

Vision-Language-Action Modelの 推論高速化に向けた性能解析

薄井 真之^{1,2,a)} 欧 亜非² 高前田 伸也^{1,2,b)}

概要: 汎用的なロボット制御を実現する手法として、Vision-Language-Action (VLA) モデルが注目を集めているが、その膨大な計算量に伴うリアルタイム性の欠如が実用上の大きな課題となっている。本研究では、推論効率の観点から優位性が期待される Diffusion VLA に焦点を当て、その高速化方針を明確にするための性能解析を行った。推論過程を VLM による Prefill フェーズと、小型 LLM による Denoise フェーズに分割し、ルーフライン解析を用いた評価を実施した結果、Prefill は計算バウンドで高いハードウェア利用効率を示す一方、Denoise は理論限界を大きく下回る極めて非効率な実行状態にあることが明らかになった。以上の解析結果を踏まえ、本研究では、Prefill には演算回数の削減を主眼としたアプローチを、Denoise には行列積の回数自体を削減する構造的な効率化を適用すべきであるという、各フェーズの計算特性に準拠した高速化の設計指針を導出した。

1. 序論

近年、多様なタスクや環境に適応可能な汎用ロボット制御の実現に向け、Vision-Language-Action Model (VLA) が大きな注目を集めている。VLA は、広範なインターネットデータで学習された視覚言語モデル (VLM) をベースとして、視覚情報と言語指示から直接ロボットの行動を出力する手法であり、従来のロボット制御手法を凌駕する高い汎化性能を有している [10]。しかし、その膨大なパラメータ数に起因する推論の遅延は、リアルタイム性が厳格に要求されるロボット制御において極めて大きな課題となっている。

現在の VLA は、主に Autoregressive VLA と Diffusion VLA の2つのアプローチに分類される [8]。Autoregressive VLA は、VLM のアーキテクチャをそのまま利用し、行動をトークンとして生成する手法である。既存の語彙のうち使用頻度の低いトークンをロボットアームの角度などの数値に対応付けることで、言語生成と同様の枠組みで行動決定を行う。一方で、Diffusion VLA は拡散モデルを用いて行動を生成する手法であり、VLM の順伝播から得られる KV キャッシュを条件として、小型の LLM を用いて拡散過程を実行する。これら2つを比較すると、Autoregressive

VLA が行動生成のために大型 LLM を 30~60 回程度繰り返して推論する必要があるのに対し、Diffusion VLA は小型 LLM を用いた 10 回程度の拡散ステップで行動を生成できる [8]。推論効率の観点において Diffusion VLA は Autoregressive VLA よりも優位であると考えられるため、本研究では Diffusion VLA に焦点を当てて解析を行う。

本研究では、Diffusion VLA のさらなる高速化指針を明確にすることを目的とし、その詳細な性能解析を行う。解析にあたっては、推論過程を VLM による視覚・言語情報の処理である Prefill フェーズと、拡散過程による行動生成である Denoise フェーズの2段階に分割し、ルーフライン解析を用いて性能のボトルネックを特定した。解析の結果、Prefill フェーズは計算バウンドな特性を示しており、ハードウェアの理論的なピーク性能 (FLOPS) に近い実効性能が得られていることが明らかになった。一方で、Denoise フェーズはメモリ帯域幅および演算性能のいずれの理論限界値も大きく下回る性能しか発揮できていないという、特異な実態が判明した。

さらに、単純な線形層を用いた追加実験により、一回あたりの行列積 (線形変換) のサイズが小さい場合、ハードウェアの利用効率が極端に低下することが示唆された。これは、Diffusion VLA におけるデノイジングプロセスが、個々の演算規模が小さいためにハードウェアの性能を引き出しきれていないことを意味する。以上の解析に基づき、本論文では VLA の高速化に対してフェーズごとに異なるアプローチが必要であることを結論づける。演算回数の削

