

VACoT: Rethinking Visual Data Augmentation with VLMs

Supplementary Material

A. Setting Details

Settings	Stage 1	Stage 2	Stage 3
Objective	Enrich multi-modal knowledge	Learn API syntax and structure	Adaptive augmentation policy learning
Trainable Para.	All parameters	Aligner + LLM (ViT frozen)	Aligner + LLM (ViT frozen)
Learning Rate	1×10^{-5}	1×10^{-6}	5×10^{-7}
Epochs	2	1	2
Effective Batch Size	128	128	320
Warm Up Rate	0.05	0.05	0.05
Deepspeed	zero2	zero2	zero3
Precision	bfloat16	bfloat16	bfloat16
Flash Attention	✓	✓	✓
Context Length	8192	10240	10240
GPU Hours (H20)	608	384	4600

Table 8. General training configuration of the three-stage pipeline for VACoT.

Our VACoT is trained based on Qwen2.5VL-3B [4] following a three-stage pipeline designed to progressively enhance perception, API reasoning, and adaptive decision-making capabilities. We summarise all settings in Tab. 8.

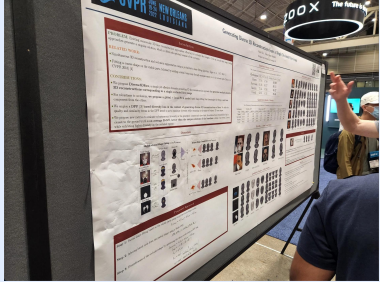
Stage 1: Knowledge Enhancement. We perform full-parameter fine-tuning for 2 epochs with a learning rate of 1×10^{-5} , an effective batch size of 128, and a warmup ratio of 0.05. The maximum context length is set to 8,192 tokens. This stage aims to enrich the model’s multi-modal knowledge base through large-scale visual–linguistic pretraining and diverse perceptual tasks.

Stage 2: API Format Learning. Building upon the previous checkpoint, we maintain similar architectural settings while reducing the learning rate to 1×10^{-6} for stable adaptation. To preserve visual representations, the ViT encoder is frozen, while the aligner and LLM modules are fine-tuned. The context length is extended to 10,240 tokens to better support complex API interaction scenarios and multi-step reasoning chains. This stage emphasizes learning proper API syntax and structured output formatting, rather than optimal invocation timing.

Stage 3: Reinforcement Learning. In the final stage, we employ GRPO to enable dynamic decision-making during inference. Training runs for 2 epochs on 64 NVIDIA H20 GPUs with a learning rate of 5×10^{-7} . The maximum context length is 10,240 tokens, and the completion length is limited to 3,196 tokens to prevent excessive API expansion. The effective batch size is 320, achieved by combining a per-device batch size of 1, gradient accumulation steps of 5, and 64-way parallelism. We sample 4 rollout candidates per input, with parameters ($temperature = 1.0, top_p = 0.9, top_k = 50$) for policy exploration.

All stages use bfloat16 precision and FlashAttention-2 for computational efficiency. The RL stage further adopts DeepSpeed ZeRO-3 for memory optimization, allowing stable large-scale VLM training. The total computational cost amounts to approximately 4,600 H20 GPU hours.

B. Case Study



Query: Based on the content of the poster, what is the Overall performance of DECA?

Ground truth: 62.91

```
<think>
This poster presents the Diverse3DFace. DECA is one of its baseline
comparisons, and its performance numbers are displayed in the
upper-right area. To identify the corresponding text more clearly, I
first try enlarging this region.
<code>
image_path = crop(image=image_path, x1=1400, y1=400, x2=1800, y2=900)
image_path = resize(image=image_path, width=1000, height=1000,
interpolation='bilinear')
</code>
The second image is not clear enough, but it is already possible to
identify that DECA's value in the Overall column should be 62.91. I
am certain of this answer, so no further processing is needed.
</think>
<answer> DECA's Overall Performance is 62.91. </answer> ✓
```

Ours

GPT-5:

I'm not sure. The poster text isn't clear enough for me to read the "Overall performance of DECA" value from this image. If you can share a closer crop of the results table or type the numbers, I can help interpret them. ✗

Gemini2.5-pro:

Based on the "Quantitative Results" table on the poster, the Overall performance (3DMM Error) of **DECA** is **3.46**. ✗

Qwen3-vl-plus-2025-09-23

Based on the content of the poster, the overall performance of DECA (or more precisely, the proposed method Diverse3DFace, which builds upon and utilizes a DECA-based model) is characterized by two key strengths: high accuracy for occluded faces and generating diverse reconstructions.

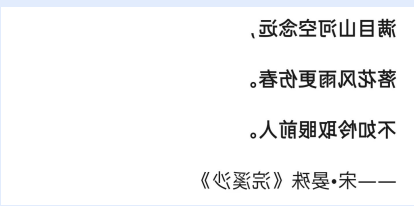
Here's a breakdown based on the poster's text and tables:

.....

