Beschreibung großer Bilddatensätze in natürlicher Sprache mit Vision-Language Models für das automatisierte Fahren

Natural-language description of large image datasets using Vision-Language Models for automated driving

Julian Truetsch[1,2], Frank Bieder[1,2] und Christoph Stiller[2]

[1] FZI Forschungszentrum Informatik, Haid-und-Neu-Str. 10, 76131 Karlsruhe
[2] Karlsruher Institut für Technologie, Kaiserstraße 12, 76131 Karlsruhe

Zusammenfassung Moderne Bildverarbeitungssysteme, insbesondere im automatisierten Fahren, werden zunehmend datengetrieben entwickelt. Das Verständnis der zugrundeliegenden, oft Millionen Bilder umfassenden Datensätze entzieht sich jedoch der manuellen Inspektion und ist über Metadaten und Labelstatistiken nur eingeschränkt zugänglich. Wir stellen einen praxisorientierten Ansatz vor, mit dem sich Bilddatensätze mit Hilfe von open-weight VLMs automatisiert in natürlicher Sprache zusammenfassen lassen. Neben der konzeptionellen Beschreibung des zweistufigen, auch für sehr große Datensätze geeigneten Proposer-Filter-Ansatzes adressiert der Beitrag auch praxisrelevante Aspekte wie Prompt-Design, GPU-Speicherbedarf, Quantisierung und strukturierte Modellausgabe mittels *guided decoding*.

Schlüsselwörter Fahrzeuge, Vision-Language Models

Abstract Modern computer vision systems, particularly in automated driving, are increasingly developed in a data-driven manner. However, understanding the underlying datasets, which often comprise millions of images, is beyond the reach of manual inspection and only partially accessible through metadata and label statistics. We present a practice-oriented approach for automatically summarizing image datasets in natural language using open-weight VLMs. In addition to a conceptual description of

the two-stage proposer-filter approach, which is also suitable for very large datasets, this paper addresses practical aspects such as prompt design, GPU memory requirements, quantization, and structured model output via *guided decoding*.

Keywords Vehicles, Vision-Language Models

1 Motivation

Mit der zunehmenden Datenorientierung in der industriellen Bildverarbeitung wird die Zusammensetzung der Trainings- und Validierungsdatensätze zu einem zentralen Qualitätsfaktor. Insbesondere Datensätze für das automatisierte Fahren sind mittlerweile zu groß, um diese noch effektiv manuell sichten zu können. In der Praxis erfolgt die Analyse großer Bilddatensätze daher überwiegend über Metadaten (Zeitstempel, GNSS-Position, Sensorkonfiguration) und vordefinierte Labels. Beide Quellen sind durch ihren festgelegten Wertebereich beschränkt und können semantische Eigenschaften, wie etwa typische Umgebungsbedingungen, häufige Objekteigenschaften oder unbeabsichtigte Biases, nur teilweise erfassen.

Aktuelle Vision-Language Models (VLMs) bieten einen alternativen Zugang: Sie können Bildinhalte in natürlicher Sprache beschreiben und sind nicht an einen vordefinierten Labelraum gebunden. Während vorhergehende Arbeiten zur *set difference captioning* [1,2] diese Fähigkeit genutzt haben, um *Unterschiede* zwischen zwei Bildmengen in natürlicher Sprache zu beschreiben, betrachten wir hier den verwandten, aber eigenständigen Fall der Beschreibung eines *einzelnen* Datensatzes.

