HW3 Report

Hsuan-Ting Lin

B11901132

1 P1: Zero-shot Image Captioning with LLaVA

1.1 "Visual Instruction Tuning" Paper Reading

The "Visual Instruction Tuning" paper introduces LLaVA, a multimodal model that connects a visual encoder with a LLM to enable general-purpose vision and language understanding [1].

Some of the key components of LLaVA are outlined as follows:

1.1.1 Architecture

The authors used the pre-trained CLIP visual encoder ViT-L/14 to obtain image embeddings, and used a simple linear layer, which is a trainable matrix, to connect image features into the word embedding space.

The projected visual embeddings are then concatenated with the text embeddings from instructions to create a unified input sequence. The Vicuna LLM processes this combined sequence as if it were a single input, enabling it to understand and generate responses based on both the image and instruction.

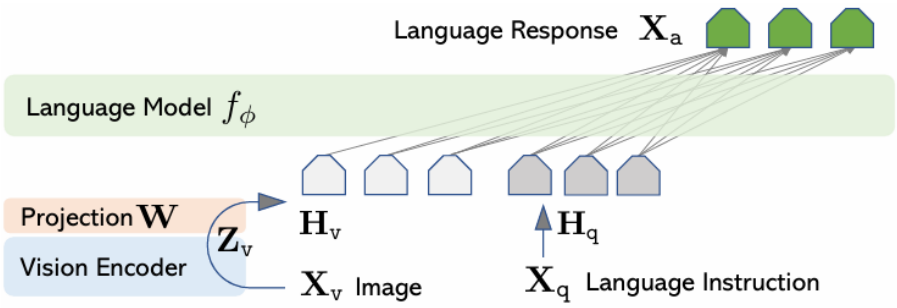

Figure 1: LLaVA network architecture

1.1.2 Training

The model is trained on a high-quality dataset of multimodal instruction pairs, including GPT-4-generated synthetic data. Inputs combine image embeddings and tokenized text instructions, while outputs are textual responses, such as captions or answers to visual questions.

A causal language modeling loss ensures that generated responses align with reference outputs. End-to-end optimization fine-tunes the entire system, enabling effective integration of image and text for tasks like image captioning and visual question answering.

1.2 Prompt-Text Analysis

Apart from the best setting, which is deterministic as I set `do_sample = False`, I experimented with two other configurations with different extent of randomness, controlled by `temperature`, `top_p` and `top_k`.

```
# Best Setting, deterministic
output = model.generate(**inputs,
    max_new_tokens=50,
    do_sample=False,
)
# Score: CIDEr: 1.1433838361756834 | CLIPScore: 0.77259521484375

# Setting 1, moderate
output = model.generate(**inputs,
    max_new_tokens=50,
    do_sample=True,
    temperature=0.7,
    top_p=0.9,
    top_k=50,
)
# Score: CIDEr: 0.93241023005383 | CLIPScore: 0.773560791015625

# Setting 2, more diverse
output = model.generate(
    **inputs,
    max_new_tokens=50,
    do_sample=True,
    temperature=0.9,
    top_p=0.95,
    top_k=100,
)
# Score: CIDEr: 0.737606350810065 | CLIPScore: 0.760635986328125
```

As `temperature`, `top_p` and `top_k` values become higher, the generation captions become more diverse but deviate from the reference captions, reducing n-gram overlap and lowering CIDEr scores due to less alignment with expected phrases. CLIP scores, which assess alignment with visual content, remains stable.

Below is a comparison of captions generated for the same image using the three configurations to illustrate the differences.

```
# Best setting
"A cat is laying on a computer keyboard."
# Setting 1
"A grey and white cat is laying on a computer keyboard."
# Setting 2
"A white cat is laying on a computer keyboard with its head on the keyboard."
```

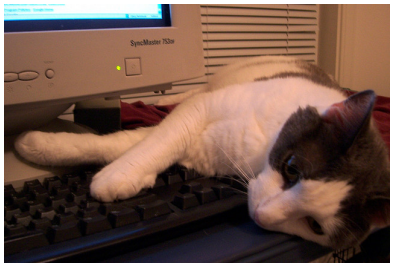

Figure 2: Image corresponding to the captions

2 P2: PEFT on Vision and Language Model for Image Captioning

2.1 Best Setting Report

I used `openai/clip-vit-large-patch14` as my vision encoder, and used a single linear layer as the projection layer. Referring to the paper, I also conducted two training stages, the first only training projection layer, and the second jointly training both decoder and projection layer.

