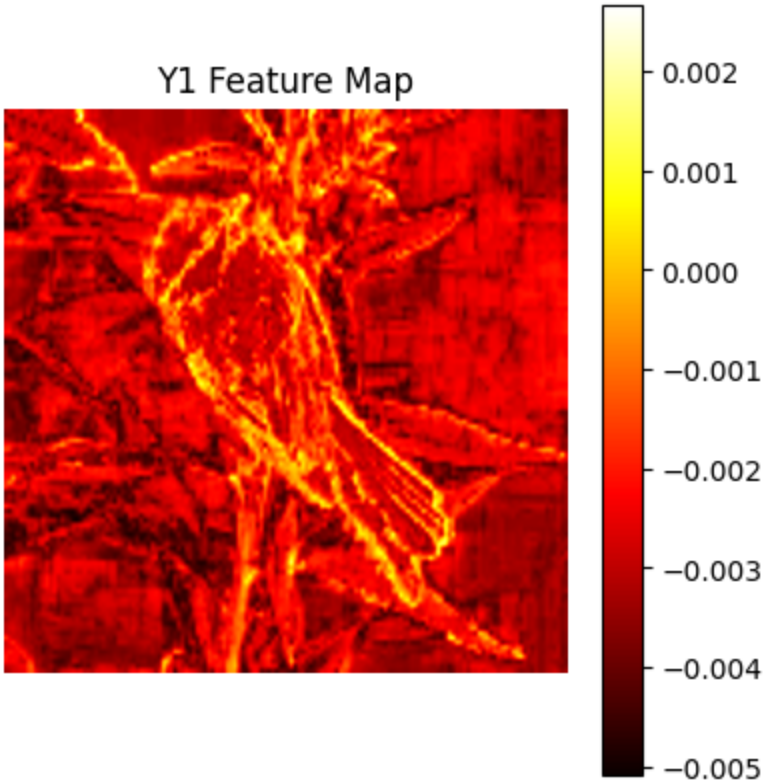
Success Segmentation Overlay
"Four kids sit by the pool enjoying popsicles together."



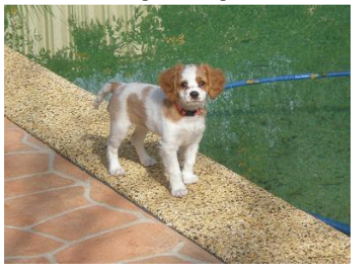
Original Image

Success Segmentation
"A small bird with black and white feathers sits on a plant."

Success Segmentation Overlay
"A small bird with black and white feathers sits on a plant."

