ChessMates: How good are VLLMs at Chess?

Rahul Chand
Stanford
rahulsc@stanford.edu

Abstract

In this project, I explore the capabilities of open-source Visual Large Language Models (VLLMs) at chess. VLLMs have shown impressive performance across various benchmarks from visual question-answering to image captioning and understanding. Though this is impressive, such benchmarks often either fail to test fine-grained visual perception of VLLMs or their reasoning capabilities. On the other hand, playing chess requires both a fine-grained visual understanding of the board and also good reasoning capabilities to select the best moves. Therefore, this project aims to quantify how good current open-source VLLMs fare at this challenging benchmark. Github: https://github.com/RahulSChand/chess_vlm

1. Introduction

Vision Large Language Models (VLLMs or VLMs) [2] have shown impressive performance across variety of benchmarks from visual question answering (i.e. given an image, answer question such as describing the objects in it), OCR (i.e. given a text-rich image correctly transcribe the contents) and reasoning (i.e. solve a puzzle or math problem in the form of an image). Though this is impressive, these tasks often test one of two things (1) The fine-grained perception ability of VLLMs, that is, how good and accurately it can pick up the details in the image or (2) Reasoning in visual space.

Chess is a game which requires both a fine-grained visual understanding to figure out which board pieces are on which squares and deep reasoning ability to figure out what the next move should be. Moreover, chess is a widely popular game played by millions. This makes it both an interesting, and more importantly a challenging benchmark to test the ability of VLLMs. On a high level, this project aims to answer the following questions

(1) How good are off the shelf VLLMs at chess.

(2) What is the increase in performance you can get via standard fine-tuning (either full or LoRA [7])

(3) In img-to-img scenario, do autoregressive models img-to-img models perform better than diffusion models.

2. Literature Review and Background

VLLMs: The emergence of LLMs [3] has revolutionized the field of AI. LLMs are autoregressive models based on transformer architecture trained on a huge corpus of data. LLMs exhibit impressive few-shot and zero-shot ability, that is, ability to perform well on tasks either not seen during pre-training or perform reasonably well after provided just a few examples. Vision Large Language Models (VLLMs) extend the strength of LLMs to images. VLLMs are often trained by tokenizing images into patches (similar to ViT [4]) and then projecting both image tokens (patches) as well as text tokens to a shared space. The model is then trained in a similar autoregresive fashion like other LLMs.

Past few years has seen the release of a number of VLLMs capable of strong visual reasoning such as GPT-4V, Gemini, Claude and Llama. Additionally, a number of open-source VLLMs (some of which are of direct interest to us since we will test them in this project) such as CogVLM [15], Pixtral [1], LlaVA [9], SmolVLM [11].

VLLM Benchmarks and Game Playing: There is a large existing literature around benchmarking of VLLMs, this section tries to provide a rough overview.

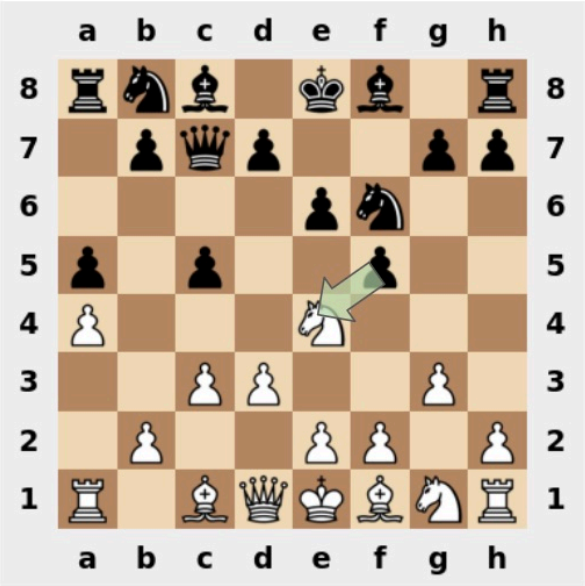
(i) Visual text understanding: Testing VLLMs ability to extract and understand visual concepts. TextVQA [14], DocVQA [12].

(ii) OCR: Fine-grained transcription of text in imgs. MM-Vet [17], OCRBench [5].

(iii) Visual reasoning: Ability to reason in both logic and visual space. MathVista [10], GQA [8].

