Mind Your Special Tokens! On the Importance of Dedicated Sequence-End Tokens in Vision-Language Embedding Models

Elio Musacchio[1,2], Giovanni Semeraro[1], Goran Glavaš[3]
[1]Department of Computer Science, University of Bari
[2]Department of Computer Science, University of Pisa
[3]WüNLP, CAIDAS, University of Würzburg
Correspondence: elio.musacchio@uniba.it

Abstract

Large Vision-Language Models (LVLMs), trained by aligning visual encoders to LLMs on extensive vision-language data, demonstrate impressive performance across a broad variety of tasks that require understanding of both visual and textual inputs. Acknowledging this, recent work proposed to post-hoc convert generative LVLMs into vision-language encoders (VLEs) via supervised contrastive learning objectives. This type of training enables LVLMs to produce better representations, i.e., *embeddings* for image and text input, used in retrieval and (semantic) similarity tasks. Having observed that this type of VLEs (i.e., LVLMs turned into encoders) commonly employ last-token pooling in downstream tasks, *without* using special sequence-end tokens, in this focused contribution, we study the effect of pooling strategies on VLEs' downstream performance. We empirically show that, in contrast to mean pooling, last-token pooling (without special sequence-end tokens) makes VLEs highly sensitive to end-of-input artifacts in fine-tuning and inference data, e.g., whether input sequences end with punctuation or newline characters. Finally, we show that introducing the special end-of-sequence token removes this sensitivity and makes VLEs robust to formatting artifacts of input text.

1 Introduction

Large Vision-Language Models (LVLMs) extend pretrained Large Language Models (LLMs) with the ability to process visual input together with text. The mainstream approach to train LVLMs is visual instruction-tuning (Liu et al., 2023; Geigle et al., 2025): here, embeddings of visual tokens—output by a pretrained visual encoder—are projected into an input embedding space of a pretrained, typically already instruction-tuned, LLM and concatenated to the embeddings of the text instruction (i.e., task description) tokens. This approach benefits from the impressive language generation and reasoning abilities of recent LLMs (Llama Team, 2024; Yang et al., 2024) and leverages it for *generative* vision-language tasks, such as image captioning or visual question answering. Given the impressive performance of LVLMs on generative tasks, very recent work investigated their post-hoc conversion into encoders (i.e., embedding models) (Jiang et al., 2025; Meng et al., 2025; Chen et al., 2025), for the benefit of non-generative vision-language tasks, such as cross-modal retrieval and semantic similarity. These vision-language encoders (VLEs), much like the text-only embedding models (Reimers and Gurevych, 2019; Wang et al., 2024a; Huang et al., 2025), need to represent the entire variable-size input into a single fixed-size embedding vector: the difference with VLEs, however, is that the sequence to be compressed is a concatenation of visual and textual tokens. To this end, two common *pooling* strategies are leveraged: (1) mean pooling averages the transformed output representations of all tokens; and (2) last-token pooling just uses the output representation of the last token as the embedding of the whole sequence.

Given that LLMs showcase major sensitivity to the formatting of the prompt (Zhuo et al., 2024; Ngweta et al., 2025), the pooling strategy, intuitively, should have a major impact on the VLEs' performance. Despite this, recent VLE works (Meng et al., 2025; Chen et al., 2025) fail to investigate this effect and simply resort to last-token pooling, without using dedicated sequence-end tokens. In this work, we analyze the effect of different pooling strategies on downstream VLE performance. We show that last-token pooling (without the sequence-end token) leads to substantial performance loss and makes the VLEs highly sensitive to the formatting artifacts at the end of the text (i.e., instruction) input, e.g., whether the input ends with punctuation or a newline character. We then show that introducing the dedicated sequence-end character, if only in task-specific fine-tuning, removes this sensitivity and makes VLEs robust to irrelevant formatting artifacts. In contrast to last-token pooling, we find mean pooling to be robust to formatting artifacts, which, in contrast to current practice, recommends it for a default pooling strategy when fine-tuning VLEs. We release our code and models at `https://github.com/m-elio/LVLMs-Encoder`.