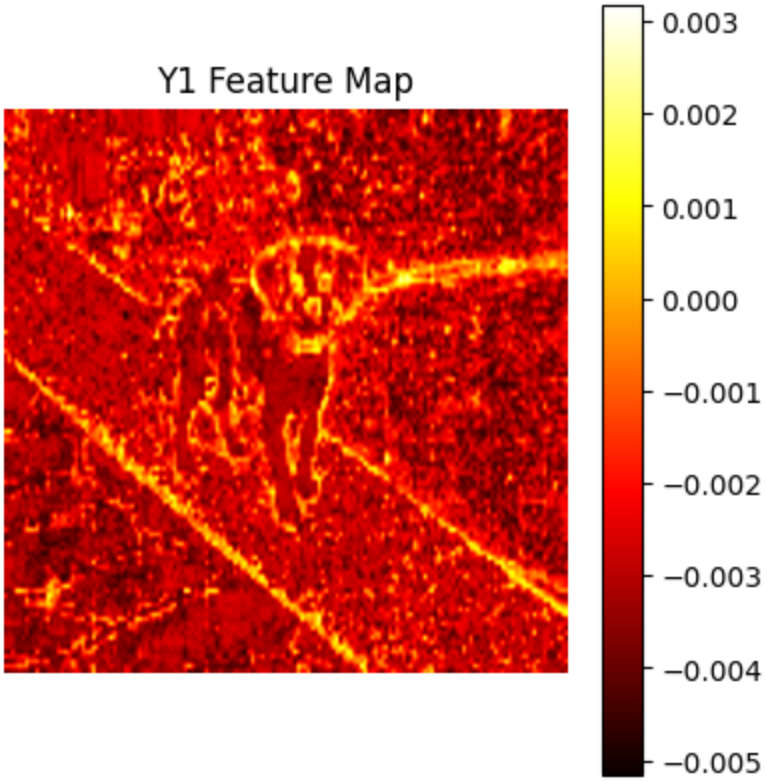


Original Image



Success Segmentation
"A puppy stands near a pool, looking at the camera."

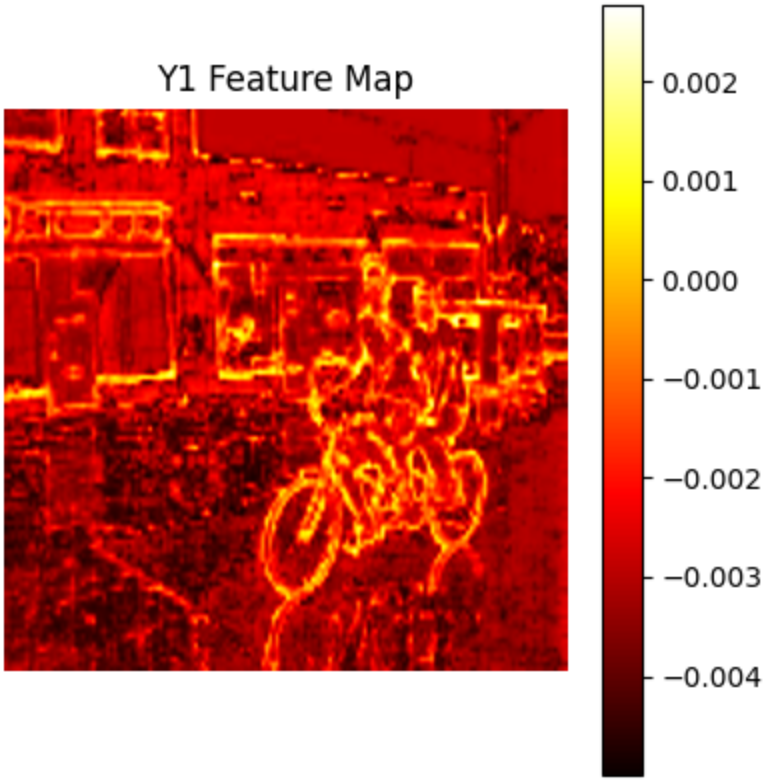
Success Segmentation Overlay
"A puppy stands near a pool, looking at the camera."



Original Image

Success Segmentation
"Two men ride a tandem bike, one smiling and waving."

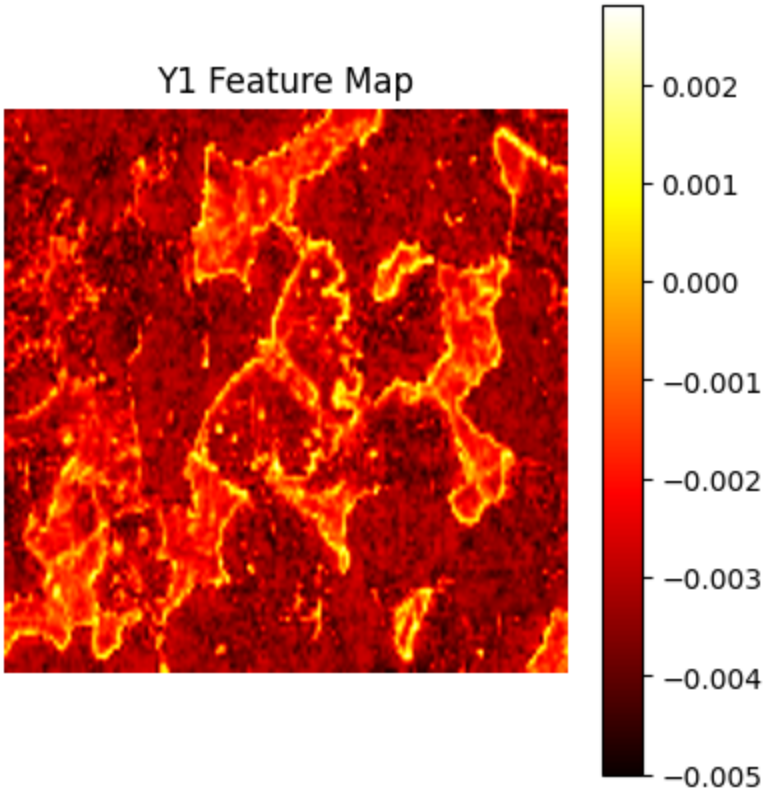
Success Segmentation Overlay
"Two men ride a tandem bike, one smiling and waving."



