Retrieval-VLA: Training-Free In-Context Adaptation for Vision-Language-Action Models

Yue Zhang[1*] Rui Wang[2*] Jiehong Lin[1] Zhongrui Wang[2] Xiaojuan Qi[1,†]
[1]The University of Hong Kong [2]Southern University of Science and Technology
u3009724@connect.hku.hk, wangr2025@mail.sustech.edu.cn, mortimer.jh.lin@gmail.com
wangzr@sustech.edu.cn xjqi@eee.hku.hk

Retrieval-VLA: Training-Free In-Context Adaptation for Vision-Language-Action Models

Supplementary Material

1. Backbones Details

We employ two state-of-the-art vision-language action (VLA) models as backbones in our retrieval framework: π_0 [2] and OpenVLA [4]. We integrate both models into our Retrieval-VLA framework without modifying their internal architecture. This demonstrates the plug-and-play nature of our method and its compatibility with a wide range of VLA backbones.Detailed descriptions are provided in Sec. 1.1 and Sec. 1.2, respectively.

1.1. Details of π_0

π_0 is a pre-trained transformer-based vision-language-action (VLA) model designed to support general-purpose robot control across diverse embodiments and tasks. It builds upon the PaliGemma vision-language model [1] and incorporates robotics-specific capabilities via architectural extensions. The model encodes RGB image observations, language instructions, and proprioceptive robot states, and outputs continuous action sequences through a flow-matching mechanism. π_0 follows the PaliGemma VLM [1] design, with the following key modifications:

- **Robotics-specific inputs and outputs:** Unlike the original PaliGemma model, which only processes vision and language inputs, π_0 introduces additional input and output token projections to handle robot-specific information. These include the robot's proprioceptive state vector q_t (e.g., joint angles) as input, and a sequence of future actions $A_t = [a_t, a_{t+1}, ..., a_{t+H-1}]$ as output. These tokens are projected into the same embedding space as the image and language tokens, allowing them to be processed jointly within the transformer;
- **Flow matching timestep encoding:** To support trajectory prediction via flow matching—a technique inspired by diffusion models—π_0 incorporates an additional multi-layer perceptron (MLP) that embeds the flow matching timestep τ into each action token. This allows the model to learn how the action distribution evolves over time during training;
- **Dual-expert transformer design:** π_0 uses a two-expert transformer architecture. The first expert is the vision-language backbone initialized from the pre-trained PaliGemma VLM, which processes image and text inputs. The second expert, referred to as the *action expert*, is a smaller transformer module that is trained from scratch. It specializes in handling the robot-specific tokens (state and action) and generates continuous action outputs. These two experts share attention layers but maintain separate feedforward and projection weights.

The vision-language backbone of π_0 is initialized from the pre-trained PaliGemma model [1], which contains 3 billion parameters. The added action expert introduces approximately 300 million additional parameters, resulting in a total model size of 3.3 billion parameters. In our experiments, we adopt π_0 as a plug-and-play backbone in our retrieval framework, denoted as Retrieval-π_0.

1.2. Details of Open-VLA

OpenVLA is a publicly available open-source framework for vision-language manipulation tasks. It supports modular architectures, including CLIP-based visual encoders and various instruction conditioning mechanisms. It is a 7B-parameter open-source VLA model designed for generalist robot control. It is trained on 970k real-world robot manipulation episodes from the Open X-Embodiment dataset [**?**], covering a wide range of robot embodiments, tasks, and scenes.

The OpenVLA model builds upon a pretrained vision-language model backbone called Prismatic [3], which fuses multi-resolution features from two vision encoders—DINOv2 [9] and SigLIP [13] with a LLaMA 2 language model [12]. The robot control task is formulated as an autoregressive token prediction task: continuous robot actions are discretized into 256 bins per dimension and represented as special tokens in the LLM vocabulary. During inference, the model receives an image and a language instruction and autoregressively predicts a sequence of action tokens, which are then decoded into a 7-dimensional control command.

All model weights, code, and training pipelines for OpenVLA are publicly available[1], making it the first open-source generalist VLA model that supports both large-scale training and flexible downstream adaptation. The pre-training storage requirements for the OpenVLA are 7.7B parameters. In our experiments, we adopt OpenVLA as a plug-and-play backbone in our retrieval framework, denoted as Retrieval-OpenVLA.