(iv) Image generation: Ability to generate images based on text description. GenEval [6]

Apart from these there is a long history of benchmarks and models around "AI playing games", e.g. the famous Deep Blue which beat Kasparov, AlphaGo that beat Lee Sedol (move 43), AlphaZero which is super-human at chess beating engines like stockfish and OpenAI Five, an AI model that competed at Dota (before OpenAI became closed AI).



```
[-4 -2 -3  0 -6 -3  0 -4]
[ 0 -1 -5 -1  0  0 -1 -1]
[ 0  0  0  0 -1 -2  0  0]
[-1  0 -1  0  0 -1  0  0]
[ 1  0  0  0  2  0  0  0]
[ 0  0  1  1  0  0  1  0]
[ 0  1  0  0  1  1  0  1]
[ 4  0  3  5  6  3  2  4]
```

Figure 1. Input image of shape 384x384 and label (right). The label is created using the following convention. **Empty squares:** 0. **White pieces:** Pawn=1, Knight=2, Bishop=3, Rook=4, Queen=5, King=6. For Black, its the same except with a negative sign. The best move of black pawn capturing white's horse (fxe4) is shown in green.

3. Methodology

This section covers the datasets, models and methods used for exploring performance of VLLMs on chess.

3.1. Dataset

Perceive task: In this task, I benchmark VLLMs on their ability to correctly read the board positions. Though there are existing chess datasets available, I decided to create my own dataset primarily so that I can control the difficulty of the benchmark. To create the dataset, I use the `python-chess` library and `PyQt` and randomly place the 32 chess pieces on the board. The board positions and label are shown in Figure 1. The training dataset has 1024 positions and evaluation has 128. The evaluation code for this task is provided in the final "Additional detials" section.

Best move task: For the task of predicting the next best moves, I first simulate a random number of legal moves (between 1 to 40) and then use `stockfish` to find the next best moves from that position. If I have played an even number of moves then the next position would be played by White and otherwise Black. I store this information and provide it as part of the prompt (i.e. tell the model if its playing as white or black). The labels (best moves) are in SAN format, e.g. in Figure 1, the best move for Black is for the pawn to take the white's horse. This is represented in SAN as **fxe4**. I generate top 10 moves for each position. Similar to above train-test split is 1024 and 128.

Why are the datasets so small? Due to a combination of different factors: task difficulty, dataset being extremely representative with low bias, task and learning difficulty.

3.2. Method

Architecture: For all my experiments, except the final ones, I use autogressive VLMs shown in Figure 2. VLMs typically take the input image, tokenize it into patches, often using ViT, (e.g. SmolVLM creates patches of size 14x14) and converts these patches into a shared embedding space with the text tokens (which are tokenized using their own tokenizer). Imagine a 140x140 input image and a text "Describe the image", we get a (103, D) input (100 tokens for the image and assuming each word in the text is tokenized separately). This is passed into the decoder (typically a Language Model) which then autoregressively produces the output ("Image of a car").

The models are trained using the logprob loss (which increases the probability of the desired token), it can be written as,

$$\mathcal{L} = -\sum_{i=1}^{N} \log p_\theta(y_i \mid x, y_{<i}) \tag{1}$$

Where y_i is the token we want to predict in the i^{th} position. x is made of both the image tokens and the text tokens. Typically when supervise finetuning VLMs, we don't want to train over the prompt, only on the completion. Therefore the above equation seperates x (prompt + image) and y (completion). This is accomplished by passing -100 as

label. The more the model gives the correct tokens higher probability, the lower $-log(p)$ is.

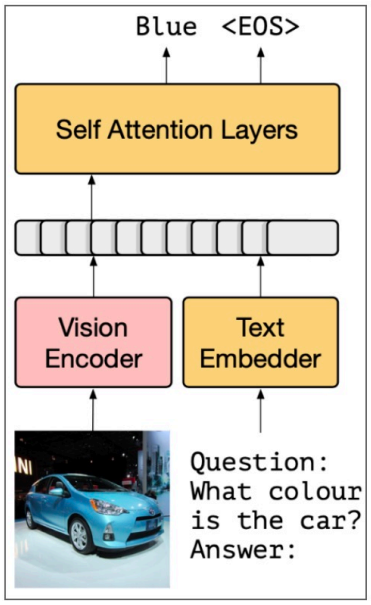


Figure 2. Autoregressive VLM architechure. Picture taken from the great Lilian Weng blog. In case of QwenVLM, the "Self-Attention Layers" is a whole LLM (Qwen LM).