Original Image

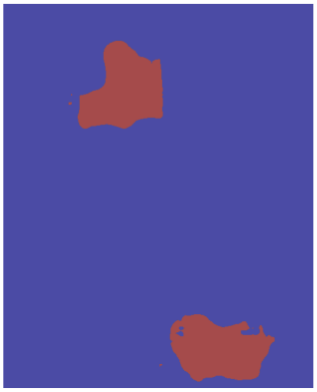
Success Segmentation
"A brown butterfly rests on green leaves."

Success Segmentation Overlay
"A brown butterfly rests on green leaves."

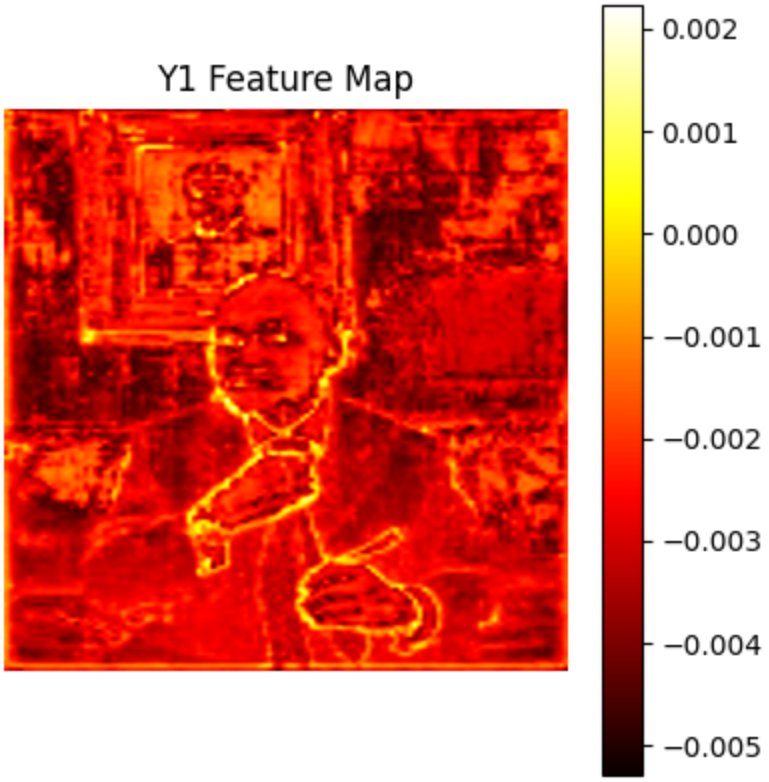


Original Image

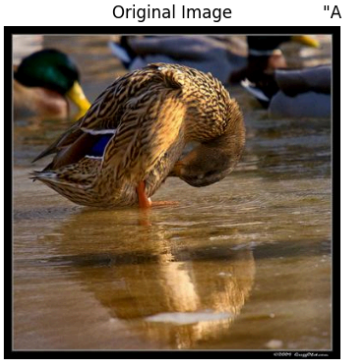
Success Segmentation
"A framed portrait of a man in a suit hangs on a wooden wall."



