

CSE 344: Computer Vision

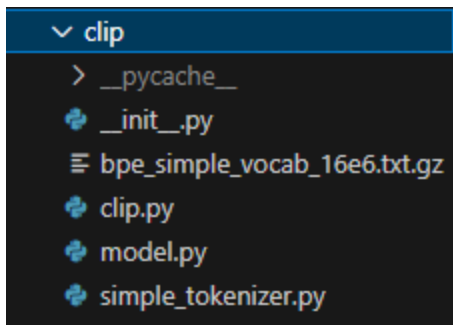
Assignment 3: Report

Aditya Aggarwal

2022028

1. CLIP (Contrastive Language-Image Pretraining)

1.1 Installation of CLIP



1.2 CLIP pretrained weights

```
model, preprocess = clip.load("ViT-B/32", device=device)
image = preprocess(Image.open(image_path)).unsqueeze(0).to(device)
text = clip.tokenize(descriptions).to(device)
```

1.3 Similarity Scores using CLIP

1. Score: 0.3352 | A fair bearded man standing in a room in white formal shirt, grey trousers and brown leather shoes holding a big grey dog in his arms.
2. Score: 0.3291 | A tall man standing in a room holding a big dog in his arms.
3. Score: 0.3145 | A man standing straight and holding a dog with both hands.
4. Score: 0.2947 | A big animal carried by a human being in a closed room.
5. Score: 0.2571 | Two living beings in front of a bookcase in a white room
6. Score: 0.2520 | A human showing empathy towards a fellow living creature by displaying care for it.

7. Score: 0.2179 | A dog in a lying position some distance above the ground.

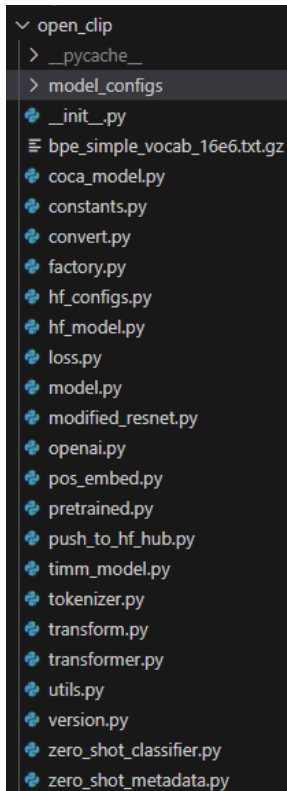
8. Score: 0.2083 | A book case kept in front of a white wall being blocked by two living entities.

9. Score: 0.1759 | A snake trying to attack an innocent child playing with her toys.

10. Score: 0.1409 | A black cat is playing with a young boy in a sunlit park, surrounded by green trees and a clear blue sky.

```
==== CLIP Ranking ====
1. Score: 0.3352 | A fair bearded man standing in a room in white formal shirt, grey trousers and brown leather shoes holding a big grey dog in his arms.
2. Score: 0.3291 | A tall man standing in a room holding a big dog in his arms.
3. Score: 0.3145 | A man standing straight and holding a dog with both hands.
4. Score: 0.2947 | A big animal carried by a human being in a closed room.
5. Score: 0.2571 | Two living beings in front of a bookcase in a white room
6. Score: 0.2520 | A human showing empathy towards a fellow living creature by displaying care for it.
7. Score: 0.2179 | A dog in a lying position some distance above the ground.
8. Score: 0.2083 | A book case kept in front of a white wall being blocked by two living entities.
9. Score: 0.1759 | A snake trying to attack an innocent child playing with her toys.
10. Score: 0.1409 | A black cat is playing with a young boy in a sunlit park, surrounded by green trees and a clear blue sky.
```

1.4 Installation of CLIPS



```
open_clip
├── __pycache__
├── model_configs
├── __init__.py
├── bpe_simple_vocab_16e6.txt.gz
├── coca_model.py
├── constants.py
├── convert.py
├── factory.py
├── hf_configs.py
├── hf_model.py
├── loss.py
├── model.py
├── modified_resnet.py
├── openai.py
├── pos_embed.py
├── pretrained.py
├── push_to_hf_hub.py
├── timm_model.py
├── tokenizer.py
├── transform.py
├── transformer.py
├── utils.py
├── version.py
├── zero_shot_classifier.py
└── zero_shot_metadata.py
```

