Assignment 1 - Evaluating efficient Vision Language Models (VLMs)

TU Wien 191.021 Introduction to Computational Sustainability, 2025 Winter Semester, Group 15

Daniel Martin Pühringer
Matr.Nr.: 01556470
e01556470@student.tuwien.ac.at

Patrick Ennemoser
Matr.Nr.: 51913717
e51913717@student.tuwien.ac.at

Dragana Sunaric
Matr.Nr.: 11814569
e11814569@student.tuwien.ac.at

***Abstract*—This report presents an evaluation of the LLaVA-1.5 model for the task of descriptive image captioning using the COCO Captions dataset. We assess both efficiency (memory, latency, throughput, model size) and quality (CIDER score) across different quantization configurations. The goal is to understand how different quantization strategies affect the trade-off between inference efficiency and caption quality.**

We evaluate three setups:

1) **FP16 (Baseline)**
2) **Weight-only INT8**
3) **Quantized vision tower to INT8**

Our experiments show that both quantized variants maintain comparable captioning quality to the FP16 baseline, achieving similar CIDEr scores ($\tilde{0}.85$). The weight-only INT8 model reduced VRAM usage by roughly 45% but suffered from higher latency and lower throughput due to runtime upcasting. In contrast, quantizing only the vision tower preserved FP16-like performance while slightly reducing efficiency gains.

I. Introduction

Large Vision-Language Models (VLMs) such as LLaVA-1.5 [1] demonstrate remarkable ability in analyzing images. However, deploying such models in real-world or resource-constrained environments poses challenges due to their large memory footprint and inference latency.

Quantization — reducing the precision of model weights or activations — offers a practical solution to improve efficiency (smaller model size, faster inference, lower GPU memory usage) at the possible expense of accuracy or generation quality.

We investigate how quantization impacts LLaVA-1.5's captioning performance on the COCO 2017 val (5k) [2] dataset.

Specifically, we compared:

1) a baseline FP16 model
2) a weight-only INT8 quantization
3) a quantized vision tower to INT8

We measure:

- Efficiency: Peak VRAM usage, latency per image-prompt, throughput (images/s), and model size.
- Quality: Using the CIDER metric [3], a standard for caption similarity.

These experiments are useful to understand the trade-offs between efficiency and quality when deploying large VLMs in practical scenarios. By systematically comparing different quantization strategies, we can identify configurations that retain most of the model's captioning quality while significantly reducing computational and memory requirements. We predict that weight-only INT8 quantization will achieve substantial efficiency gains with minimal loss in captioning performance, whereas quantizing the vision tower will yield higher efficiency improvements but a more noticeable drop in CIDER score.

The full source code for this evaluation is available on Github:
https://github.com/dsunaric/Assignment-1-Efficiency-VLMs

II. Methods

A. Model

We use LLaVA-1.5 [1], a multimodal large language model that integrates a vision encoder (ViT [4]) and a language decoder based on LLaMA [5]. The model is pre-trained to align image and text representations and fine-tuned for image captioning tasks.

We tested and compared the following quantization configurations:

1) **FP16 (Baseline)**
   Standard inference in half-precision (FP16).
   Tool / Method: Native PyTorch.

2) **Weight-only INT8**
   Quantize attention and MLP weights to INT8 while keeping the vision tower in FP16.
   Tool / Method: bitsandbytes [6].

3) **Quantized vision tower to INT8**
   Quantize vision components (ViT) to INT8.

Tool / Method: bitsandbytes [6].

B. Dataset

We used the COCO 2017 validation dataset, which consists of 5,000 images. All images in the dataset already have a resolution smaller than 1024 pixels on the longer side, so no resizing was required. The dataset provides reference captions, which we used to compare against the captions generated by our model.

For our experiments, we selected a subset of 100 images. To ensure a consistent and deterministic selection process, we sorted all images in ascending order by their file names and chose the first 100 images from the resulting list.

C. Evaluation Metrics

The evaluation was conducted across two main categories: efficiency and quality.

Efficiency Metrics:

- Peak VRAM (MiB): VRAM refers to Video RAM, which is dedicated to the GPU, and is controlled by the GPU.
- Latency (s/image): The average inference time required to process a single image.
- Throughput (img/s): The number of images processed per second.
- Model size (GB): The disk size of the quantized model.