2 Relevant Work

Given the impressive generative performance of LLMs, efforts to post-hoc adapt them into encoder models for discriminative tasks quickly emerged. BehnamGhader et al. (2024) introduced LLM2Vec, which converts an LLM into an encoder by (i) removing future-token masking (i.e., allowing bidirectional contextualization) (ii) and continued model training via masked token prediction and sequence-level self-supervised contrastive training. Schmidt et al. (2024) extended LLM2Vec by aligning a massively multilingual MT encoder to the LLM2Vec model.

In the vision-language realm, the early embedding models like CLIP (Radford et al., 2021) and SigLIP (Zhai et al., 2023) trained aligned text and vision encoders from scratch. More recent work leveraged pretrained LLMs to bootstrap the training of CLIP-like models (Huang et al., 2024). Following the trend in the text-only models, most recent work focused on post-hoc converting LVLMs, as generative models, into encoders (i.e., VLEs). To this end, VLM2Vec (Jiang et al., 2025) introduced a contrastive learning framework and also MMEB, a benchmark for evaluating VLEs. Subsequently, Meng et al. (2025) extended VLM2Vec with support for videos and visual documents; whereas mmE5 (Chen et al., 2025) made it massively multilingual by training on multilingual data with hard negatives, synthesized by a state-of-the-art commercial LLM.

Evaluations of earlier, CLIP-like vision-language encoders studied models' robustness to realistic permutations in visual and textual input (Tu et al., 2023; Wang et al., 2024b; Zhang et al., 2024). We argue that for VLEs based on LLMs—notorious for their sensitivity to non-semantic prompt variation (Zhuo et al., 2024; Ngweta et al., 2025)—a study of models' sensitivity to the formatting of the input is paramount. In this work, we thus investigate such sensitivity, and in particular in relation to VLEs' pooling strategies.

3 Pooling and Formatting for VLEs

As illustrated in Figure 1, VLEs compress the sequence of visual and text tokens of arbitrary length into a single embedding, where the visual tokens—projected into the input embedding space of the LLM—are commonly prepended to the textual tokens. Observing that recent VLEs (Jiang et al., 2025; Meng et al., 2025; Chen et al., 2025) just use the output representation of the last token as the embedding of the whole sequence, we study the effect of two design decisions on the downstream VLE performance: (1) inclusion of the dedicated token that denotes the end of the input sequence and (2) pooling strategy: mean vs. last-token pooling.

Mean pooling, where the sequence embedding is obtained by averaging the contextualized embeddings of tokens, is arguably the most common pooling approach for encoder models. However, for VLEs based on autoregressive LLMs (i.e., with causal self-attention)—mean pooling is a poor choice, since only the representation of the last token will have "seen" the whole input. This is why recent LLM-based VLEs (Meng et al., 2025; Chen et al., 2025) resort to last-token pooling (Muennighoff, 2022; Neelakantan et al., 2022) instead: the output representation of the last token is taken as the embedding of the entire input sequence.

Although text-based encoder models commonly use the explicit sequence-end token (`EOS`) (Neelakantan et al., 2022; Warner et al., 2025), LLM-based VLEs choose not to bound their input this way. This means that they represent an image-text input sequence with the contextualized representation of whichever token ended up being the last one in the text instruction—often a punctuation token or a newline character.

We question these two coupled design decisions of LLM-based VLEs, namely (i) last-token pooling based on (ii) naturally occurring last input token. To this end, in a fully controlled setup, we train VLEs from instruction-tuned LLMs, benchmarking, additionally, (1) removal of the causal masking (i.e., enabling bidirectional contextualization) with *mean pooling* and (2) introduction of a dedicated sequence-end token (`<EOS>`).