Evaluation: I use a mix of open-source (**SmolVLM2-2.2B-Instruct**, **llava-1.5-7b-hf**, **Qwen2-VL-7B-Instruct**, **Pixtral-12B**, **llama-3.2-11b-vision**) and closed-source models (**GPT-4.1-mini** and **GPT-4.1**).

For evaluation, I pass board position as input and parse the output into a 8x8 matrix for "Perceive" task and extract the SAN notation string for the "Best move" task.

Finetuning: For finetuning, I work with only **SmolVLM2-2.2B-Instruct**, **llava-1.5-7b-hf**, **Qwen2-VL-7B-Instruct** since other models are either too big or too expensive to SFT. For all experiments, I do full SFT (no lora) and mask the prompt + input image during SFT loss.

Qualitative Analysis: For the third part of the project, I try to see if given a board image can a image-to-image model generate what the board would look like after the best move was played. I qualitatively compare how diffusion based models (**stable-diffusion-xl-refiner**) perform compared to auto-regressive/mixed architechure (GPT-4o image generation).

4. Results

4.1. Perceive task (Table 1)

What does the † legend mean, and how to interpret the results of SmolVLM/llava?

† indicates that the model outputs the initial (starting) chessboard position (provided as an example in the prompt Prompt Link) for most or all test images. This happens because (i) These VLMs (like SmolVLM) cannot distinguish subtle differences between board images and overall "see" each input as just the same chessboard (ii) They are poor at instruction following. As a result, their outputs are often either invalid (unparsable) or simply repeat the prompt example matrix. However, since the initial chess position shares many pieces with most test positions (e.g., always 8 white pawns), this constant output alone gives a non-trivial baseline accuracy ($\sim$7%).

Model	Accuracy
SmolVLM 2.2b	7.2†
llava-7b-hf	7.2†
Qwen2.5 7b VL	6.3
Llama 13b	2.0
Pixtral 12b	4.5
SmolVLM 2.2b + SFT	7.8
llava-7b-hf + SFT	0.0
Qwen2.5 VL + SFT	100
GPT-4.1-mini	57.3
GPT-4.1	32.7

Table 1. Perceive task accuracy. Some results are suprising and counter-intuitive (e.g. Why SFT makes it worse sometimes while for Qwen makes it perfect?). I provide a thorough explanation in the "Analyzing Results" setction. † means output is constant.

Why do better models like Qwen2.5 perform worse?

Qwen2.5 is better at differentiating between input images and does not just copy the prompt. It attempts to generate a new 8x8 matrix for each board, reflecting some level of understanding. However, its outputs are often imperfect (e.g., too many kings, too few pawns), leading to lower accuracy under strict evaluation, even though the outputs are more responsive to the input. In contrast, models like SmolVLM, by always outputting the initial board, "accidentally" match the correct pieces more often due to overlap with the test data, despite not actually "seeing" the input.

What about Llama/Pixtral?

Although Llama and Pixtral are best in their class, my results (available in the GitHub repo) show they have similar limitations as Qwen. They can distinguish different board images, but their outputs are often similarly flawed (e.g., missing pawns, not outputting black pieces etc.). Llama, in particular, makes frequent formatting errors, such as incorrect or missing brackets, requiring lot of manual cleaning.

Results on closed source models

I evaluated GPT-4.1 (strongest general VLM) and GPT-4.1-mini. Both achieved relatively high accuracy, with mini surprisingly outperforming the full version. However, even these models (likely with 100B+ parameters) fail to cross 60% accuracy, highlighting the difficulty of this task.

How does SFT help? Qwen is 100% ??

SFT (Supervised Fine-Tuning) training curves in the Appendix. I also provide more disucssion about it in the "Training Dynamics" section. After SFT, SmolVLM shifts from repeating the same board to predicting from a small set of boards configurations, suggesting it struggles with the task and settles on a handful of "modes" to minimze its loss. For llava, SFT made outputs unstable and unparsable (e.g., missing/extra brackets, wrong matrix sizes etc.).

Qwen after SFT performs the best, achieving perfect accuracy on the test set. I verified that no test board matches any training board. Notably, Qwen's training dynamics resemble "grokking-like" [13] phase transition behavior, which I discuss in detail later.

