StaMo: Unsupervised Learning of Generalizable Robot Motion from Compact State Representation

Supplementary Material

1. Implementation Details

1.1. Implementation Detail of StaMo

Our StaMo encoder experiments were conducted on Ubuntu 22.04 LTS with PyTorch 2.8, using publicly available pre-trained weights such as DinoV2 and Stable Diffusion 3 for model initialization. Unless otherwise noted, all main experiments were carried out on standard public datasets under a consistent hardware and software setup. Training was performed with the AdamW optimizer, an initial learning rate of 3e-5 with a cosine decay scheduler, a global batch size of 128, and a weight decay of 1e-3. For evaluation, PSNR and SSIM were adopted as the primary metrics to assess the quality of image reconstruction. All experiments were run on 8 NVIDIA H100 GPUs, with a typical training duration of around ten days, and we fixed the random seed to 33 to ensure reproducibility.

Unless otherwise specified, all main experiments are conducted using data from publicly available datasets, LIBERO and DROID. We strictly adhere to their original release protocols and only perform essential preprocessing steps, without involving any private or sensitive information.

1.2. Implementation Detail of RDT Policy

For diffusion-based policy co-training, we utilize a more scalable DiT block [8] as the backbone, building upon the work of RDT-1B [4]. Extracted language and visual features are incorporated as conditioning inputs via cross-attention layers within the DiT block.To enable the joint learning of action-less video data and robot data within a unified policy model, we construct two sets of MLP networks. These networks map continuous latent motion and robot actions into a shared embedding space, and subsequently map them back to their respective original spaces.During the training phase, we adopt the DDPM scheduler over 1000 diffusion steps, employing a glide cosine scheduling scheme (specifically, the $squaredcoscap$ v2 variant). For inference, we leverage the DPM-Solver++ [6] with an analogous glide cosine scheduler, but with a significantly reduced sampling budget of only 5 steps. Finally, to capture the temporal dependencies of actions and ensure real-time dynamic adaptability during policy execution, we set an action/motion chunk size of 8 for both training and inference.

The diffusion-based policy training was configured using the AdamW optimizer [5] with a base learning rate of 5e-4 and an effective batch size of 256, running for a total of 100 training epochs.For the model's architecture, visual features are extracted using DINOv2 [7] with a ViT-B [2] backbone, while language features are extracted via BERT [3]. The core policy model is built with 12 layers, 16 Multi-Head Self-Attention (MHSA) heads, and a hidden dimension of 768. An action/motion chunk size of 8 is used. The network mappings include an Action projector, a Latent motion projector, an Action head, and a Latent motion head.

2. Effect of Incorporating the DiT Prior

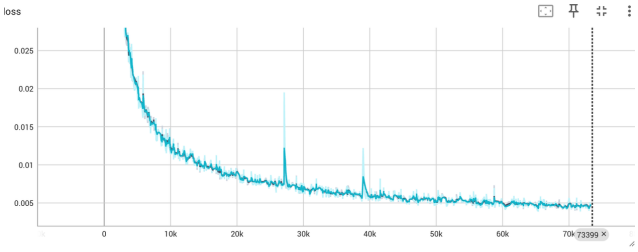

Figure 1. Training loss curve of our model **with** the DiT prior

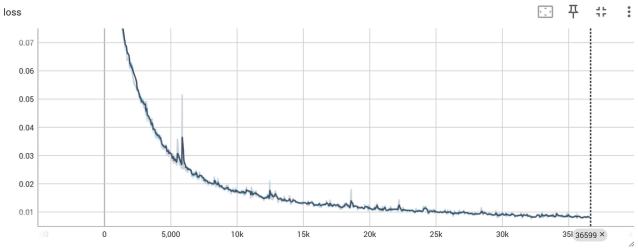

Figure 2. Training loss curve of our model **without** the DiT prior

To further investigate the impact of architectural priors on training dynamics, we conducted an ablation study evaluating whether incorporating a DiT prior can accelerate convergence during training. As shown in Fig 1 2, introducing the DiT prior consistently improves optimization efficiency: the model achieves a significantly lower training loss within the same number of steps, and the convergence curve becomes noticeably smoother and steeper in the early training phase.