4 Experiments

We first test (§4.1) the sensitivity to prompt formatting for three existing VLEs: VLM2VEC-QWEN2VL-2B, VLM2VEC-QWEN2VL-7B, and MME5-MLLAMA-11B-INSTRUCT: all three use

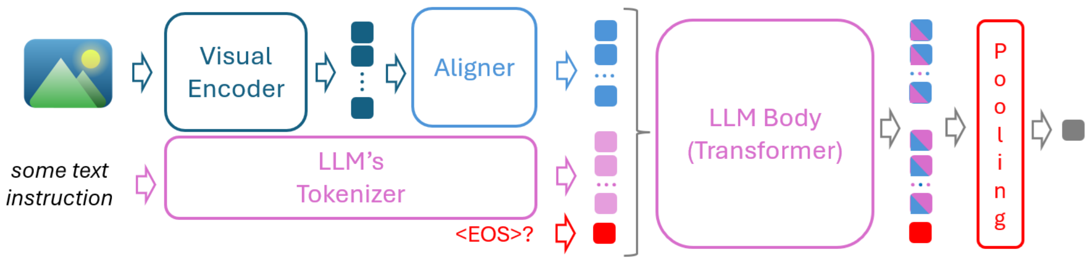

Figure 1: Illustration of formatting and pooling in VLEs (i.e., LVLMs turned into encoders) examined in this work. Visual tokens are projected (aligned) into the embedding space of the LLM (light blue) and prepended to the text tokens of the instruction (i.e., task description). Contextualized embeddings (pink-blue and blue-pink), output by the LLM's Transformer body, are pooled into an embedding vector representing the whole sequence. In red: design decisions under investigation in this work: (1) inclusion of a dedicated sequence-end token and (2) pooling strategy: mean pooling vs. last-token pooling.