¹ 東京大学
The University of Tokyo

² 理化学研究所
RIKEN

a) usui-masayuki394@g.ecc.u-tokyo.ac.jp

b) shinya@is.s.u-tokyo.ac.jp

減が求められる Prefill フェーズに対しては、疎性の活用やフレーム間類似性に基づく冗長性の排除が有効である。一方で、Denoise フェーズに対しては、行列積の数そのものを減らすためにレイヤー数や拡散ステップ数の削減といった構造的な効率化が不可欠である。本研究で得られた知見は、モデルの精度を維持しつつ、真に実用的なリアルタイム VLA を構築するための重要な設計指針を与えるものである。

2. 背景と関連研究

2.1 VLA のモデルアーキテクチャと推論過程

図 1 に VLA のモデルアーキテクチャの比較を示す。

図 2 に Autoregressive VLA の推論過程を示す。Autoregressive VLA は VLM と同一のモデルアーキテクチャを持つが、(出力側の) トークナイザーが異なり、言語ではなく行動を出力するようになっている。具体的には、既存の語彙のうち使用頻度の低いトークンをロボットアームの角度などの数値に直接結びつける方式 [7], [10] や、専用の特別なトークン化を利用する方式 [8] が存在する。このうち、FAST [8] で提案されるトークン化手法は圧縮の工程を含むため、RT-2 [10] や OpenVLA [7] における単純なトークン化と比較して、トークン数を削減することが可能である。また、(FAST によるトークン化を採用した) Autoregressive VLA は Diffusion VLA よりも学習が高速であると報告されている [8]。しかし、Autoregressive VLA は一般的な LLM や VLM と同様に Autoregressive Decoding に基づいて行動を生成することから、推論は Diffusion VLA の方が高速であると指摘されている [8]。なお、訓練では Autoregressive VLA を、推論では Diffusion VLA を用いるような方式も提案されている [4]。

図 3 に Diffusion VLA の推論過程を示す。Diffusion VLA では Diffusion Model (拡散モデル) によって行動を生成する。文献によっては Diffusion Model ではなく Flow Matching によって説明されていることもあるが、Diffusion Model と Flow Matching はほとんど等価である [6]。Diffusion VLA における推論では、VLM に相当する部分 (図の左側) の順伝播によって KV キャッシュ (アテンション層の Key と Value) を取得し、それをもとに小型の LLM (図の右側) を用いた (逆) 拡散過程によって出力となる行動を生成する。本論文では、前半のキーとバリューを得る段階を Prefill、後半の行動を生成する段階を Denoise と呼称する。Denoise フェーズにおいて、Prefill フェーズで得られたキーとバリューから、 π_0 [5] では Self-Attention によって情報を引き出す一方で、SmolVLA [9] では Self-Attention と Cross-Attention を組み合わせた方式で情報を引き出すという違いがある。

Autoregressive VLA と Diffusion VLA の推論の効率性について、Autoregressive VLA では Autoregressive Decod-

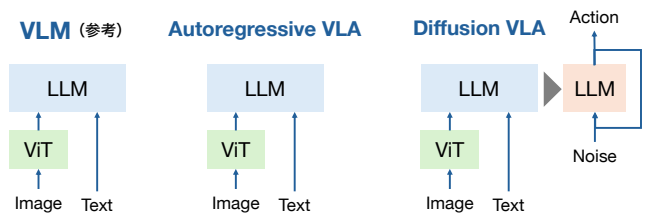


図 1 VLA のモデルアーキテクチャの比較。参考として VLM のモデルアーキテクチャも示している。Autoregressive VLA と VLM は同一のモデルアーキテクチャを持つ (トークナイザーは異なる)。Diffusion VLA の図解に 2 つの LLM が登場するが、それらはそれぞれ別の重みを有している (青色で示される LLM と橙色で示される LLM は別物ということ)。

