1. CLIP (Contrastive Language-Image Pretraining)

Overview

The task involves performing a detailed study and comparative analysis between CLIP (Contrastive Language-Image Pretraining) and CLIPS (an enhanced CLIP framework) on a given image of a human and a dog.

---

1. Installation of CLIP Dependencies

This step installs all the necessary dependencies to run CLIP using the transformers, torch libraries. The package installation is executed silently without printing unnecessary output

```
!pip install transformers torch torchvision  --quiet
```

2. Download CLIP Model and Load Weights

Here, the code imports the `CLIPProcessor` and `CLIPModel` from the `transformers` library and loads the pre-trained weights for the `clip-vit-base-patch32` model. This model is capable of processing both images and text for similarity tasks.

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

3. Generate CLIP Similarity Scores

This step involves loading an image and a set of 10 textual descriptions. After preprocessing the image and text using the `CLIPProcessor`, the model generates

embeddings for both the image and text. The cosine similarity is computed between the image and text embeddings.

```
A man holding a dog --> Similarity Score: 30.5882
A woman holding a dog --> Similarity Score: 26.2149
A man holding an elephant --> Similarity Score: 26.6754
A woman holding an elephant --> Similarity Score: 21.1037
A man cooking in a kitchen --> Similarity Score: 21.0087
A beautiful sunset --> Similarity Score: 16.3746
A person sitting on a bench --> Similarity Score: 20.0851
A dog and its owner --> Similarity Score: 27.6654
A city skyline at night --> Similarity Score: 12.8141
A Room full of toys --> Similarity Score: 19.5991
A dog looking at a human --> Similarity Score: 25.2832
```

4. CLIPS Dependency Installation

This step installs all the necessary dependencies to run CLIP using the transformers, torch and open_clip libraries. The package installation is executed silently without printing unnecessary output

5. Download CLIPS Model and Load Weights

After installing the required dependencies, this step loads the CLIPS model and tokenizer. It uses `create_model_from_pretrained` to fetch the pre-trained weights for the CLIPS model, specifically the ViT-L-14-CLIPS-224.

```
# Load model
model, preprocess = create_model_from_pretrained("hf-hub:UCSC-VLAA/ViT-L-14-CLIPS-224-Recap-DataComp-1B")
tokenizer = get_tokenizer("hf-hub:UCSC-VLAA/ViT-L-14-CLIPS-224-Recap-DataComp-1B")
```

6. Generate CLIPS Similarity Scores

Similarly, this part processes the same image and text using the CLIPS model, calculating the cosine similarity between image and text embeddings.

```
A man holding a dog --> Similarity Score: 16.3031
A woman holding a dog --> Similarity Score: 10.1649
A man holding an elephant --> Similarity Score: 16.4388
A woman holding an elephant --> Similarity Score: 10.8382
A man cooking in a kitchen --> Similarity Score: 6.4267
A beautiful sunset --> Similarity Score: 3.7355
A person sitting on a bench --> Similarity Score: -0.2385
A dog and its owner --> Similarity Score: 15.3522
A city skyline at night --> Similarity Score: -0.2308
A Room full of toys --> Similarity Score: 6.8924
A dog looking at a human --> Similarity Score: 12.7015
```

7. Result Analysis

Upon evaluating the similarity scores for a sample image of a human and a dog across 11 textual descriptions, we observe notable differences in how CLIP and CLIPS interpret and score the semantic alignment between text and image:

- **CLIP** consistently yields **higher similarity scores** across all captions, especially for relevant descriptions like “A man holding a dog” (30.59) and “A dog and its owner” (27.67). This indicates that CLIP is strongly attuned to visual-textual alignment for real-world image-text pairs, likely due to its training on vast amounts of natural image-caption data from the internet.

- **CLIPS**, although an enhanced version of CLIP designed for robustness using synthetic captions, produces **lower and more conservative similarity scores**. While it still identifies relevant captions such as “A man holding an elephant” and “A dog and its owner” as more similar, the scores are generally much smaller (e.g., “A man holding a dog” scores 16.30).

- Some descriptions even receive **negative or near-zero scores** in CLIPS (e.g., “A person sitting on a bench”: -0.24), suggesting that CLIPS may enforce stricter or more discriminative matching, possibly due to its enhanced synthetic training regime that emphasizes precision over recall.