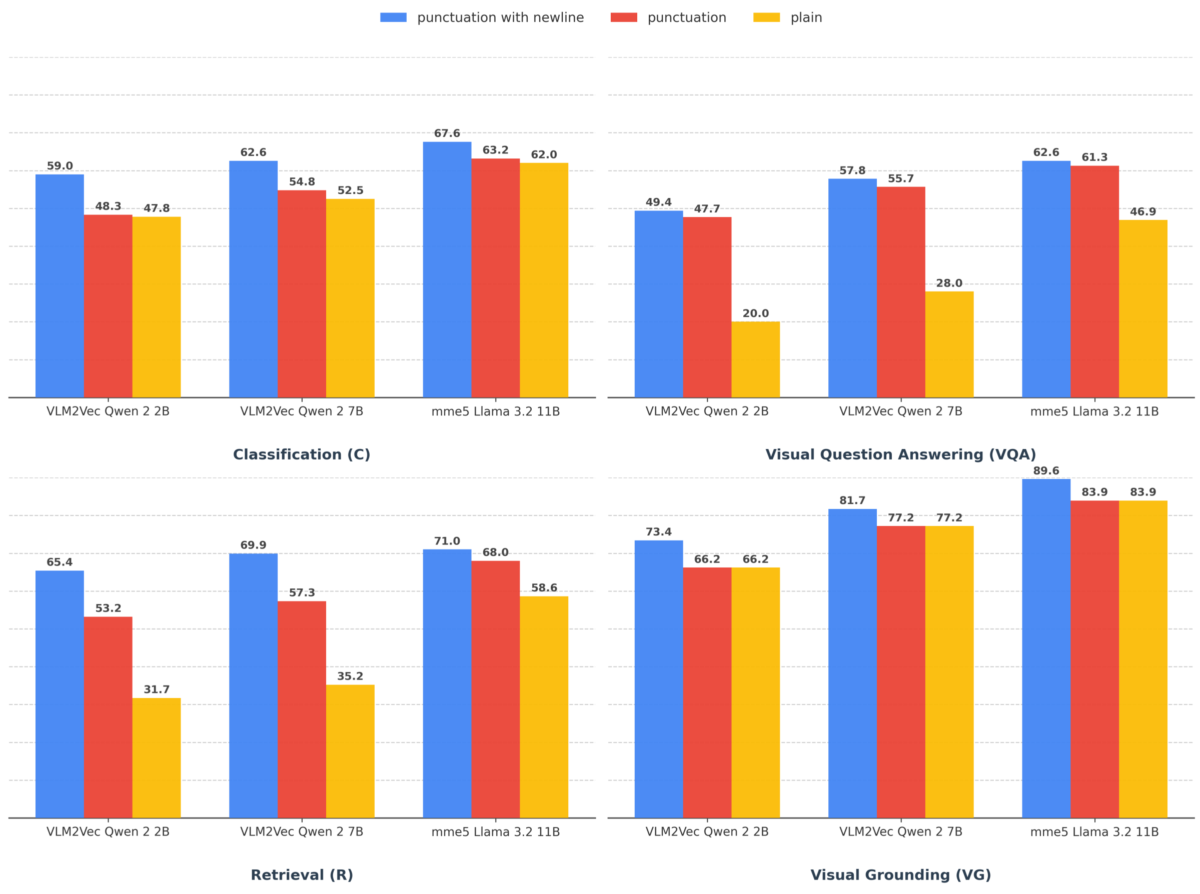

Figure 2: Performance of three off-the-shelf VLE models on the MMEB, for three different formatings of the test instances: results shown for each of the four task categories.

last-token pooling *without* appending the special sequence-end (<EOS>) token. We then train our own LLM-based VLEs (§4.2), starting from Qwen2.5-VL-3B-Instruct as the LLM backbone, seeking to isolate the effects of (1) removal of causal mask and mean pooling and (2) dedicated <EOS> token. We provide the training details in Appendix A. We evaluate on MMEB (Jiang et al., 2025), which has become the default benchmark for evaluating LLM-based VLEs. MMEB encompasses 37 vision-language embedding tasks, grouped into four task categories: *Classification* (C), *Visual Question Answering* (VQA), *Retrieval* (R) and *Visual Grounding* (VG).

In both §4.1 and §4.2, we experiment with three different formats of how the instruction (i.e., the

			Task Group			
Train	**Pooling**	**Format**	C	VQA	R	VG
Strip	Last, **EOS**	PU+NL	58.3	55.5	63.1	81.7
		PUNCT	58.1	55.7	62.0	81.2
		PLAIN	57.9	53.8	$\underline{59.0}$	81.1
	Last (no EOS)	PU+NL	59.6	54.2	60.8	81.2
		PUNCT	57.8	54.6	59.3	80.9
		PLAIN	57.4	$\underline{47.1}$	$\underline{51.0}$	81.0
	Mean	PU+NL	58.7	54.5	62.3	83.6
		PUNCT	57.6	54.4	62.3	83.4
		PLAIN	57.6	53.6	61.0	83.4
No Strip	Last, **EOS**	PU+NL	57.9	55.8	61.4	83.1
		PUNCT	57.8	55.6	60.8	82.8
		PLAIN	57.9	$\underline{50.9}$	58.6	82.8
	Last (no EOS)	PU+NL	58.4	54.2	61.2	81.4
		PUNCT	56.3	53.6	$\underline{56.7}$	79.9
		PLAIN	$\underline{54.3}$	$\underline{34.7}$	$\underline{41.1}$	79.9
	Mean	PU+NL	58.7	54.7	62.8	84.3
		PUNCT	59.1	54.6	62.6	84.0
		PLAIN	59.1	53.8	62.1	84.0

Table 1: Performance of VLEs we trained from Qwen2.5-VL-3B-Instruct, controlling for (1) pooling: *Mean* pooling (removed causal masking) vs. *Last*-token pooling; and (2) appending the <EOS> token in last-token pooling. Average performance (P@1) for all four MMEB task groups. Underline: performance drop ≥ 3 P@1 points w.r.t. the formatting with best performance for the same training setup and pooling strategy.

input) ends: (a) with punctuation (full stop for all task groups except VQA, where we use a question mark instead) *and* a newline character (PU+NL), (b) with punctuation only (PUNCT), where newline characters (if present in the data) are stripped away), and (c) without punctuation and without newline characters (PLAIN). We note that all three formats naturally occur in the MMEB dataset.

Finally, while training our own VLEs from Qwen2.5-VL-3B-Instruct (§4.2), we consider two different training setups: In Strip, we remove the trailing newline characters from the training instances; whereas in No-Strip we do not modify the instances, i.e., we keep them as originally formatted in MMEB training portions (with most instances containing the trailing newline character).