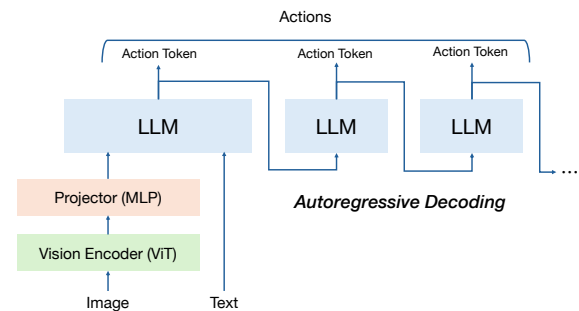


図 2 Autoregressive VLA の推論過程の図示。過去の出力トークンを入力とする Autoregressive Decoding (自己回帰デコーディング) によって行動を生成していることから、Autoregressive VLA という名称になっている。

ing による行動生成のために大型の LLM (数十億パラメータ) を 30~60 回順伝播しなければならない一方、Diffusion VLA は Diffusion Model による行動生成のために小型の LLM (数億パラメータ) を 10 回程度順伝播すればよいことから、Autoregressive VLA よりも Diffusion VLA の方が推論効率に優れる [8]。本質的には、Autoregressive VLA は単一の大きな LLM を利用している一方、Diffusion VLA は (Prefill フェーズで用いられる) 大きな LLM と (Denoise フェーズで用いられる) 小さな LLM を使い分けていることが要因となっている。加えて、FAST [8] による効率的な行動トークン化を採用したとしても、Autoregressive VLA の行動トークン数は Diffusion VLA のサンプリングステップ数を上回ってしまうことも要因である。それゆえ、本研究では、より推論効率に優れる Diffusion VLA の方を性能解析の対象とした。具体的には π_0 [5] をベースとしている。

3. 実験と議論

3.1 実行環境

NVIDIA A100 40GB PCIe を搭載した研究室のサーバ上で全てのプログラムを実行した。NVIDIA A100 40GB PCIe の仕様上、演算性能は bfloat16 で 312 TFLOPS, HBM の帯域幅は 1555 GB/s であり、Roofline Model の Ridge Point は 200 程度となる [1]。したがって、Arithmetic In-

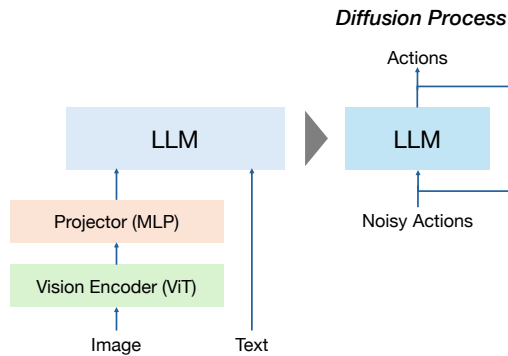


図 3 Diffusion VLA の推論過程の図示。Diffusion Model（拡散モデル）によって行動を生成していることから、Diffusion VLA という名称になっている。ここで、左右の二つの LLM は異なる重みを持つ別のモデルであって、前半のモデルから後半のモデルへ KV キャッシュ（アテンション層の Key と Value）を渡すことで、情報を伝達している。

tensity が 200 以下ならば Memory Bound であり、200 以上ならば Compute Bound であるだろうということが大雑把には言える。

3.2 実装

openpi [2] と LeRobot [3] のコードベースをもとにして π_0 [5] のモデルを構築した。元のコードには (KV キャッシュの無駄なコピーなど) 冗長な部分が存在しているため、より簡潔な実装に置き換えた。ただし、モデルの本質的な計算は変更していない。PyTorch で構築されたモデルであり、torch.compile によって高速化されている。torch.compile のオプションには fullgraph=True および mode="reduce-overhead" を指定し、CUDA Graphs を利用して CUDA カーネル起動時のオーバーヘッドを削減するようにしている。