2. Benchmark Details

2.1. LIBERO Benchmark Details

The LIBERO benchmark [6] is a recently proposed simulation suite for evaluating language-conditioned robotic manipulation. It is built on the Franka Panda robotic arm, and

[1] https://openvla.github.io

Table 1. LIBERO tasks details. The language instruction templates and the total number of tasks per suite are listed.

Suite	Instructions	Tasks
Spatial	*pick up the OBJ SPATIAL_REL and place it on the TARGET*	10
Object	*pick up the FOOD and place it in the CONTAINER*	10
Goal	*open/close the CONTAINER* *open the DRAWER and put the OBJ inside* *put the OBJ on/in the TARGET* *push the OBJ to the POSITION of the TARGET* *turn on the APPLIANCE*	10
Long	*put both OBJ1 and OBJ2 in the CONTAINER* *turn on the APPLIANCE and put the OBJ on it* *put the OBJ in the CONTAINER/APPLIANCE and close it* *place OBJ1 on TARGET1 and OBJ2 on TARGET2/at REL of TARGET2* *pick up the OBJ and place it in the caddy COMPARTMENT*	10

features a diverse set of manipulation tasks grounded in natural language and multimodal observations. Each task have RGB images from two viewpoints: a front-view camera providing a third-person perspective of the scene; a wrist-view camera mounted on the robot's end-effector, offering a first-person perspective. LIBERO is structured into four task suites, each designed to evaluate specific aspects of language-conditioned manipulation:

- **Spatial**: Tasks that require understanding of spatial relations (e.g., "place the red block to the left of the green one"). These tasks involve placing the same object in varying target positions.
- **Object**: Tasks that involve object grounding (e.g., "move the yellow mug to the top shelf"). These tasks emphasizes manipulating diverse objects within a consistent scene layout.
- **Goal**: Tasks that specify abstract high-level goals (e.g., "make a stack of three blocks"). These tasks includes heterogeneous operations such as opening containers, placing objects, or turning on appliances, all performed within a fixed environment.
- **Long**: Long-horizon tasks composed of multiple steps or subgoals (e.g., "open the drawer and place the spoon inside"). These tasks consists of ten extended tasks that require completing multiple subgoals across different scenes.

Each suite contains 10 distinct tasks, and each task includes 500 demonstrations, totaling 5,000 demonstrations per suite. Tab. 1 outlines the suites of the LIBERO benchmark.

2.2. CALVIN Benchmark Details

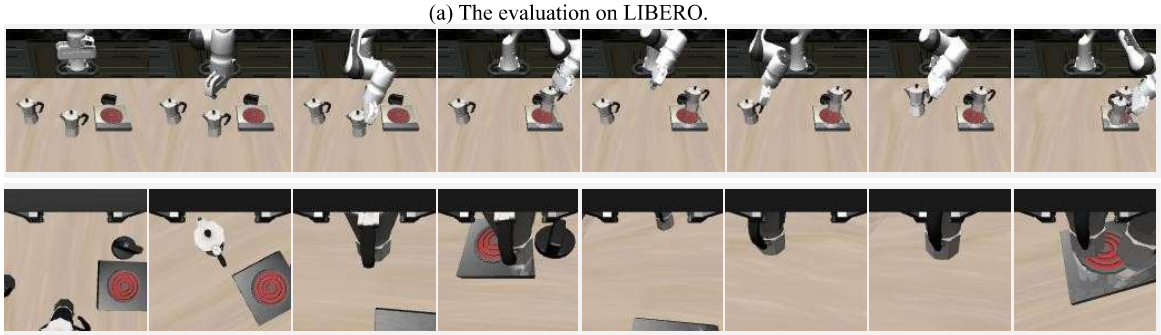
CALVIN [8] is a simulated benchmark designed to evaluate long-horizon, language-conditioned robotic manipulation using only onboard sensing. CALVIN includes four structurally similar but visually distinct tabletop environments: **A**, **B**, **C**, and **D**. Each environment contains:

- A 7-DOF Franka Emika Panda arm with a two-finger parallel gripper.
- A table with manipulable objects including a sliding door, a drawer, a light switch, a button, and three colored blocks.
- RGB-D cameras mounted statically.
- Proprioceptive sensors and a vision-based tactile sensor.