Quality Metric:

- CIDEr [3]: Measures the similarity between the generated captions and the reference captions.
- It is estimated using codecarbon, which uses nvidia-smi under the hood

Energy Metric:

- This is a bonus metric and measures the CO2eq emissions in kg for the inference of the model

D. Experiment Setup

All experiments were conducted on Google Colab [7], using Python 3.12 and a NVIDIA L4 GPU. The model was evaluated across different maximum token limits of 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 tokens to analyze how caption length influences performance. To ensure reproducibility, we maintained the same set of prompts and data order across all runs.

The following prompt format was used for inference:

```
USER: (image)
Generate a caption for this image
ASSISTANT:
```

This structure follows the LLaVA model's conversational input format, which expects instructions in a dialogue-like exchange between a "USER" and an "ASSISTANT."

III. Results

A. Quantitative Evaluation

We compare three model configurations: FP16, weight-only 8-bit, and a partially quantized vision tower (INT8 vision encoder, FP16 elsewhere). In the FP16 model, all computations run natively in floating-point. In the quantized vision tower, only the vision encoder uses INT8 weights, which reduces the need for upcasting during the forward pass compared to a fully weight-only 8-bit model that must convert all INT8 weights to FP16.

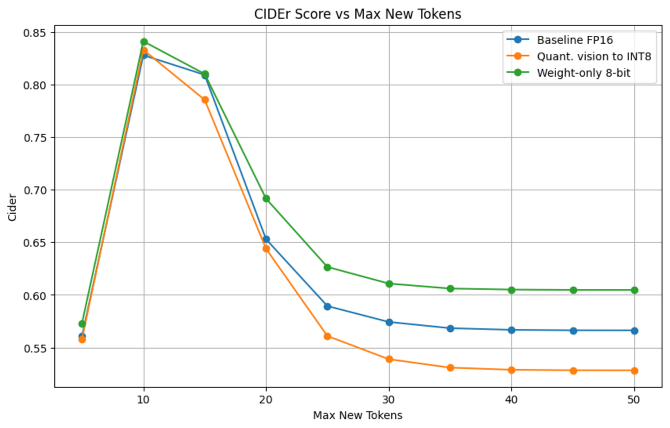

Fig. 1: CIDEr score comparison across models.

Figure 1 shows that all three models achieve similarly high CIDEr scores, around 0.85. Peak performance occurs at 10 tokens for each model, with scores decreasing from 10 to 30 tokens and plateauing afterwards. These results indicate that both quantizing the vision encoder and using weight-only 8-bit quantization do not noticeably degrade captioning quality.

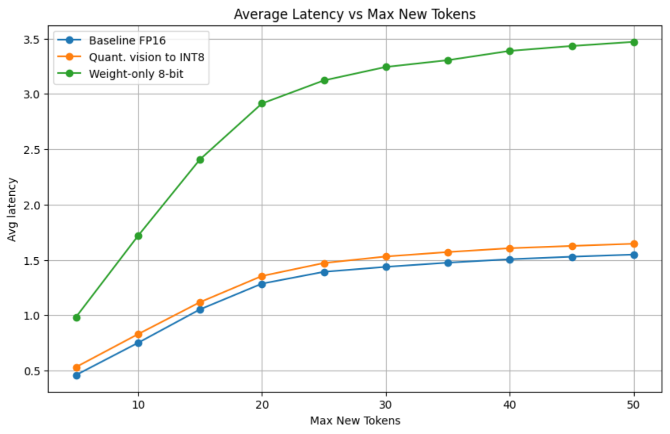

Fig. 2: Latency comparison across models.

Latency trends, shown in Figure 2, mirror these findings. The weight-only 8-bit model experiences the highest latency, increasing with token count and leveling off after 20 tokens. This reflects the computational cost of upcasting all INT8 weights to FP16. In contrast, the quantized vision tower only upcasts the vision encoder, maintaining latency similar to the FP16 baseline.