2 Vision-Language Models und Self-Hosting

VLMs verbinden die Sprachfähigkeiten großer Sprachmodelle (LLMs) mit der visuellen Verarbeitung von Bildencodern und können textuelle Anweisungen (*Prompts*) gemeinsam mit einem oder mehreren Bildern als Eingabe verarbeiten [3]. Neben kommerziellen Modellen, die nur über kostenpflichtige APIs zugänglich sind, hat sich in den letzten Jahren eine leistungsfähige Familie an *open-weight* Modellen eta-

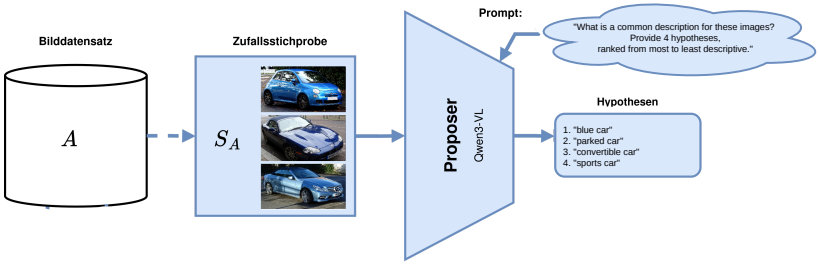


Abbildung 1: Einstufiger Ansatz zur Beschreibung eines Bilddatensatzes A in natürlicher Sprache. Der *Proposer* generiert gerankte Beschreibungshypothesen aus zufälligen Teilmengen $S_A \subset A$.

bliert, deren Gewichte frei heruntergeladen und lokal ausgeführt werden können. In unserer Implementierung verwenden wir `Qwen3-VL` [4] als VLM und LLM und `gpt-oss` [5] als reines Sprachmodell für Evaluierungszwecke. Die lokale Ausführung ist insbesondere dann wichtig, wenn sensitive Daten, wie sie in vielen industriellen Aufnahmen vorliegen, aus regulatorischen oder wettbewerblichen Gründen das Unternehmen nicht verlassen dürfen.

Für die effiziente Ausführung dieser Modelle wurde eine Reihe verschiedener Frameworks entwickelt; wir verwenden in diesem Beitrag das weit verbreitete und auch für kommerzielle Anwendungen geeignete Framework `vLLM` [6]. Es unterstützt eine Vielzahl gängiger Modellarchitekturen sowie Optimierungen wie *continuous batching* und *PagedAttention*, die für einen hohen Durchsatz bei der Verarbeitung großer Bildmengen erforderlich sind. Die Modelle lassen sich mit wenigen Zeilen Code über eine OpenAI-kompatible HTTP-Schnittstelle oder direkt über die Python-API ansprechen, was die Integration in bestehende Datenpipelines deutlich vereinfacht.

3 Beschreibung von Bilddatensätzen

Die direkte Verarbeitung aller Bilder eines Datensatzes durch ein VLM ist durch das begrenzte Kontextfenster aktueller VLMs nicht möglich. Stattdessen wird (ggf. wiederholt) eine zufällige Teilmenge $S_A \subset A$ einer festen Größe gezogen und zusammen mit einem Prompt,

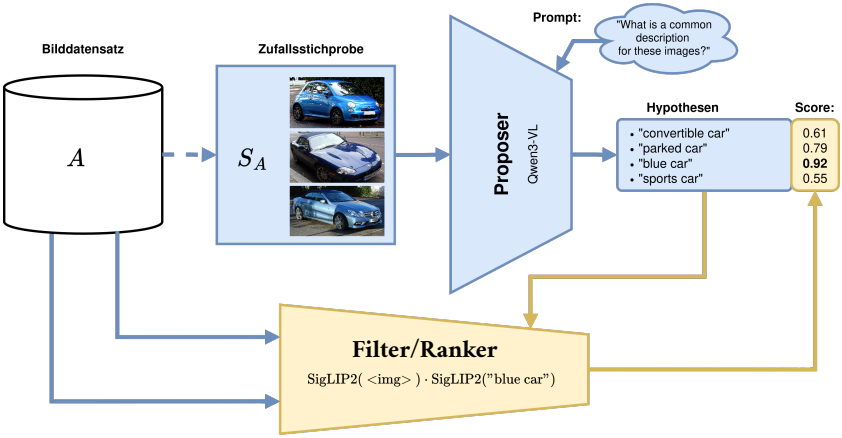


Abbildung 2: Zweistufiger Ansatz zur Beschreibung eines Bilddatensatzes A in natürlicher Sprache. Der *Proposer* generiert Beschreibungshypothesen aus zufälligen Teilmengen $S_A \subset A$; der *Filter/Ranker* verifiziert für jede Hypothese, auf welchen Anteil des vollständigen Datensatzes A sie zutrifft.

