Evaluating BLIP Models for Image Captioning

Sai Anoushka K
CS298 Dr. Chris Pollett

Why Model Selection matters?

Real-time captioning requires a balance between:

- Accuracy (Descriptive & Contextually Relevant Captions)
- Speed (Low Latency for Immediate User Feedback)
- Scalability (Handling Multiple API Requests Efficiently)

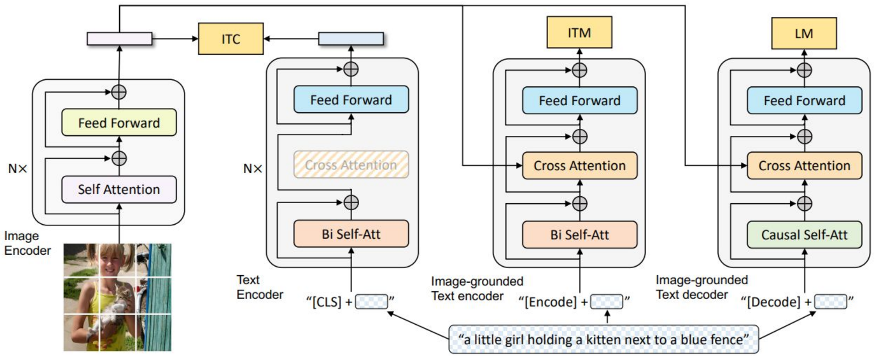
Trade-offs:

- More complex models are slower but provide better captions
- Faster models may compromise on descriptive accuracy

BLIP Model Architecture

- BLIP (Bootstrapped Language-Image Pretraining) is a multimodal model designed for vision-language tasks, including image captioning.
- The architecture integrates three core objectives:
  - Image-Text Contrastive (ITC) – Aligns images and text in a shared representation space.
  - Image-Text Matching (ITM) – Determines whether an image and text pair are semantically related.
  - Language Modeling (LM) – Generates descriptive captions based on visual input.

BLIP Model Architecture



Model Components

- **Image Encoder:** Uses self-attention and feed-forward layers to extract feature representations from the input image.
- **Text Encoder:** Processes text using a transformer-based architecture, encoding words into meaningful representations.
- **Cross-Attention Mechanism:** Enables interaction between the visual and textual modalities, refining contextual understanding.
- **Bi-directional and Causal Self-Attention Layers:**

Bi-directional Self-Attention: Helps in joint learning of vision and language.

Causal Self-Attention: Used in the text decoder to generate captions word by word.

- **Final Caption Generation:** After encoding, the model decodes a meaningful sentence that best describes the image.

Methodology

- **Models Evaluated:**
  - BLIP-Base (Salesforce/blip-image-captioning-base)
  - BLIP-Large (Salesforce/blip-image-captioning-large

- **Evaluation Metrics:**
  - Caption Coherence & Accuracy (Descriptive quality)
  - Inference Time (Speed of response)
  - Model Suitability for VisionMate

- **Test Setup:**
  - 4 sample images from Pexels
  - Measured performance using Hugging Face's transformers library

```python
import time
import requests
import matplotlib.pyplot as plt
from PIL import Image
from transformers import BlipProcessor, BlipForConditionalGeneration

# Load BLIP models (Base and Large)
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model_base = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
model_large = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large")

# Sample images for testing
image_urls = [
    "https://images.pexels.com/photos/1108099/pexels-photo-1108099.jpeg",  # Dogs in a field
    "https://images.pexels.com/photos/34950/pexels-photo.jpg",  # train track
    "https://images.pexels.com/photos/3777572/pexels-photo-3777572.jpeg",  # man with a laptop
    "https://images.pexels.com/photos/112326/pexels-photo-112326.jpeg"  # Mountain Landscape
]

# Function to process an image and generate captions
def generate_caption(image_url, model, processor):
    response = requests.get(image_url, stream=True)
    image = Image.open(response.raw).convert("RGB")

    inputs = processor(images=image, return_tensors="pt")

    start_time = time.time()  # Measure inference time
    output = model.generate(**inputs)
    caption = processor.decode(output[0], skip_special_tokens=True)
    inference_time = time.time() - start_time  # Compute time taken

    return image, caption, inference_time
```

```python
# Display results
fig, axes = plt.subplots(len(image_urls), 3, figsize=(12, len(image_urls) * 4))

for i, image_url in enumerate(image_urls):
    # Generate captions
    image, caption_base, time_base = generate_caption(image_url, model_base, processor)
    _, caption_large, time_large = generate_caption(image_url, model_large, processor)

    # Plot images and captions
    axes[i, 0].imshow(image)
    axes[i, 0].axis("off")
    axes[i, 0].set_title("Original Image", fontsize=10)

    axes[i, 1].imshow(image)
    axes[i, 1].axis("off")
    axes[i, 1].set_title(f"Base: {caption_base}\n(Time: {time_base:.2f}s)", fontsize=10)

    axes[i, 2].imshow(image)
    axes[i, 2].axis("off")
    axes[i, 2].set_title(f"Large: {caption_large}\n(Time: {time_large:.2f}s)", fontsize=10)

plt.tight_layout()
plt.show()
```

Results

- BLIP-Base: "Two pup sitting in a field of flowers."
- BLIP-Large: "There are two dogs sitting in the grass with flowers in the background."
- Inference Time:
  Base: 7.38s
  Large: 13.64s

Results

- BLIP-Base: "A train track with trees and bushes in the background."
- BLIP-Large: "There is a train track that is surrounded by trees and bushes."
- Inference Time:
  - Base: 5.30s
  - Large: 13.14s

Performance Comparison

Criteria	BLIP Base	BLIP Large
Caption Accuracy	Generates clear and concise captions, but may miss finer details.	Captions are more detailed and descriptive, especially for complex scenes.
Inference Time	Faster (**5-7s per image**)	Slower (**12-15s per image**)
Computational Load	Requires fewer resources, making it efficient for real-time applications.	Higher memory and processing power requirements.
Suitability	Ideal for real-time captioning due to speed and efficiency.	Better for offline or batch processing where accuracy is the priority.

Conclusion

- BLIP-Base offers the best trade-off between accuracy and speed
- BLIP-Large, while more descriptive, is too slow for real-time use
- Next step: Setup Front-end