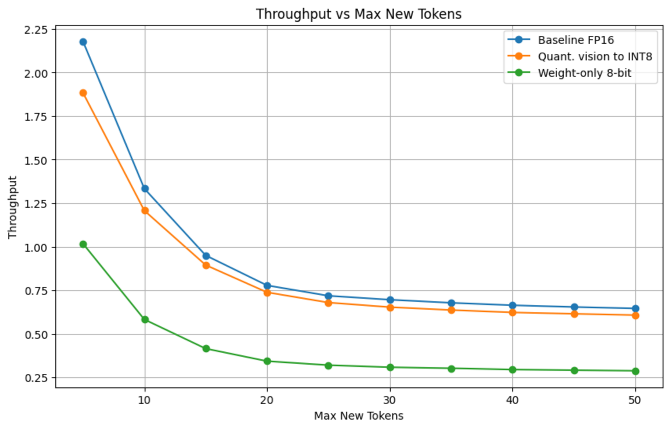

Fig. 3: Throughput comparison across models.

Figure 3 further highlights throughput differences. The weight-only 8-bit model achieves the lowest throughput due to its higher latency, while the quantized vision tower sustains higher throughput by limiting upcasting to a portion of the model.

Table I summarizes memory, storage, and estimated emissions of the configurations tested. The quantized vision tower consumes the most VRAM (14,846 MiB) and disk space (56,686 MiB). The weight-only 8-bit model requires roughly half the VRAM of the FP16 configuration (7,538 MiB), while disk usage is similar across all three models.The weight-only 8-bit model requires less VRAM than FP16 because storing all weights in INT8 reduces memory usage. Despite this, CO_2 emissions are similar for FP16 and the quantized vision tower, whereas the weight-only 8-bit models show higher emissions, likely due to the longer inference times and additional GPU computation overhead caused by upcasting.

B. Qualitative Evaluation

This section focuses on the results of one single image at a time and highlights the differences in token size and the differences of each model.

1) Image for evaluation: The evaluation uses image 4, which shows a stop sign. In addition, the evaluation also uses an image of teddy bears 5.

The CIDEr score is the main score for our quantitative evaluation, as it can be seen in section III-A. This score cannot be used for this qualitative evaluation due to technical restrictions (it cannot be calculated for one single image). However it uses the ground truth captions for its calculation, which can be shown here to gain a deep understanding of the data.

To provide an example, the ground truth captions (which are downloaded with the dataset) for image 4 are as follows:

- 'A stop sign is mounted upside-down on it's post. '
- 'A stop sign that is hanging upside down.'
- 'a stop sign put upside down on a metal pole '
- 'A stop sign installed upside down on a street corner'

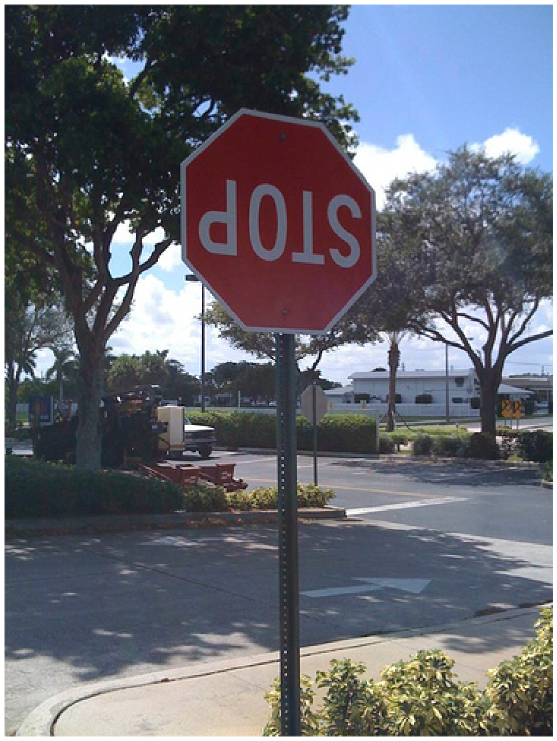

Fig. 4: Image of a stop sign upside-down (from COCO dataset) used for qualitative evaluation.

2) Evaluation of one model among different token sizes: Depending on the token size, the image of an upside-down stop sign (image 4) receives a different caption. In table II it can be observed that the more tokens are being used, the more descriptive the caption gets. However this only holds to a certain point, when the maximum length of the description is already reached. For image 4, this point is reached at 20 tokens, which can be found in table II. This is why the latency also does not increase after this point either.