der zur Generierung mehrerer Beschreibungshypothesen auffordert, an das VLM übergeben. Mehrere Wiederholungen liefern einen Pool an Kandidaten-Beschreibungen. Alternativ können Bildbeschreibungen (Captions) zunächst einzeln erzeugt und anschließend von einem reinen LLM zu Hypothesen aggregiert werden.

Bei einem solchen stichprobenbasierten Ansatz stellt sich die Frage, ob die auf einer einzelnen Bilder-Stichprobe erhaltenen Beschreibungen tatsächlich auf den Datensatz in seiner Gesamtheit zutreffen, oder nur das Resultat von Sampling-Bias sind. Je diverser der Datensatz, desto schwieriger ist es, den Datensatz durch eine einzelne Stichprobe adäquat abzubilden. Das Risiko, durch Sampling-Bias verzerrte Stichproben zu erhalten steigt zudem unter Rauschen [1, 2], also wenn einzelne Bilder *nicht* zu einer übergreifenden Datensatzbeschreibung passen, unabhängig davon, ob es sich dabei um natürliche Ausreißer oder Fehler bei der Zusammenstellung des Datensatzes handelt.

Wir untersuchen daher neben einem einstufigen, auf einer einzelnen Stichprobe basierenden Ansatz (Abb. 1), auch zweistufige Architekturen (Abb. 2), bestehend aus einem *Proposer*, der, wie zuvor beschrie-

ben, ausgehend von zufälligen Teilmengen Beschreibungshypothesen generiert, sowie einem nachgelagerten *Ranker* oder *Filter*, der diese Hypothesen über den vollständigen Datensatz verifiziert. Der einstufige Ansatz gibt hingegen direkt die auf der Stichprobe vom Proposer-VLM generierten Hypothesen als Datensatzbeschreibung aus.

3.1 Ranker

Der zweistufige Ansatz mit einem Ranker ist die direkte Adaption der VisDiff-*set difference captioning*-Methodik [1,2], wird hier jedoch auf den Ein-Datensatz-Fall übertragen und entsprechend angepasst.

Der Ranker berechnet für jede Hypothese eine Metrik, die beschreibt, wie gut diese Hypothese den Datensatz als Gesamtes beschreibt. Entscheidend dabei ist, dass diese Metrik alle Bilder des Datensatzes berücksichtigt, und nicht nur die Bilder der Proposer-Stichprobe. Die Hypothesen werden dann entsprechend dieser Metrik sortiert, und die top-N-Hypothesen werden als Datensatzbeschreibung ausgegeben. Im einfachsten Fall wird jedes Bild einzeln von einem VLM mit der Hypothese verglichen, ein Score berechnet wie gut die Hypothese zu dem Bild passt, und dieser Score über den gesamten Datensatz gemittelt.

Wir betrachten zunächst zwei Varianten, die beide direkte Adaptionen des VisDiff-Rankers [1] für Beschreibungen eines einzelnen Datensatzes sind: (i) ein generativer Ranker, der das VLM fragt, ob die jeweilige Hypothese zutrifft; (ii) ein eingebettungsbasierter Ranker mit `SigLIP 2` [7], der Hypothese und Bilder in einen gemeinsamen Vektorraum projiziert und die Kosinusähnlichkeit als Score interpretiert.

