# Deep Learning for Computer Vision HW3

contributed by < weihsinyeh (https://github.com/DLCV-Fall-2024/dlcv-fall-2024-hw3-weihsinyeh) >
資訊所 葉惟欣
學號: R13922043

## Problem 1: Zero-shot Image Captioning with LLaVA

- Zero-shot: You cannot do any finetuning for the pretrained model.
- In this problem, you only need to evaluate the pretrained LLaVA on image captioning task.
- Input: (1)image (2)language instruction (3)generation config
- Output: caption
- Please use the model “llava-hf/llava-1.5-7b-hf” in the transformers package
- Goal: Choose a proper instruction and generation config to obtain better caption

### Problem 1: Grading - Report (9%)

#### Paper reading(3%)

Please read the paper “Visual Instruction Tuning” (https://arxiv.org/pdf/2304.08485) and briefly describe the important components (modules or techniques) of LLaVA.

Visual Instruction Tuning 這篇論文提出 LLaVA end-to-end trained large multimodal model。LLM 可以作為多個任務的幫手，建立一個從視覺特徵 mapping 到語言特徵的機制。LLaVA 提出 visual instruction-tuning，將 instruction tuning 擴展到多個語言與視覺共用的 latent space 上，將他們 align 。也提出一個 pileline ，輸入圖片 image 與其描述 text 的 pair ，輸出合適的 instruction format。使用 CLIP 做為 visual encoder 與 Vicuna 作為 language decoder 去開發一個 large multimodal model (LMM)。

- GPT-assisted Visual Instruction Data Generation
  用 GPT-4 在廣泛存在的圖像配對數據上去收集多模態指令。將圖像對於一張圖像($X_v$)及其相關標註($X_c$)，自然可以生成一組問題($X_q$)，目的是指導助手描述圖像內容。將 image 與 description 放在：**人類** $X_q\ X_v <STOP>$**助手**：$X_c <STOP>$ 的格式中。為了增加回應的多樣性，以及加強推理能力。使用 GPT-4 與 ChatGPT 作為 teacher model。同時使用兩種 labe Captions 從多個角度描述視覺場景以及Bounding boxes 用來定位場景中的物體，每個框包含物體的概念及其空間位置。
  - Conversation：設計一個助手與人類對話內容圍繞這張圖片的場景。回答的語氣表現得像助手正在觀看圖像並回答問題。問題涵蓋圖像視覺內容的多個方面，包括物體類型、物體計數、物體行為、物體位置及物體間的相對位置。
  - Detailed description：為圖像生成豐富且全面的描述，創建一個問題列表，旨在實現此目標。讓 GPT-4 整理問題列表，並請 GPT-4 對於每張圖像以及搭配隨機抽取列表中的一個問題後，請GPT-4生成詳細描述。
  - Complex reasoning：透過具複雜推理的 image dataset。讓模型會逐步推理。

- Visual Instruction Tuning
  使用pretrained CLIP 作為 image 的 encoder，該編碼器提供 visual representation。在我們的實驗中，會考慮最後一層Transformer層前後的網格特徵。我們考慮使用一個簡單的線性層來將圖像特徵映射到語言嵌入空間。具體來說，我們應用一個可訓練的 project layer 將 image token 轉到 text token，讓兩者用同樣的 embedding space 。對於每個圖像，生成多輪對話。將這些對話組成一個序列，所有的回答視為助手的回應，其後每一輪選擇合適的對話方式，我們就得到了統一格式的多模態指令跟隨序列，兩階段的 instruction-tuning procedure：
  - Stage 1 ： Pre-training for Feature Alignment.
    在訓練中，不訓練 visual encoder 和LLM，只訓練 projection layer 。This stage can be understood as training a compatible visual tokenizer for the frozen LLM
  - Stage 2： Fine-tuning End-to-End.
    只訓練 projection layer 和LLM的預訓練權重。