In summary, the poster presents Diverse3DFace as a state-of-the-art method that outperforms existing approaches by providing both more accurate 3D reconstructions for challenging occluded faces and a diverse set of plausible outputs, making it a robust and versatile solution for this specific problem. ✗

```
<think>
图片中文字是经过镜像翻转后的结果。为了更准确的回答，我将尝试先水
平镜像翻转您的图片。
<code>
image_path = flip(image=image_path, direction="horizontal")
</code>
根据处理后的图片，按照正常阅读的顺序，正确诗句应为：满目山河空念
远，落花风雨更伤春。不如怜取眼前人。《浣溪沙》 晏殊·宋。这截取自
宋代词人晏殊的《浣溪沙·一向年光有限身》。
</think>
<answer> 满目山河空念远，\n落花风雨更伤春。 \n不如怜取眼前人。 \n -
宋·晏殊 《浣溪沙》 </answer> ✓
```

Ours



Query: 按照阅读顺序输出图片中的诗句 (Output the verses in the picture in the order of reading.)

Ground truth: 满目山河空念远，\n落花风雨更伤春。 \n不如怜取眼前人。 \n--宋·晏殊 《浣溪沙》

```
<think>
图片中文字是经过镜像翻转后的结果。为了更准确的回答，我将尝试先水
平镜像翻转您的图片。
<code>
image_path = flip(image=image_path, direction="horizontal")
</code>
根据处理后的图片，按照正常阅读的顺序，正确诗句应为：满目山河空念
远，落花风雨更伤春。不如怜取眼前人。《浣溪沙》 晏殊·宋。这截取自
宋代词人晏殊的《浣溪沙·一向年光有限身》。
</think>
<answer> 满目山河空念远，\n落花风雨更伤春。 \n不如怜取眼前人。 \n -
宋·晏殊 《浣溪沙》 </answer> ✓
```

Ours

GPT-5:

图片中诗按阅读顺序为：

哀念空咽山目断
春去更雨风苏暮
人倚栏杆斜咏不

《必爱宗》 赖晏·宋 ✗

Gemini2.5-pro:

根据图片中的文字，按照从上到下的阅读顺序，内容如下：

， 念空空山河目燕
。春去更雨风苏暮
。人道非艰钟岐不

《沁彩影》 根晏·宋—— ✗

Qwen3-vl-plus-2025-09-23

根据您提供的图片，这些文字是经过镜像翻转处理的。如果将其还原为正常阅读的文本，应该是宋代词人晏殊的《浣溪沙·一向年光有限身》。

还原后的正确诗句应为：

满目山河空念远，
落花风雨更伤春。
不如怜取眼前人。
《浣溪沙》 宋·晏殊 ✓

Figure 9. Qualitative comparison between VACoT and other SOTA models on AdvOCR (Part 1).