3.2 Experimente

Wir evaluieren sowohl den einstufigen als auch die zweistufigen Ansätze auf dem AD-Diff-Bench-Benchmark [2]. AD-Diff-Bench ist ursprünglich für die Aufgabe *set difference captioning* [1] konzipiert, bei der Unterschiede zwischen Bilder-Set-Paaren gefunden werden sollen. Jedes Set-Paar des Benchmark-Datensatzes verfügt bereits über eine Beschreibung des ersten Sets des Paares, wir verwenden für unsere Evaluation entsprechend jeweils nur das erste Set eines Set-Paares.

Der Benchmark-Datensatz besteht aus drei Splits (Abb. 3): `web-scraped`, `annotation-filtered` und `CLIP-filtered`. Der Split

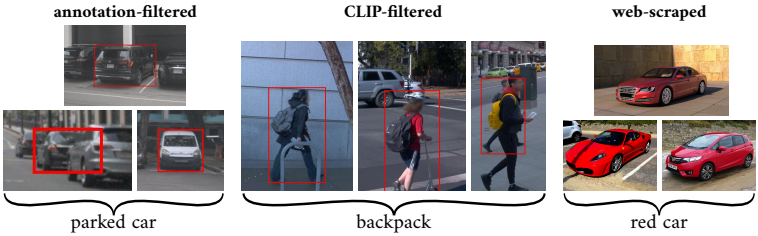


Abbildung 3: Beispielbilder mit Ground-Truth-Beschreibung. jeweils eines zu beschreibenden Bilder-Sets für jeden der drei Splits (web-scraped, annotation-filtered, CLIP-filtered) des Benchmark-Datensatzes AD-Diff-Bench [2].

Tabelle 1: Einstufige und zweistufige Verfahren ohne Domänen-Referenz, acc@1 (Mittel über drei Seeds, Streuung $\leq .039$). *Einstufig* (B2): VLM ohne Ranker (1×100 Bilder); *Zweistufig*: VLM oder LLM auf Captions als Proposer (3×20 Bilder, Qwen3-VL-30B-A3B), dann Ranker (mittlere SigLIP2-Kosinusähnlichkeit zur Zielmenge (SigLIP2-Ranker, B1 und B3); oder gemittelte VLM-Bewertung per Bild (VLM-Ranker, B4, Qwen3.5-35B-A3B, nur annotation-filtered)).

		annotation-filtered (60)	web-scraped (180)
B2	einstufig	**.625**	**.604**
B1	zweistufig (VLM) + SigLIP2-Ranker	.447	**.599**
B3	zweistufig (LLM) + SigLIP2-Ranker	.358	**.613**
B4	zweistufig (VLM) + VLM-Ranker	.431	–

`annotation-filtered` besteht aus Bildern aus Datensätzen für das automatisierte Fahren (KITTI [8], NuImages [9], Waymo Open Perception [10]), gefiltert in unterschiedliche Sets durch Datensatz-Label und Metadaten. `CLIP-filtered` basiert auf denselben Datensätzen, verwendet aber CLIP [11], um Bilder anhand ihrer Kosinusähnlichkeit zu vordefinierten Schlagwörtern im CLIP-Embeddingraum in unterschiedliche Sets zu sortieren, die Ground-Truth-Beschreibung ist dann das jeweilige Schlagwort. `web-scraped` basiert auf mittels Bing-Bildersuche aus dem Internet gescrapten Bildern, der jeweilige Suchbegriff definiert in diesem Fall das Ground-Truth-Label.

Unsere Evaluation basiert nur auf den Splits `annotation-filtered` (`ann60`) und `web-scraped`, da `CLIP-filtered` zu Gunsten des SigLIP2-basierten Rankers und Filters verzerrt ist. Sowohl bei CLIP als auch

bei SigLIP2 handelt es sich um kontrastiv auf Web-Alt-Texten trainierte Dual-Encoder. Diese Ähnlichkeit in Architektur und Trainingsdaten lässt geteilte Fehlermodi erwarten, weshalb eine Überschätzung der Genauigkeit eines SigLIP2-basierten Ansatzes auf einem mit CLIP konstruierten Benchmark zu befürchten ist.