#### Prompt-text analysis (6%)

Please come up with two settings (different instructions or generation config). Compare and discuss their performances
下方實驗都使用這樣的 conversation config。只是替換 content 中的 "text"

```
conversation = [
{
"role": "user",
"content": [
{"type": "text", "text": "Focus only on the primary subject and its core action in this image. Keep the caption short and clear. Avoid any extra details."},
{"type": "image"},
],
},
]
```

我原先參考 Hugging face 的 prompt，他會生出過長的句子。所以我就開始跟他說只要關注在重要的物件上即可。同時避免生成沒有用的細節。同時我也發現 subject 換成 object 會效果會變差，代表很多時候描述的東西不一定只是物件，而是任何主題出現在圖片中都有可能為描述的對象。

##### prompt : What are these?

| CIDEr | CLIPScore |
| --- | --- |
| 0.3331355094260031 | 0.7363873291015625 |

**prompt : Write a concise caption for this image focusing only on the main subject and action, avoiding unnecessary details.**

| CIDEr | CLIPScore |
| --- | --- |
| 1.0998791927463725 | 0.7646685791015625 |

**prompt : Focus only on the primary subject and its core action in this image. Keep the caption short and straightforward.**

| CIDEr | CLIPScore |
| --- | --- |
| 1.121081622968281 | 0.774183349609375 |

**prompt : Focus only on the primary subject and its core action in this image. Keep the caption short and clear. Avoid any extra details.**

| CIDEr | CLIPScore |
| --- | --- |
| 1.1392361473280939 | 0.7710675048828125 |

**Reference**

https://github.com/NielsRogge/Transformers-Tutorials/blob/master/LLaVa/Inference_with_LLaVa_for_multimodal_generation.ipynb (https://github.com/NielsRogge/Transformers-Tutorials/blob/master/LLaVa/Inference_with_LLaVa_for_multimodal_generation.ipynb)
https://huggingface.co/llava-hf/llava-1.5-7b-hf (https://huggingface.co/llava-hf/llava-1.5-7b-hf)

## Problem 2: Evaluation metrics report (10%)

### 1. Report your best setting and its corresponding CIDEr & CLIPScore on the validation data. Briefly introduce your method. (TA will reproduce this result) (5%)