- Overall, **CLIP appears more lenient and broadly associative**, making it useful for general-purpose zero-shot recognition tasks. In contrast, **CLIPS**

demonstrates more selective and cautious behavior, which might make it better suited for tasks requiring fine-grained or accurate semantic matching.

In conclusion, both models show the ability to identify semantically relevant captions, but **CLIP offers broader matching**, while **CLIPS leans toward higher precision and robustness**, potentially sacrificing recall in the process.

2. Visual Question Answering using BLIP

Overview

Visual Question Answering (VQA) is a multi-modal AI task where a system answers natural language questions about visual content, such as images. It combines image understanding and natural language processing, enabling models to produce coherent textual answers for visual queries.

---

1. BLIP: Bootstrapping Language-Image Pre-training

Refer to the research paper and GitHub repository of **BLIP: Bootstrapping Language-Image Pre-training** (Salesforce/BLIP GitHub).

- All dependencies were installed using pip.

```
from PIL import Image
import requests
import torch
from transformers import BlipProcessor, BlipForQuestionAnswering
```

- The pretrained model `Salesforce/blip-vqa-base` was downloaded via Hugging Face's `transformers` library.

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

- The model is designed specifically for VQA tasks and fine-tuned to generate natural language answers based on a given image and question.

2 – 3 Generating Visual Answers

Using the **sample image of a man holding a dog**, the model was prompted with the following two questions:

Q1: "Where is the dog present in the image?"

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

Q2: "Where is the man present in the image?"

- **Answer:** *"living room"*

4. Output Analysis

Both responses generated by the BLIP VQA model are **accurate and contextually appropriate**:

- The answer "in man's arms" correctly captures the **spatial relationship** and **interaction** between the man and the dog, reflecting a nuanced understanding of the image.

- The answer "living room" for the man's location correctly identifies the **environment** or **setting**, demonstrating the model's ability to infer background elements, not just foreground subjects.

These outputs showcase BLIP's ability to interpret complex visual scenes and deliver natural, concise responses that are grounded in both the visual and linguistic context.

3. BLIP vs CLIP

Overview

This task involves a comparative analysis between **BLIP** and **CLIP** (including **CLIPS**) in the context of **image captioning** and **semantic alignment evaluation**. Used a set of sample images to generate captions with BLIP and measured the alignment accuracy using CLIP-based similarity scores.

1. Load Pretrained BLIP for Image Captioning

Utilized the model `Salesforce/blip-image-captioning-base`, a transformer-based architecture designed for generating high-quality natural language captions based on visual input. This model is pretrained on large-scale datasets and fine-tuned specifically for image-to-text generation tasks.

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

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

2. Generate Captions for Each Image

A folder of 10 images and generated one caption per image using the BLIP model. Below are a few sample outputs:

```
{'/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000004.jpg': 'a small dog running across a green field',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000022.jpg': 'a small dog standing on a stone ledge',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000023.jpg': 'a man riding a bike down a wet street',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000026.jpg': 'a man in a suit and tie sitting on a couch',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000018.jpg': 'a family sitting in a pool with a towel',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000003.jpg': 'a small dog walking on a green carpet',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000019.jpg': 'a small bird sitting on a plant',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000030.jpg': 'a duck drinking water from a pond',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000034.jpg': 'a coffee machine with two cups on it',
 '/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000025.jpg': 'a brown butterfly sitting on a green plant'}
```

The captions were detailed, context-aware, and relevant to the image content, showcasing the strong generative ability of the BLIP captioning model.

3. CLIP-Based Evaluation of Caption Accuracy

Using `openai/clip-vit-base-patch32` evaluated the semantic similarity between the BLIP-generated caption and the corresponding image using **cosine similarity**. Sample similarity scores:

```
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000004.jpg | CLIP Similarity: 32.7026
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000022.jpg | CLIP Similarity: 31.0347
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000023.jpg | CLIP Similarity: 30.8345
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000026.jpg | CLIP Similarity: 28.8953
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000018.jpg | CLIP Similarity: 31.3365
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000003.jpg | CLIP Similarity: 31.5660
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000019.jpg | CLIP Similarity: 28.9383
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000030.jpg | CLIP Similarity: 30.5252
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000034.jpg | CLIP Similarity: 27.9613
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000025.jpg | CLIP Similarity: 28.9163
```