Model	Top-1	Top-3	Top-5	Top-10
SmolVLM 2.2b	7.08	14.96	21.25	29.99
llava 7b hf	7.08	11.02	13.38	18.89
Qwen2.5 7b VL	11.81	18.89	24.40	34.64
Llama 13b vision	2.00	3.90	5.40	7.03
pixtral 12b	7.81	14.06	17.90	27.30
SmolVLM + SFT	4.72	4.72	5.51	7.08
Qwen2.5 + SFT	3.90	14.06	15.625	18.75
GPT-4.1-mini	20.40	33.80	42.50	54.00
GPT-4.1	43.75	64.00	73.00	78.57

Table 2. Best move prediction. Top-k here equals to if my predicted move was in the top k best moves according to stockfish label. Again results are suprising, I provide an explanation of what is happening in the resutls section.

4.2. Best move task (Table 2)

How to read the table?

Top-1 and Top-3 metrics are the most meaningful. Top-5 and Top-10 scores are often inflated because, in most mid-game positions, common moves like advancing central pawns (e.g., e4, d4) are frequently among the top options. As a result, even models that repeatedly suggest such moves can achieve relatively high Top-10 accuracy, making this metric less meaningful.

Performance of open-source models

As with the perceive task, smaller models tend to produce only a few distinct outputs across all positions (see Table 3). For instance, on 128 test images, SmolVLM's responses are limited to just e4, d4, e5, d5. Since central pawn moves are often reasonable, this yields non-trivial accuracy. The same pattern holds for llava. Qwen is even more restricted, choosing only between e5 and e4, but is better at selecting which to use when.

Llama-13b-vision outputs are often unparseable, because it frequently produces multiple SAN strings or excessive CoT tokens, leading to poor performance. Pixtral, in contrast, gives correctly formatted answers with reasoning, but its predictions rarely match the ground truth.

Results on closed source models

GPT-4.1 performs best, in this case as expected better than GPT-4.1-mini. Most GPT-4.1 outputs are without CoT reasoning, so prompting for a step-by-step solutions might further improve results. I am slightly suprised by the nearly 45% accuracy, as I had imagined this task to be much tougher. A more challenging benchmark would be to use games played by grandmasters, where moves are less trivial and predictable, this would better test model capabilities.

What about SFT? Why does it not work?

SFT on both SmolVLM and Qwen actually reduces performance. For SmolVLM, SFT causes the model to settle on new "modes" (e.g., always predicting h6, Nxd4, e4), which perform worse than the previous "central pawn pushing" outputs. This drop occurs even as training loss decreases. The reasons for this, and for Qwen's results, are discussed in the next section

Prompt for Perceive task

You are a Vision Language Model specialized in interpreting data from chess board images. Your task is to accurately describe the chess board position in the image using a 8x8 grid output. Each type of chess piece, both black and white, is represented by a unique number:
Empty squares: 0
White pieces: Pawn=1, Knight=2, Bishop=3, Rook=4, Queen=5, King=6
Black pieces: Pawn=-1, Knight=-2, Bishop=-3, Rook=-4, Queen=-5,King=-6
From the provided chessboard image, convert the visible board into this 8x8 matrix format. For example, the initial chess position would be represented as:
[[-4, -2, -3, -5, -6, -3, -2, -4],
[-1, -1, -1, -1, -1, -1, -1, -1],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[1, 1, 1, 1, 1, 1, 1, 1],
[4, 2, 3, 5, 6, 3, 2, 4]]
Ensure that your output strictly follows this format

Prompt for Best Move task

You will play as white in this position.
Return the move in SAN notation. Return the move in the following format boxed{move}. For example, if the best move is e4, return boxed{e4}
Answer:

5. Training Dynamics and Analysis

What is happening with SFT?

Recall, that these models are trained using the next token prediction negative logprob loss.

$$\mathcal{L} = -\sum_{i=1}^{N} \log p_\theta(y_i \mid x, y_{<i}) \tag{2}$$

Here, p_i denotes the model-assigned probability of the correct token at position i. As shown in Figure 3, SmolVLM's loss decreases and eventually saturates around 0.01. However, as the loss decreases, the model continues to produce only a single output for all inputs, effectively getting stuck in one "mode". This occurs because the model can reduce its loss by increasing the probability of, for example, the token "0," without ever making it the most likely token. Thus, the loss drops even though the predictions remain incorrect. My hypothesis is that, due to its visual backbone's inability to distinguish input images, the model becomes trapped in a "basin" of the loss landscape. SmolVLM fails to get out of this basin.