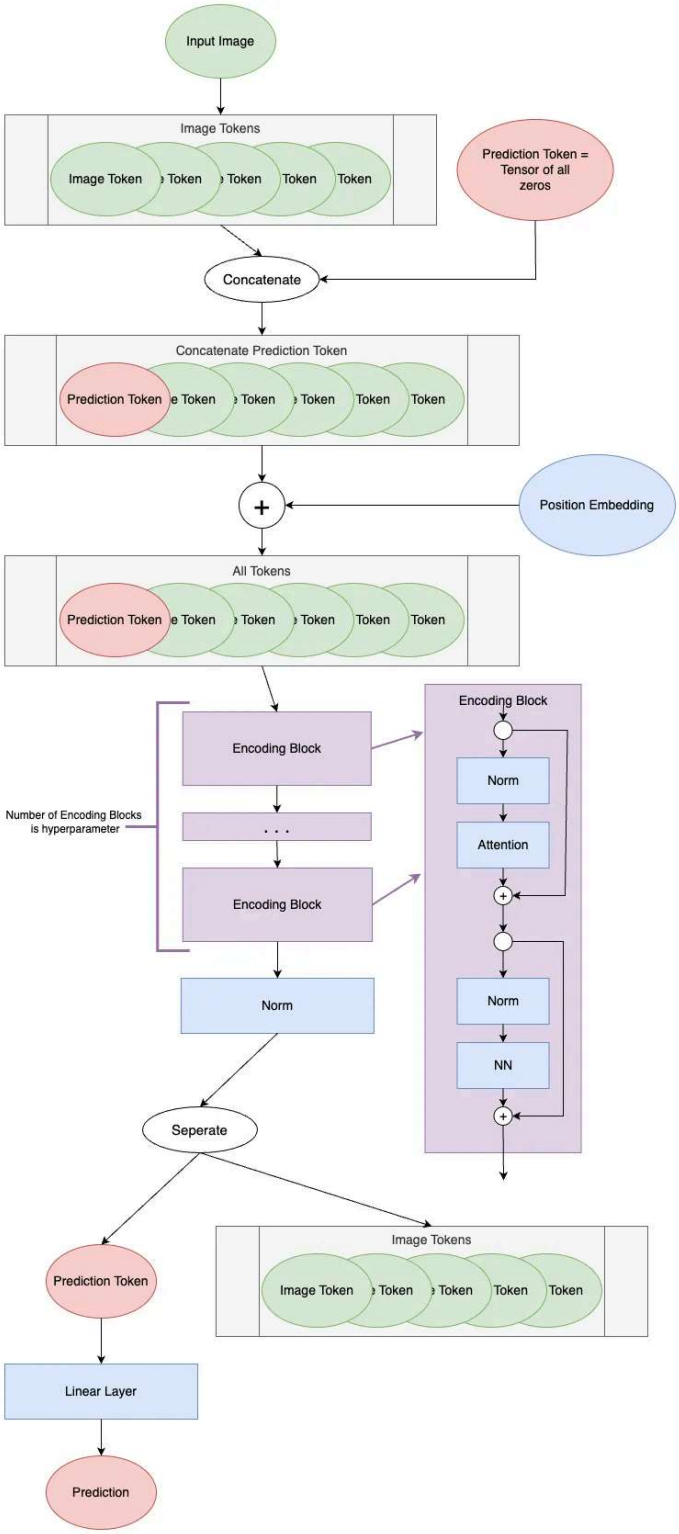
- ModelName = 'vit_gigantic_patch14_clip_224.laion2b'
- Trainable parameters (model_trainable_params): 7630112
- Batch Size : 32
- Learning rate : 0.0008
- Optimizer : AdamW
- Weight decay: 0.005
- T_max: 3
- Loss Function : nn.CrossEntropyLoss(reduction='mean')
- Projection Dropout : 0.1
- Lora Setting
  - rank : 32
  - lora_alpha=64
  - lora_dropout=0.1

| Epoch | CIDEr | CLIPScore |
| --- | --- | --- |
| 3 | 0.9005304286055559 | 0.7211166381835937 |

我參考了這篇貼文：https://discuss.huggingface.co/t/combine-word-embedding-with-visual-features-from-vit-model/25849 (https://discuss.huggingface.co/t/combine-word-embedding-with-visual-features-from-vit-model/25849) 其提到了將 text token 與 visual token 直接連在一起。但 concatenate text tokens 與 visual tokens 後，我一直都沒有辦法訓練成功。後來我把每個 decoder 的每個 tensor 都 print 出來，發現原先進去的 text token 的 embedding's tensor value 都為 0 附近。但 concatenate image token 後其 embedding's tensor value 都為負很多，接著我將其分布的 mean std 與 variance 後，發現兩者差很多：

```
# decoder's forward function
def forward(self, visual_embeds: Tensor, x: 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)
        # Check 1 :
        print(f'After token embedding: {x.mean()}, {x.min()}, {x.max()}')
        x = self.transformer.wte(x) + self.transformer.wpe(pos)
        x = torch.cat([visual_embeds, x], dim=1)
        # Check 2 :
        print(f'After concat: {x.mean()}, {x.min()}, {x.max()}')
        # Check 3 :
        print(visual_embeds.mean(), visual_embeds.min(), visual_embeds.max())
        x = self.transformer.ln_f(self.transformer.h(x))
        x = self.lm_head(text_output)
        # Check 4 :
        print(f'After transformer: {x.mean()}, {x.min()}, {x.max()}')
        return x
```

我發現這個 Check 4 的值太負，也就是這個輸出的結果被 image token dominate 。經過嘗試 Visual Embeddings Scale 沒有變更好。我看到這篇文章 Vision Transformers, Explained (https://towardsdatascience.com/vision-transformers-explained-a9d07147e4c8)