Under identical data, optimizer, and batch-size settings, training with the DiT prior converges to a comparable or

better loss value using only 200 epochs, whereas the baseline without the DiT prior requires substantially more steps to reach a similar level. This demonstrates that the structural prior provided by DiT facilitates more stable gradient flow and more effective representation learning, ultimately yielding faster and more reliable convergence.

3. Action Linear Probing Details

Our linear probing dataset is constructed by sampling from the LIBERO dataset. We performed a randomized sampling process across all available trajectories to create a dataset of tuples $(\mathbf{I}_n, \mathbf{I}_{n+k}, \mathbf{A}_n)$, as described in the main paper. A total of approximately 20000 samples are collected, which are then partitioned into training and validation sets with a 95%/5% split, respectively. This process is repeated independently for each of our four evaluated action horizons, where $k \in \{1, 2, 4, 8\}$. To ensure a fair comparison, the same random seed are used for data sampling and splitting. Notably, to accommodate the LAPA baseline which requires task description input, each sampled tuple also includes the corresponding language instruction. For the $k = 1$ comparison, the exact same initial frames $(\mathbf{I}_n)$ are maintained through using same random seed across all methods.

Following the standard linear probing protocol, we use a simple Multi-Layer Perceptron (MLP) as the probe to decode the action representations. The MLP consists of a single hidden layer with 128 units and a ReLU activation function. The architecture is intentionally kept minimal to ensure that the measured performance primarily reflects the quality of the input representation rather than the capacity of the probe itself. During training, the upstream encoders (i.e., *StaMo* and DINOv2) are kept frozen, and gradients are only computed and backpropagated for the MLP's parameters.

To visually substantiate the rich motion information encoded within our latent actions, we conduct a qualitative replay experiment in the LIBERO simulation environment. For each of the ten tasks in the LIBERO-10 benchmark, we utilize all the ground-truth initial frames $(\mathbf{I}_n)$ and final frames $(\mathbf{I}_{n+k})$ from the demonstrations. These two frames are encoded into a single latent actions (Δz) using our frozen *StaMo* encoder. Here we use $k = 8$ in our experiment.

The latent actions are then fed into the corresponding pretrained MLP probe to predict the entire sequence of k actions. To lower the accumulation of small prediction errors, we only execute the first action in every sequences. The results, visualized in Figure 3, show that even the actions decoded by this simple MLP are sufficient to successfully execute the corresponding tasks to some degree. We also observe that the primary failure mode occurred in tasks requiring precise grasping, where the predicted actions sometimes cause the gripper to descend to an insufficient depth.

4. Real World Setting

Our robotic setup includes a single Franka Research3 arm equipped with a UMI [1] grippper. A third-person-view RealSense D435 camera is mounted in a fixed position to capture environmental observations at a resolution of 640×480 pixels. Following the implementation in publicly available code [1], we use a 3D mouse for teleoperated data collection. Our system operates at 20 Hz, which is a moderate reduction from the native 100 Hz control frequency to balance training efficiency and motion continuity. Actions are defined as absolute end effector poses in SE(3), using 3D position and quaternion orientation. Our task definition are list below:

1) Pick up the [TOY NAME]: The robot is required to pick up the specific toy, including the toy snake, the toy eggplant, and the toy chicken, resulting in a total of three tasks (with no distractions).

2) Put the toy into the basket: The robot need to pick up the toy(no other distractions) and put it into the basket.

3) Open the drawer: The robot is required to approach the drawer and pull it open.

4) Put all cups into the basket: We place two cups with different colors on the table and set a few other things(toys) as distractions. The robot need to sequentially put two cups into the basket.

5) Put the toy into the drawer: We place a toy on the table and set a few other things as distractions. The robot need to sequentially open the drawer, pick up the specific toy and put it into the drawer, and close the drawer.

6) Stack all cups: Three cups of varying sizes are placed on a tabletop. The robot's task is to perceive their relative sizes and then stack them in descending order (from largest to smallest).drawer, and close the drawer.

5. More Visualization Results

5.1. Simulation Results

5.1.1. Visual Reconstruction Results

In Figure 4, we demostrated more visual reconstruction results from the same episode. The first and last frames are reconstructed from ground-truth images using the *StaMo* encoder, and the intermediate frames are obtained by linearly interpolating between the latent state tokens of these endpoints. The resulting transitions illustrate smooth and continuous motion of both the robotic arm and the manipulated objects, demonstrating that the latent space captures meaningful and coherent dynamics.