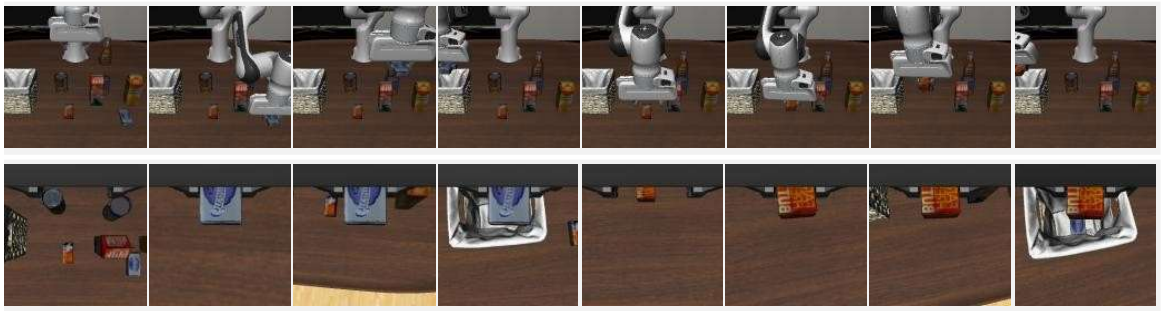
The benchmark defines 34 atomic manipulation tasks (e.g., open drawer, push red block right, turn on light) that can be composed into long-horizon task sequences. These tasks are distributed across four distinct environments that differ in texture, object positions, and scene layout to encourage generalization. While the robot and camera positions remain fixed, all static elements (e.g., drawer, button, switch) are placed differently in each environment. To evaluate generalization, CALVIN adopts the ABC→D protocol, which challenges agents to generalize both perceptually and spatially to novel environments. To further assess compositional generalization, CALVIN introduces the Long-Horizon Multi-Task Language Control (LH-MTLC) setting, where agents must follow 1,000 unique 5-step natural language instruction chains. Each chain consists of five feasible and non-redundant subtasks that can be completed sequentially from a predefined initial state. Example instructions include: "Open the drawer → Pick up the red block → Place it in the drawer → Close the drawer → Press the green button."

At inference time, the robot starts from a neutral pose, and transitions to the next instruction only if the current one is successfully completed. This ensures agents must rely solely on language and onboard perceptions, without bias from initial state-task correlation.

(a) The evaluation on LIBERO.



Put both moka pots on the stove.



Put both the cream cheese box and the butter in the basket.

(b) The evaluation on CALVIN environment D.