其在進到 linear layer 也就是下方程式碼 lm_head (Linear Layer) 前，在經過 transformer 的 ln_f (Layer Normalize) 後會將 text token 與 visual token 分開。並只將 text token

```
self.transformer = nn.ModuleDict(dict(
        wte=lora.Embedding(cfg.vocab_size, cfg.n_embd, r=16),
        wpe=lora.Embedding(cfg.block_size, cfg.n_embd, r=16),
        h = nn.Sequential(*[Block(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=16)
```

也因此我改成下方程式碼。

```
def forward(self, visual_embeds: Tensor, x: 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)
    x = self.transformer.wte(x) + self.transformer.wpe(pos)
    x = torch.cat([visual_embeds, x], dim=1)
    x = self.transformer.ln_f(self.transformer.h(x))
    # Take only the text embeddings as the prediction token to Linear layer
    text_output = x[:, visual_embeds.size(1):]
    x = self.lm_head(text_output)
    return x
```

但是後來我發現如果直接兩個模態的 token ，也就是 image token 與 text token 一起做 normalize 的話，會犧牲掉 text token，原因同上，因為兩者的數值範圍差很大。所以後來我就改為 normalize 前就將 text token 與 image token 分開，再拿 text token 去預測。

**2. Report 2 different attempts of LoRA setting (e.g. initialization, alpha, rank…) and their corresponding CIDEr & CLIPScore. (5%, each setting for 2.5%)**

**Setting 11**

```
Batch size: 32
Learning rate: 0.001
Projection dropout: 0.1
LoRA dropout: 0.1
Weight decay: 0.005
T_max: 3
alpha = 32
```

```
$ CUDA_VISIBLE_DEVICES=1 python3 evaluate.py --pred_file /project/g/r13922043/hw3_output/P2
s/val --annotation_file /project/g/r13922043/hw3_data/p2_data/val.json
PTBTokenizer tokenized 123146 tokens at 601252.16 tokens per second.
PTBTokenizer tokenized 24072 tokens at 194123.90 tokens per second.
CIDEr: 0.8107021322957735 | CLIPScore: 0.7139019775390625
rank 16 alpha 32
```

| Epoch | CIDEr | CLIPScore |
|---|---|---|
| 4 | 0.8107021322957735 | 0.7139019775390625 |

```
Batch size: 32
Learning rate: 0.0008
Projection dropout: 0.1
LoRA dropout: 0.1
Weight decay: 0.005
T_max: 3
alpha = 64
rank 16 alpha 64
```

| Epoch | CIDEr | CLIPScore |
|---|---|---|
| 3 | 0.9005304286055559 | 0.7211045837402343 |

## Problem 3 Visualization of Attention in Image Captioning (26%)

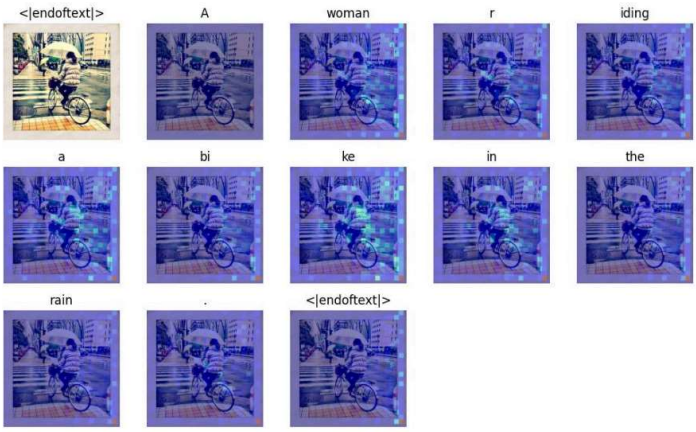
In this problem, you have to analyze your image captioning model in problem 1 and 2 by visualizing the self-attention between images tokens and generated captions' tokens.
use your best model in problem 2
Given an input image, your model would be able to generate a corresponding caption sequentially, and you have to visualize the self-attention between the image patches and each predicted word in your own caption.
The attention weights you have to visualize are in the self attention block in decoder
Self-attention weights in the last decoder layer (or the second to last) have better visulization results.
You should modify the code in problem 2 to get the attention weights for visualization.