3) Evaluation of one token size among all three different experiment configurations with same caption: Depending on the experimental configuration, the image of a stop sign 4 could potentially receive a different caption with the same token amount. However, we observed that this is not the case (see table III) and that the three captions are the same for each of our configurations. This also matches the observation that the CIDEr scores are very similar, as described in chart 1.

4) Evaluation of one token size among all three different experiment configurations with different caption: In contrast to the stop sign image above (image 4) we found a suitable example highlighting the possible deviation in captions for each experiment. Image 5 shows a group of teddy bears, which are described differently across the three experiments; see table IV.

IV. Discussion

The evaluation indicates that the quantization strategy primarily affects latency, throughput, and resource usage

Configuration	Peak VRAM (MiB)	Disk Usage (MiB)	Emissions (kg CO_2eq)
Quantized Vision Tower (L4 GPU)	14,846.11	56,539.10	0.00666
Weight-Only 8-Bit (L4 GPU)	7,538.43	55,663.21	0.01404
FP16 (L4 GPU)	13,976.20	55,538.21	0.00610

TABLE I: Hardware and environmental metrics for the tested model configurations on L4 GPU.

Max new tokens	Latency	Generated caption for image 4
5	0.434s	A stop sign is ups
10	0.732s	A stop sign is upside down, with the
15	1.038s	A stop sign is upside down, with the word ”STOP”
20	1.276s	A stop sign is upside down, with the word ”STOP” written backwards.
25	1.276s	A stop sign is upside down, with the word ”STOP” written backwards.
30	1.276s	A stop sign is upside down, with the word ”STOP” written backwards.
35	1.277s	A stop sign is upside down, with the word ”STOP” written backwards.
40	1.278s	A stop sign is upside down, with the word ”STOP” written backwards.
45	1.278s	A stop sign is upside down, with the word ”STOP” written backwards.
50	1.278s	A stop sign is upside down, with the word ”STOP” written backwards.

TABLE II: Generated captions of the FP16 (Baseline) configuration with different token sizes and latency measured in seconds

Experiment configuration	Generated caption for image 4 with 15 max token
FP16 (Baseline)	A stop sign is upside down, with the word ”STOP”
Weight-only INT8	A stop sign is upside down, with the word ”STOP”
Quant Vision Tower	A stop sign is upside down, with the word ”STOP”

TABLE III: Generated captions for stop sign image 4 of all three different experiment configurations under a max token size of 15

Experiment configuration	Generated caption for image 5 with 30 max token
FP16 (Baseline)	A group of teddy bears are sitting on a bed, with one teddy bear sitting on top of another.
Weight-only INT8	A group of teddy bears are piled on top of each other, with one teddy bear in the center.
Quant Vision Tower	A group of teddy bears are sitting on a bed, with one teddy bear sitting on top of another teddy bear.

TABLE IV: Generated captions for teddy bears image 5 of all three different experiment configurations under a max token size of 30

rather than captioning quality. The weight-only 8-bit model still exhibits higher latency and lower throughput compared to the FP16 baseline and the partially quantized vision tower. This behavior is explained by the runtime upcasting of INT8 weights to FP16 during computation, which increases temporary memory usage and processing time.

The quantized vision tower, where only the vision encoder is quantized, reduces upcasting overhead, resulting in latency and throughput closer to the FP16 model. However, all three models achieve comparable CIDEr scores, suggesting that neither weight-only nor partial quantization substantially degrades captioning quality. Emission estimates follow a similar pattern: the weight-only 8-bit model produces approximately double the CO_2 output due to prolonged inference, while the FP16 and quantized vision tower models remain similar in efficiency.

The evaluation has several limitations. It was conducted in a constrained hardware environment using Google Colab L4 and T4 GPUs and under small-scale testing conditions, which may not fully reflect performance at larger batch sizes or in production deployments. From an ethical perspective, LVLMs are not inherently robust and can be easily misled by manipulated or adversarial inputs, potentially leading to biased or inaccurate outputs.