Das Evaluationsprotokoll ist an das von AD-Diff-Bench angelehnt; das open-weights LLM `gpt-oss` [5] vergleicht die ausgegebene Datensatzbeschreibung mit der Ground-Truth-Beschreibung auf semantische Äquivalenz und gibt eine Bewertung von `0` (keine Übereinstimmung), `1` (teilweise Übereinstimmung) oder `2` (semantisch äquivalent) aus. Strukturierte Modellausgabe mittels *guided decoding* garantiert, dass das LLM niemals mit einer anderen Ausgabe als `0`, `1` oder `2` antworten kann. Den Prompt für gpt-oss, der ursprünglich auf einem manuell gelabelten Hypothesen-Datensatz getunt wurde, haben wir mit geringfügigen Anpassungen übernommen. Ein detaillierter, strukturierter Prompt, sowohl mit praktischen Beispielen als auch einem differenzierten Regelwerk hat sich dabei als am effektivsten herausgestellt. Proposer-VLM/LLM verwenden einen deutlich kürzeren Prompt, der neben klaren Anweisungen ebenfalls Beispiele enthält.

Sowohl für den einstufigen als auch den zweistufigen Ansatz verwenden wir in allen Experimenten `Qwen3-VL` [4] als Proposer-VLM/LLM. Im zweistufigen Ansatz mit VLM-Proposer (LLM-Proposer) werden für drei Stichproben von jeweils 20 Bildern (Captions) vom Proposer-VLM (Proposer-LLM) jeweils 10 Hypothesen generiert. Der einstufige Ansatz verwendet nur eine Stichprobe von 100 Bildern und generiert fünf Datensatzbeschreibungen. Der SigLIP2-Ranker des zweistufigen Ansatzes lädt die Modellvariante `SigLIP 2 Giant`. Der alternative VLM-Ranker basiert auf `Qwen3.5-35B-A3B` und verwendet wie `gpt-oss` guided decoding für ein garantiert eingehaltenes, automatisch parsebares Ausgabeformat.

Der zweistufige Ansatz liefert auf `annotation-filtered` deutlich schlechtere Ergebnisse als der einstufige Ansatz (Tab. 1). Auf `web-scraped` erzielen dagegen alle getesteten Ansätze ähnliche Ergebnisse. Es ist anzunehmen, dass dies an dem Domain-Shift zwischen web-scraped (Bilder aus dem Internet) und annotation-filtered (Straßenverkehr) liegt: SigLIP2 ist auf aus dem Internet gescrapten Bildern und deren Alt-Texts trainiert, und die Embeddings sind entsprechend den a-priori Wahrscheinlichkeiten von Bildern und Texten im Inter-

net angepasst, nicht spezifisch auf die Domäne Straßenverkehr. Der LLM-Proposer ist auf `annotation-filtered` deutlich schlechter als der VLM-Proposer, und vergleichbar auf `web-scraped`. Im Folgenden untersuchen wir daher nur Varianten mit VLM-Proposer.

Der Einfluss von guided decoding auf den VLM-Ranker ist vernachlässigbar. Von 617.697 Ranking-Schritten *ohne* guided decoding trat nur zweimal eine Verletzung des im Prompt festgelegten Ausgabeformats auf. Der VLM-Ranker ist um mehr als einen Faktor drei langsamer als der SigLIP2-Ranker. Dies liegt darin begründet, dass generative VLMs autoregressiv arbeiten, für jedes Bild also mehrere Forward-Passes durch das Modell erfolgen müssen, anstatt nur ein Forward-Pass bei einem Embedding-Model wie SigLIP2. Da die Genauigkeit zudem leicht unter der des SigLIP2-Rankers liegt, wird im Folgenden in allen Experimenten der SigLIP2-Ranker (oder Filter) verwendet.