**Report 1**

Given five test images ([p3_data/images/]), and please visualize the predicted caption and the corresponding series of attention maps in your report with the following template: (20%, each image for 2%, you need to visualize 5 images for both problem 1 & 2)

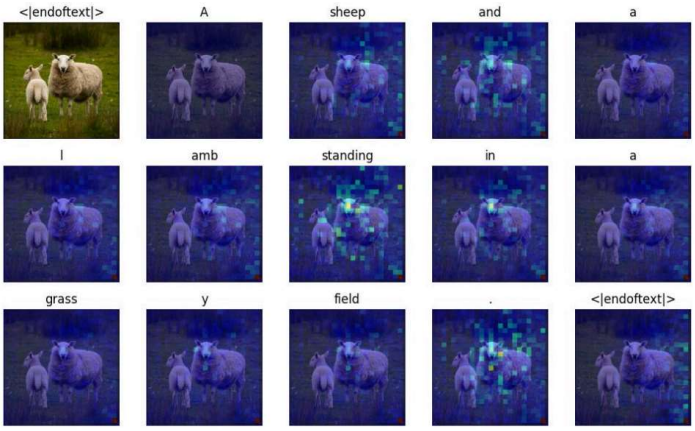
**PB1**

**bike**

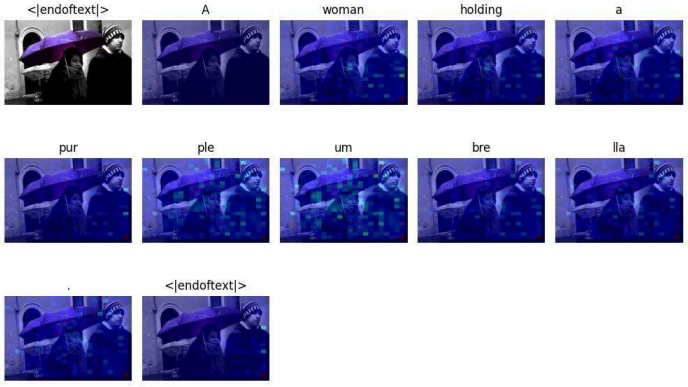