4. CLIPS-Based Evaluation of Caption Accuracy

Evaluation Using CLIPS (a fine-tuned version of CLIP with enhanced grounding). Sample results:

```
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000004.jpg | CLIPS Similarity: 19.1549
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000022.jpg | CLIPS Similarity: 15.1255
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000023.jpg | CLIPS Similarity: 16.9236
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000026.jpg | CLIPS Similarity: 12.7573
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000018.jpg | CLIPS Similarity: 14.8474
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000003.jpg | CLIPS Similarity: 18.1808
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000019.jpg | CLIPS Similarity: 16.9768
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000030.jpg | CLIPS Similarity: 16.5172
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000034.jpg | CLIPS Similarity: 15.6065
/kaggle/input/cv-assgn3/samples/ILSVRC2012_test_00000025.jpg | CLIPS Similarity: 16.7442
```

5. Alignment Metrics

While cosine similarity is the most direct metric, several other evaluation strategies can help assess alignment between visual and textual modalities:

Metric	Description	When to Use
Cosine Similarity	Measures angle similarity between embeddings.	Quick estimation of semantic match. Used in both CLIP and CLIPS.
BLEU / METEOR / ROUGE	Compare n-gram overlap between generated and reference captions.	When you have ground truth captions for comparison.
CIDEr / SPICE	Focus on content words and semantic correctness.	Best for evaluating human-likeness and relevance in image captioning.
Human Evaluation	Manual review for fluency and correctness.	Ideal when dealing with subjective or ambiguous visuals.
Visual Grounding Accuracy	How well the caption corresponds to actual localized image regions.	When captions need to refer to specific objects or parts in the image.

Example:

- *Cosine similarity* is ideal when evaluating how well a generated caption semantically aligns with the image in vector space.
- *BLEU* would be appropriate when benchmarking BLIP captions against a gold-standard dataset like MS COCO.
- *Visual grounding* is useful in sensitive applications like autonomous driving or medical imaging where exact object reference is critical

4. Referring Image Segmentation (RIS)

Overview

Referring Image Segmentation (RIS) is a multimodal task where a model segments a region of interest in an image based on a natural language description, e.g., *"the cat sitting on the sofa"*. The LAVT (Language-Aware Vision Transformer) model enables this by fusing image and text features using advanced transformer architectures.

---

1. Setup & Pre-trained Weights

- Cloned the official LAVT-RIS repository.
- Updated LAVT to return Y1 feature during forward Pass
- Installed required dependencies as specified in the README.

```
!pip install h5py
!pip install timm
!pip install mmcv==1.3.12
!pip install mmsegmentation==0.17.0
```

- Downloaded the following pre-trained models:
    - LAVT checkpoint for RefCOCO.
    - Swin Transformer backbone: `swin_base_patch4_window12_384_22k.pth`.

2. Segmentation with Provided References

- Loaded 10 sample images.
- Used the provided natural language references for each image.
- Performed segmentation using the LAVT model.

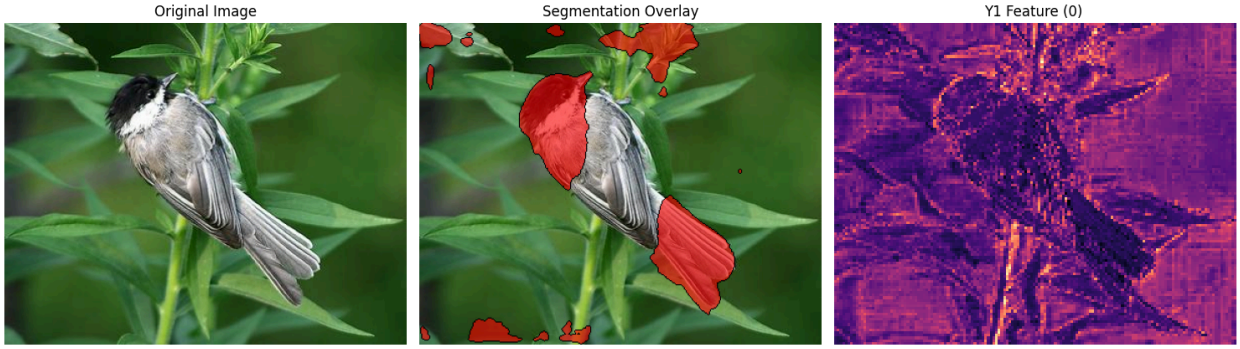
3. Y1 Feature Map

- Modified the model architecture to return `x_c1` as Y1.
- Extracted and plotted the first channel of the Y1 feature map.
- Y1 maps offer insight into early-stage spatial attention.

