Fashion Image Recommendation Model for Natural Language Search Queries

COSE474-03

Deep Learning Term Project Final Presentation

Team 10

Contents

Task Overview: **Image-Text Retrieval for Fashion**

Develop a model to recommend fashion images based on natural language queries.

Enhance shopping experiences with visual results matching users' request.

Dataset: DeepFashion-MultiModal

Human fashion dataset with 44,096 full body human images and their corresponding textual descriptions

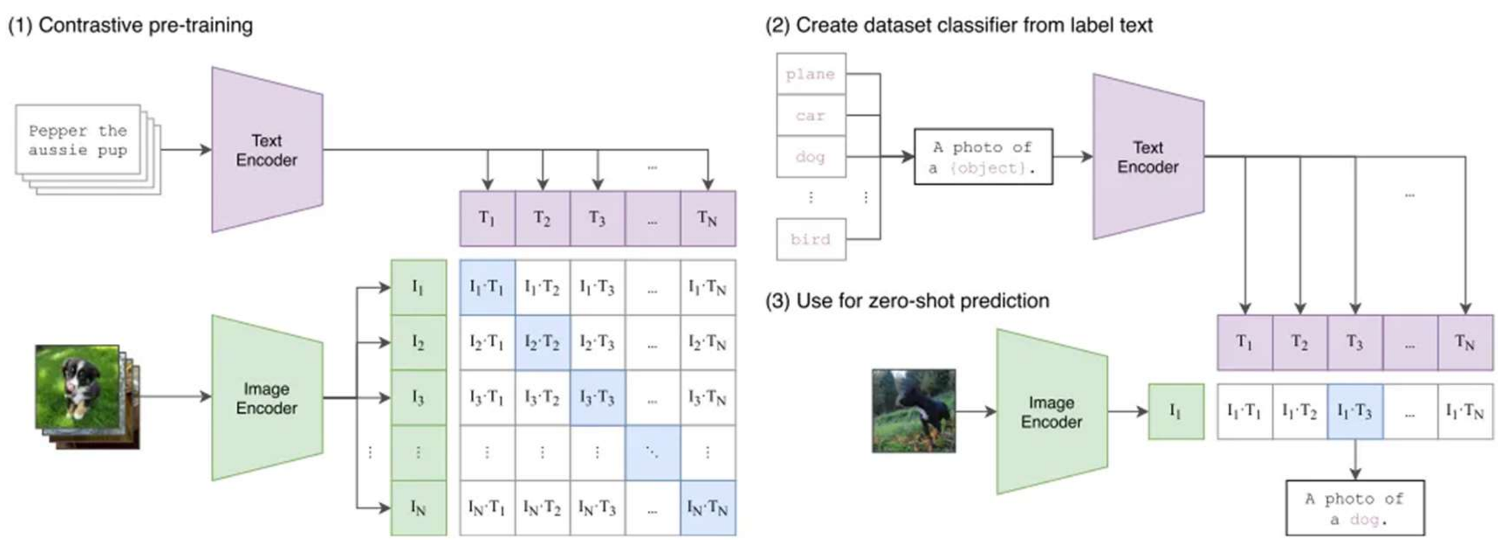
human image textual descriptions

From Jiang, Y., Yang, S., Qiu, H., Wu, W., Chen, C. L., & Liu, Z. (2022). DeepFashion-MultiModal. Retrieved November 13, 2024.

Baseline Model: **CLIP** (Contrastive Language-Image Pre-Training)

- A neural network trained on a variety of (image, text) pairs
- Dual-encoder = simultaneous image and text encoder
- Able to predict the most relevant text snippet, given an image, without directly optimizing for the task

Baseline Model: **CLIP** (Contrastive Language-Image Pre-Training)



From Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. https://arxiv.org/abs/2103.00020

Reasons for Choosing CLIP

- Multi-modal understanding (text-to-image retrieval)
- More efficient at zero-shot transfer
- More robust at task shift

Code for Baseline Model

```
class Model(nn.Module):

def __init__(self):
super().__init__()
self.model, _ = clip.load('ViT-B/32',device=device)

def forward(self, imgs, tokens):
image_features = self.model.encode_image(imgs)
match_text_features = self.model.encode_text(tokens)
image_features = image_features / image_features.norm(dim=-1, keepdim=True)
match_text_features = match_text_features / match_text_features.norm(dim=-1, keepdim=True)
similarity_match = image_features @ match_text_features.T
return similarity_match

def encode_text(self, tokens):
return self.model.encode_text(tokens)

def encode_image(self, imgs):
return self.model.encode_image(imgs)
```

Loss Function: **Cross-Entropy Loss**

```
def compute_loss(similarity_match, labels):
loss1 = F.cross_entropy(similarity_match, labels)
loss2 = F.cross_entropy(similarity_match.T, labels)
loss = (loss1 + loss2) / 2
return loss
```

loss1 focuses on finding the most similar image given a text.
loss2 focuses on finding the most similar text given an image.

