Vision-Language Models and Multimodal Post-Training

Applied Deep Learning — Chapter 19

Yin Yang

Foundation Books Project

Roadmap

The Running Example and the Contract

From Encoder to Image Tokens

Prompt-Only Baseline

Error Taxonomy

Multimodal Instruction Tuning

LoRA for VLM Adaptation

Reinforcement Post-Training

Evaluation, Reporting, Cost

Summary

The Running Example and the Contract

What a Vision-Language Model Does

A VLM consumes **image + text** and emits **text**. Running example: an image contains a small math expression, table, chart, or diagram. The user asks the model to answer in `<answer>` tags. **Desired output for** $7 + 5 \times 3$**:**

```
<think>Multiplication first: 5*3=15. 7+15=22.</think>
<answer>22</answer>
```

Small enough to debug; rich enough to fail in many ways.

Define the Contract First

Before tuning anything, pin down:

- input format (image + text + chat template)
- required answer format (`<answer>` tags?)
- parser (regex? whitespace tolerance? units?)
- evaluator (exact match? equivalence rules?)

"The answer is twenty-two" may be correct but fail an automated evaluator expecting `<answer>22</answer>`. The model can look smart while the evaluator correctly disagrees.

Two Different Numbers

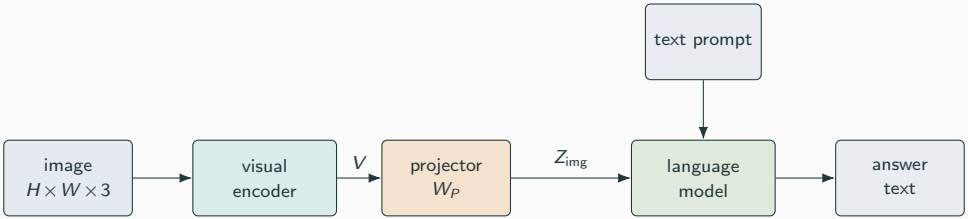
100 held-out visual math prompts, parser finds valid tags in 82, of which 61 are correct.

- **Valid-format rate:** $82/100 = 82\%$
- **Exact-match (over all):** $61/100 = 61\%$
- **Exact-match (over parseable):** $61/82 \approx 74.4\%$

Different statements — formatting failures vs reasoning failures call for different fixes. Report both.

From Encoder to Image Tokens

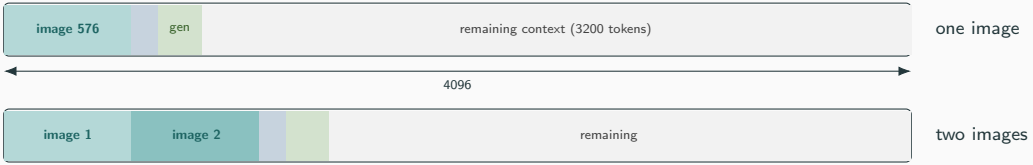
The Bridge: Image Tokens



Linear projector: $Z_{\text{img}} = VW_P + \mathbf{1}b_P^\top$, $W_P \in \mathbb{R}^{d_v \times d_{\text{LM}}}$. Image tokens occupy context positions and participate in attention like text.

Image Tokens Cost Context

Context budget = max input + generated tokens. Example: 4096-token budget.



Visual detail competes with prompt length and generation length.

VLM Architecture Families

LLaVA visual instruction tuning ties image + instruction + response.

BLIP-2 keeps the image and language giants frozen; learns a small Q-Former between them.

CLIP encoders reused as visual checkpoints; alignment is in embedding space.

The bridge controls what evidence the LM ever sees. If the bridge drops a key value, the LM may fluently explain the wrong evidence.

Prompt-Only Baseline

Freeze the Baseline

Before any fine-tuning, run the pretrained model with:

- fixed checkpoint
- fixed image preprocessing (resize, normalize, color)
- fixed prompt wording
- fixed decoding (greedy / temperature / max tokens)
- fixed parser
- fixed held-out prompts

Without this, an improvement after tuning may simply be a better prompt or a fixed parser.

Decoding Choices Affect the Score

- **Greedy:** highest-probability next token. Deterministic.
- **Sampling (temperature, top-p):** useful for diversity, but adds noise to evaluation.
- **Max new tokens:** 16 may cut a reasoning trace short; 512 may add latency and irrelevant text.

Visual math has a definite answer, so start with greedy unless there is a specific reason to sample.

Error Taxonomy

Five Distinct Failure Types

A single accuracy number hides them.

Type	Example	First intervention
Perception	reads 5 as 6	higher resolution; OCR-aware checkpoint
Layout	uses wrong table row	doc-aware model; prompt examples
Instruction follow	omits `<answer>` tags	format SFT or format reward
Reasoning	reads correctly, computes $7+15=21$	reasoning examples; correctness reward
Parser	answer correct, parser rejects extra text	parser unit tests

Parser Errors Become Training Signals

In RL, every parser bug is part of the reward function. Cases worth testing:

- multiple `<answer>` tags
- answer with units (`22 cm` vs 22)
- leading whitespace, trailing comma
- numerical equivalence (22 vs `22.0`)

Test rewards like code — once RL starts, every edge case is part of the objective.

Leakage Errors

Contamination between train, validation, and test:

- same rendered expression in training and final eval
- validation set used repeatedly to tune prompt + reward
- rendered images from the same generator in both splits

A larger model only makes a contaminated score more expensive. The fix is to separate data and decisions *before* the next run.

Multimodal Instruction Tuning

Supervised Fine-Tuning, Now Multimodal

Pair (image, instruction, target response). Loss only on response tokens.

$$\mathcal{L}_{\text{VLM-SFT}}(\theta) = -\frac{1}{\sum_t m_t} \sum_t m_t \log p_\theta(y_t \mid x_{\text{img}}, u, y_{<t})$$

where $m_t = 1$ on assistant-response positions, 0 on image, instruction, and padding. Mask errors can make the model learn the prompt, the padding, or nothing useful.