**girl**

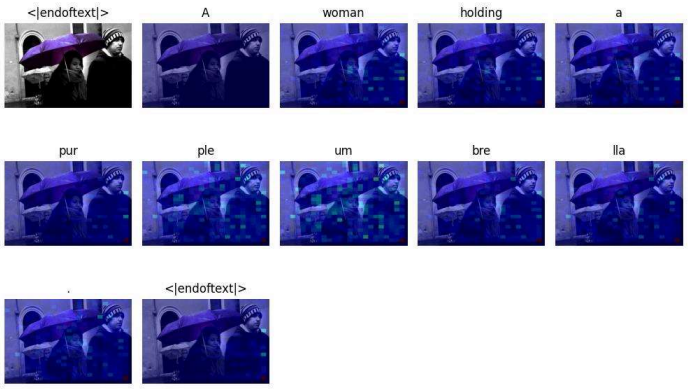

**sheep**




**sky**

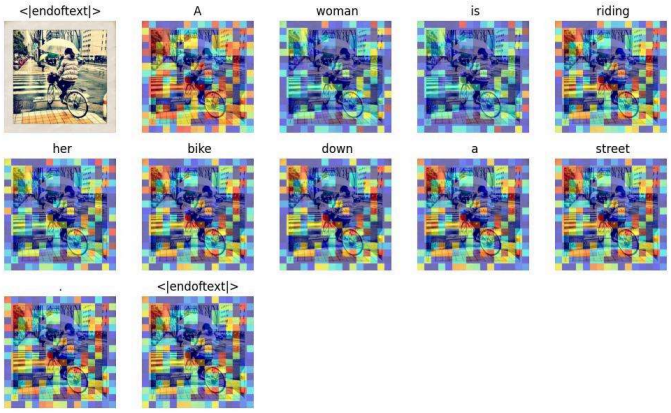


**umbrella**




**PB2**

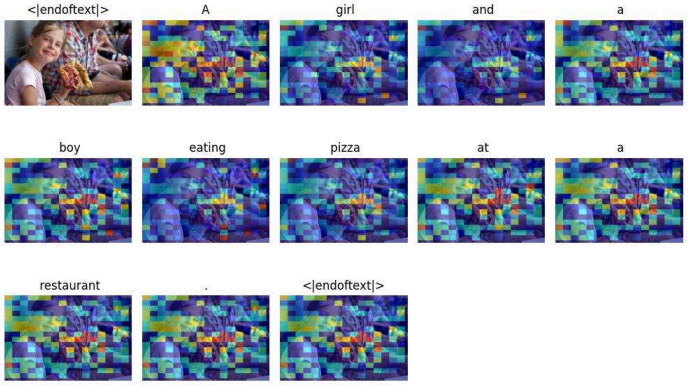
**bike**



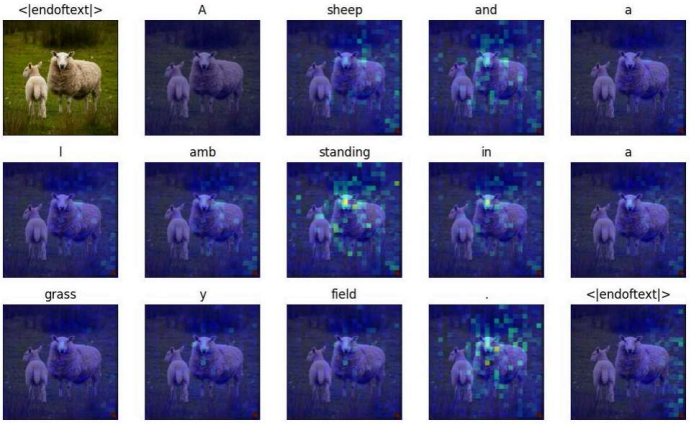

**Deep Learning for Computer Vision HW3**

HackMD(https://hackmd.io?utm_source=view-page&utm_medium=logo-nav)