Figure 3. Loss curve for SmolVLM SFT on perceive task. The loss saturated at around 0.01

For Qwen, the loss is shown in Figure 4. Here the model suffers from similar problem as SmolVLM until it suddenly has a "groking-like" phase transition [13] and comes out of the basin, once it does that it is able to produce different outputs for different images, is not stuck in a set of "modes" and also the loss decrease rapidly and saturates few steps after. Infact, evaluating the model just before and after this "grokking-like" phase transition step, results in a large difference in accuracy. Its as if the model suddenly learns. I see the same behavior for Qwen 7b across different runs.

What about SFT for best move?

Why does SFT not work for the best move task then? First, the task in inherently more difficult than transcribing board positions. I don't discuss SmolVLM here since if that model can't learn to transcribe board pieces using SFT then

Figure 4. Loss curve for Qwen 7b VL on perceive task compared to SmolVLM. Notice the sudden sharp drop around step 200. Though not clear from the image, the loss for red becomes much smaller (0.00001) as compared to green (0.01)

Model	Unique	Frequency
GT	99	Too Long
SmolVLM	4	{'e4':6, 'd4':18, 'd5':102, 'e5':2}
+ SFT	3	{'e4':22, 'Nxd4':94, 'h6':12}
Qwen7b	2	{'e4': 53, 'e5': 75}
+ SFT	44	Too Long

Table 3. Diversity and frequency of model outputs before and after SFT. GT = Ground Truth. Unique = Number of unique outputs for the test set which has 128 images. Frequency = How the outputs are distributed. So for Qwen7b (before SFT) it only outputs either e4 or e5 for all the 128 input images. Using e4 53 times and e5 75 times. This shows have models have these "modes" they are stuck in

what hope does it have to learn the best move. The orignal Qwen VL loss graph for SFT is shown in Figure 5. As we can see from the figure, the loss remains huge and decreases only slightly.

To understand why, I decided to see what happens if I train on a smaller training set. I ran Qwen on 8, 16, 32 and 64 datapoints. The results for which are shown in Figure 8. One thing to observe here is that it takes longer and longer for the model to overfit (or get a reasonably low loss). It takes 39 epochs for 16 points, 95 epochs for 32 points and 441 (!!) epochs for 64 points.

This suggests to me that the model's difficulty goes beyond simply needing more data or training time. Predicting the best move requires not only accurate recognition of the board but also a strategic reasoning on top of it, which the model is not capable of. Moreover, the supervision is sparse, in the sense that the label is only made of at most 2-3 tokens. And there is little overlap between the structure of the different labels (e.g. a slight change in board position will result in a completely different best move whereas for

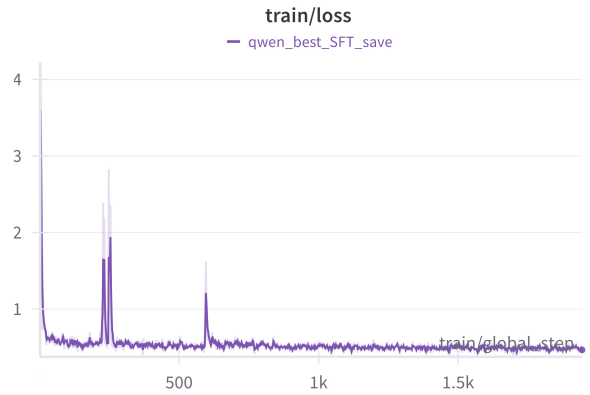


Figure 5. Loss curve for Qwen SFT on best move task. The model never has a "grok-like" phase transition. The loss saturates quickly and the model gives only a few set of outputs for all inputs.

the perceive task the labels remain largely similar). Therefore each gradient update pushes the model in different direction, so much so that it takes 400+ epochs to even overfit on 64 points.

6. Image to Image models

For my final experiments, I qualitatively test if img-to-img models can do the following

(i) Given a chess board image input produce what the board will look like if the best move was played.

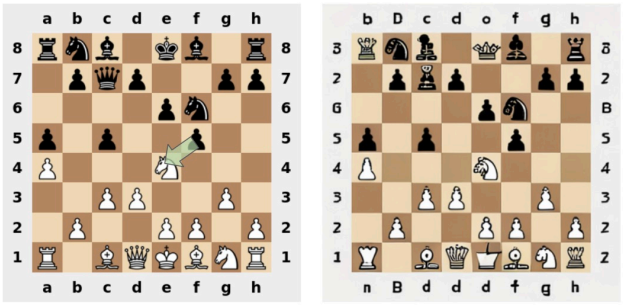


Figure 6. (Left) Input image. (Right) Output of diffusion model when asked to produce how the board will look like after the best move by black has been played. Diffusion can't reason, just produces the same image with slightly different style on pieces. Please note the green arrow is not part of input just for demonstrative purpose.

