1) **Image Captioning**

a) Compare the performance of 2 selected different pre-trained models in generating captions, and use the one you find the most effective for later problems.

我比較了兩種 pre-trained models 分別為 blip2-opt-6.7b-coco 以及 blip2-opt-2.7b，比較結果發現，blip2-opt-6.7b-coco 生成的 generated_text 文字描述較為精確詳細

下圖為 blip2-opt-6.7b-coco 產生的文字:

```
"generated_text": "a group of jellyfish swimming in the ocean under blue water",
"generated_text": "a puffin swimming in a pool of water with rocks and grass",
"generated_text": "a couple of two starfish are in an aquarium next to a rock"
```

下圖為 blip2-opt-2.7b 產生的文字:

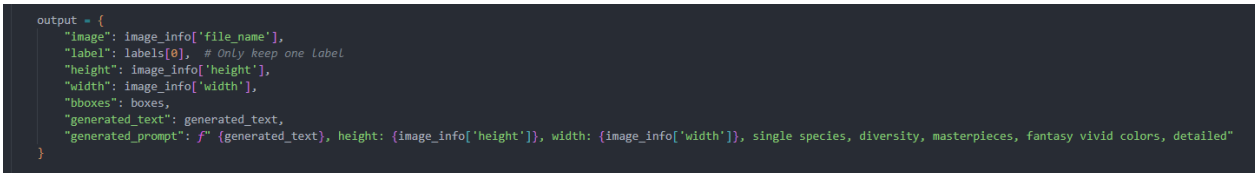
```
"generated_text": "jellyfishs in the aquarium", "generated_prompt"
"generated_text": "a puffin is swimming in the water"
"generated_text": "two starfishs in an aquarium",
```

從以上兩者同一張圖片生成的文字中可以看到前者所生成的文字對背景環境描述更加完整生動，所以選擇 blip2-opt-6.7b-coco 為比較有效的 pre-trained models 來處理後面的問題

(這邊作法是寫一個 BLIP-2 Inference 的 code, BLIP-2.py（裡面可以直接選取 7＊20 張較適合 GLIGEN inference 的照片）

b) Design 2 templates of prompts for later generating comparison

這裡如圖中我在生成文字的部分多設計了一個 prompt 後面加上一些資訊或是圖片描述，之後可以透過程式選擇要使用純文字生成的 template(generated_text)或是使用加上了一些額外的 prompt 的 template(generated_prompt) 去丟給 GLIGEN 生成圖片，這邊之所以只多設計一個 prompt 去執行後面的題目原因為，在初期嘗試的時候發現不管後面怎麼加文字，效果都差不多差，而且是很差，為了更好地去做區別，所以選擇了原始生成的 prompt 以及多加文字的 prompt 當作兩個 templates 做後續比較。



```
output = {
    "image": image_info['file_name'],
    "label": labels[0],  # Only keep one label
    "height": image_info['height'],
    "width": image_info['width'],
    "bboxes": boxes,
    "generated_text": generated_text,
    "generated_prompt": f" {generated_text}, height: {image_info['height']}, width: {image_info['width']}, single species, diversity, masterpieces, fantasy vivid colors, detailed"
}
```

2) Text-to-Image Generation

a) Use 2 kinds of generated prompts from Problem 1(b) to generate images. (text only!)

這裡以 generated_text 當成輸入的部分我把它設定成表格中的 Template#1，而以 generated_prompt 為輸入的部分我將它設定成 Template#2