3.3 実行時間の計測方法

実行時間の計測対象となるコードブロックを、ウォームアップとして 10 回実行してから、計測用に 100 回実行し、その実行時間を 100 で割った値を最終的な実行時間とする。このとき、ウォームアップを必ず除かないと、torch.compile による JIT コンパイル時間が入ってきて、計測結果が全くとおかしくなる。また、torch.cuda.synchronize による CPU と GPU の間の同期も必要である。さらに、torch.compile を利用している場合は、デッドコード削除に近い現象が発生し、モデルの出力を外部で全く利用していないと、CUDA カーネルの実行がスキップされる。そのため、モデルの出力の和を取って表示させるようなコードを追加している。

3.4 Diffusion VLA の実行時間の分析

推論過程全体の実行時間は 50 ms であった。ViT の実行時間（図 1 の ViT に対応する部分の実行時間）は 10 ms、

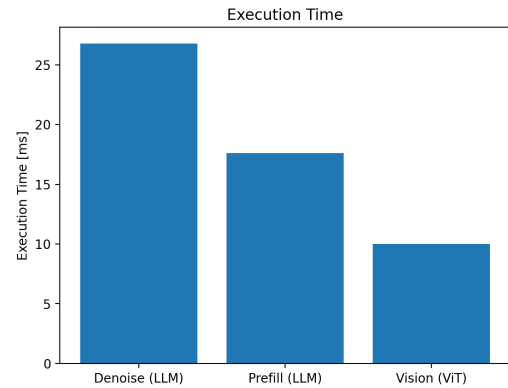


図 4 各段階の実行時間の比較

Prefill のための LLM の推論時間は 17.6 ms、Denoise のための LLM の推論時間は順伝播 1 回あたり 2.78 ms であった（Denoise の順伝播は 10 回繰り返すことに注意）。各段階の実行時間を足し合わせると全体の実行時間を若干上回ってしまうが、これは小さく分割することで CUDA Graphs による最適化が効きにくくなってしまいうことと、計測自体にオーバーヘッドが存在することが原因と考えられる。

図 4 に各段階の実行時間の比較を示す。Denoise フェーズの LLM の推論時間が最も大きな割合を占めている。これは単純にモデルの順伝播を 10 回も繰り返しているためと考えられる。Prefill フェーズの LLM の推論時間は次に大きな割合を占めている。これは Prefill フェーズの LLM は Denoise フェーズの LLM や ViT と比較してパラメータ数が大きいためと考えられる。

図 5 にルールラインモデルによる各段階の比較を示す。ここで、ピーク性能とピーク帯域幅については節 3.1 で言及したスペックを採用し、演算回数 (FLOPs) とデータ転送量 (Bytes) は理論的に算出している。例えば、形状が (seq_len, hidden_size) のテンソルを入力とし、形状が (seq_len, intermediate_size) のテンソルを出力とする線形層の場合、bfloat16 を想定すると、演算回数は $2 * \text{seq_len} * \text{hidden_size} * \text{intermediate_size}$ 、データ転送量は $2 * \text{hidden_size} * \text{intermediate_size}$ となる。このような計算を全ての層に対して実行した上でそれらを足し合わせると、全体の演算回数 (FLOPs) とデータ転送量 (Bytes) を求めることができる。なお、実行時間は実測値である。図 5 を参照すると、Prefill フェーズは演算性能 (FLOPS) による理論限界に近い性能が出ている。図では少し見にくいだが、190 TFLOPS とスペック値 (312 TFLOPS) の 6 割程度の性能が出ている。一方で、Denoise フェーズは演算性能 (FLOPS) による理論限界とメモリ帯域幅 (Bytes/s) による理論限界の両方を大きく下回る性能しか出していない。この現象はルールラインモデルの枠組みでは説明できないため、追加の実験を行った。それについて次の節で説明する。