RESULTS :

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

Original Image | Segmentation Overlay | Y1 Feature (0)



ILSVRC2012_test_00000022.jpg | Reference: The sad dog standing beside the pool

Original Image | Segmentation Overlay | Y1 Feature (0)

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

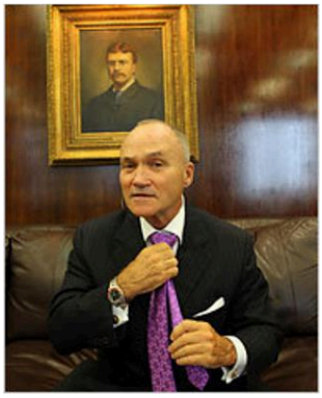
Original Image | Segmentation Overlay | Y1 Feature (0)

ILSVRC2012_test_00000025.jpg | Reference: The butterfly in the picture

Original Image | Segmentation Overlay | Y1 Feature (0)

ILSVRC2012_test_00000026.jpg | Reference: The mang wearing a suite and tie

Original Image

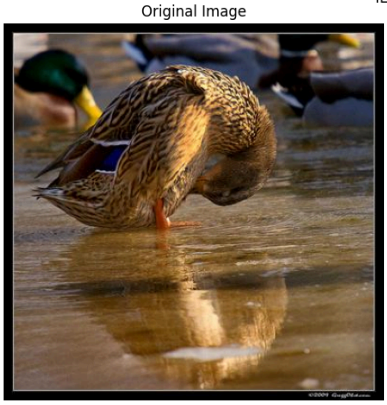


Segmentation Overlay

Y1 Feature (0)

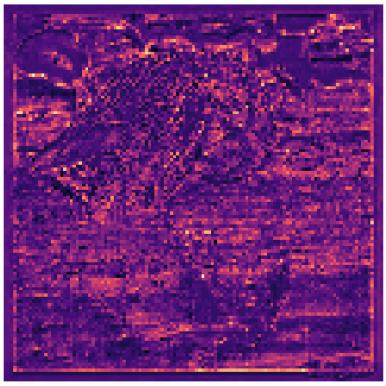
ILSVRC2012_test_00000030.jpg | Reference: The duck in the picture

Original Image



Segmentation Overlay

Y1 Feature (0)

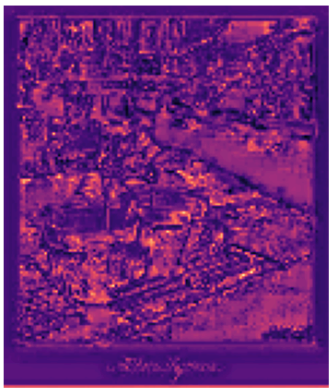


ILSVRC2012_test_00000034.jpg | Reference: The white coffee cups on the coffee machine

Original Image

Segmentation Overlay

Y1 Feature (0)