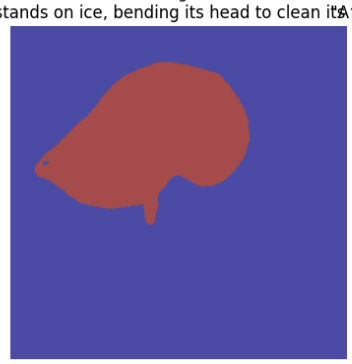
Success Segmentation Overlay
"A framed portrait of a man in a suit hangs on a wooden wall."



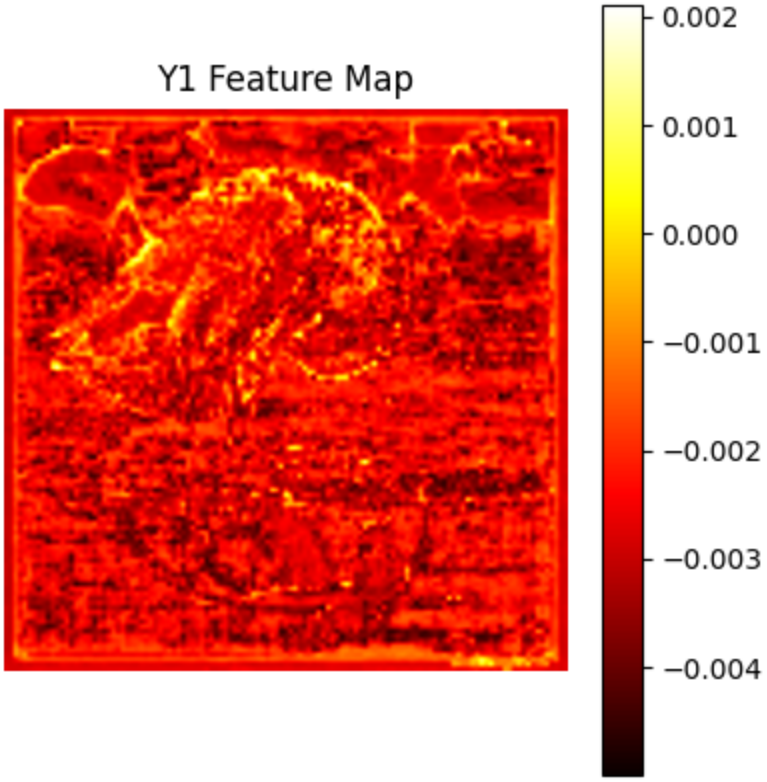
Original Image



Success Segmentation
"A duck stands on ice, bending its head to clean its feathers."

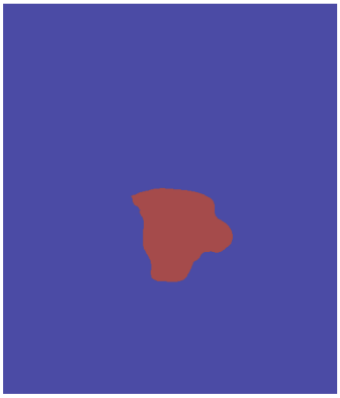


Success Segmentation Overlay
"A duck stands on ice, bending its head to clean its feathers."


Y1 Feature Map
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004

Original Image

Success Segmentation
"Espresso pours into two cups from a coffee machine."



Success Segmentation Overlay
"Espresso pours into two cups from a coffee machine."