[1] https://github.com/UT-Austin-RPL/deoxys_control

5.1.2. Motion Transfer Results

As illustrated in Fig. 5, we take latent tokens from three different task scenarios and form new representations using the linear operation tokens(3) + tokens(2) – tokens(1). The reconstructed images derived from these synthesized tokens reveal that the encoder's latent space behaves approximately linearly: semantic attributes relevant to task configuration can be added, removed, and recombined in a meaningful way. Notably, this linearity holds across two distinct tasks, suggesting that the learned representation is not only smooth but also sufficiently structured to enable reliable transfer between domains. Such behavior provides strong evidence that the encoder captures disentangled and composable features that generalize beyond individual demonstrations.

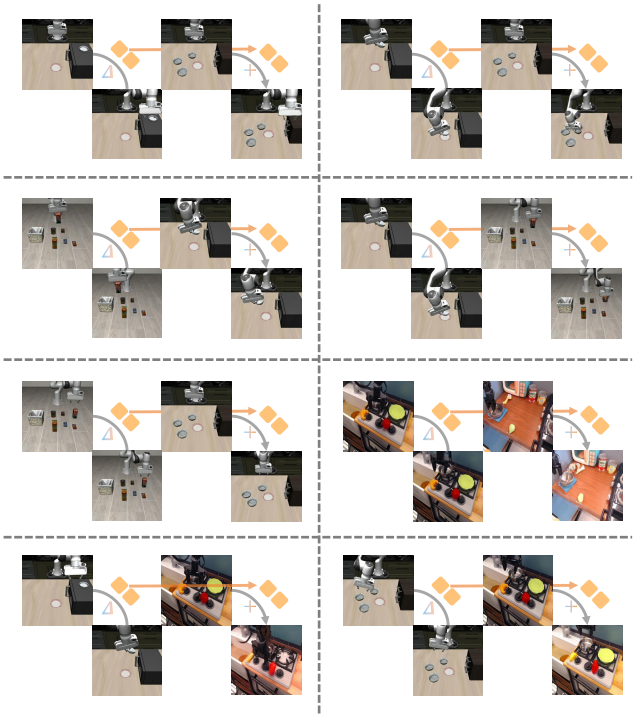

Figure 5. **Transfer linear interpolation experiment with the *StaMo* encoder.** The left and right panels illustrate different task scenarios, where reconstructions are obtained by tokens(3) + tokens(2) – tokens(1), demonstrating the linear interpolation property of latent representations during transfer.

5.2. Real World Results

We show in Figure 6 additional demonstrations of the OpenVLA model equipped with the *StaMo* world-modeling capability performing manipulation in the real world. With the integration of *StaMo*, our system exhibits notably stronger manipulation performance while introducing only minimal overhead to inference speed.