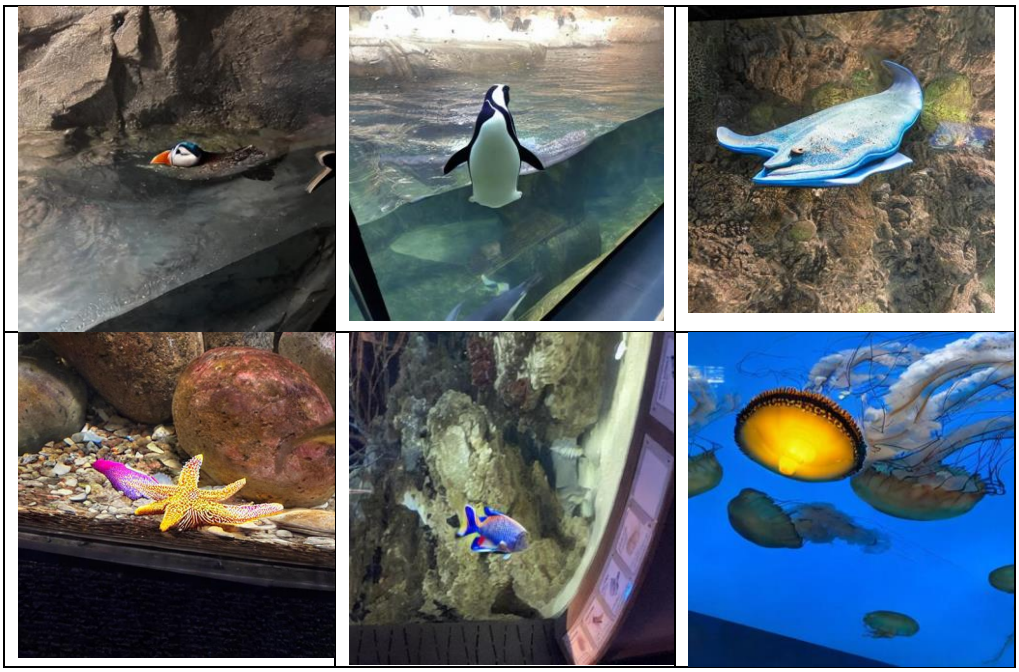
Template#1 生成的圖片的部分:

Template#2 生成的圖片的部分:

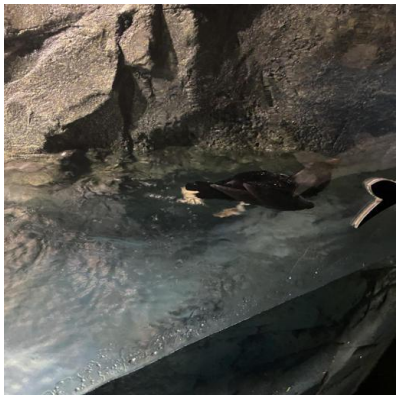
選擇圖片的時候都選擇同個檔案名稱做比較發現，其實在後面多加了一些字的情況下所生出來的圖片肉眼看起來也不一定會比較好，但整

體看下來，多加了一些字之後，整個畫面比較鮮明豐富，而且感覺合理性也夠，確實也有符合我給的 prompt，所以我一開始認為是比較好，把 Template#2 作為 better-generating results 當作 Text grounding 和 image grounding 文字部分的輸入生成圖片後在去分別訓練 DINO，不過到後面發現 Template#1 生成的圖片的 FID 比 Template#2 的好，而且 Template#1 生成的圖片丟進 DINO 做 training 也是比 Template#2 的好上很多(後面有做比較)，這樣的結果我想應該是因為我雖然多給了額外的 prompt，但是可能給的詞彙用語反而會影響真實度，就算肉眼看起來相對合理，可是以結果來說，多加的贅詞可能效果還會比原本差，不過 Image grounding 是用什麼 Template 去當文字輸入對後續 FID 以及 DINO 的 AP 差別就不大了(最後也有解釋可能的原因)

b) Select the prompts for better-generating results, and perform image grounding generation. (text + image)



在 Image grounding 的部分(我是使用 Image grounding for inpainting)，用跟 Text grounding 相同檔名的圖片去做比較發現，確實生成圖片的部分更加符合實際情況，而且也不失圖片資訊的豐富度，例如左上第一張拿原始圖片來做比較



```
"generated_text": "a puffin swimming in a pool of water next to a rock wall"
```