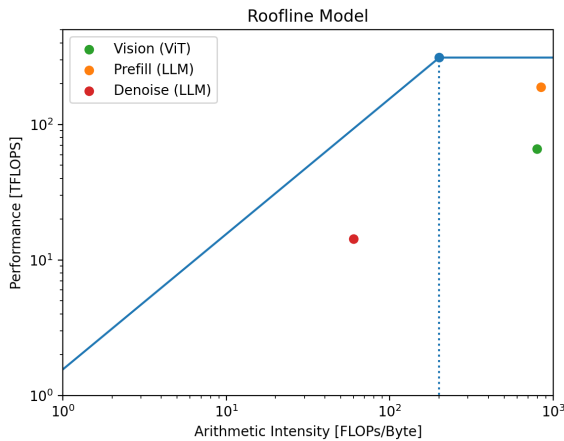


図 5 ルーフラインモデルによる各段階の比較

3.5 単純な線形層による追加実験

形状が $(\text{sequence_length}, \text{hidden_size})$ のテンソルを入力とし、形状が $(\text{sequence_length}, \text{hidden_size})$ のテンソルを出力とする線形層の場合、形状が $(\text{sequence_length}, \text{hidden_size})$ のテンソル（行列）と形状が $(\text{hidden_size}, \text{hidden_size})$ のテンソル（行列）を掛けることになる。このような設定において sequence_length と hidden_size を変化させながら、実行時間を測定した。このとき、bfloat16を想定すると、演算回数は $2 * \text{sequence_length} * \text{hidden_size}^{**2}$ 、データ転送量は $2 * \text{hidden_size}^{**2}$ となり、Arithmetic Intensity は sequence_length に一致する。

図 6 に直接の演算性能（FLOPS）を、図 7 にルーフラインモデルによる理論限界で正規化した相対値を示す。 hidden_size が大きい場合にはルーフラインモデルによる理論限界に近い性能が出ている一方で、 hidden_size が小さい場合にはルーフラインモデルによる理論限界を大きく下回る性能しか出していないことがわかる。

Prefill フェーズの LLM におけるハイパーパラメータは、 $\text{hidden_size}=2048$ および $\text{intermediate_size}=16384$ となっていて、Denoise フェーズの LLM のハイパーパラメータは、 $\text{hidden_size}=1024$ および $\text{intermediate_size}=4096$ となっている。LLM の内部の MLP の内部の線形層のサイズは $(\text{hidden_size}, \text{intermediate_size})$ である。演算回数とデータ転送量が等しくなるように換算した場合、Prefill フェーズは図 6 と 7 の $\text{hidden_size}=4096$ （赤）と $\text{hidden_size}=8192$ （紫）の間に、Denoise フェーズは図 6 と 7 の $\text{hidden_size}=2048$ （緑）に対応する。このことを念頭に図 6 と 7 を見直してみると、節 3.4 において、Prefill フェーズが理論限界に近い性能が出ていた一方で、Denoise フェーズが理論限界を大きく下回る性能しか出なかった原因は、Prefill フェーズは各層のサイズ（本節の実験では hidden_size ）が大きく、Denoise フェーズは各層のサイズ（本節の実験では hidden_size ）が小さいため