1.5 CLIPS Pretrained Weights

```
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')
model = model.to(device)
```

1.6 Similarity Scores using CLIPS

1. Score: 0.2335 | A tall man standing in a room holding a big dog in his arms.
2. Score: 0.2299 | A fair bearded man standing in a room in white formal shirt, grey trousers and brown leather shoes holding a big grey dog in his arms.
3. Score: 0.1909 | A man standing straight and holding a dog with both hands.
4. Score: 0.1649 | A big animal carried by a human being in a closed room.
5. Score: 0.1282 | Two living beings in front of a bookcase in a white room
6. Score: 0.0919 | A human showing empathy towards a fellow living creature by displaying care for it.
7. Score: 0.0765 | A book case kept in front of a white wall being blocked by two living entities.
8. Score: 0.0625 | A dog in a lying position some distance above the ground.
9. Score: 0.0310 | A snake trying to attack an innocent child playing with her toys.
10. Score: -0.0459 | A black cat is playing with a young boy in a sunlit park, surrounded by green trees and a clear blue sky.

```
===== CLIPS Ranking =====
1. Score: 0.2335 | A tall man standing in a room holding a big dog in his arms.
2. Score: 0.2299 | A fair bearded man standing in a room in white formal shirt, grey trousers and brown leather shoes holding a big grey dog in his arms.
3. Score: 0.1909 | A man standing straight and holding a dog with both hands.
4. Score: 0.1649 | A big animal carried by a human being in a closed room.
5. Score: 0.1282 | Two living beings in front of a bookcase in a white room
6. Score: 0.0919 | A human showing empathy towards a fellow living creature by displaying care for it.
7. Score: 0.0765 | A book case kept in front of a white wall being blocked by two living entities.
8. Score: 0.0625 | A dog in a lying position some distance above the ground.
9. Score: 0.0310 | A snake trying to attack an innocent child playing with her toys.
10. Score: -0.0459 | A black cat is playing with a young boy in a sunlit park, surrounded by green trees and a clear blue sky.
```

1.7 CLIP vs CLIPS

Inference:

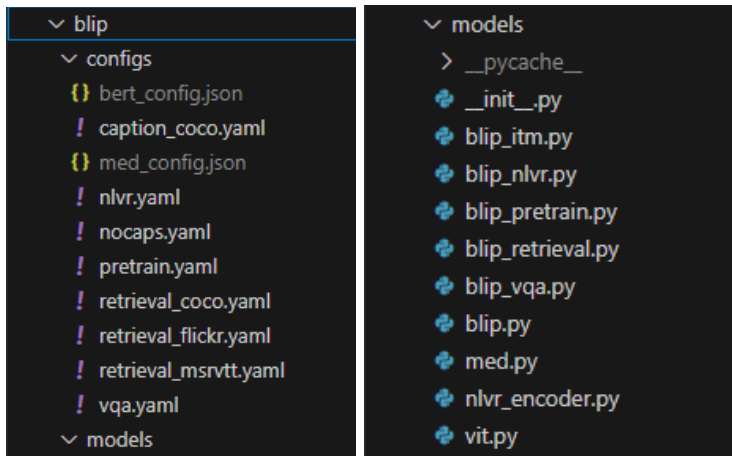
CLIP assigns higher similarity scores than CLIPS, showing stronger alignment with detailed captions. Both models correctly rank relevant descriptions highest and unrelated ones lowest, but CLIP prefers more specific, visually grounded sentences. CLIPS even gave negative scores to the last sentence, maybe due to opposite pairs in the sentence and the true textual description (dog ↔ cat, man ↔ boy).

Reasoning:

CLIP likely captures fine-grained visual-text features, making it better at matching detailed descriptions. CLIPS seems optimized for general semantic alignment, scoring simpler captions higher and penalizing overly specific or abstract ones. CLIPS also assigns negative score to contradictory sentences.