Format Tuning

A small consistent SFT set can raise valid-format rate from 60% to near 100% without improving the model's reasoning.

- Useful: the model becomes *evaluable*.
- Misleading: same exact-match accuracy may now look like a perception gain.

Build the validation prompt set to pull failure types apart: small symbols, table layouts, easy format-only checks.

LoRA for VLM Adaptation

Where Do You Place the Adapter?

Adapter placement is an experimental decision, not cosmetic.

Language layers teach answer style and reasoning patterns; visual evidence unchanged.

Vision layers / projector change what evidence reaches the LM; raise memory and overfit risk.

Combined more flexibility, more parameters, more failure modes.

Report: selected modules, rank, trainable parameter count, precision, hardware.

Trainable Percentage

7B-parameter VLM with 20M trainable LoRA parameters:

$$\frac{20{,}000{,}000}{7{,}000{,}000{,}000} \times 100 \approx 0.286\%.$$

Rank-16 adapter on a 4096×4096 projection adds

$$4096 \cdot 16 + 16 \cdot 4096 = 131{,}072 \text{ trainable parameters}$$

for that one matrix. Small relative to the frozen projection — but worth recording.

Reinforcement Post-Training

Reward Functions for Visual Math

Many answers are easy to score: parse the answer, compare with the label. Simple reward:

$$R(y) = R_{\text{correct}}(y) + R_{\text{format}}(y).$$

- correct + valid tags $\Rightarrow$ 1.2
- correct without tags $\Rightarrow$ 1.0
- wrong with valid tags $\Rightarrow$ 0.2
- wrong without tags $\Rightarrow$ 0.0

If format reward is too large, the model learns to wrap pretty tags around bad answers.

GRPO with Image-Conditioned Prompts

Group several completions for the *same* (image, question), normalize rewards within the group. Group-relative advantage:

$$A_i = \frac{R_i - \bar{R}_{\text{group}}}{s_{\text{group}} + \epsilon}.$$

Clipped policy update with KL penalty (DeepSeekMath-style):

$$\max_{\theta} \frac{1}{G} \sum_i \sum_t \min\left(r_{i,t} A_i,\ \text{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon) A_i\right) - \beta\ \text{KL}_{i,t}\,.$$

Reward-Hacking Risks (Visual Edition)

- Format reward too high $\Rightarrow$ pretty tags around wrong numbers.
- Parser accepts malformed text $\Rightarrow$ malformed text gets reinforced.
- Training prompts leak into eval $\Rightarrow$ memorization scored as skill.
- Visually ambiguous crops $\Rightarrow$ rewards reflect the parser's confusion, not the model's vision.

Build adversarial reward unit tests: wrong-but-tagged, right-but-untagged, empty answer, multiple tags, layout-dependent reading order.

Evaluation, Reporting, Cost

Two Numbers, Together

Exact-match and valid-format must be reported separately. Reference run on 30 visual-math prompts (Qwen2.5-VL-3B-Instruct):

Run	Change	Exact	Valid	tok/s
baseline	default contract	28/30	24/30	16.17
strict prompt	answer contract	28/30	30/30	17.17

Same exact-match, cleaner contract — a real gain in evaluability, not reasoning.

Latency, Throughput, Memory

All in the same table as accuracy. Example: 96 output tokens in 4.8s after 0.7s first-token latency.

- generation throughput: $96/4.8 = 20$ tok/s
- end-to-end: $96/5.5 \approx 17.5$ tok/s

3B-parameter VLM in fp16: roughly $3{\times}10^9 \cdot 2\,\mathrm{B} = 6$ GB for base weights alone. 4-bit quantization ~ 1.5 GB before activations and KV-cache.

Context Cost is Quadratic

Increasing interaction from 896 to 1472 positions changes attention work by about

$$(1472/896)^2 \approx 2.70 \times .$$

Reasons to compress visual evidence (querying module, smaller patch grids, pooling tokens) are practical, not aesthetic.

What a VLM Report Should Contain

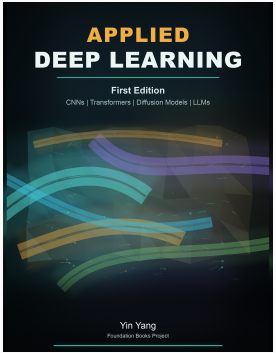
- **Data:** source, prompt count, split rule, generation procedure.
- **Run:** checkpoint, image size, preprocessing, prompt template, decoding.
- **Scoring:** parser tests, exact-match and valid-format rules, reward weights.
- **Adaptation:** method, LoRA rank, target modules.
- **Result:** validation *and* final test, latency, throughput, peak memory.
- **Errors:** perception, layout, format, reasoning, parser breakdown.

Summary

Summary

- A VLM joins a visual encoder, a projector, and a language model. Image tokens cost context.
- The contract (input, parser, metric) is part of the model.
- Failures separate into perception, layout, format, reasoning, parser, and leakage — each calls for a different fix.
- Adaptation: prompt-only baseline $\rightarrow$ multimodal SFT $\rightarrow$ LoRA $\rightarrow$ reinforcement post-training.
- Test rewards like code; report exact-match and valid-format together; scope claims to the data.

Next: when the output becomes spatial — boxes (object detection), masks (segmentation).



Applied Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/applied-deep-learning`
Code	`github.com/foundation-books/applied-deep-learning-code`
Print	`amazon.com/dp/B0GZF7QRSH`
Kindle	`amazon.com/dp/B0GZF9221X`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.