LoRA layers are employed on every linear layer in decoder, resulting in total trainable parameters count 7.14 M, including LoRA and projection parameters. Teacher forcing is used to train caption generation, where the inputs to the decoder are "image embeddings + EOS + caption + EOS + padding", and cross entropy loss is calculated on the parts I want model to predict, which is "caption + EOS".

Some hyperparameters and training configurations of the best setting are given below:

```
# hyperparameters
lora_rank = 32
lora_alpha = 64
batch_size = 12

# training stage 1
num_epochs = 2
optimizer = torch.optim.Adam([p for p in model.projection.parameters()], lr=1e-3)

# training stage 2
num_epochs = 3
optimizer = torch.optim.Adam([p for p in model.projection.parameters()], lr=1e-3)
scheduler = torch.optim.lr_scheduler.CosineAnnealingWarmRestarts(optimizer, T_0=834,
    T_mult=2, eta_min=1e-4) # 834 steps per epoch
```

Using the checkpoint with lowest val loss to generate captions in a greedy fashion on val set, the evaluation scores are CIDEr: 0.9727208148433776 | CLIPScore: 0.7366921997070313, which is over the strong baseline.

2.2 Different Attempts on LoRA Settings

I experimented with LoRA ranks 8 and 16, while keeping other hyperparameters and configurations the same. As per the paper "LoRA: Low-Rank Adaptation of Large Language Models" [2], tuning alpha is the same as tuning the learning rate, so I did not scale alpha in order to achieve fair comparision.

The checkpoints with lowest val loss are used to generate captions also in a greedy method, the results are shown below:

```
# r = 32, trainable params = 7.14 M
CIDEr: 0.9727208148433776 | CLIPScore: 0.7366921997070313
# r = 16, trainable params = 3.96 M
CIDEr: 0.988539124884642 | CLIPScore: 0.735296630859375
# r = 8, trainable params =  2.38 M
CIDEr: 0.9716935887338448 | CLIPScore: 0.7331080627441406
```

The performance is stable across ranks, with CIDEr and CLIP scores showing minimal variation. This result aligns with the conclusion in the original LoRA paper, where they found that LoRA achieves competitive performance even with low-rank configurations, demonstrating that model updates during fine-tuning have a low intrinsic rank.

3 P3: Visualization of Attention in Image Captioning

3.1 Visualization on Test Images Using Attention Maps

3.1.1 Attention Visualization Using llava-hf/llava-1.5-7b-hf

Using the LLaVA-1.5-7B-HF model and the inference configuration from P1, I visualized the attention maps between the generated caption tokens and the five test images. The heatmaps below represent these attention patterns.

I experimented with attention weights from both the last and second-to-last layers, using both maximum and mean aggregation across attention heads. While some results, such as those for umbrella.jpg and girl.jpg, successfully highlight the main objects in the images, the patterns in other cases appear less distinct and often similar across tokens. This suggests that the model may not consistently ground each token in specific visual features, possibly relying more on global image representations or the input prompt. The relatively uniform patterns for some tokens could also stem from the use of higher layers, where attention often integrates global and contextual information. As a result, finer token-level differences may become less pronounced. This is especially evident in cases where auxiliary or less visually grounded tokens exhibit attention patterns that are difficult to differentiate.

Figure 3: Visualization of bike.jpg (left) and girl.jpg (right)