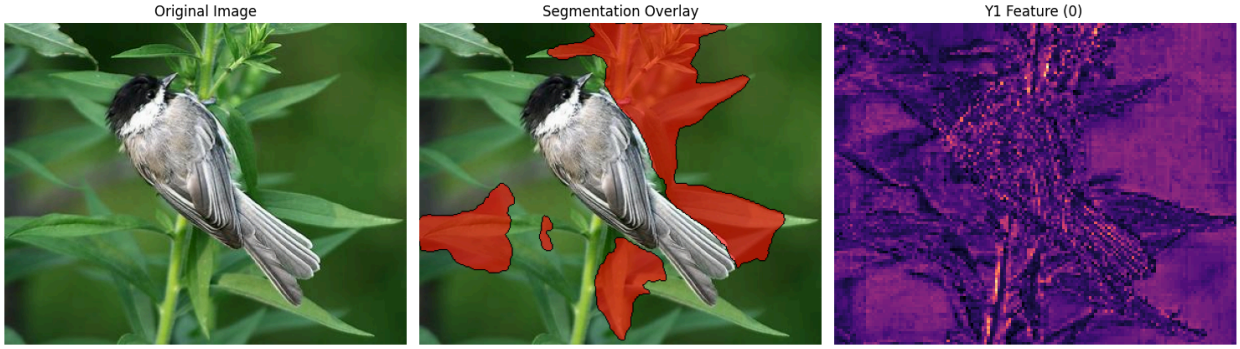


4: Custom Reference & Failure Cases

- Created misleading or unrelated natural language descriptions for each image.
- Ran inference again and observed degraded or failed segmentation results.

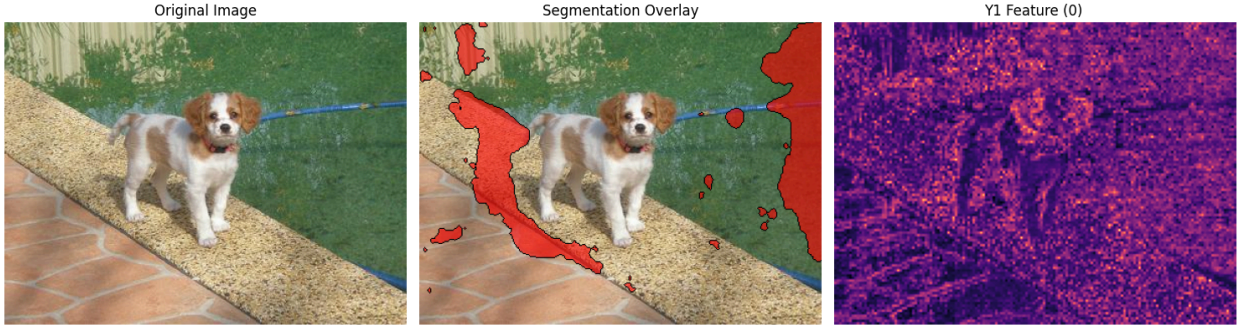
ILSVRC2012_test_00000019.jpg | Reference: A bowl of fresh fruit on a table

Original Image | Segmentation Overlay | Y1 Feature (0)



ILSVRC2012_test_00000022.jpg | Reference: A computer keyboard with colored lights

Original Image | Segmentation Overlay | Y1 Feature (0)

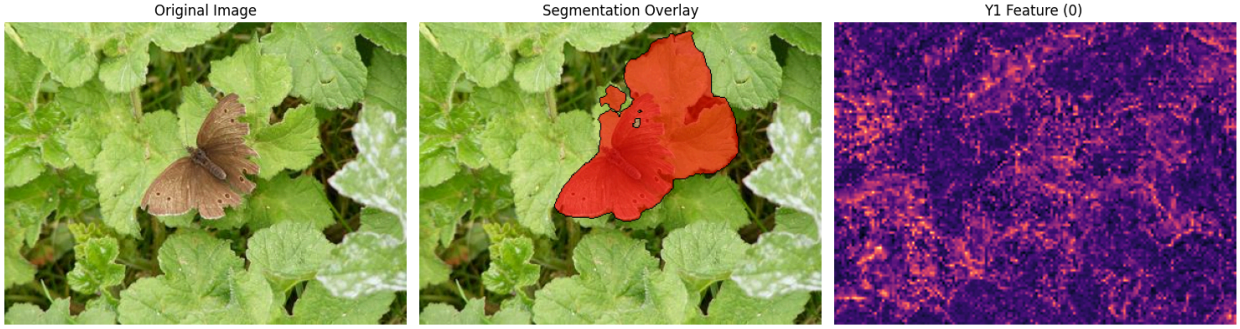


ILSVRC2012_test_00000023.jpg | Reference: A fish swimming in the ocean

Original Image | Segmentation Overlay | Y1 Feature (0)

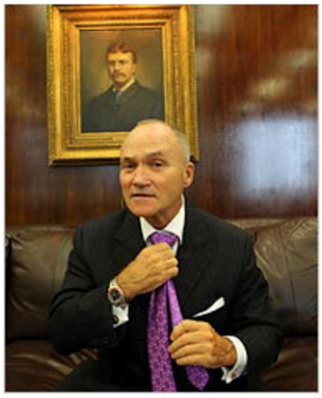
ILSVRC2012_test_00000025.jpg | Reference: A red sports car on a racetrack