4.1 SotA VLEs: Sensitivity to Formatting

Figure 2 shows the performance of the three off-the-shelf LLM-based VLEs on MMEB, for the three different formats of the (same) test instances. All models exhibit the best performance when instances end with *punctuation and the newline character* and consistently and significantly drop in performance across all task groups when test instances lack the newline character or both punctuation and newline. The performance drop on VQA and Retrieval tasks is dramatic, reaching 30 P@1 points in many cases. We believe that such sensitivity to tiny (and semantically irrelevant) formatting differences is a result of the absence of <EOS> token in these models, which led models to overfit to frequent end-of-input artifacts in the training data, i.e., punctuation and newline characters.

4.2 Mean Pooling and <EOS> as Last Token

Table 1 shows the MMEB performance (for the same three test formats as in §4.1) of our VLE models trained with Qwen2.5-VL-3B-Instruct, where we control for pooling (*Mean* vs. *Last*-token) and appending <EOS> in last-token pooling. First, for Last-token pooling *without* <EOS>, we see the same behavior as for the off-the-shelf models in §4.1: performance varies dramatically across formats of test instances, i.e., when we remove newlines and punctuation (PUNCT and PLAIN), with most pronounced losses for VQA and Retrieval: e.g., in *No Strip* training, the performance of the last-token pooling model on PLAIN-formatted instances is over 20 P@1 points lower than the performance on the *PU+NL*-formatted test instances, for both VQA and Retrieval.

Both our interventions—(1) bidirectional contextualization and mean pooling (Mean) as well as (2) last token pooling with the dedicated <EOS> token (Last, <EOS>)—almost completely remove this sensitivity and lead to much more stable performance across all three formats of test instances: for Last, EOS, the difference in performance between PU+NL and PLAIN is 4.9 P@1 points for VQA and 2.8 P@1 points for retrieval; for Mean, the respective differences are even smaller, mere 0.9 (VQA) and 0.7 (Retrieval) P@1 points. We additionally visually showcase how the embeddings of the same instances change depending on the input-end formatting in Appendix B.

Finally, we observe that stripping training instances (*Strip*) also increases robustness, but it is less effective than adding <*EOS*> or doing Mean pooling. Without the EOS token, last-token pooling still leads to substantial performance drops on *PLAIN* instances, even with *Strip*ped training (-7 and -10 points compared to PU+NL on VQA and Retrieval, respectively).

4.3 Ablation Studies

We perform additional experiments for our models when using Mean pooling and Last-token pooling

Pooling	Format	Task Group C	VQA	R	VG
Mean, **Bi**	`PU+NL`	58.7	54.5	62.3	83.6
	`PUNCT`	57.6	54.4	62.3	83.4
	`PLAIN`	57.6	53.6	61.0	83.4
Mean, **Causal**	`PU+NL`	56.7	45.5	58.2	80.3
	`PUNCT`	56.9	45.8	58.1	80.6
	`PLAIN`	56.9	41.4	57.3	80.6

Table 2: Results for the ablation study on attention masking when using `Mean` pooling.

Pooling	Format	Task Group C	VQA	R	VG
Last, **EOS**	`PU+NL`	58.3	55.5	63.1	81.7
	`PUNCT`	58.1	55.7	62.0	81.2
	`PLAIN`	57.9	53.8	59.0	81.1
Last, **Random** (no EOS)	`PU+NL`	57.2	53.0	59.9	81.4
	`PUNCT`	57.8	53.3	59.8	80.7
	`PLAIN`	57.9	53.2	57.3	80.7

Table 3: Results for the ablation study on random formatting when using `Last`-token pooling.

without `<EOS>`.

For `Mean` pooling, we test whether bidirectional contextualization is required, or if this pooling strategy can be used with causal attention masking directly. We train a model using `Mean` pooling and causal attention masking in the `Strip` training setup. Results are reported in Table 2. Overall, results showcase major performance drop when using causal attention, in particular for the VQA, R and VG task categories. This is consistent w.r.t. results obtained by BehnamGhader et al. (2024).