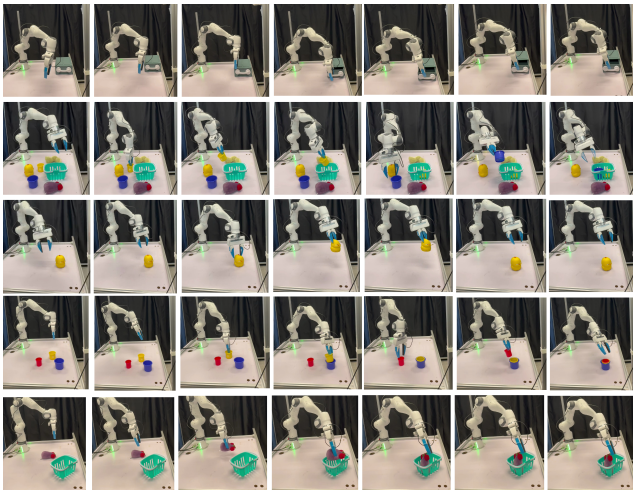

Figure 6. **Real world experiment results.**

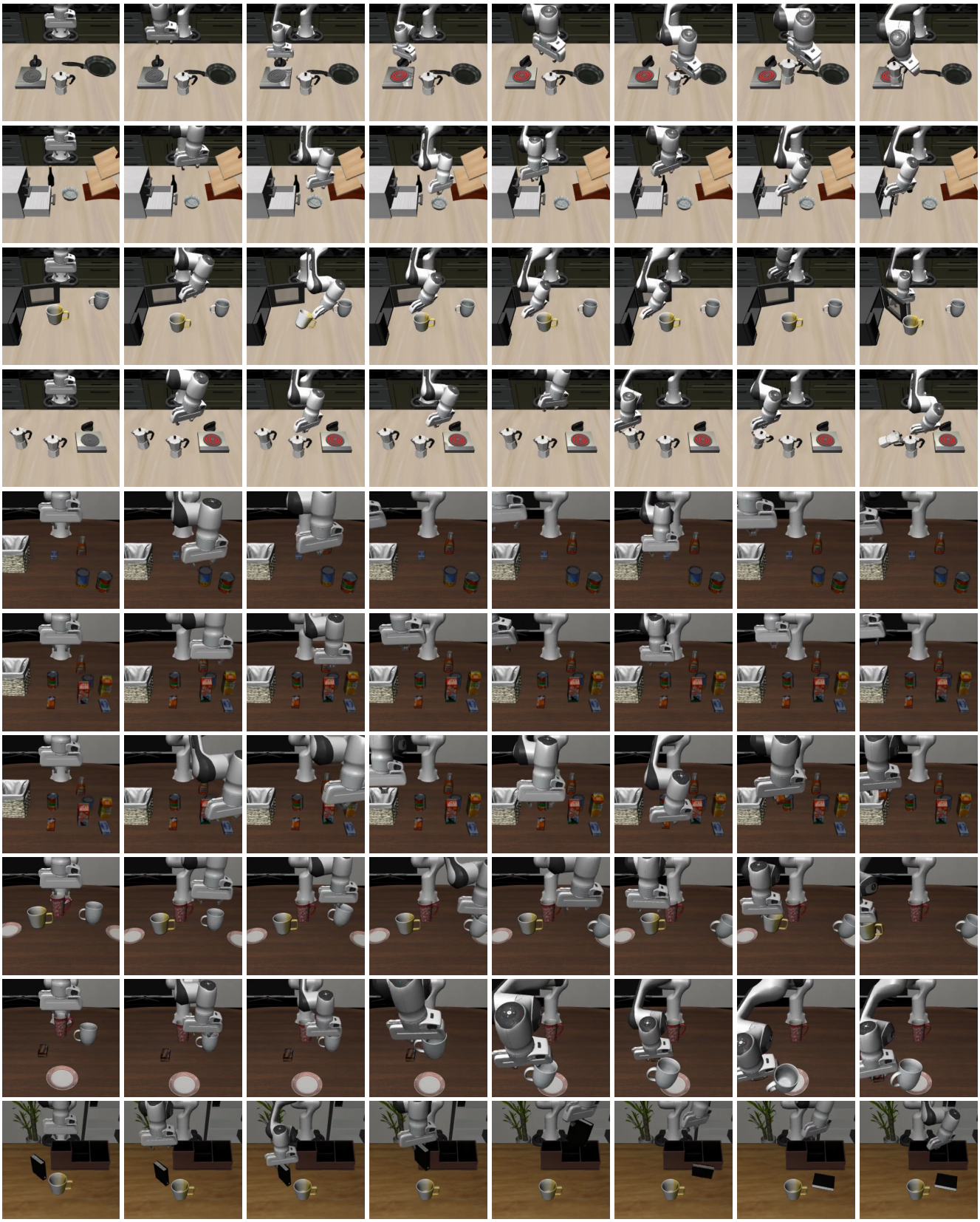

Figure 3. **MLP-predicted actions replay.** Actions are predicted by an MLP probe from our *StaMo* latent action representation and replayed in LIBERO-10 tasks. The results show that our latent actions are remarkably effective, encoding functionally coherent and executable motions.