2. VQA (Visual Question Answering)

2.1 Installation



```
def blip_answer_question(image_path, question):
    device = "cuda" if torch.cuda.is_available() else "cpu"
    image_size = 480

    # Load image
    image = load_image(image_path=image_path, image_size=image_size, device=device)

    # Load BLIP VQA model
    model_url = 'https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model_base_vqa_capfilt_large.pth'
    model = blip_vqa(pretrained=model_url, image_size=image_size, vit='base')
    model.eval()
    model = model.to(device)

    # Inference
    with torch.no_grad():
        answer = model(image, question, train=False, inference='generate')

    return answer[0]
```

2.2 & 2.3 Question-Answering

```
load checkpoint from https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model_base_vqa_capfilt_large.pth
Q1: Where is the dog present in the image?
A1: in man's arms

load checkpoint from https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model_base_vqa_capfilt_large.pth
Q2: Where is the man present in the image?
A2: living room
```

2.4 Comments

The answers are accurate with respect to the questions that were asked associated with the sample image. However, the answers tend to be very short and not descriptive at all.

3. BLIP vs CLIP

3.1 Pretrained Weights - Image Captioning

```
def generate_caption(image_path):
    device = "cuda" if torch.cuda.is_available() else "cpu"
    image_size = 480

    # Load and preprocess image
    image = load_image(image_path=image_path, image_size=image_size, device=device)

    # Load BLIP captioning model
    model_url = 'https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model_base_capfilt_large.pth'
    model = blip_decoder(pretrained=model_url, image_size=image_size, vit='base')
    model.eval()
    model = model.to(device)

    # Generate caption
    with torch.no_grad():
        caption = model.generate(image, sample=True, top_p=0.9, max_length=20, min_length=5)

    return caption[0]
```

3.2, 3.3 & 3.4 Image Captioning and Similarity Scores

image	caption	clip_score	clips_score
ILSVRC2012_test_00000003.jpg	a dog walking across the show floor	0.3076	0.1885
ILSVRC2012_test_00000004.jpg	a dog in the grass	0.2849	0.1498
ILSVRC2012_test_00000018.jpg	three little kids relaxing in a pool	0.3035	0.1514
ILSVRC2012_test_00000019.jpg	a bird that is perched on a plant	0.2832	0.1575
ILSVRC2012_test_00000022.jpg	the dog	0.24	0.1109
ILSVRC2012_test_00000023.jpg	a man that is riding his bike	0.2747	0.149
ILSVRC2012_test_00000025.jpg	a butterfly on green leaves	0.2703	0.1666
ILSVRC2012_test_00000026.jpg	a man in a suit and tie on a brown couch	0.292	0.1386
ILSVRC2012_test_00000030.jpg	two different ducks walking on the water	0.2869	0.1225
ILSVRC2012_test_00000034.jpg	a coffee machine making two cups	0.291	0.148

3.5 Discussion on Evaluation Metrics

CLIP Cosine Similarity: This metric measures how closely the image and caption embeddings align in CLIP space. A high similarity score means the generated caption semantically matches the image well. It's useful when reference captions are unavailable.