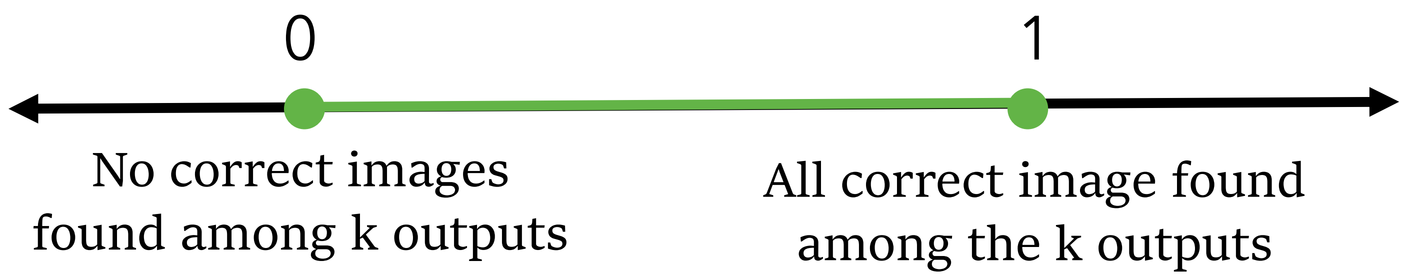
Evaluation Metric: **Recall@K**

$$\text{Recall@K} = \frac{\text{Number of Queries with Correct Match in Top-K}}{\text{Total Number of Queries}}$$

Recall@1

Recall@5

Recall@10



Data Preprocessing & Splitting

- **Splitting Train and Test Data**
  - Split the dataset into **train** and **test** sets (8:2).
  - Set a **fixed random seed** to **ensure consistent results during testing.**
- **Text Truncation**

```
image_caption_pairs = [(key, value) for key, value in captions.items()]
# Set fixed seed
torch.manual_seed(42) # Fix seed
# Split dataset by 8:2
train_size = int(0.8 * len(image_caption_pairs))
test_size = len(image_caption_pairs) - train_size
train_pairs, test_pairs = random_split(image_caption_pairs, [train_size, test_size])
texts = clip.tokenize(texts, truncate=True).to(device)
```

Results from Baseline Model

Model	Recall@1	Recall@5	Recall@10
Baseline (Pretrained CLIP)	0.0001	0.0007	0.0020

It seems significantly low.

Does this mean the baseline model is not performing well in image-to-text retrieval?

Experiment in Baseline Model

Query: "a red dress" →

It seems to be doing the task well.
Why, then, was the recall so low?

First Query & Image Review of Test Dataset in Baseline Model

First Query in the Test Dataset

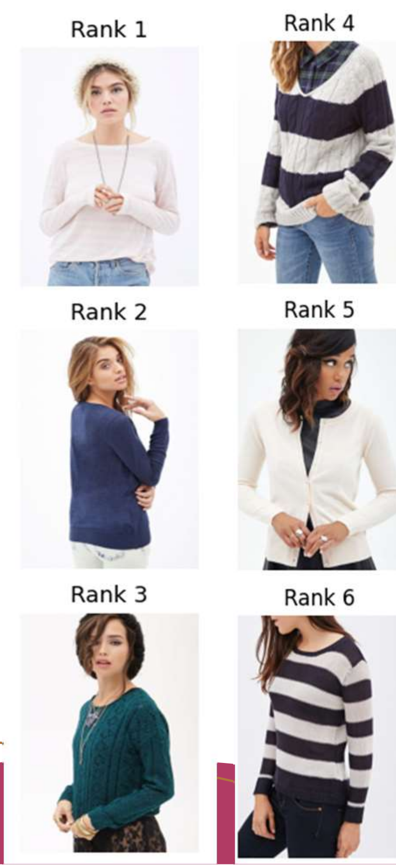
*"The sweater this person wears has **long sleeves** and it is with cotton fabric and **solid color patterns**. The neckline of the sweater is crew. This person wears **long pants, with denim fabric** and solid color patterns. **The outer clothing** this person wears is with cotton fabric and pure color patterns. This female has neckwear. There is a ring on her finger."*

Correct Image for the First Query

First Query & Image Review of Test Dataset in Baseline Model

Top Retrieved Images for the Query

*"The sweater this person wears has **long sleeves** and it is with cotton fabric and **solid color patterns**. The neckline of the sweater is crew. This person wears **long pants, with denim fabric** and solid color patterns. **The outer clothing** this person wears is with cotton fabric and pure color patterns. This female has neckwear. There is a ring on her finger."*



Results for the query execution:

- The top retrieved images were **relevant but not exact** matches.
- Even after checking the top 50 results, the correct image was not found.