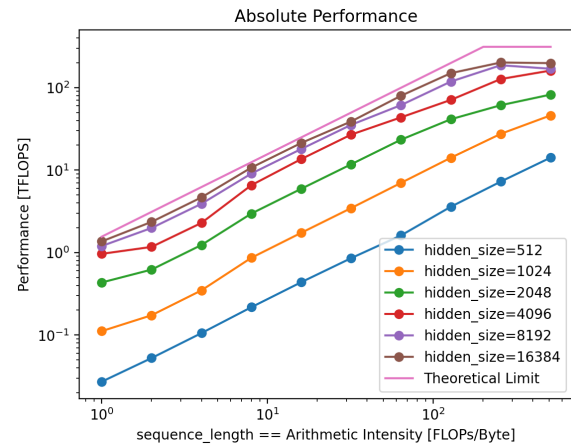


図 6 単純な線形層（行列積）の性能. 演算性能（FLOPS）を直接プロットしている。

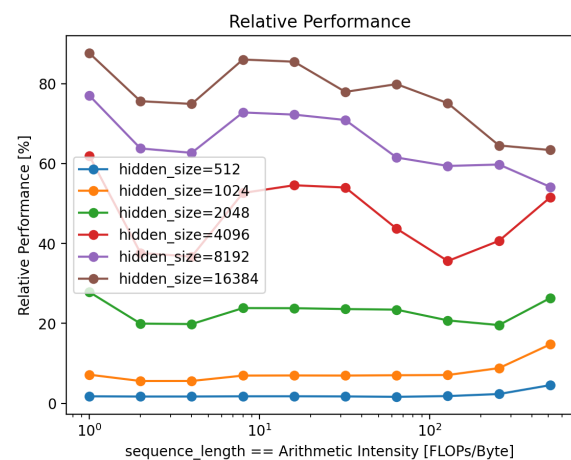


図 7 単純な線形層（行列積）の性能. ルーフラインモデルによる理論限界で正規化した（割った）相対的な演算性能をパーセント表示で示している。

ある考えられる。実際、モデルのハイパーパラメータの hidden_size と intermediate_size の積を計算すると、Prefill が 2^{25} で Denoise が 2^{22} と 8 倍もの差がある。

4. 結論と今後の展望

Diffusion VLA に分類される π_0 [5] の性能解析を行い、推論過程の段階によって異なる性能特性を持つことを明らかにした。特に、Prefill フェーズは理論限界に近い性能が出ている一方で、Denoise フェーズは理論限界を大きく下回る性能しか出していないことが明らかになった。なぜ Denoise フェーズにおいて理論限界を大きく下回る性能しか出していないのかを明らかにするために、単純な線形層による追加実験を行い、各層のサイズが小さい場合に計算効率が低下することを明らかにした。Prefill フェーズは各層のサイズが大きい一方、Denoise フェーズは各層のサイズが小さいために、Denoise フェーズでは Prefill フェーズと比べて計算効率が低下しているのである。この発見をもとにすると、Prefill フェーズでは演算回数（FLOPS）を削減

するような手法, 例えば疎性やフレーム間の冗長性に基づく計算回数の削減, が高速化に有効である一方, Denoise フェーズではレイヤー数やサンプリングステップ数を削減するような手法, 例えば Layer Skipping 等, が有効であるという指針が得られる. 各段階の計算特性に応じた高速化の方法が求められることを本研究は示唆している.

Denoise フェーズにおいて各層のサイズが小さいことにより計算効率が低下していることまでは本研究で明らかになったが, ではなぜ各層のサイズが小さい場合に計算効率が低下するのかを今後の研究で明らかにしたい. GPU 内部の演算器が埋まりにくいことや, 遅延隠蔽 (Latency Hiding) が効きにくいことなどが理由として挙げられるが, 実際にプロファイラなどを用いて調査したい. そこから発展して, サイズが小さい場合でも高速に処理できるカーネルや (コンピュータ) アーキテクチャを提案することができれば最善である. また, 技術的には, 図 5 では Prefill フェーズが理論限界の 15 パーセント程度の性能しか出しておらず, 図 6 と図 7 では緑線でも 20 パーセント程度の性能が出ていて, この 15 パーセントと 20 パーセントの差異がどこから来ているのかも明らかにしたい. 恐らくは Layer Normalization 等のオーバーヘッドが大きいのではないかと考えている.

謝辞 本研究の一部は JST CREST JPMJCR21D2 の支援による.

参考文献

- [1] <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf>.
- [2] <https://github.com/Physical-Intelligence/openpi>.
- [3] <https://github.com/huggingface/lerobot>.
- [4] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025.
- [5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π : A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025.
- [6] Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin Patrick Murphy, and Tim Salimans. Diffusion models and gaussian flow matching: Two sides of the same coin. In *The Fourth Blogpost Track at ICLR 2025*, 2025.
- [7] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025.
- [8] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025.
- [9] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025.
- [10] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huang Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023.