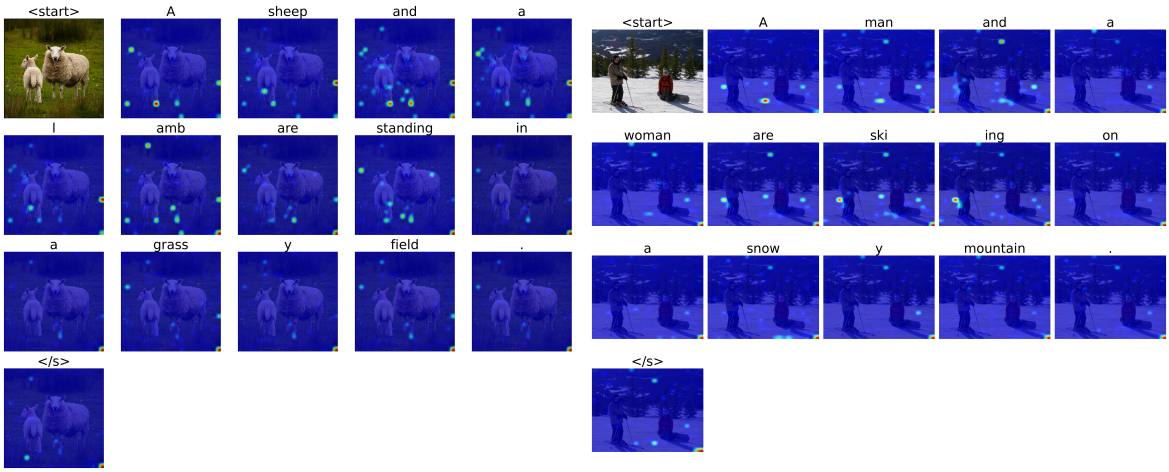

Figure 4: Visualization of sheep.jpg (left) and ski.jpg (right)

Figure 5: Visualization of umbrella.jpg

3.1.2 Attention Visualization Using Model in P2

Using my model in P2, the heat maps below are the attention visualization between generated caption tokens and the 5 test images.

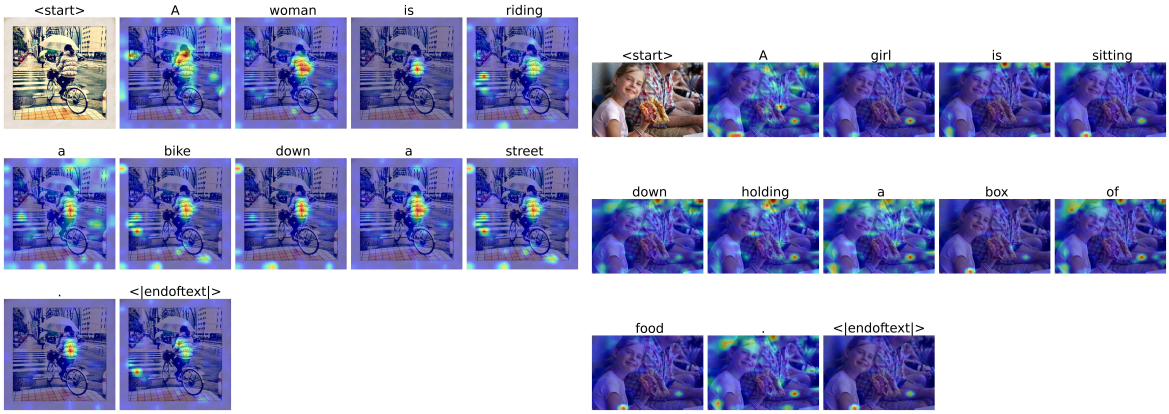

Figure 6: Visualization of bike.jpg (left) and girl.jpg (right)

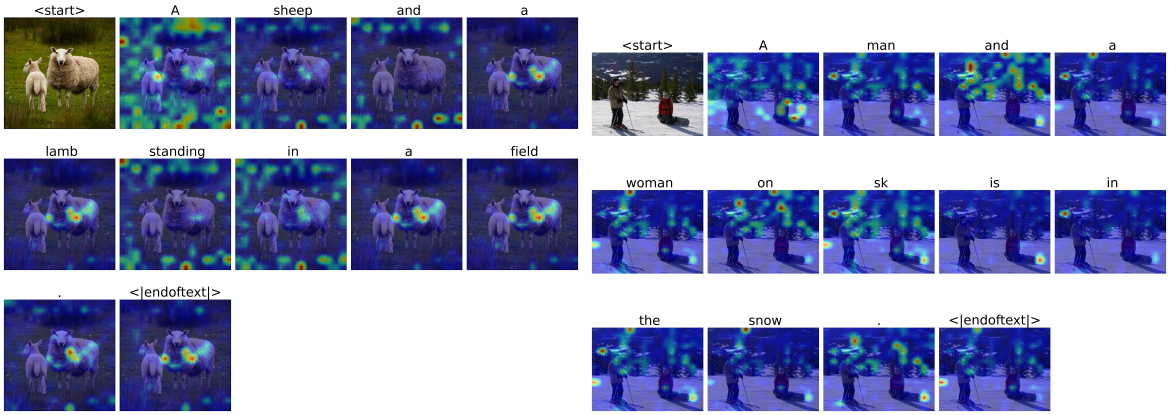

Figure 7: Visualization of sheep.jpg (left) and ski.jpg (right)