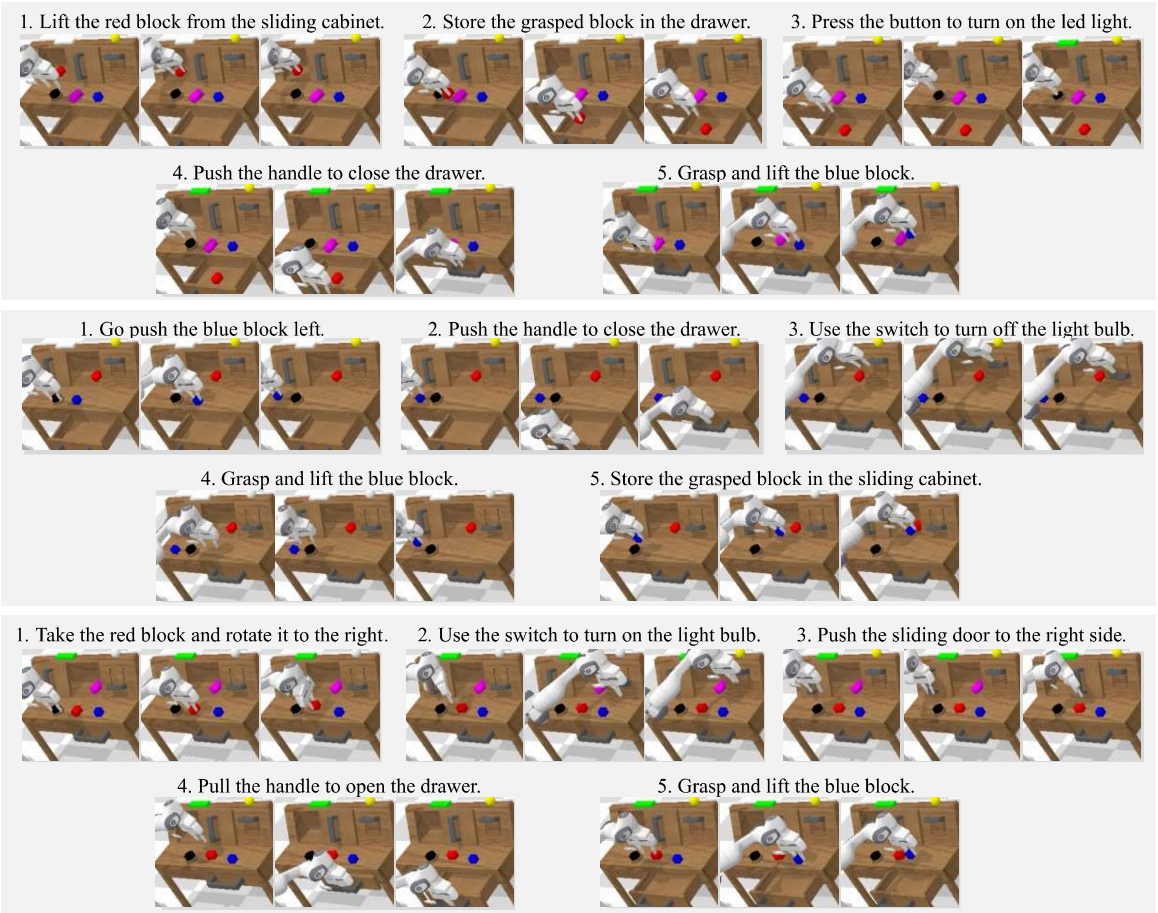


Figure 1. Task execution examples for LIBERO and CALVIN-D using Retrieval-VLA.

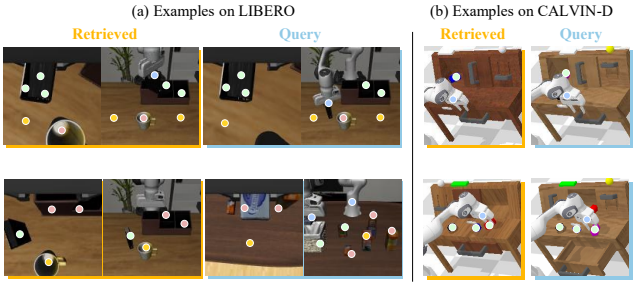


Figure 2. Visualization of Query and Retrieved keypoints during Retrieval-VLA test in the (a) LIBERO and (b) CALVIN-D environment.

3. Construction of Retrieval Demonstrations

For each task, we collect a small set of human or scripted demonstrations to serve as the retrieval memory during inference. Specifically, we collect 20 demonstrations per task for Libero, and 15–20 demonstrations per task for CALVIN following the ABC→D protocol, where demonstrations are sampled from environments A, B, and C, and evaluation is conducted zero-shot on the unseen environment D. Each demonstration consists of a language instruction, RGB(-D) observation, and corresponding action, and is preprocessed offline to extract object-centric keypoints using the Grounded-SAM-based pipeline. These demonstrations are stored in a unified key-value format and cached for fast retrieval during test-time inference. Fig. 1 presents additional qualitative examples of Retrieval-VLA executing tasks in the LIBERO and CALVIN Domain D settings. Fig. 2 presents a visualization of the query images used during test-time retrieval and the corresponding retrieved demonstration samples.

4. Hyper-parameter Details

Tab. 2 shows the important hyper-parameters in Retrieval-VLA framework, that influence retrieval quality, keypoint representation, and final action generation. This section provides detailed explanations of these parameters and the rationale behind their selection.

4.1. Instruction Embedding and Similarity

For aligning the input instruction x_t^{text} with stored demonstrations, we compute semantic similarity between instructions using cosine distance over token embeddings. The instruction embeddings v^{text} and v_i^d are obtained from the native tokenizer and embedding stack of the pre-trained VLA model (e.g., LLaMA [12] in OpenVLA backbone or Gemma [11] in π_0 backbone). No finetuning is applied to the language encoder. We find that cosine similarity provides sufficient alignment quality for semantic retrieval, and no additional threshold is used since we always select the most similar demonstration.

Table 2. Summary of major hyperparameters used in Retrieval-VLA.