Original Image | Segmentation Overlay | Y1 Feature (0)



ILSVRC2012_test_00000026.jpg | Reference: A plate of pancakes with syrup

Original Image

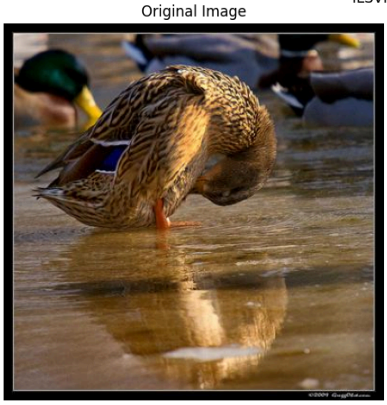


Segmentation Overlay

Y1 Feature (0)

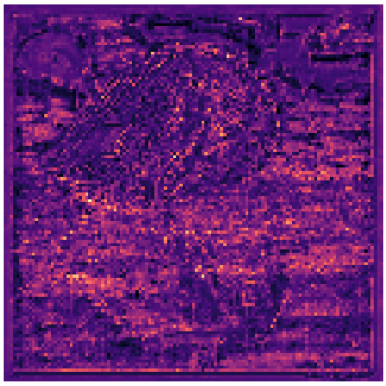
ILSVRC2012_test_00000030.jpg | Reference: A rocket launching into space

Original Image



Segmentation Overlay

Y1 Feature (0)

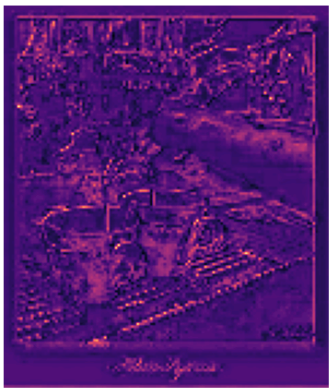


ILSVRC2012_test_00000034.jpg | Reference: A basketball player dunking

Original Image

Segmentation Overlay

Y1 Feature (0)



5. Image as reference Segmentation

Overview

In this task, we explore **Matcher**, a novel perception system that leverages vision foundation models to perform segmentation based on image references—**without any additional training**. This approach is called **one-shot segmentation**, where a single reference image is used to segment similar objects in a target image.

Matcher fuses features from **DINOv2** and **SAM** to generate accurate, training-free segmentations, allowing it to generalize to unseen objects and domains.

1: Setup & Pretrained Weights

- Cloned the Matcher repository: https://github.com/aim-uofa/Matcher
- Installed all dependencies:
  - `detectron2`, `POT`, and Matcher's requirements.
- Downloaded pretrained models:
  - `dinov2_vitl14_pretrain.pth`
  - `sam_vit_h_4b8939.pth`
  - `swint_only_sam_many2many.pth`
- Verified that Matcher loads and runs correctly on a CUDA-enabled device.

```
[ ]:
!git clone https://github.com/facebookresearch/detectron2.git
!pip install -e detectron2

!git clone https://github.com/aim-uofa/Matcher.git
```