7. Conclusion and Future Work

For this project I wanted to see how VLMs fare at chess. From my experiments these are some of my takeaways

(i) Open source VLMs, specifically smaller ones still have some way to go. Small VLMs consistently performed

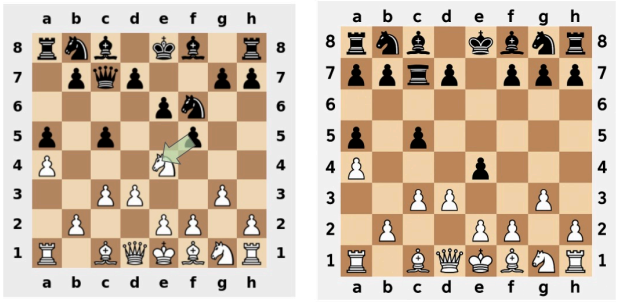


Figure 7. (Left) Input image. (Right) Output of GPT-4.1 when asked to produce how the board will look like after the best move by black has been played. GPT-4.1 suprisingly actually works. Though it destroys some other board pieces such as the black queen at c7 is replaced by a rook.

poor and apart from Qwen VL were also unstable to train. Moreover, training VLMs on task that require both precision and reasoning is really challenging.

(ii) Closed source models such as GPT-4.1 simultaneously show surprisingly strong (better than average human) performance (such as predicting the best move ∼45% of time) while also showing worse than human performance on the transcribing board position task (an average human can easily get 100% on this while GPT-4.1 only gets ∼40%).

This is a sign of the times we are living in and has been noted as Moravec's Paradox ("It is comparatively easy to make computers exhibit adult level performance on intelligence tests, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility")

Future work: Apart from creating better robust benchmarks (such as the idea of using board positons from grandmaster games), this project's future work involves (given the compute resources) testing how bigger open source VLMs fare, such as finetuning Pixtral models. Also a more thorough investigation into the SFT training observations made in previous sections would also be interesting.

8. Hyper Parameters and Appendix

```
def eval(pred, gt):
    # gt is ground truth.
    # pred is predicted output
    # Both are 8x8 numpy arrays.
    mask = gt != 0  # non-zero mask
    correct = pred[mask] == gt[mask]
    acc = np.mean(correct)
    return acc
```

9. Contributions & Acknowledgments

During the literature review I found out that a recently published paper in ICLR'25, "ARE LARGE VISION LAN-

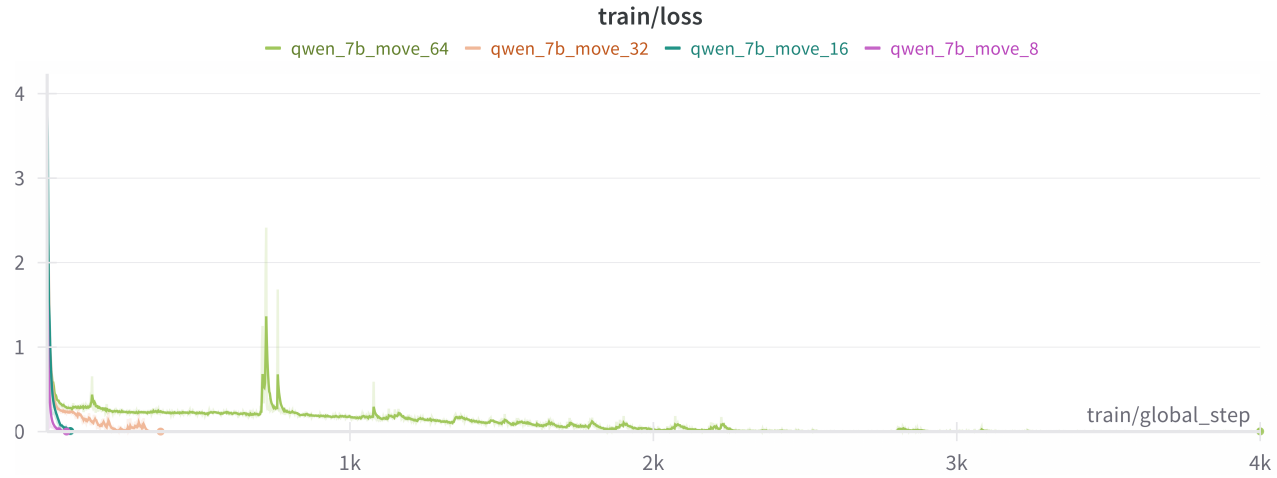


Figure 8. Comparison of how long it takes for Qwen 7b VL to get a reasonably low loss on different sized datasets (of sizes 8, 16, 32 and 64).

