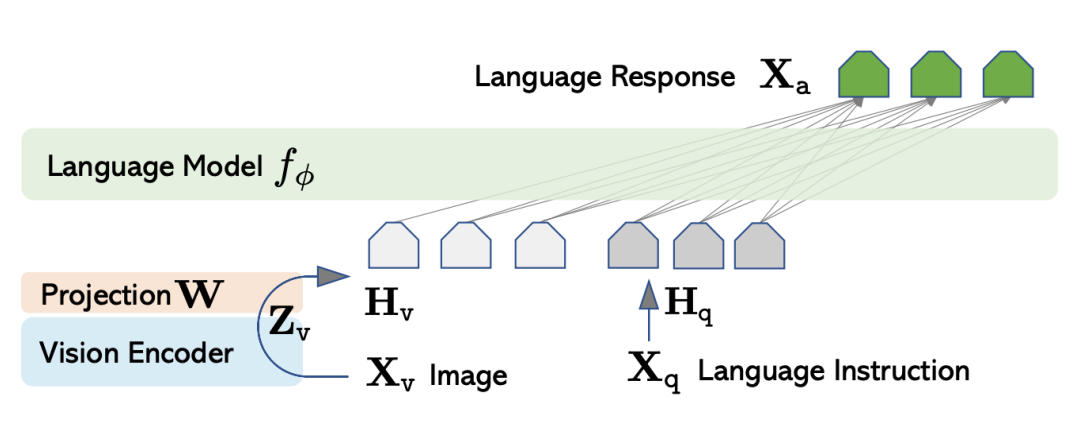
DLCV-HW3

- Student ID: B09611007

Problem 1: Zero-shot Image Captioning with LLaVA

1. Read the paper "Visual Instruction Tuning" and briefly describe the important components (modules or techniques) of LLaVA.

- **Visual Encoder:** Uses CLIP to extract image features.
- **Language Model:** Utilizes Vicuna as the language decoder.
- **Linear Projection Layer:** Projects visual features into the language embedding space, connecting image features with the language model in the same semantic space.
- **Multi-turn Dialogue Generation:** Supports coherent multi-turn dialogues, helping the model maintain context across interactions.
- **Instruction-Tuning Data:** Leverages GPT-4 to generate multimodal instruction-following data, including dialogues, descriptions, and reasoning tasks.



2. Please come up with two settings (different instructions or generation config). Compare and discuss their performances.

	Setting A	Setting B
Prompt	"Write a description for the photo in one sentence describing: the exact subject, its accurate action, and precise visual context."	"Write a description for the photo in one sentence describing: the exact subject, its accurate action, and precise visual context. **For example, "A busy intersection with an ice cream truck driving by.**"
Config	"num_beams": 5, "max_new_tokens": 40, "do_sample": True, "temperature": 0.27, "top_p": 0.85, "no_repeat_ngram_size": 4, "repetition_penalty": 1.35, "length_penalty": 0.91, "early_stopping": True	(The same)
CIDEr	1.175	1.251
CLIPScore	0.781	0.784
Comparison / Discussion	Guiding the model toward detailed and accurate descriptions, prioritizing specificity and completeness.	Providing an example helps the model align with a specific descriptive style, reducing ambiguity and promoting consistency. **CIDEr scores improve** because the example guides the model to generate descriptions with **vocabulary and structure closer to ground truth.**

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

1. Report your best setting and its corresponding CIDEr & CLIPScore on the validation data. Briefly introduce your method. (TA will reproduce this result)

- **ImageCaptioning Model:** To generate descriptive captions for images. The model integrates visual and textual features and to complete image-to-text captioning tasks.
- **Image Encoder:** A pretrained ViT model `vit_huge_patch14_clip_224.laion2b` extracts visual features from images, which are then resized to match the dimensions expected by the decoder.

```
# Encoder (pretrained ViT model)
self.encoder = timm.create_model(
    "vit_huge_patch14_clip_224.laion2b", pretrained=True, num_classes=0
)
self.feature_resize = nn.Linear(1280, 768)
```

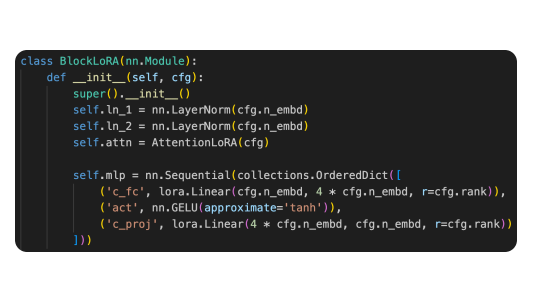
- **Tokenizer:** A Byte-Pair Encoding (BPE) tokenizer converts text captions into token IDs for training and decodes generated token sequences back into text during inference.
- **Decoder:** A decoder with self-attention layers generates captions by conditioning on the image features and previously generated tokens.

```
def forward(self, x: Tensor, encoder_feature: Tensor):

    x = torch.narrow(x, 1, 0, min(x.size(1), self.block_size))
    pos = torch.arange(x.size()[1], dtype=torch.long, device=x.device).unsqueeze(0)

    # Get word and position embeddings for x
    x = self.transformer.wte(x) + self.transformer.wpe(pos)

    # Concat visual features with text embeddings
    x = torch.cat((encoder_feature, x), dim=1)

    for block in self.transformer.h:
        x = block(x)

    x = self.lm_head(self.transformer.ln_f(x))
    return x
```