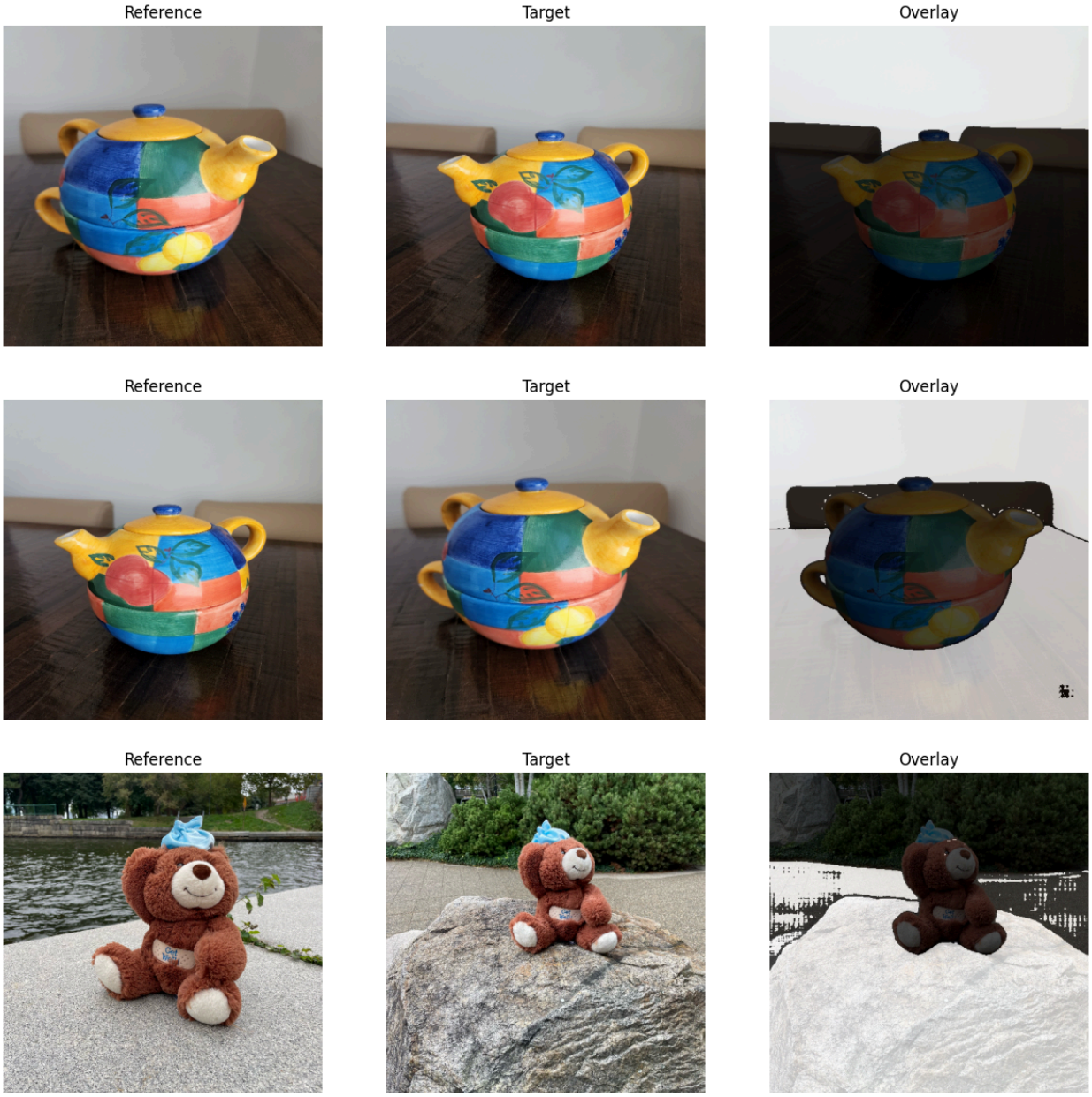
```
[ ]:
%cd Matcher
!pip install -r requirements.txt
!pip install POT
```

```
[ ]:
# .models directory exists
!mkdir -p models

# Download the file into the models directory
!wget -P models https://dl.fbaipublicfiles.com/dinov2/dinov2_vitl14/dinov2_vitl14_pretrain.pth
!wget -P models https://dl.fbaipublicfiles.com/segment_anything/sam_vit_h_4b8939.pth
!wget -P models https://github.com/UX-Decoder/Semantic-SAM/releases/download/checkpoint/swint_only_sam_many2many.pth
```

2: One-Shot Segmentation with Reference Image

- **Dataset**: Each folder contains 2 similar images.
- For each folder:
    1. Use **Image A as reference**, segment **Image B**.
    2. Use **Image B as reference**, segment **Image A**.
- The Matcher model uses an in-context image mask (simulated with a centered mask) to segment the target image based on visual similarity.