4 Beschreibung von Bilddatensätzen unter Domain-Shift

Wenn die schlechteren Ergebnisse des SigLIP2-Rankers auf dem `annotation-filtered` Split im Vergleich zum `web-scraped` Split im Domain-Shift von Bildern aus Datensätzen für das automatisierte Fahren zu den SigLIP2-Trainingsdaten begründet sind, bietet sich eine Standardisierung der SigLIP2-Kosinusähnlichkeiten auf einen innerhalb der Domäne gleichen Mittelwert und Varianz an: $z_H = (\bar{s}_{\text{Ziel},H} - \mu_{\text{Ref},H})/\sigma_{\text{Ref},H}$, mit Mittelwert $\mu_{\text{Ref},H}$ und Standardabweichung $\sigma_{\text{Ref},H}$ der Kosinusähnlichkeit der Hypothese H auf einer Domänen-Referenz, also einem ausreichend großen Datensatz für die Domäne repräsentativer Bilder. Für AD-Diff-Bench nutzen wir, entsprechend den zwei verschiedenen Domänen in AD-Diff-Bench (Internet-Bilder und Straßenverkehr) zwei Domänen-Referenzen, jeweils aus den Bildern des entsprechenden Datensatz-Splits (`annotation-filtered` bzw. `web-scraped`). Bilder aus der Zielmenge werden aus den Domänen-Referenzen entfernt.

Dass der SigLIP2-Ranker das Ranking von Proposer-Hypothesen in der set difference captioning Aufgabe klar verbessert [2], aber nicht für Beschreibungen einzelner Sets, motiviert eine weitere Familie an Verfahren, die die Domänen-Referenz als Referenzset im Sinne des set difference captioning nutzt. Eine Hypothese wird umso höher

gerankt, je mehr sie sich vom Domänen-Hintergrund unterscheidet. Die VisDiff- [1] und AD-Diff- [2] Ranker verwenden als Vergleichsmetrik zwischen den beiden Sets den AUROC zwischen den Kosinusähnlichkeitsverteilungen der jeweiligen Hypothese H auf beiden Sets. Wir untersuchen zudem zwei weitere Metriken für den Abstand zwischen den beiden Sets: die KL-Divergenz und die p-Werte eines Permutationstests zwischen den Kosinusähnlichkeitsverteilungen.

4.1 Filter

Ein alternativer Ansatz zum Ranking aller Proposer-generierten Hypothesen ist ein Filter-Schritt als zweite Stufe nach dem Proposer. Die Reihenfolge der vom Proposer ausgegebenen Hypothesen wird in diesem Fall nicht geändert, stattdessen werden nur diejenigen Hypothesen aussortiert, die nicht auf den Datensatz in seiner Gesamtheit zutreffen, die also Resultat von Sampling-Bias sind. Im Allgemeinen ist es schwierig, einen konkreten Schwellwert für die mittlere Kosinusähnlichkeit festzulegen, was die Anwendung dieses Verfahrens ohne Referenzdatensatz nicht erlaubt. Ist aber eine Domänen-Referenz verfügbar, so lassen sich für den Filter dieselben Metriken wie für den Ranker anwenden (AUROC, KL-Divergenz, Permutationstest).

4.2 Experimente

Tabelle 2 zeigt, dass sowohl Ranking über standardisierte mittlere Kosinusähnlichkeit, als auch Ranking über AUROC, KL-Divergenz, oder Permutations-p zwischen Zielmenge und Domänen-Referenz die Genauigkeit des Rankers auf dem Datensatz mit Domain-Shift (`annotation-filtered`) im Vergleich zum Ranking ohne Domänen-Referenz (`zweistufig (VLM)`) wesentlich verbessert, und dort nahezu die Genauigkeit des einstufigen Ansatzes erreicht. Unter 20% Domänenrauschen, wenn also 20% der Zielmenge durch zufällig aus derselben Domäne gesampelte Bilder ausgetauscht werden, verstärkt sich dieser Trend noch weiter. Einstufiger Ansatz und Domänen-Referenz-Ansätze sind dann als gleichwertig zu betrachten. Ein eindeutiger Vorteil des Rankers mit Domänen-Referenz gegenüber dem einstufigen Ansatz ergibt sich jedoch auch in diesem Fall nicht.