- Greedy search -> CIDEr: **0.943** & CLIPScore: **0.731**
- Beam search (2 beams) -> CIDEr: 1.001 & CLIPScore: 0.724
- Beam search (3 beams) -> CIDEr: 1.028 & CLIPScore: 0.722
- Increasing the number of beams can boost the CIDEr score but may reduce the CLIPScore.
  - This is because more beams help the model generate captions closer to the reference, thus raising the CIDEr.
  - However, the model may over-focus on reference similarity at the expense of semantic alignment with the image, leading to a lower CLIPScore.
- The implementation was partly adopted from Lucashien (https://github.com/Lucashien/NTU-2023Fall-DLCV/tree/main/hw3) and pha (https://github.com/pha123661/NTU-2022Fall-DLCV/tree/master/HW3).

2. Report 2 different attempts of LoRA setting (e.g. initialization, alpha, rank...) and their corresponding CIDEr & CLIPScore.

- I added LoRA on all `Embedding` and `Linear` layers in the decoder (params: 4.8M)

DecoderLoRA	BlockLoRA
	

DecoderLoRA:

```
class DecoderLoRA(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.cfg = cfg
        self.block_size = cfg.block_size

        # Define transformer components with LoRA
        self.transformer = nn.ModuleDict(dict(
            wte=lora.Embedding(cfg.vocab_size, cfg.n_embd, r=cfg.rank),
            wpe=lora.Embedding(cfg.block_size, cfg.n_embd, r=cfg.rank),
            h=nn.Sequential(*[BlockLoRA(cfg) for _ in range(cfg.n_layer)]),
            ln_f=nn.LayerNorm(cfg.n_embd)
        ))

        self.lm_head = lora.Linear(cfg.n_embd, cfg.vocab_size, bias=False, r=cfg.rank)
        self.transformer.wte.weight = self.lm_head.weight
```

BlockLoRA:

```
class BlockLoRA(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.ln_1 = nn.LayerNorm(cfg.n_embd)
        self.ln_2 = nn.LayerNorm(cfg.n_embd)
        self.attn = AttentionLoRA(cfg)

        self.mlp = nn.Sequential(collections.OrderedDict([
            ('c_fc', lora.Linear(cfg.n_embd, 4 * cfg.n_embd, r=cfg.rank)),
            ('act', nn.GELU(approximate='tanh')),
            ('c_proj', lora.Linear(4 * cfg.n_embd, cfg.n_embd, r=cfg.rank))
        ]))
```

- Result

rank=8, alpha=8	rank=16, alpha=16
CIDEr=0.844	CIDEr=0.851
CLIPScore=0.722	CLIPScore=0.717

Problem 3: Visualization of Attention in Image Captioning

1. Given five test images, visualize the predicted caption and the corresponding series of attention maps

Image Captioning in Problem 1:

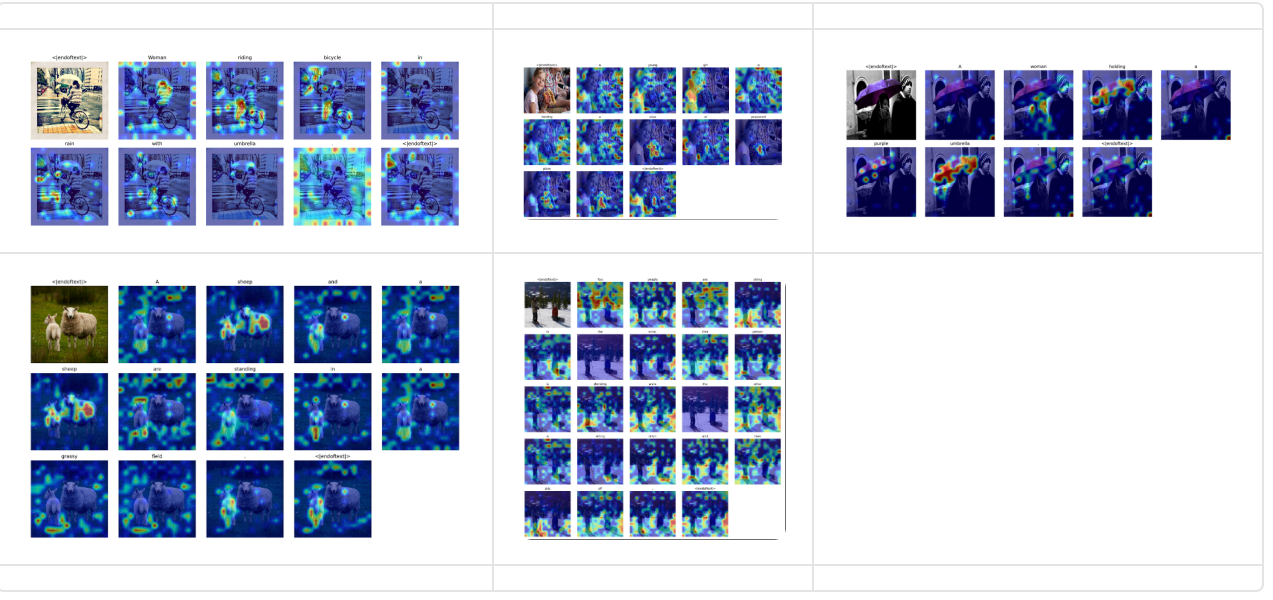
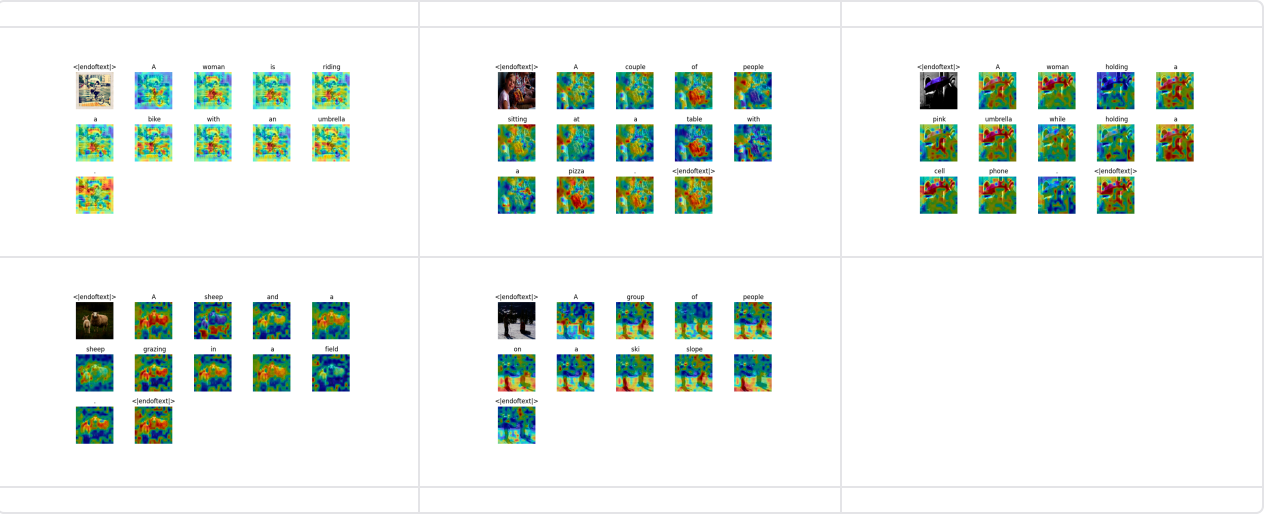
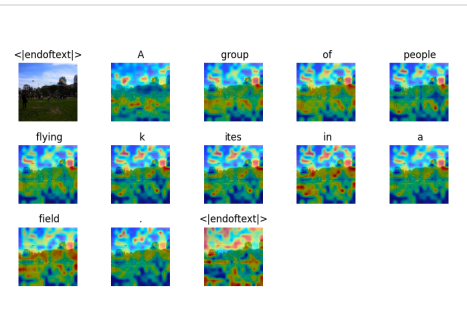
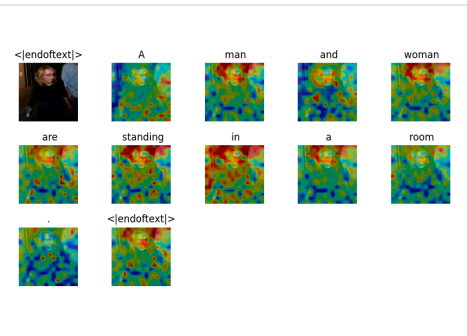


Image Captioning in Problem 2:



2. Visualize top-1 and last-1 image-caption pairs and report its corresponding CLIPScore in the validation dataset of problem 2.

Top-1 (CLIPScore: 1.027)	Last-1 (CLIPScore: 0.377)
	
'000000001086.jpg'	'000000001323.jpg'

3. Analyze the predicted captions and the attention maps for each word according to the previous question. Is the caption reasonable? Does the attended region reflect the corresponding word in the caption?

- I think it reasonable to a certain extent.
- The attention performs impressively well for prepositions like "on" and "in" (e.g. "in" in the '000000001323.jpg'). However, for nouns, the attended regions still lack alignment with the actual content of the image (e.g. both "man" and "woman" shown in '000000001323.jpg').