For `Last`-token pooling, we test whether randomly applying one of the three formattings for each instance in the train set can lead to improved stability. Results are reported in Table 3. The model trained with random formatting *without* <*EOS*> performs worse w.r.t. the model trained *with* the <*EOS*> token. Specifically, the model trained with <*EOS*> performs better in the VQA and VG task categories.

5 Conclusion

Recent LLM-based vision-language encoders (VLEs) embed the sequence of visual and text tokens via last-token pooling, without a dedicated end-of-sequence (<EOS>) token. We empirically show that this leads to tremendous sensitivity to minor formatting changes at the end of the input, such as missing punctuation or a lack of trailing newline characters. We find that (1) last-token pooling *with* a dedicated <EOS> token and (2) mean pooling (VLE trained by removing the causal mask), both successfully mitigate this sensitivity to semantically irrelevant changes in input formatting.

Limitations

We highlight three main limitations of this work. First, our investigation is limited only to a narrow set of formatting variations that occur at the very end of the input (i.e., instruction): presence/removal of punctuation and presence/removal of a trailing newline character. Nonetheless, we show that even such minor formatting changes at test time can trigger performance collapse in VLEs that resort to last-token pooling without the <EOS> token. Second, our evaluation encompasses (only) three open VLEs of moderate size, from 2B to 11B parameters; this prevents extrapolation of our findings to larger and more capable models. Finally, our findings pertain only to the English language, since we evaluate the models on the MMEB benchmark, available only in English. We thus cannot generalize our findings to other natural languages.

Acknowledgments

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JURECA at Jülich Supercomputing Centre (JSC) (Jülich Supercomputing Centre, 2021). We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI (CUP H97G22000210007) under the NRRP MUR program funded by the NextGenerationEU. The work of Goran Glavaš was supported by the Alcatel-Lucent Stiftung and Deutsches Stiftungszentrum through the grant "Equitably Fair and Trustworthy Language Technology" (EQUIFAIR, Grant Nr. T0067/43110/23).

References

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. LLM2Vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*.

Haonan Chen, Liang Wang, Nan Yang, Yutao Zhu, Ziliang Zhao, Furu Wei, and Zhicheng Dou. 2025. mme5: Improving multimodal multilingual embeddings via high-quality synthetic data. *Preprint*, arXiv:2502.08468.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online, August 6, 2021*, pages 316–321. Association for Computational Linguistics.

Gregor Geigle, Florian Schneider, Carolin Holtermann, Chris Biemann, Radu Timofte, Anne Lauscher, and Goran Glavaš. 2025. Centurio: On drivers of multilingual ability of large vision-language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2831–2881, Vienna, Austria. Association for Computational Linguistics.

Weiquan Huang, Aoqi Wu, Yifan Yang, Xufang Luo, Yuqing Yang, Liang Hu, Qi Dai, Xiyang Dai, Dongdong Chen, Chong Luo, and Lili Qiu. 2024. LLM2CLIP: powerful language model unlocks richer visual representation. *CoRR*, abs/2411.04997.

Yongxin Huang, Kexin Wang, Goran Glavaš, and Iryna Gurevych. 2025. Modular sentence encoders: Separating language specialization from cross-lingual alignment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2167–2187, Vienna, Austria. Association for Computational Linguistics.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. 2025. Vlm2vec: Training vision-language models for massive multimodal embedding tasks. In *ICLR*.

Jülich Supercomputing Centre. 2021. JURECA: Data Centric and Booster Modules implementing the Modular Supercomputing Architecture at Jülich Supercomputing Centre. *Journal of large-scale research facilities*, 7:A182.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Llama Team. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Rui Meng, Ziyan Jiang, Ye Liu, Mingyi Su, Xinyi Yang, Yuepeng Fu, Can Qin, Zeyuan Chen, Ran Xu, Caiming Xiong, Yingbo Zhou, Wenhu Chen, and Semih Yavuz. 2025. Vlm2vec-v2: Advancing multimodal embedding for videos, images, and visual documents. *CoRR*, abs/2507.04590.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, and 1 others. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Lilian Ngweta, Kiran Kate, Jason Tsay, and Yara Rizk. 2025. Towards LLMs robustness to changes in prompt format styles. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 529–537, Albuquerque, USA. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Fabian David Schmidt, Philipp Borchert, Ivan Vulić, and Goran Glavaš. 2024. Self-distillation for model stacking unlocks cross-lingual NLU in 200+ languages. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6724–6743, Miami, Florida, USA. Association for Computational Linguistics.

