Deep Learning for Computer Vision – HW #3

Student ID : R13943118
Name : 林玠志

Problem 1: Zero-shot inference with LLaVA

1. Report the accuracy of LLaVA w/o VCD (3-1) and w VCD (3-2)

LLaVA w/o VCD	LLaVA w VCD
61.7%	64.5%

2. Describe the underlying mechanism of Visual Contrastive Decoding (VCD) and how it helps mitigate object hallucinations in Large Vision-Language Models

Visual Contrastive Decoding (VCD) is a zero-shot decoding strategy that guards a multimodal language model against hallucinating objects that aren't in the image. It runs two forward passes at every decoding step:

- **Clean view**: Use the original image; this produces the logits the model would normally rely on, which capture both genuine visual cues and language priors.
- **Distorted view**: Feed a heavily perturbed version of the same image (e.g., with diffusion noise). Real visual evidence gets destroyed, so the model's output here reflects mostly language priors.

While decoding, VCD compares these two logits. Tokens that stay high even when the visual input is corrupted are likely driven by the model's priors rather than the image. VCD subtracts a scaled portion of the distorted-logits from the clean ones (controlled by α) and masks tokens whose clean logits barely exceed the distorted ones (controlled by β). This contrastive filtering suppresses hallucination-prone tokens while leaving visually grounded tokens intact, reducing instances such as the model asserting "there's a spoon" when no spoon exists in the image.

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)

Best setting & score :

Pretrained vision encoder	vit_large_patch14_clip_224.openai
Projector	nn.Linear(1024, 1024)

Transform	`def build_transform(image_size: int = 224) -> transforms.Compose:`
	`  return transforms.Compose(`
	`  [`
	`  transforms.Resize((image_size, image_size), interpolation=transforms.InterpolationMode.BICUBIC),`
	`  transforms.ToTensor(),`
	`  transforms.Normalize(mean=CLIP_MEAN, std=CLIP_STD),`
	`  ]`
	`  )`
	`CLIP_MEAN = (0.48145466, 0.4578275, 0.40821073)`
	`CLIP_STD = (0.26862954, 0.26130258, 0.27577711)`

Epoch	10
Learning rate	5e-4
Weight decay	1e-6
Loss	torch.nn.functional.cross_entropy
optimizer	AdamW
LoRA rank	16
LoRA alpha	16.0
LoRA dropout	0.1
LoRA target	q_proj, v_proj
LoRA train bias	False
CIDEr	1.088
CLIPScore	0.745

We follow the assignment's LLaVA-style recipe: a frozen CLIP ViT encoder produces patch embeddings, we map those into the decoder's hidden space with a learnable projector, and a frozen Qwen3 decoder generates captions.

LoRA is injected by wrapping each targeted decoder attention projection (default q_proj/v_proj) with a LoRALinear. That layer exposes the frozen base weight plus a rank-r low-rank update (B·A) whose parameters remain trainable; every other decoder weight stays frozen.

The projector is a single nn.Linear(embed_dim, hidden_size) aligned to the decoder's embedding width (1024). It's the only component bridging vision features into the language space, so we keep its weights trainable during fine-tuning.

For training we freeze the vision encoder and decoder backbone, update only the projector and LoRA adapters. Captions are tokenized with the provided tokenizer, and we concatenate projected vision tokens with shifted text embeddings. We train using cross-entropy on caption[1:], mix precision/gradient clipping, and checkpoint just the trainable tensors plus the config so inference can reconstruct the full model with the frozen base weights.

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

All the parameters are same as parameters specified in previous section, except LoRA rank, and LoRA alpha.

LoRA rank	LoRA alpha	CIDEr	CLIPScore
16	16.0	1.088	0.745
32	32.0	1.038	0.745