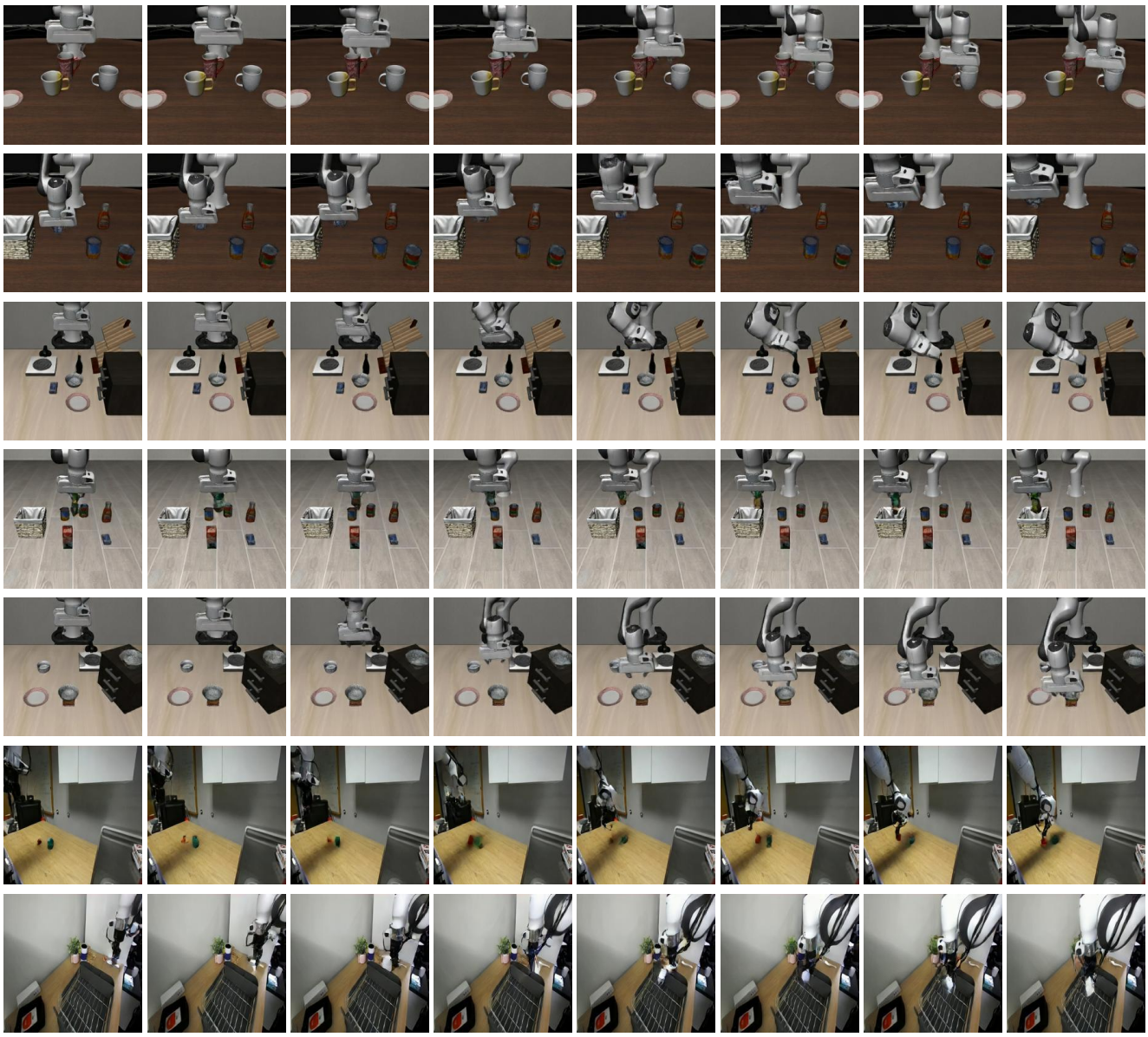

Figure 4. **Visual reconstruction results from the same episode.** The first and last frames are reconstructed from ground-truth images using the *StaMo* encoder, while the intermediate frames are generated by linearly interpolating between the latent state tokens of the two endpoints. The transitions show that both the robotic arm and the objects move in a continuous and smooth manner.

References

[1] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024. 2

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. 1

[3] Mikhail V Koroteev. Bert: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*, 2021. 1

[4] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024. 1

[5] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2017. 1

[6] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, pages 1–22, 2025. 1

[7] 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

[8] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 1