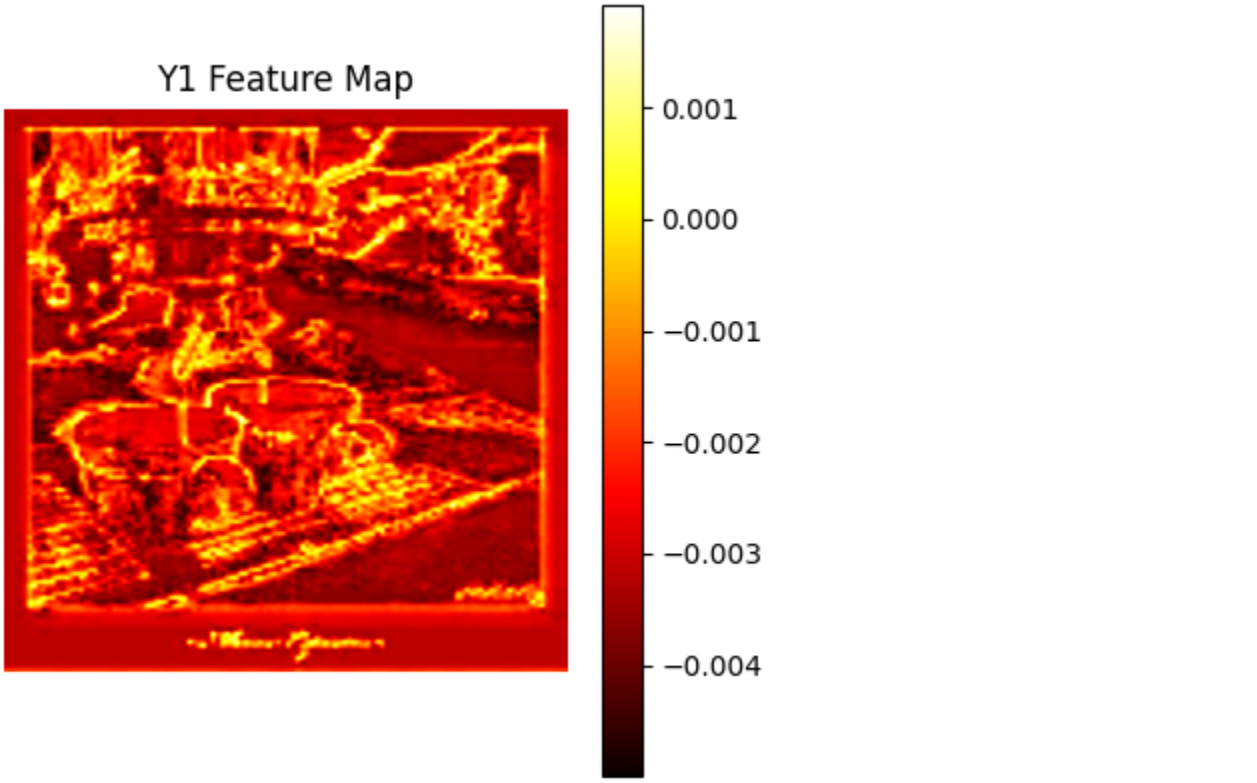


Image	Reference Text	Failure Observed
00000003.jpg	A well-groomed dog walks proudly at a dog show.	None. The segmentation captures the full shape and pose of the dog accurately.
00000004.jpg	A fluffy dog runs through the grass on a sunny day.	The segmentation under-segments the dog, especially missing parts of its legs and body.
00000018.jpg	Four kids sit by the pool enjoying popsicles together.	Only one child is segmented; the others are completely missed.
00000019.jpg	A small bird with black and white feathers sits on a plant.	Only the bird's head is partially segmented; the body is missed. Extra regions in the background are falsely segmented.
00000022.jpg	A puppy stands near a pool, looking at the camera.	Segmentation is generally good but slightly cuts off the puppy’s tail and parts of legs.
00000023.jpg	Two men ride a tandem bike, one smiling and waving.	Segmentation isolates only one person, missing the other and the bicycle completely.
00000025.jpg	A brown butterfly rests on green leaves.	Good segmentation. The butterfly is captured clearly with accurate shape.
00000026.jpg	A framed portrait of a man in a suit hangs on a wooden wall.	Fails completely — the segmentation targets the man sitting below, not the portrait.
00000030.jpg	A duck stands on ice, bending its head to clean its feathers.	The overall shape is captured, but the duck’s bent neck and head details are lost.