Reference

Target

Overlay

Reference

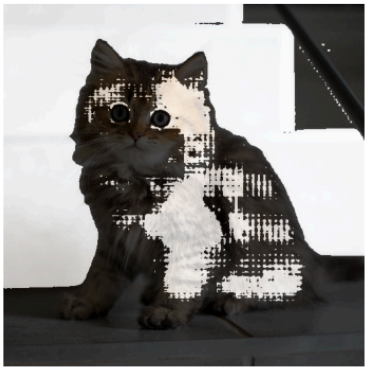
Target

Overlay

Reference

Target

Overlay



Reference

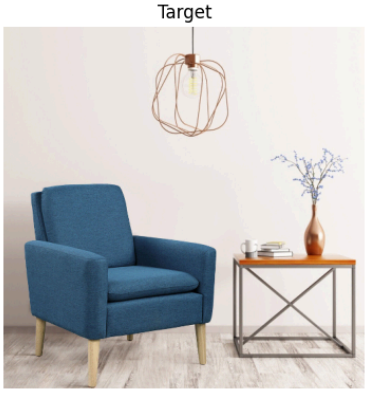
Target

Overlay

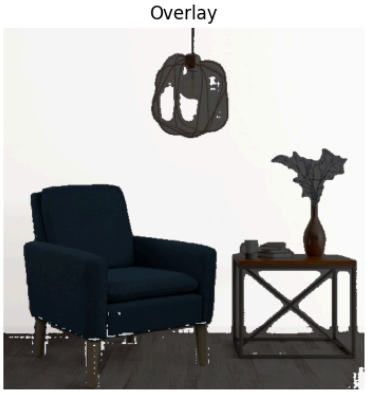
Reference	Target	Overlay
		
		
		
		

Reference

Target



Overlay



Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

Overlay

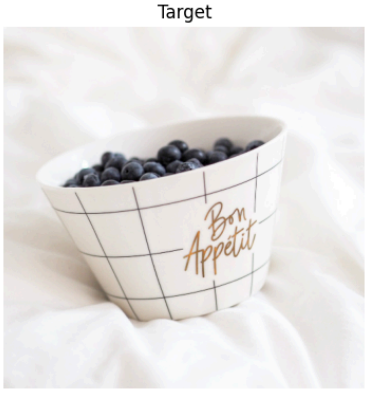
Reference

Target

Overlay

Reference

Target

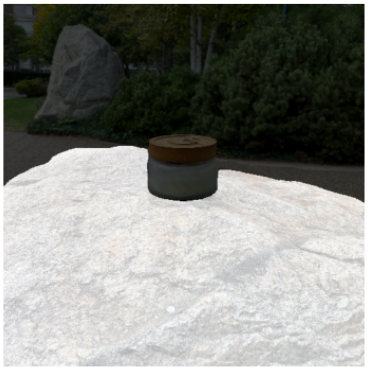


Overlay

Reference

Target

Overlay



Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

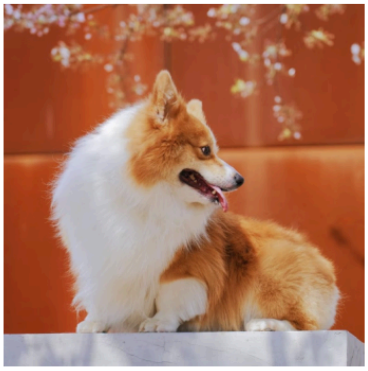
Overlay

Reference

Target

Overlay

Reference

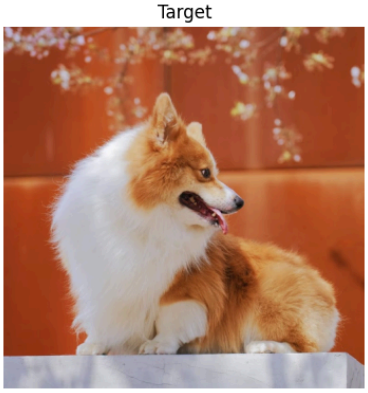


Target

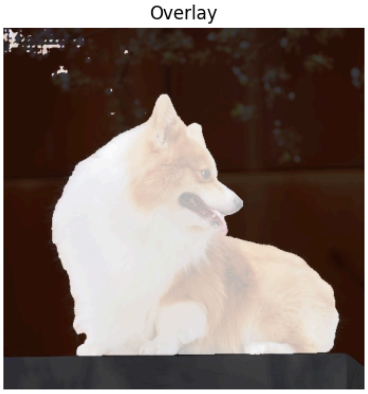
Overlay

Reference

Target



Overlay



Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

Overlay

Reference

Target

Overlay