3) Table of your performance based on FID

	Text grounding		Image grounding
prompt	Template #1	Template #2	Template #?
FID	145.734	152.225	95.028

```
(yolov8) weiwei@weiwei:/data/CVPDL3/pytorch-fid$ python -m pytorch_fid ./resized_picture ../GLIGEN/generation_samples/text_prompt1
100%|                                                                                                | 3/3 [00:01<00:00,  2.93it/s]
100%|                                                                                                | 3/3 [00:00<00:00,  4.59it/s]
FID:  145.73447668328345
```

```
(yolov8) weiwei@weiwei:/data/CVPDL3/pytorch-fid$ python -m pytorch_fid ./resized_picture ../GLIGEN/generation_samples/text_prompt2
100%|                                                                                                | 3/3 [00:01<00:00,  2.70it/s]
100%|                                                                                                | 3/3 [00:00<00:00,  3.85it/s]
FID:  152.22529883993408
```

```
Downloading: "https://github.com/mseitzer/pytorch-fid/releases/download/fid_weights/pt_inception-2015-12-05-6726825d.pth" to /home
/weiwei/.cache/torch/hub/checkpoints/pt_inception-2015-12-05-6726825d.pth
100%|                                                                                              | 91.2M/91.2M [00:03<00:00, 27.3MB/s]
100%|                                                                                                | 3/3 [00:01<00:00,  2.34it/s]
100%|                                                                                                | 3/3 [00:00<00:00,  4.07it/s]
FID:  95.02752724780606
```

4) Table of the improvement of your detection model from HW1 after data Augmentation

(checkpoint0011_4scale.pth)

	Before Data Augmentation	After Data Augmentation (Text grounding)	After Data Augmentation (image grounding)
$AP_{[50:5:95]}$	0.508	0.273	0.302

Before Data Augmentation:

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.508
```

After Data Augmentation (Text grounding)(Template #1 result)

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.273
```

After Data Augmentation (Text grounding)(Template #2 result)