Tabelle 2: Ranking und Filtern der Kandidaten, ohne und mit Domänenrauschen (20% jeder Zielmenge durch Bilder aus der Domäne ersetzt). Filter behalten obere 50%, sofern nicht anders angegeben; *Permutations-p*: $p \leq 0.05$. Mittel über drei Seeds; ΔB1 und ΔB2 sind gepaarte Differenzen gegenüber B1 bzw. B2. **Fett**: Bestes Verfahren des Splits; unterstrichen: bestes im Abschnitt. B3 wurde nicht mit Rauschen ausgewertet.

		Methode	annotation-filtered (60) acc@1	ΔB1	ΔB2	web-scraped (180) acc@1	ΔB1	ΔB2
ohne Rauschen	Basel.	B2 einstufig	.625	+.178	+.000	**.604**	+.004	+.000
		B1 zweistufig (VLM)	.447	+.000	−.178	**.599**	+.000	−.004
		B3 zweistufig (LLM)	.358	−.089	−.267	**.613**	+.012	+.008
	Ranker mit Dom.-Ref.	Standardisierung	.547	+.100	−.078	.568	−.031	−.035
		AUROC	.564	+.117	−.061	.581	−.019	−.022
		KL-Divergenz	.597	+.150	−.028	.582	−.017	−.021
		Permutations-p	.539	+.092	−.086	.579	−.021	−.024
	Filter mit Dom.-Ref.	Standardisierung	**.653**	+.206	+.028	.576	−.023	−.027
		AUROC	**.667**	+.219	+.042	.575	−.024	−.028
		KL-Divergenz (> 25%)	**.675**	+.228	+.050	.581	−.019	−.022
		Permutations-p	**.647**	+.200	+.022	.577	−.022	−.026
mit 20% Rauschen	Basel.	B2 einstufig	.603	+.222	+.000	**.569**	−.003	+.000
		B1 zweistufig (VLM)	.381	+.000	−.222	**.572**	+.000	+.003
	Ranker mit Dom.-Ref.	Standardisierung	.558	+.178	−.044	**.568**	−.005	−.002
		AUROC	**.592**	+.211	−.011	.531	−.041	−.038
		KL-Divergenz	**.600**	+.219	−.003	**.568**	−.005	−.002
		Permutations-p	.550	+.169	−.053	**.571**	−.001	+.002
	Filter mit Dom.-Ref.	Standardisierung	**.642**	+.261	+.039	.568	−.005	−.002
		AUROC	**.636**	+.256	+.033	.563	−.009	−.007
		KL-Divergenz (> 25%)	**.642**	+.261	+.039	**.581**	+.008	+.011
		Permutations-p	.622	+.242	+.019	.558	−.014	−.011

Der Filter-Ansatz mit Domänen-Referenz hingegen erzielt sowohl mit als auch ohne Domänen-Rauschen die besten Ergebnisse auf `annotation-filtered`, mit einer höheren Genauigkeit als der einstufige Ansatz für alle getesteten Metriken.

Alle Domänen-Referenz-Ansätze, sowohl Ranker- als auch Filter-basiert, sind auf `web-scraped`, dem Datensatz ohne Domain-Shift, erwartungsgemäß nahezu neutral. Im Vergleich der verschiedenen Metriken zueinander erzielt KL-Divergenz die besten Ergebnisse, und Stan-

dardisierung und Permutations-p die schwächsten, sowohl für Ranker als auch Filter, mit ähnlichen Trends mit und ohne Domänenrauschen.