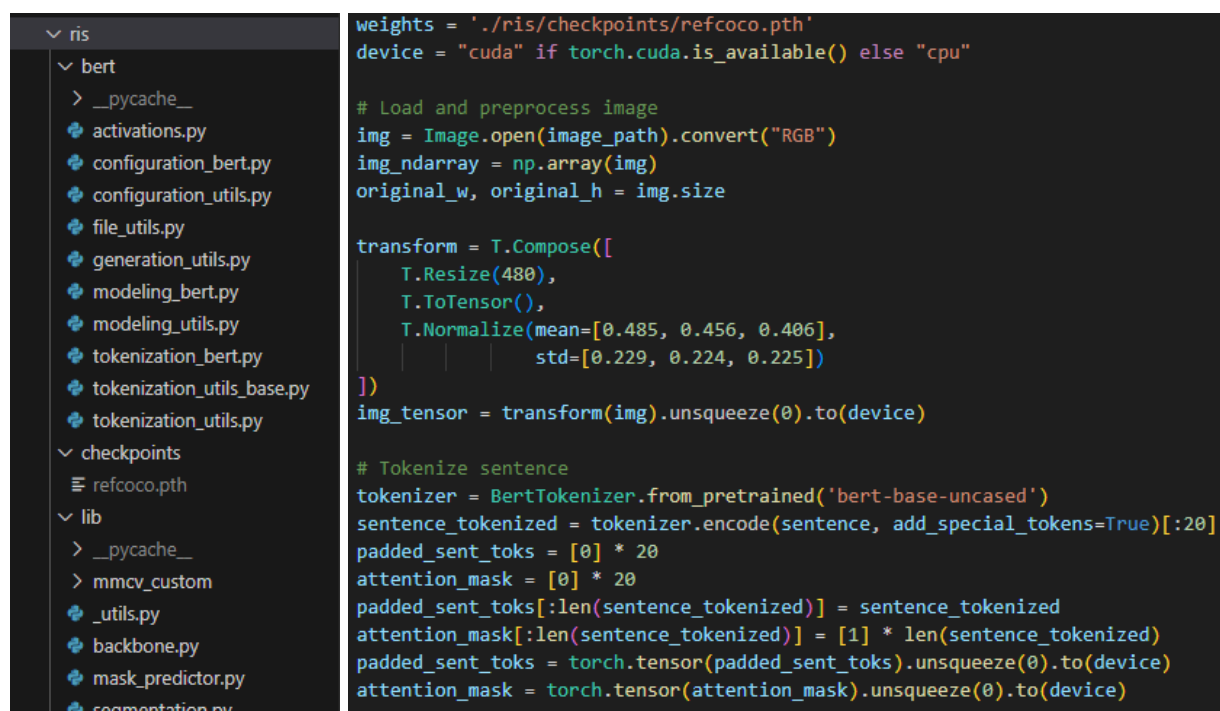
Text-Based Metrics (BLEU, ROUGE, METEOR, CIDEr): These compare BLIP-generated captions to ground-truth captions using word overlap or similarity. They're helpful for evaluating linguistic quality but may penalize correct captions that differ in phrasing.

CLIP-R (CLIP-Reward): CLIP-R is trained on human preferences to score captions based on how well they describe an image. It's ideal for ranking or refining generated captions when multiple candidates are available.

Human Evaluation: Manual review captures nuances like fluency, emotional tone, and contextual relevance. Though slower, it's the most reliable method for evaluating caption quality, especially for complex images.

4. RIS (Referring Image Segmentation)

4.1 Installation and Weights



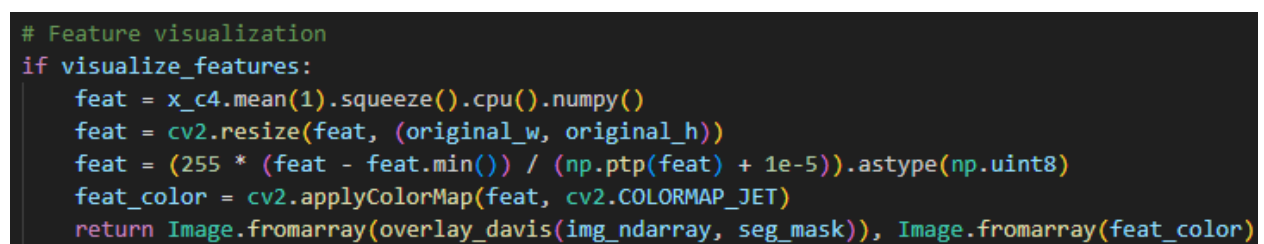
```
weights = './ris/checkpoints/refcoco.pth'
device = "cuda" if torch.cuda.is_available() else "cpu"

# Load and preprocess image
img = Image.open(image_path).convert("RGB")
img_ndarray = np.array(img)
original_w, original_h = img.size

transform = T.Compose([
    T.Resize(480),
    T.ToTensor(),
    T.Normalize(mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225])
])
img_tensor = transform(img).unsqueeze(0).to(device)

# Tokenize sentence
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
sentence_tokenized = tokenizer.encode(sentence, add_special_tokens=True)[:20]
padded_sent_toks = [0] * 20
attention_mask = [0] * 20
padded_sent_toks[:len(sentence_tokenized)] = sentence_tokenized
attention_mask[:len(sentence_tokenized)] = [1] * len(sentence_tokenized)
padded_sent_toks = torch.tensor(padded_sent_toks).unsqueeze(0).to(device)
attention_mask = torch.tensor(attention_mask).unsqueeze(0).to(device)
```