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.019
```

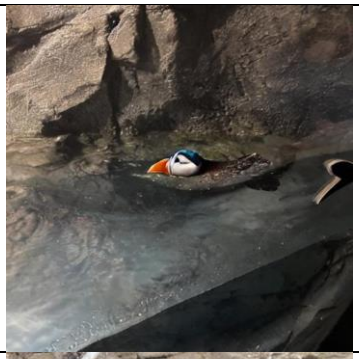
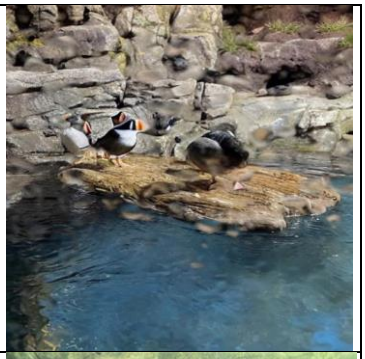
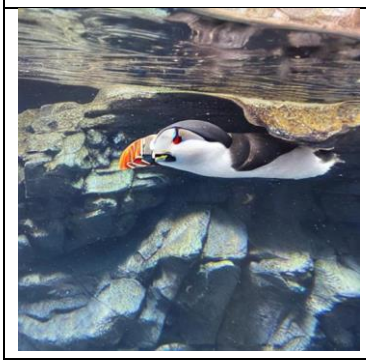
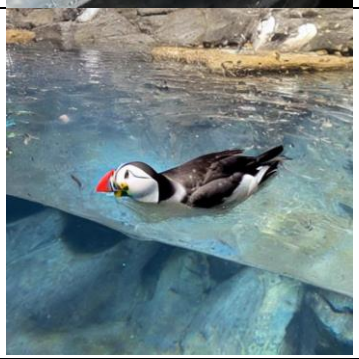
After Data Augmentation (image grounding)

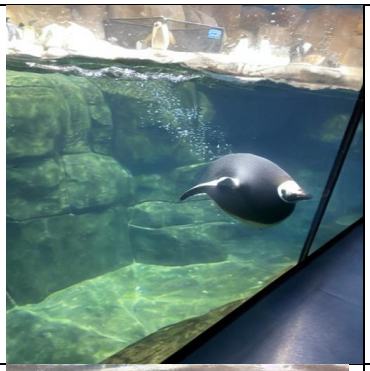
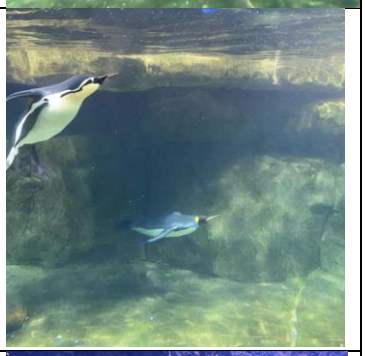
```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.302
```

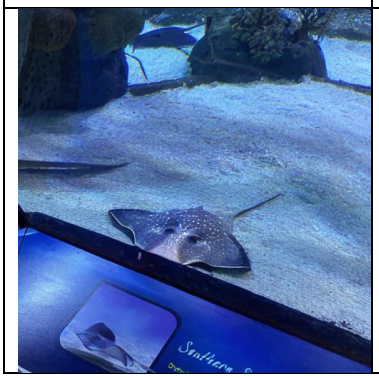
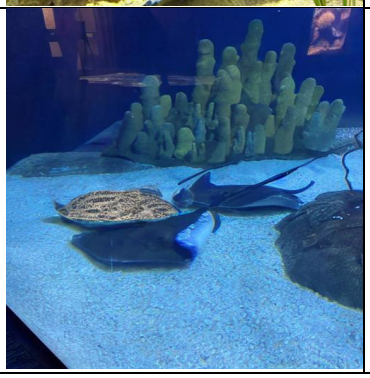
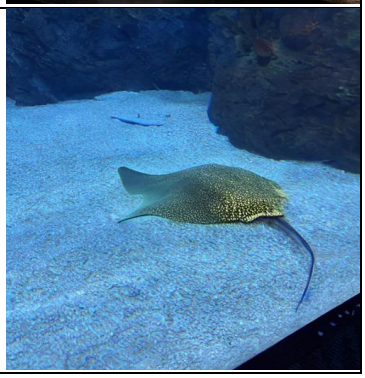
其實從這些結果就可以看出相比 Augmentation 前，不管是 Text grounding 或是 image grounding 的 Augmentation 都不增反減很多，而且其實若是單獨比較 Text grounding 跟 image grounding 的話會發現他們之間是差不了太多的，這就代表對於我 GLIGEN 生成的圖片 140 張再加上 140 張同樣檔名的原始圖片總共 280 張圖片其實都沒有辦法真正達到資料平衡，反而是增加了許多不必要的 noise，原因我想就是因為從 GLIGEN 生成的圖片中，總是會有幾張奇怪的圖像生成，這些圖像的生成反而會在與原始圖像融合之後使得訓練品質不佳，就算資料數是平衡了沒錯，可是產生的 noise 卻遠遠大過平衡帶來的好處，而這其實可以從 Text grounding 跟 image grounding 之間的差距可以證明，因為前面有圖片比較，以人類視覺來看，以及 FID 比較都可以看出，無論如何 image grounding 生成的圖像質量還是比 Text grounding 好，只是在 DINO 的訓練資料集裡卻都稱不上是好的生成圖像，所以才會造成兩者雖然有些微落差，可是放大到跟原始資料集比較，則兩者都很差。

5) Visualization

show the best 5 images for each category (35 images in total!)

puffin		
		

penguin		
		
fish		
		
jellyfish		

		
shark		
		
stingray		
		

starfish