00000034.jpg	Espresso pours into two cups from a coffee machine.	Segmentation roughly outlines the cups but merges them together and misses handles or pouring detail.

5. 1.

Setup and Import Dependencies

Generate | + Code | + Markdown

```python
import os
import sys
import torch
import numpy as np
import matplotlib.pyplot as plt
import cv2
from PIL import Image

print(f"CUDA available: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"CUDA device: {torch.cuda.get_device_name(0)}")

# Add Matcher directory to path
matcher_path = os.path.join(os.getcwd(), "Matcher")
if matcher_path not in sys.path:
    sys.path.append(matcher_path)
    print(f"Added {matcher_path} to sys.path")

# Add segment_anything to path
sam_path = os.path.join(os.getcwd(), "Matcher", "segment_anything")
if sam_path not in sys.path:
    sys.path.append(sam_path)
    print(f"Added {sam_path} to sys.path")

# Add utils to path
utils_path = os.path.join(os.getcwd(), "Matcher", "utils")
if utils_path not in sys.path:
    sys.path.append(utils_path)
    print(f"Added {utils_path} to sys.path")
```

[2] ✓ 3.9s

```
CUDA available: False
Added c:\Users\vikra\OneDrive\Desktop\CSE344-CV\A3\Q5\Matcher to sys.path
Added c:\Users\vikra\OneDrive\Desktop\CSE344-CV\A3\Q5\Matcher\segment_anything to sys.path
Added c:\Users\vikra\OneDrive\Desktop\CSE344-CV\A3\Q5\Matcher\utils to sys.path
```

```python
# Load DINOv2 model
print("Loading DINOv2 model...")
dinov2_vitl14 = torch.hub.load('facebookresearch/dinov2', 'dinov2_vitl14')
dinov2_vitl14 = dinov2_vitl14.eval()
if torch.cuda.is_available():
    dinov2_vitl14 = dinov2_vitl14.cuda()
print("DINOv2 model loaded successfully!")
```