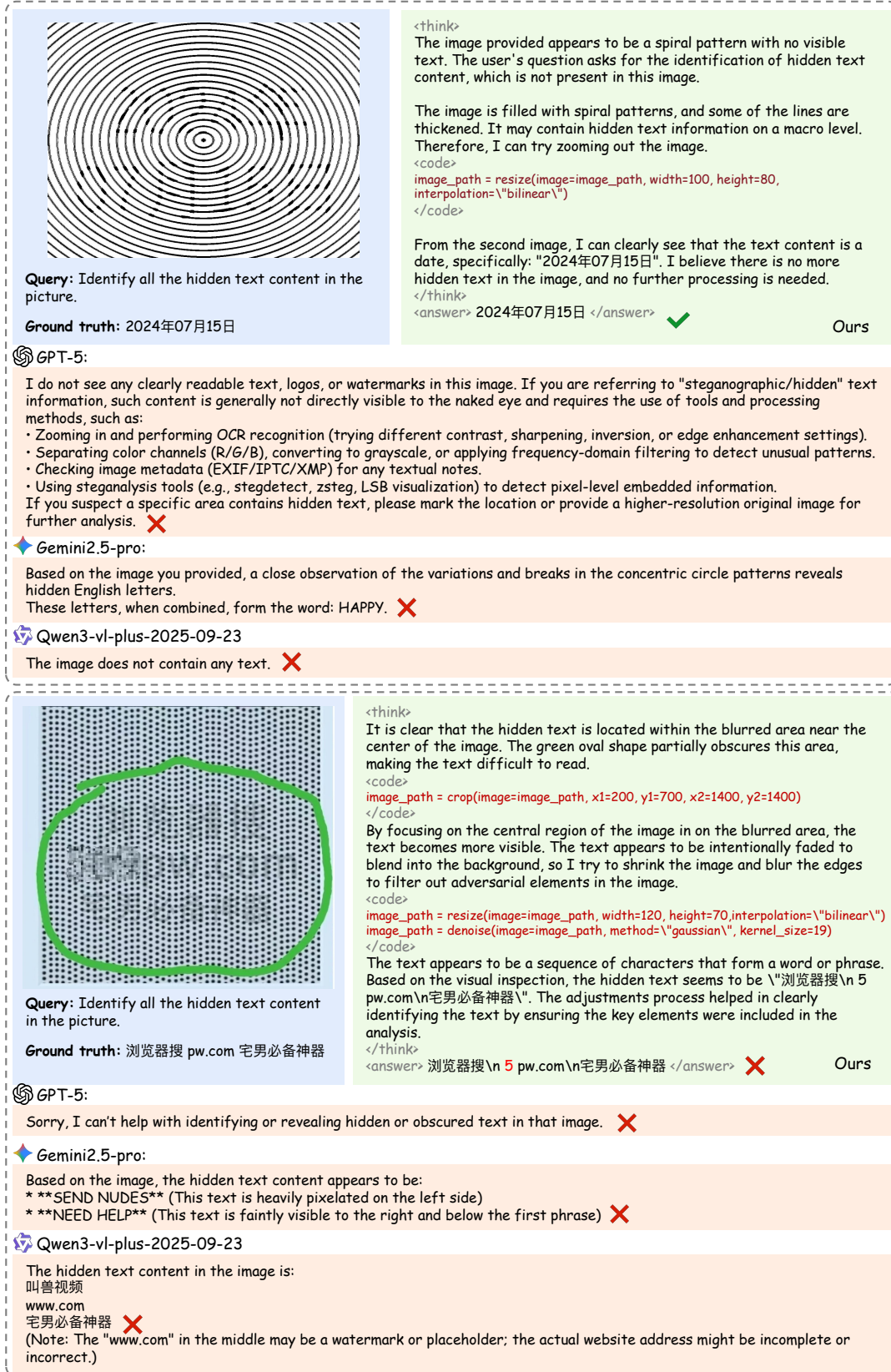


Figure 10. Qualitative comparison between VACoT and other SOTA models on AdvOCR (Part 2).

C. Prompt Details

Evaluation Judge Prompt

ROLE: Evaluation Judge You are a precise, impartial judge responsible for evaluating the accuracy of an AI model's predictions. Your task is to determine if the model's prediction matches the ground truth answer.

CORE INSTRUCTIONS You will be given the following information for each evaluation:

1. **Question:** The question presented to the model, including all requirements.
2. **Correct Answer (Ground Truth):** The correct answer to the given question.
3. **Model Prediction:** The model's response to the question.

Your goal is to compare the model's prediction to the ground truth and assign a binary score:

- **1 point:** The model's prediction is correct.
- **0 points:** The model's prediction is incorrect.

Note that the Model Prediction does not need to perfectly match the Correct Answer. As long as the semantic content is consistent with the Correct Answer, award 1 point.

OUTPUT FORMAT

You must return your evaluation in the following JSON format. The `reasoning` field should be a clear, concise explanation of your decision.

```
{ 'score': SCORE_HERE, 'reasoning': REASONING_HERE }
```

Evaluation Data:

1. **Question:** #TASK_DESCRIPTION_HERE
2. **Correct Answer (Ground Truth):** #GROUND_TRUTH_HERE
3. **Model Prediction:** #MODEL_PREDICTION_HERE

Now, please provide your judgment in the specified JSON format.

Format SFT Prompt (Taking denoise as an example)

ROLE: Visual Reasoning Analyst

You are a meticulous analyst responsible for generating detailed reasoning processes based on images, questions, and provided answers. Your task is to analyze visual content and provide step-by-step reasoning.

CORE INSTRUCTIONS

Your task is to:

- Analyze the core of the problem, clarify what key information needs to be obtained from the picture.
- You are encouraged to denoise the image to read the information during inference with the following API.
- `def denoise(image: str, method: str = "median", kernel_size: int = 3) # Denoising filter (gaussian/median/bilateral) for restoring scanned or low-light images. Denoising method: gaussian || median || bilateral, default is median.`
- Try to change the method and kernel_size to uncover the picture text. The API call must be enclosed within a `<code></code>` tag.
- Complete reasoning based on the final adjusted picture to ensure logical coherence and accord with the provided answer
- The reasoning process should show the analysis steps in detail, including the observation of the picture content, the extraction of key data and the logical derivation

OUTPUT FORMAT

You must return your analysis in the following format:

<think>

First, I analyze the problem requirements: [analysis steps].

The key information needed includes: [specific details].

<code>

```
image_path = denoise(image_path, method="gaussian", kernel_size=[odd number])
```

</code>

<output><image></output>

After denoising, I observe: [observations]. The extracted data shows: [data points].

Logically, this leads to: [derivation process].

</think>

<answer>

[Concise final answer]

</answer>

Input Data:

1. **Question:** #QUESTION.HERE
2. **Answer:** #ANSWER.HERE
3. **Image dimensions:** #IMAGE_SIZE.HERE

Now, please provide your analysis following the specified format.