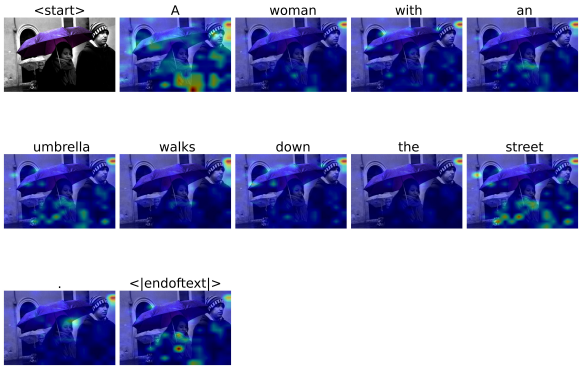

Figure 8: Visualization of umbrella.jpg

Using the attention weights in the last layer, averaged through heads, the attention maps seem to reflect the main objects in the photos. However, the differences are not too obvious for different tokens. This may be due to the model relying heavily on global features, rather than focusing on specific patches for individual tokens. Additionally, the averaging across multiple attention heads could dilute token-specific attention patterns.

3.2 Visualization of Val Images According to CLIP scores

Using the best model in part2, the heat maps of the top-1 and last-1 image-caption pairs according to CLIP score is showed below.

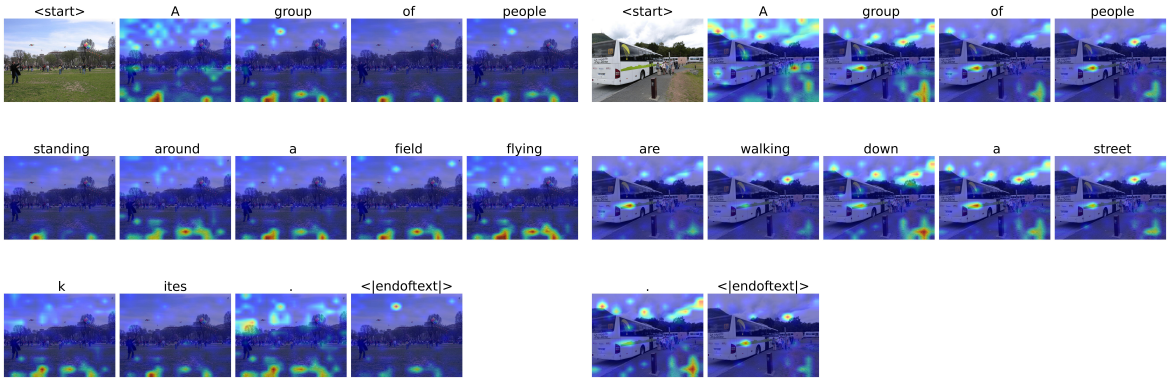

Figure 9: Highest score: 0.989990234375 (left), Lowest score: 0.43701171875 (right)

Both captions are fairly reasonable without hallucination, as the predicted caption with the higher CLIP score reflects the image accurately, capturing finer details and context that align well with the visual content. On the other hand, the caption with the lower score provides only a coarse description, lacking specific elements or nuances present in the image.

Attention visualization for both images does not reflect the corresponding words particularly well. This observation suggests that the self-attention mechanism between text rather than image embeddings may play a more significant role than anticipated in this caption-generating model.

Acknowledgments

Some of the code in this project was generated with assistance from `ChatGPT` [3] and `Claude` [4]. Their contributions include code for model training, model evaluation, and general Python scripting.

References

[1] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[3] OpenAI, "Chatgpt: Gpt-4 model," `https://chat.openai.com`, 2024, `https://chat.openai.com`.

[4] Anthropic, "Claude: A large language model by anthropic," `https://www.anthropic.com`, 2024, `https://www.anthropic.com`.