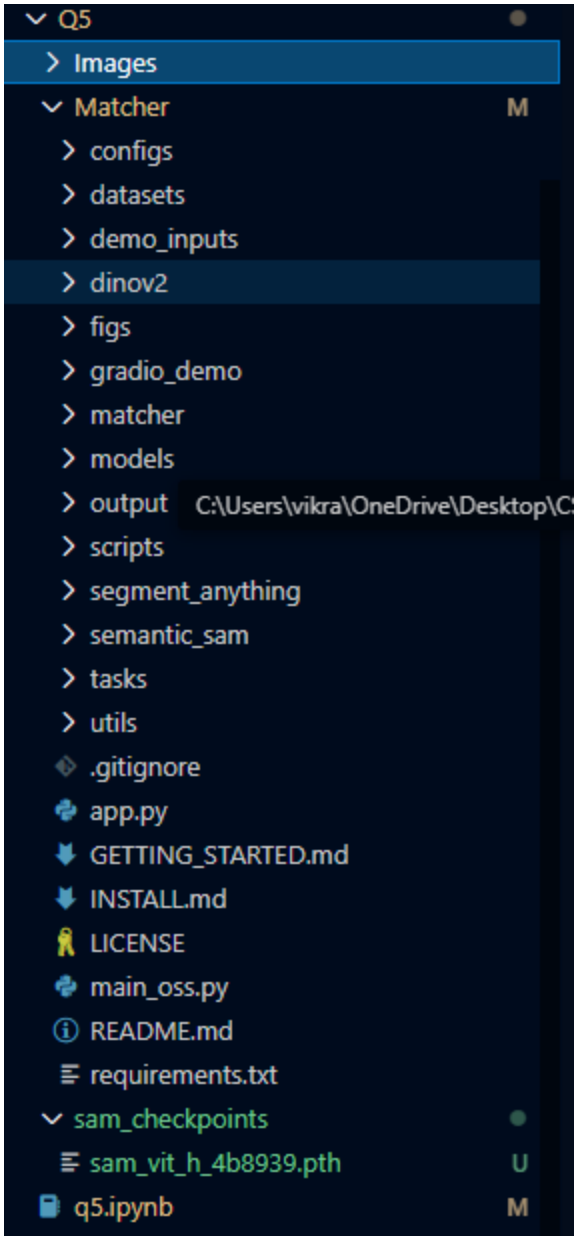
✓ 7.8s

```
Loading DINOv2 model...
Using cache found in C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\swiglu_ffn.py:51: UserWarning: xFormers is not available (SwiGLU)
  warnings.warn("xFormers is not available (SwiGLU)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\attention.py:33: UserWarning: xFormers is not available (Attention)
  warnings.warn("xFormers is not available (Attention)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\block.py:40: UserWarning: xFormers is not available (Block)
  warnings.warn("xFormers is not available (Block)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\swiglu_ffn.py:51: UserWarning: xFormers is not available (SwiGLU)
  warnings.warn("xFormers is not available (SwiGLU)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\attention.py:33: UserWarning: xFormers is not available (Attention)
  warnings.warn("xFormers is not available (Attention)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\block.py:40: UserWarning: xFormers is not available (Block)
  warnings.warn("xFormers is not available (Block)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\swiglu_ffn.py:51: UserWarning: xFormers is not available (SwiGLU)
  warnings.warn("xFormers is not available (SwiGLU)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\attention.py:33: UserWarning: xFormers is not available (Attention)
  warnings.warn("xFormers is not available (Attention)")
C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main\dinov2\layers\block.py:40: UserWarning: xFormers is not available (Block)
  warnings.warn("xFormers is not available (Block)")
DINOv2 model loaded successfully!
```

```python
# Download SAM model if not already available
import requests

# Create directory for SAM checkpoint
sam_checkpoint_dir = "sam_checkpoints"
os.makedirs(sam_checkpoint_dir, exist_ok=True)

# Download SAM model (vit_h)
sam_url = "https://dl.fbaipublicfiles.com/segment_anything/sam_vit_h_4b8939.pth"
sam_path = os.path.join(sam_checkpoint_dir, "sam_vit_h_4b8939.pth")

if not os.path.exists(sam_path):
    print(f"Downloading SAM checkpoint to {sam_path}...")
    with requests.get(sam_url, stream=True) as r:
        r.raise_for_status()
        with open(sam_path, 'wb') as f:
            for chunk in r.iter_content(chunk_size=8192):
                f.write(chunk)
    print("SAM checkpoint downloaded.")
else:
    print("SAM checkpoint already exists.")
```

✓ 0.6s

```
SAM checkpoint already exists.
```


Q5
Images
Matcher
M
configs
datasets
demo_inputs
dinov2
figs
gradio_demo
matcher
models
output
C:\Users\vikra\OneDrive\Desktop\C
scripts
segment_anything
semantic_sam
tasks
utils
.gitignore
app.py
GETTING_STARTED.md
INSTALL.md
LICENSE
main_oss.py
README.md
requirements.txt
sam_checkpoints
sam_vit_h_4b8939.pth
U
q5.ipynb
M

```python
# Initialize models
print("Initializing SAM model...")
sam = get_sam_model(checkpoint=sam_path)
print("SAM model initialized!")

print("\nInitializing Feature Matching model...")
matching_model = get_feature_match_model()
print("Feature Matching model initialized!")
```

9] ✓ 16.7s

```
Using cache found in C:\Users\vikra/.cache\torch\hub\facebookresearch_dinov2_main
Initializing SAM model...
SAM model initialized!

Initializing Feature Matching model...
Feature Matching model initialized!
SAM model initialized!

Initializing Feature Matching model...
Feature Matching model initialized!
```

Generate + Code