4.2 & 4.3 Segmentation and Feature Map Visualization



```
# Feature visualization
if visualize_features:
    feat = x_c4.mean(1).squeeze().cpu().numpy()
    feat = cv2.resize(feat, (original_w, original_h))
    feat = (255 * (feat - feat.min()) / (np.ptp(feat) + 1e-5)).astype(np.uint8)
    feat_color = cv2.applyColorMap(feat, cv2.COLORMAP_JET)
    return Image.fromarray(overlay_davis(img_ndarray, seg_mask)), Image.fromarray(feat_color)
```

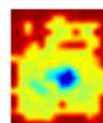
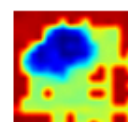
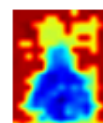
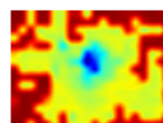
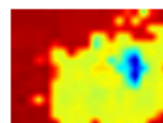
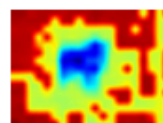
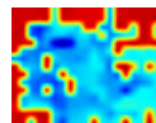
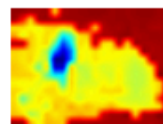
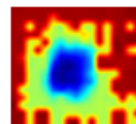
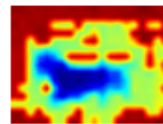
Original



Segmented



Feature Map



4.4 Failure Case of RIS

ILSVRC2012_test_00000003.jpg : "The tail of the dog"

ILSVRC2012_test_00000004.jpg : "The nose of the dog"

ILSVRC2012_test_00000018.jpg : "The shorts worn by the boy on left"

ILSVRC2012_test_00000019.jpg : "The white tail of the bird"

ILSVRC2012_test_00000022.jpg : "The four legs of the dog"

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

ILSVRC2012_test_00000025.jpg : "The right wing of the butterfly in the picture"

ILSVRC2012_test_00000026.jpg : "The pink tie"

ILSVRC2012_test_00000030.jpg : "The head of the duck in the picture"

ILSVRC2012_test_00000034.jpg : "The handle of the coffee cups on the coffee machine"

Reasoning for failure:

In all of these text descriptions, the instruction for segmentation was not for the entire entity but a part of the entity. This is a case where the RIS model fails to perform. In some cases, it included extra area than required (ILSVRC2012_test_00000003.jpg), in some cases it segmented a completely different area than expected (ILSVRC2012_test_00000019.jpg), and in some cases the image wasn't segmented at all (ILSVRC2012_test_00000025.jpg). It can be concluded that the RIS model struggles to segment a part of the entity in the image despite very clear and unambiguous textual description.

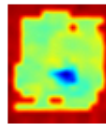
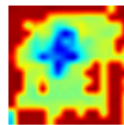
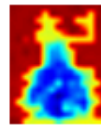
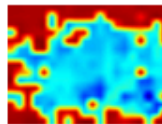
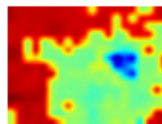
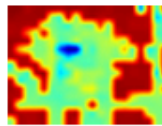
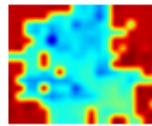
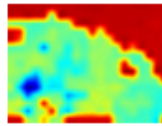
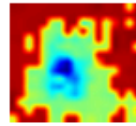
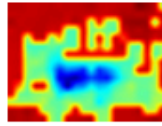
Original



Segmented



Feature Map



References

<https://github.com/UCSC-VLAA/CLIPS>

[GitHub - openai/CLIP: CLIP \(Contrastive Language-Image Pretraining\). Predict the most relevant text snippet given an image](#)

[GitHub - mlfoundations/open_clip: An open source implementation of CLIP.](#)

[GitHub - salesforce/BLIP: PyTorch code for BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation](#)

[GitHub - yz93/LAVT-RIS](#)