Weijie Tu, Weijian Deng, and Tom Gedeon. 2023. A closer look at the robustness of contrastive language-image pre-training (CLIP). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Qizhou Wang, Yong Lin, Yongqiang Chen, Ludwig Schmidt, Bo Han, and Tong Zhang. 2024b. A sober look at the robustness of clips to spurious features. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference.

In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 Technical Report. *CoRR*, abs/2407.10671.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986.

Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, and Jiaqi Wang. 2024. Long-clip: Unlocking the long-text capability of CLIP. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LI*, volume 15109 of *Lecture Notes in Computer Science*, pages 310–325. Springer.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976, Miami, Florida, USA. Association for Computational Linguistics.

A Training Details

We train the models using the MMEB training set, limiting the instances for each task to 10,000. We limit the number of instances since there are some tasks with less than 10,000 instances. This ensures proper distribution and representation for all tasks. Training is performed for 1,000 steps. We follow the same hyperparameters as LLM2Vec for supervised contrastive training.

For the model with *mean pooling* we perform an additional pre-training step following the LLM2Vec methodology. We perform masked next token prediction with bi-directional attention enabled. We used a 32,000 instances subset of the Wikipedia-based Image Text (WIT) dataset as train set[1]. We follow the same hyperparameters as the LLM2Vec methodology.

Training is performed on one NVIDIA A100 GPU with 40 GB VRAM. Training on MMEB takes 20 hours, while the pre-training step for mean pooling takes 3 hours. To perform training on MMEB on a single GPU, the GradCache library (Gao et al., 2021) is used.

[1]wikimedia/wit_base

B Embedding Plots

In Figure 3 and Figure 4 we showcase embeddings plots obtained using t-SNE on the A-OKVQA task for the *punctuation* and *plain* formattings respectively. The plots refer to the Qwen2.5-VL-3B-Instruct model trained in the `Strip` setting using last-token pooling without adding the <*EOS*> token. While the query and target embeddings in the *punctuation* formatting are close to each other, in the *plain* formatting query and target embeddings tend to be more clustered.

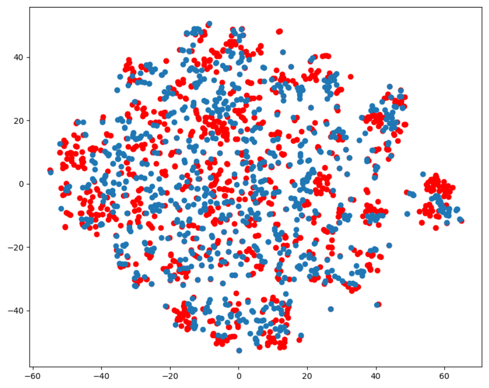

Figure 3: Embeddings plot for the model trained in the *Strip* setting for the *punctuation* formatting on the A-OKVQA task. Red dots are query embeddings, while blue ones are target embeddings.

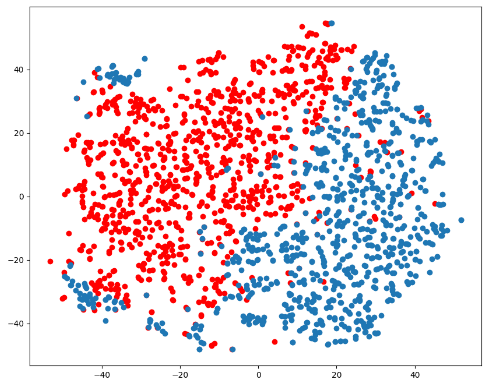

Figure 4: Embeddings plot for the model trained in the *Strip* setting for the *plain* formatting on the A-OKVQA task. Red dots are query embeddings, while blue ones are target embeddings.