Negative Sampling Strategies for Composed Image Retrieval (CIR)

1. Strategies

Category-Discrepant Negatives

- Select negatives that are visually similar to the target image but differ in the key attribute specified by the modification text.

Attribute-Mismatch Negatives

- Choose negatives that match most attributes except the one(s) specified by the modification text.

Soft Negatives (Partial Attribute Match)

- Select negatives that match some, but not all, of the target attributes.

In-Batch Negatives

- Use other samples in the same batch as negatives (standard, but often too easy).

Semantic Negatives

- Pick negatives that are semantically close in the embedding space but do not satisfy the modification.

---

2. Where to Apply Negative Sampling

- **Triplet/Contrastive Loss Construction:**
  - When building (anchor, positive, negative) triplets for training, use the above strategies to select negatives.
- **Batch Formation:**
  - During each training batch, ensure a mix of easy (random) and hard (category/attribute-based) negatives.
- **Curriculum Learning:**
  - Start with easier negatives, then gradually introduce harder negatives as training progresses.

---

3. Recommended Strategies for This Project

- **Category-discrepant and attribute-mismatch negatives** are most effective, as they directly exploit the new metadata and attribute extraction.

- Use in-batch negatives as a supplement, but focus on the hard negatives for the biggest performance gains.

---

4. Impact on Output and Workflow

Aspect	Standard Negatives	Hard Negatives (Category/Attribute)
Training Speed	Faster	Slightly slower (more computation)
Model Discrimination	Coarse	Fine-grained, attribute-aware
Retrieval Accuracy	Good for easy cases	Much better for subtle, attribute changes
User Experience	Sometimes off-target	Consistently matches requested modification
Inference Pipeline	Unchanged	Unchanged

- **Output:** More accurate, attribute-aware retrieval results.
- **Training:** More challenging, but leads to better generalization and fine-grained control.

---

5. How to Implement in Code

1. **Extract Attribute/Category Information**
    - For each sample, extract the relevant category/attribute from the modification text (using a BERT-based extractor or rule-based method).
2. **Find Hard Negatives**
    - For each (reference image, modification text, target image) triplet:
        - Category-discrepant: Find an image visually similar to the target but without the required attribute.
        - Attribute-mismatch: Find an image matching most attributes but differing in the specified one(s).
3. **Compute Loss Using Hard Negatives**
    - Use these hard negatives in your contrastive/triplet loss.

Pseudocode Example

```
for idx, (reference_images, target_images, captions) in enumerate(train_bar):
    attributes = extract_attributes_from_text(captions)
    hard_negatives = []
    for i in range(len(reference_images)):
        neg = find_hard_negative(reference_images[i], attributes[i], dataset)
        hard_negatives.append(neg)
```

```
    hard_negatives = torch.stack(hard_negatives).to(device)
    reference_features = clip_model.encode_image(reference_images)
    target_features = clip_model.encode_image(target_images)
    negative_features = clip_model.encode_image(hard_negatives)
    loss = custom_triplet_loss(reference_features, target_features, negative_features)
```

Summary: - Focus on hard negatives (category-discrepant, attribute-mismatch) for best results. - Integrate negative sampling logic in the training loop of `traintest2.py`. - Expect improved fine-grained retrieval and better generalization.