	Value
LR	5e-5
Batch size	8
Optimizer	Adam
precision	bfloat16
warmup ratio	0.03
LR Scheduler	constant

Table 4. SFT hyper-paramters. I use different epochs. For both SmolVLM and Qwen were traiend for 10 epochs for the perceive task and for 50 epochs for the best move task.

GUAGE MODELS GOOD GAME PLAYERS?" [16]. To be fair, I provide how my project differs from them.

[16] also explores VLLMs ability at different games including chess. Parts of this paper therefore borrow from it (particularly their chess UI code to help generate dataset: Github link). For sake of fairness, I describe how this project differs from it

(1) Biggest difference is this project finetunes VLLMs for tasks such as reading the board and finding the best move to see the improvement over off-the-shelf VLLMs, whereas [16] doesn't finetune any models.

(2) [16] has 2 chess tasks, reading the board, finding a legal move. This paper introduces another task of finding the best move.

(3) [16] strictly studies Img-to-Text models (models that can read both text and image but only output text). This paper takes it a step further and does a brief study on Img-to-Img models.

All the training and evaluation was done with the help and generosity of the GPUs on the SAIL cluster.

References

[1] P. Agrawal, S. Antoniak, E. B. Hanna, B. Bout, D. Chaplot, J. Chudnovsky, D. Costa, B. D. Monicault, S. Garg, T. Gervet, S. Ghosh, A. Héliou, P. Jacob, A. Q. Jiang, K. Khandelwal, T. Lacroix, G. Lample, D. L. Casas, T. Lavril, T. L. Scao, A. Lo, W. Marshall, L. Martin, A. Mensch, P. Muddireddy, V. Nemychnikova, M. Pellat, P. V. Platen, N. Raghuraman, B. Rozière, A. Sablayrolles, L. Saulnier, R. Sauvestre, W. Shang, R. Soletskyi, L. Stewart, P. Stock, J. Studnia, S. Subramanian, S. Vaze, T. Wang, and S. Yang. Pixtral 12b, 2024.

[2] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[5] L. Fu, B. Yang, Z. Kuang, J. Song, Y. Li, L. Zhu, Q. Luo, X. Wang, H. Lu, M. Huang, Z. Li, G. Tang, B. Shan, C. Lin, Q. Liu, B. Wu, H. Feng, H. Liu, C. Huang, J. Tang, W. Chen, L. Jin, Y. Liu, and X. Bai. Ocrbench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning, 2024.

[6] D. Ghosh, H. Hajishirzi, and L. Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment, 2023.

[7] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021.

[8] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering, 2019.

[9] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning, 2023.

[10] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts, 2024.

[11] A. Marafioti, O. Zohar, M. Farré, M. Noyan, E. Bakouch, P. Cuenca, C. Zakka, L. B. Allal, A. Lozhkov, N. Tazi, V. Srivastav, J. Lochner, H. Larcher, M. Morlon, L. Tunstall, L. von Werra, and T. Wolf. Smolvlm: Redefining small and efficient multimodal models, 2025.

[12] M. Mathew, D. Karatzas, and C. V. Jawahar. Docvqa: A dataset for vqa on document images, 2021.

[13] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

[14] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. Towards vqa models that can read, 2019.

[15] W. Wang, Q. Lv, W. Yu, W. Hong, J. Qi, Y. Wang, J. Ji, Z. Yang, L. Zhao, X. Song, J. Xu, B. Xu, J. Li, Y. Dong, M. Ding, and J. Tang. Cogvlm: Visual expert for pretrained language models, 2024.

[16] X. Wang, B. Zhuang, and Q. Wu. Are large vision language models good game players?, 2025.

[17] W. Yu, Z. Yang, L. Li, J. Wang, K. Lin, Z. Liu, X. Wang, and L. Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities, 2024.