V. Conclusion

Partial quantization of the vision tower provides an effective compromise between accuracy, latency, and energy efficiency. In contrast, full weight-only quantization introduces computational overhead from upcasting, reducing both performance and sustainability benefits. Future work could extend this evaluation by testing larger batch sizes to better reflect real-world workloads, examining how image preprocessing and

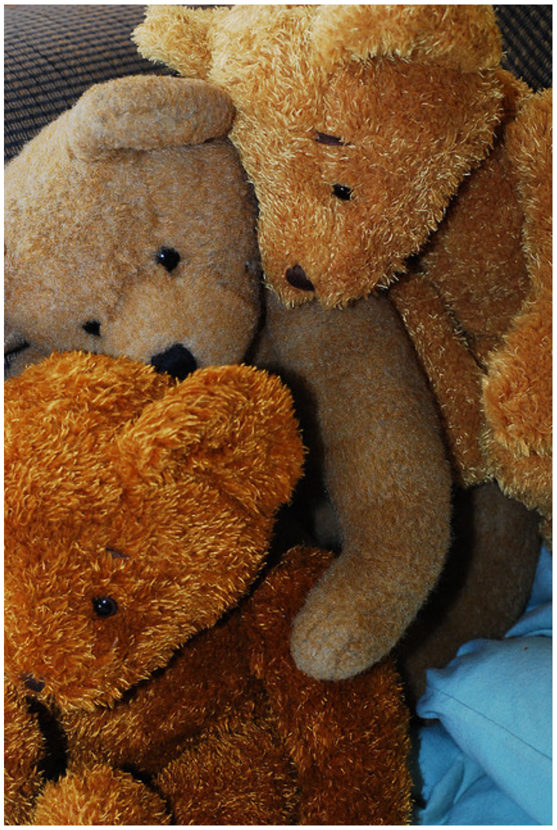
Fig. 5: Image of a group of teddy bears (from COCO dataset) used for qualitative evaluation.

augmentation operations affect captioning quality, and benchmarking the same configurations on newer hardware to assess the consistency of quantization behavior across architectures.

Appendix A
Appendix: Experimental Details

A. Dataset Information

- **COCO 2017 Validation Images** http://images.cocodataset.org/zips/val2017.zip
- **COCO 2017 Annotations** http://images.cocodataset.org/annotations/annotations_trainval2017.zip

The evaluation used the first 100 images from the COCO 2017 validation dataset, selected deterministically by sorting all file names in ascending order and taking the first 100 entries.

B. Package Versions

The following Python package versions were used during experimentation:

- `transformers==4.57.1`
- `accelerate==1.11.0`
- `einops==0.8.1`
- `tqdm==4.67.1`
- `gradio==5.49.1`
- `pycocoevalcap==1.2`
- `codecarbon==3.0.8`
- `bitsandbytes==0.45.0`
- `onnx==1.17.0`
- `onnxruntime==1.20.1`
- `onnxruntime-tools==1.7.0`
- `onnxruntime-gpu==1.20.1`

C. Hardware Specifications

All experiments were executed on a Google Colab environment with the following hardware configuration:

- **GPU:** NVIDIA L4
  - 22.5 GB GDDR6 VRAM
  - 7424 NVIDIA CUDA cores
  - Architecture: Ada Lovelace
- **CPU:** Virtualized Intel Xeon (2 vCPUs, 2.3 GHz)
- **System Memory:** 12.7 GB RAM
- **Storage:** 107.7 GB temporary disk
- **Operating System:** Ubuntu 20.04 LTS

D. Environment Notes

All models were evaluated using Python 3.12 on Google Colab. The setup ensured consistent hardware and software configurations across all experiments to minimize variability in performance and emissions measurements.

References

[1] J. Huang, J. Zhang, K. Jiang, H. Qiu, and S. Lu, “Visual instruction tuning towards general-purpose multimodal model: A survey,” 2023. [Online]. Available: https://arxiv.org/abs/2312.16602

[2] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft coco: Common objects in context,” 2015. [Online]. Available: https://arxiv.org/abs/1405.0312

[3] R. Vedantam, C. L. Zitnick, and D. Parikh, “Cider: Consensus-based image description evaluation,” 2015. [Online]. Available: https://arxiv.org/abs/1411.5726

[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. [Online]. Available: https://arxiv.org/abs/2010.11929

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “Llama: Open and efficient foundation language models,” 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[6] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, “8-bit optimizers via block-wise quantization,” 2022. [Online]. Available: https://arxiv.org/abs/2110.02861

[7] Google, “Google Colaboratory,” https://colab.research.google.com/, 2025, accessed: 2025-11-11.