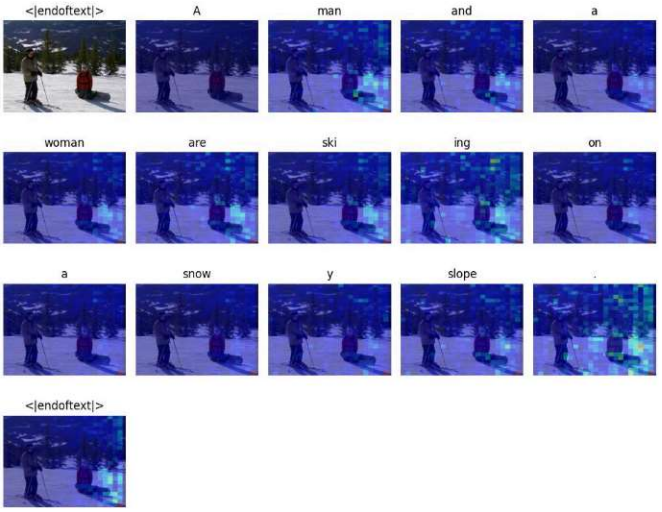
**girl**



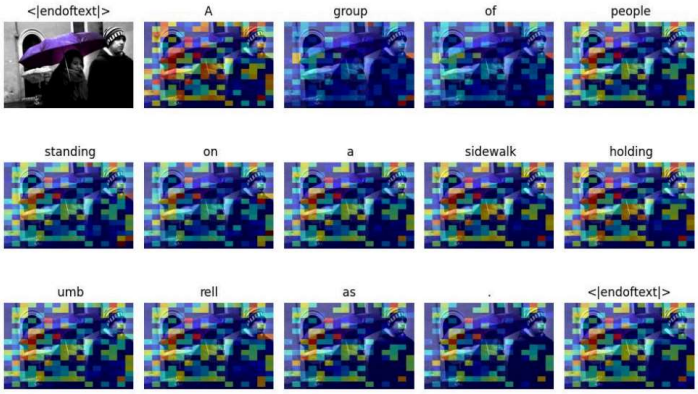

**sheep**




**sky**



**umbrella**



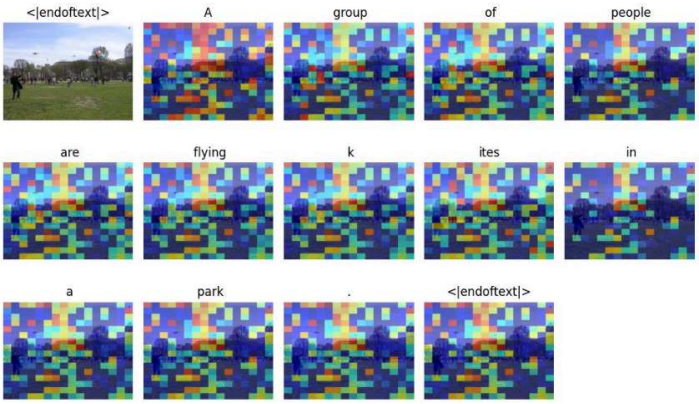

### Report 2

According to CLIPScore, you need to:
i. visualize top-1 and last-1 image-caption pairs
ii. report its corresponding CLIPScore in the validation dataset of problem 2. (3%)

| rank | file name | Column 3 |
|---|---|---|
| 1 | 000000001086 | 1.019287109375 |
| last | 000000000693 | 0.3631591796875 |

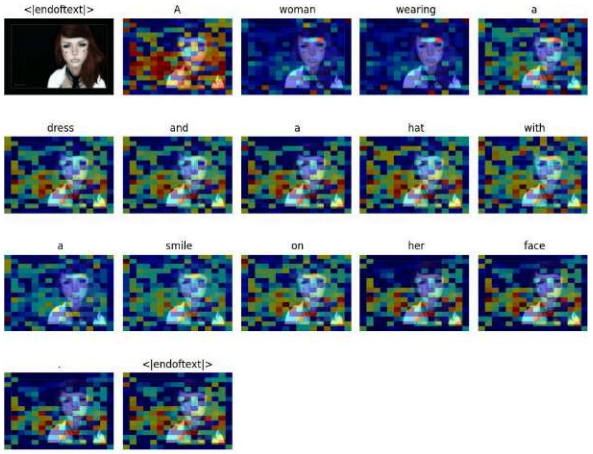
**000000001086**

"000000001086": "A group of people are flying kites in a park."



**000000000693**

"000000000693": "A woman wearing a dress and a hat with a smile on her face."




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? (3%)

我覺得attention maps 在第一張途中，因為風箏很多，然後天空也很大，人又很多，所以 attention map 總體關注的範圍很大，attention 關注的部分在中間居多，但其實也有捕捉到很多左右的資訊，我認為介係詞 in 關注的地方為中間很合理。因此他的生出來的 caption 最好。然而第二張圖，女生就在右邊而已，在 her 的時候可以看得比較清楚，attention 開始關注右邊的圖。然而像是一開始的 woman 只有少部份有關注到那個女生。

第一張圖的 caption 很合理，沒有任何一個字有問題。而最低分的那張圖，女生看不出有沒有穿 dress，也沒有 smile，也沒有帽子。不合理。

## Reference :

https://huggingface.co/blog/lora (https://huggingface.co/blog/lora)
Pruning and Merging of Tokens for Efficient VL Models: A Review
(https://medium.com/@vedantpalit10/pruning-and-merging-of-tokens-for-efficient-vl-models-a-review-5fa833a0c7e6)