Inference

1. Similar but not exact matches

2. Lack of training on fashion-specific text

3. Recall calculation on entire dataset:
   - Recall on the entire dataset (e.g., 40,000 items) is very low, as only the top match counts.
   - For batch datasets (e.g., 32 items), recall values are higher because the model only needs to match a smaller subset. (we observed a Recall@5 of 0.5934)

Improvement Methods for the Baseline Model:

1. **Fine-Tuning with DeepFashion-Multimodal Dataset**
   Equips the model with domain-specific knowledge, allowing it to better handle queries related to fashion.

2. **Adding Cross-Modal Attention to CLIP's Dual Encoder**
   Allows the model to capture more intricate dependencies between the two modalities, potentially leading to a significant improvement in performance.

Improvement Method 1:
Fine-Tuning with the DeepFashion-Multimodal Dataset

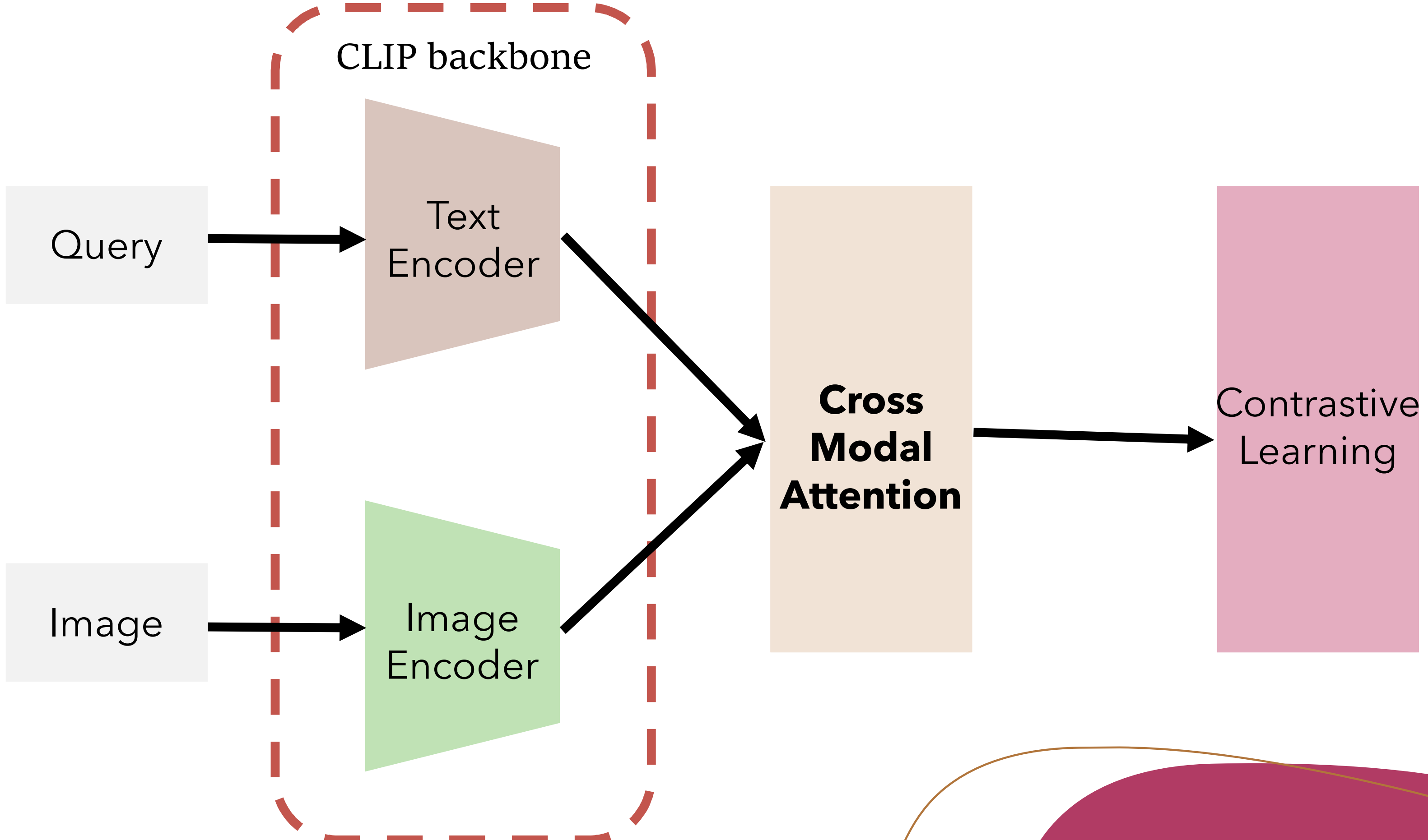
```python
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-5)
num_epochs = 1
temperature = 0.1
# Loss 저장 리스트
batch_losses = []
epoch_losses = []
model.train()
for epoch in tqdm(range(1, num_epochs + 1), desc="Epoch"):
    total_loss = 0
    for images, texts,_ in tqdm(train_dataloader, desc=f"Epoch {epoch + 1}/{num_epochs}"):
        images = images.to(device)
        texts = clip.tokenize(texts, truncate=True).to(device)
        optimizer.zero_grad()
        similarity_match = model(images, texts )
        labels = torch.arange(len(images)).to(device)
        loss = compute_loss(similarity_match, labels)

        # 역전파 및 최적화
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
        batch_losses.append(loss.item())
    epoch_losses.append(total_loss / len(train_dataloader))
    print(f"Epoch {epoch}/{num_epochs}, Loss: {total_loss / len(train_dataloader):.4f}")
```