Hyperparameter	**Value / Strategy**
Instruction similarity metric	Cosine similarity
Number of masks	4–32 manually defined
Points_per_side	2–64 manually defined
Pred_iou_thresh	0.80-0.98 manually defined
Stability_score_thresh	0.80-0.98 manually defined
Min_mask_region_area	256-8192 manually defined
Keypoints per object	Adaptive (typically 1–8)
Interpolation temperature λ	5.0– 30.0

4.2. Grounded-SAM: Number of Object Masks

To extract keypoints, we build upon the Grounded-SAM [10] pipeline, which integrates Grounding DINO [7] for object detection and Segment Anything Model (SAM) [5] for mask generation. This combination enables high-quality, open-vocabulary object segmentation, which we leverage to obtain task-relevant keypoints. In our implementation, we use the official pretrained Grounded-SAM model [10]. In our usage, we first manually define the set of object categories expected per image, which determines the number of masks to be generated. Then, Grounding DINO is used to extract object-level features automatically based solely on the RGB image input. These detections are passed to the SAM module to produce the corresponding segmentation masks, typically resulting in 4 to 32 object masks per frame.

4.3. Keypoint Clustering: K-means Radius and Cluster Count

For each object mask m_i, we apply K-means clustering to the visual feature patches inside the mask. We adopt an adaptive strategy based on spatial extent and feature variance. Specifically, we use n clusters (1-8). The clustering algorithm iterates until every feature point lies within a radius r of a cluster center. This leads to 1 to 8 keypoints per object, adaptively capturing object complexity without over-segmentation.

4.4. Interpolation Temperature Parameter λ

To fuse the retrieved action a^{d} with the base prediction a_t^{base}, we use the exponential interpolation equation:

$$a_t^{\text{new}} = e^{-\lambda d} \cdot a^{\mathrm{d}} + (1 - e^{-\lambda d}) \cdot a_t^{\text{base}} \tag{1}$$

where $d = \mathcal{R}(k_t^{\text{base}}, k^{\mathrm{d}})$ is the average Euclidean distance between keypoints. The temperature parameter λ controls the influence of the retrieved demonstration. We tune this hyperparameter separately for different VLA backbones and benchmarks:

- OpenVLA + LIBERO: $\lambda = 5.0 - 20.0$
- OpenVLA + CALVIN: $\lambda = 5.0 - 20.0$
- π_0 + LIBERO: $\lambda = 10.0 - 30.0$
- π_0 + CALVIN: $\lambda = 10.0 - 30.0$

Higher values of λ make the interpolation more sensitive to distance, resulting in stronger adaptation when the retrieved example is close.

4.5. Observation and Action Space

Observations and Action Space: The proprioceptive observation includes the end-effector's pose and gripper state. The action can be represented in one of two ways: either as 7-dimensional joint velocities, or as a 7-dimensional task-space vector comprising the end-effector's delta position (x, y, z), delta orientation (roll, pitch, yaw), and gripper aperture.

Kinematic Transformation: The two action representations are interchangeable through forward and inverse kinematics (FK/IK).

Image Preprocessing: All images are resized to 224×224 with padding.

5. Evaluation Metrics

For the LIBERO benchmark, we evaluate performance using the **success rate** on each of the four evaluation suites (Spatial, Object, Goal, and Long), as well as the **average success rate** across all tasks. A task is considered successful if the final state satisfies the success condition defined in the environment. The success rate is computed as:

$$\text{Success Rate} = \frac{\text{Successful Episodes}}{\text{Total Episodes}} \times 100\% \tag{2}$$

The **Average Success Rate** is obtained by taking the mean success rate across all tasks within a suite:

$$\text{Average Success Rate} = \frac{1}{N} \sum_{i=1}^{N} \text{Success Rate}_i \tag{3}$$

where N is the total number of tasks in the evaluation suite.

For the CALVIN benchmark, we follow standard evaluation protocols and report two metrics:

- **Success Rate per Instruction Chain:** The percentage of multi-step instruction chains (each consisting of 5 subtasks) where all 5 subtasks are completed successfully.
- **Average Completed Sequence Length:** The average number of subtasks successfully completed per instruction chain, ranging from 0 to 5. This metric reflects partial success and provides a finer-grained assessment of policy robustness.

All evaluations are conducted in a zero-shot setting without any task-specific finetuning.

References

[1] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024. 1

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 1

[3] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024. 1

[4] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 1

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 4

[6] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023. 1

[7] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024. 4

[8] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3): 7327–7334, 2022. 2

[9] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al.

Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1

[10] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024. 4

[11] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. 4

[12] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1, 4

[13] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023. 1