5 Hardwareanforderungen

Die Modelle wurden parallel mit vLLM-Docker-Containern auf fünf Nvidia RTX 6000 Ada GPUs mit jeweils 48 GiB GPU-Speicher verteilt. Das Proposer-VLM/LLM Qwen3-VL-30B-A3B (30 Mrd. Parameter) belegt unquantisiert in bf16 (ca. 2 Byte per Parameter) 59 GiB GPU-Speicher allein für die Modellgewichte, und wurde daher tensorparallel auf zwei GPUs verteilt deployed. Das Evaluations-LLM gpt-oss-120b (120 Mrd. Parameter) ist MXFP4-quantisiert und benötigt etwas mehr als ein halbes Byte pro Parameter, insgesamt 69 GiB an GPU-Speicher für Gewichte, und wird daher ebenfalls verteilt auf zwei GPUs deployed. SigLIP2-giant misst insgesamt ca. 1,8 Mrd. Parameter, 1,1 Mrd. Parameter für den Vision-Encoder und 0,7 Mrd. für den Text Encoder. Damit benötigt es in bf16 nur 3,5 GiB für die Modellgewichte, und passt auf eine GPU. Qwen3.5-35B-A3B (35 Mrd. Parameter) wurde in FP8-Quantisierung deployed, damit benötigt es bei ungefähr gleicher Parameterzahl nur etwas mehr als die Hälfte des GPU-Speichers von Qwen3-VL (34 GiB), und passt damit auf eine GPU.

Zusätzlicher GPU-Speicher wird bei allen autoregressiven Modellen für den KV-Cache benötigt, abhängig von der Anzahl an Bildern und der Länge des Textes der vom Modell gleichzeitig verarbeitet wird. Bei moderaten Kontextlängen kamen alle unsere Modelle mit wenigen GiB KV-Cache aus, auch gpt-oss als einziges Modell mit durch aktiviertes Reasoning erhöhtem KV-Cache-Bedarf. Um wiederholte Benchmark-Sweeps zu beschleunigen, wurden alle Modellausgaben (Texte und Embeddings) mittels Valkey [12] gecached.

6 Fazit

In diesem Beitrag konnten wir demonstrieren, wie sich bereits mit einem einfachen, einstufigen VLM-Ansatz große Bilddatensätze in natürlicher Sprache zusammenfassen lassen. Ein zweistufiger Proposer-Ranker-Ansatz führt, anders als bei Beschreibungen von Unterschieden zwischen Datensätzen [1, 2], für diese Anwendung im

Allgemeinen nicht zu besseren Ergebnissen. Für den in der Praxis häufigen Fall von Daten mit hohem Domain-Shift zu den Trainingsdaten der verwendeten VLMs, wie etwa bei Anwendungen für das automatisierte Fahren, führt ein zweistufiger Proposer-Filter-Ansatz dagegen zu robusteren Beschreibungen als ein einstufiger Ansatz.

Literatur

1. L. Dunlap *et al.*, "Describing differences in image sets with natural language," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024.
2. J. Truetsch, F. Hauser, C. Stiller, and F. Bieder, "Understanding autonomous driving datasets by describing differences between image subsets in natural language," 2026, arXiv:2609.03677.
3. H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2023.
4. S. Bai *et al.*, "Qwen3-VL technical report," 2025, arXiv:2511.21631.
5. OpenAI, "Introducing gpt-oss," OpenAI Blog. https://openai.com/index/introducing-gpt-oss/, 2025.
6. W. Kwon *et al.*, "Efficient memory management for large language model serving with PagedAttention," in *Proc. 29th Symp. Oper. Syst. Princ.*, 2023.
7. M. Tschannen *et al.*, "SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features," 2025, arXiv:2502.14786.
8. A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, 2013.
9. H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu *et al.*, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.
10. P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.
11. A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021.
12. The Linux Foundation, "Valkey," https://valkey.io, 2026.