Before		After
Recall@1: 0.0001 Recall@5: 0.0007 Recall@10: 0.0020	After 1 epoch training	Recall@1: 0.0001 Recall@5: 0.0011 Recall@10: 0.0022

There was a slight increase in Recall at 1, 5, and 10.

Improvement Method 2: Adding Cross-Modal Attention to CLIP's Dual Encoder

Model Architecture:



Improvement Method 2: Adding Cross-Modal Attention to CLIP's Dual Encoder

Code:

```
class CrossModalModel(nn.Module):
def __init__(self, embed_dim=512, num_heads=8, num_layers=6):
super().__init__()
# Transformer for Sequence-wise Attention
encoder_layer = nn.TransformerEncoderLayer(d_model=embed_dim, nhead=num_heads, batch_first=True)
self.transformer = nn.TransformerEncoder(encoder_layer, num_layers=num_layers)
def encode_image(self, imgs):
# CLIP 이미지 인코더 호출
image_features = self.clip_model.encode_image(imgs)
def encode_text(self, tokens):
# CLIP 텍스트 인코더 호출
text_features = self.clip_model.encode_text(tokens)
def forward(self, imgs, tokens):
cls_token_image = torch.zeros_like(image_features) # (Batch, 1, Embed)
cls_token_text = torch.zeros_like(text_features) # (Batch, 1, Embed)
combined_seq = torch.cat([image_seq, text_seq], dim=1) # (Batch, Seq+Seq+2, Embed)
# Pass through Transformer for Cross-Modal Attention
cross_modal_output = self.transformer(combined_seq) # (Batch, Seq+Seq+2, Embed)
```

Training was conducted on the train set for this model, adjusting the weights of the attention mechanism.

Improvement Method 2: Adding Cross-Modal Attention to CLIP's Dual Encoder

Results:

Before		After
Recall@1: 0.0001 Recall@5: 0.0007 Recall@10: 0.0020	After 1 epoch training	Recall@1: 0.0007 Recall@5: 0.0018 Recall@10: 0.0029

There was a slight increase in Recall@1, 5, and 10.

Additional Experiment

CLIP is a pre-trained model, meaning it generally performs well at retrieving images based on queries.

But what would happen if we **randomly initialize the weights** of CLIP?

```python
def initialize_weights_randomly(model):
    """
    모델의 모든 가중치를 랜덤 초기화
    """
    for layer in model.parameters():
        if layer.requires_grad:  # 학습 가능한 레이어만 초기화
            nn.init.normal_(layer, mean=0.0, std=0.02)  # 정규분
initialize_weights_randomly(model)
```

Additional Experiment

Query ⟶ *"The sweater this person wears has **long sleeves** and it is with cotton fabric and **solid color patterns**. The neckline of the sweater is crew. This person wears **long pants, with denim fabric** and solid color patterns. **The outer clothing** this person wears is with cotton fabric and pure color patterns. This female has neckwear. There is a ring on her finger."*

Correct Image for the Query

Top Retrieved Images for the Query
after **randomly initializing the weights**

⟶ We can see that it **does not output the correct image** for the query

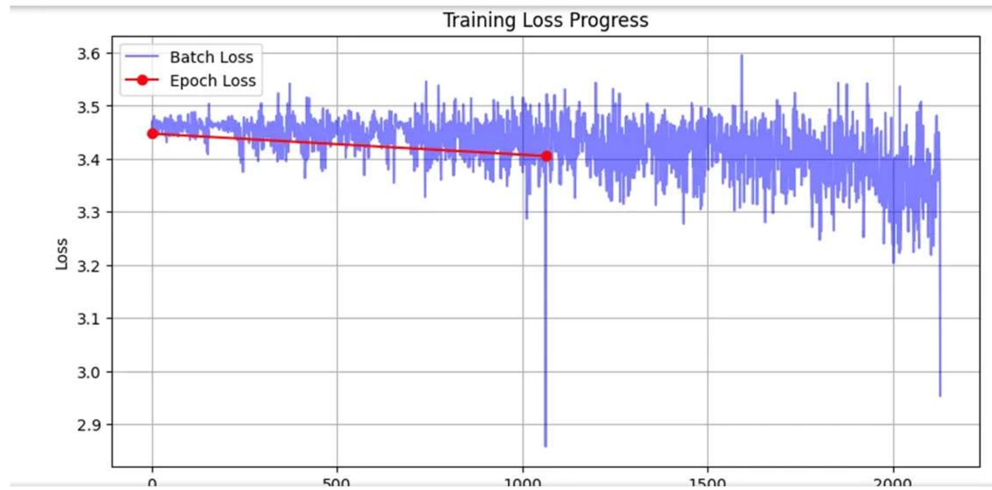
Additional Experiment

*"The sweater this person wears has **long sleeves** and it is with cotton fabric and **solid color patterns**. The neckline of the sweater is crew. This person wears **long pants, with denim fabric** and solid color patterns. **The outer clothing** this person wears is with cotton fabric and pure color patterns. This female has neckwear. There is a ring on her finger."*

```
# 시각화 함수 호출
display_images(corrected_image_paths, output_dir, title="Top-5 Images after Training")
```

Top-5 Images after Training

After training on the fashion train set (1 epoch)



Conclusion

Model	Recall@1	Recall@5	Recall@10
Baseline (Pretrained CLIP)	0.0001	0.0007	0.0020
Fine-tuning with DeepFashion Dataset	0.0001	0.0011	0.0022
Adding Cross-Modal Attention	**0.0007**	**0.0018**	**0.0029**

The recall value was **highest when cross-modal attention was added**.
This suggests that **enhancing the interaction between images and text is an effective way to improve** the model's performance.

Limitations & Difficulties

- **Limited epochs:** due to GPU shortage and slow CPU performance
- **GPU shortage**: While running on GPU (via Elice) for 100 epochs, a GPU shortage error occurred, halting further execution. As a result, only 1 epoch was completed before the error.
- **CPU performance**: After the GPU was no longer available, we attempted to continue training by increasing the number of epochs on the CPU. However, it took about 8 hours per epoch on the CPU, which made it difficult to increase the number of epochs.
- **Disk space shortage**: Due to limited disk space (30 GB), it was not possible to train the model on the originally planned larger dataset.
- **Loss = NaN issue**: Encountered NaN values in the loss during training.

Possible Improvements

- **Increase epochs using GPU**
- **Optimizing training**: Implementing more efficient training techniques, such as mixed-precision training or data augmentation
- **Dataset management**: Using techniques like dataset pruning or selecting smaller, more relevant subsets of the data for training
- **Focus on image-text preprocessing**: Improve the preprocessing of both image and text data to ensure higher quality input for the model, which can lead to better overall results.

References

1. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. https://arxiv.org/abs/2103.00020

2. Jiang, Y., Yang, S., Qiu, H., Wu, W., Chen, C. L., & Liu, Z. (2022). DeepFashion-MultiModal. Retrieved November 13, 2024.

3. Liu, H., Xu, S., Fu, J., Liu, Y., Xie, N., Wang, C.-C., Wang, B., & Sun, Y. (2021). CMA-CLIP: Cross-Modality Attention CLIP for Image-Text Classification.

4. [Scene-Text Aware Image and Text Retrieval with Dual-Encoder](https://aclanthology.org/2022.acl-srw.34) (Miyawaki et al., ACL 2022)

Q&A

Thank you