# $\pi_0$: A Vision-Language-Action Flow Model for General Robot Control

## Physical Intelligence

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong,
Anna Walling, Haohuan Wang, Ury Zhilinsky
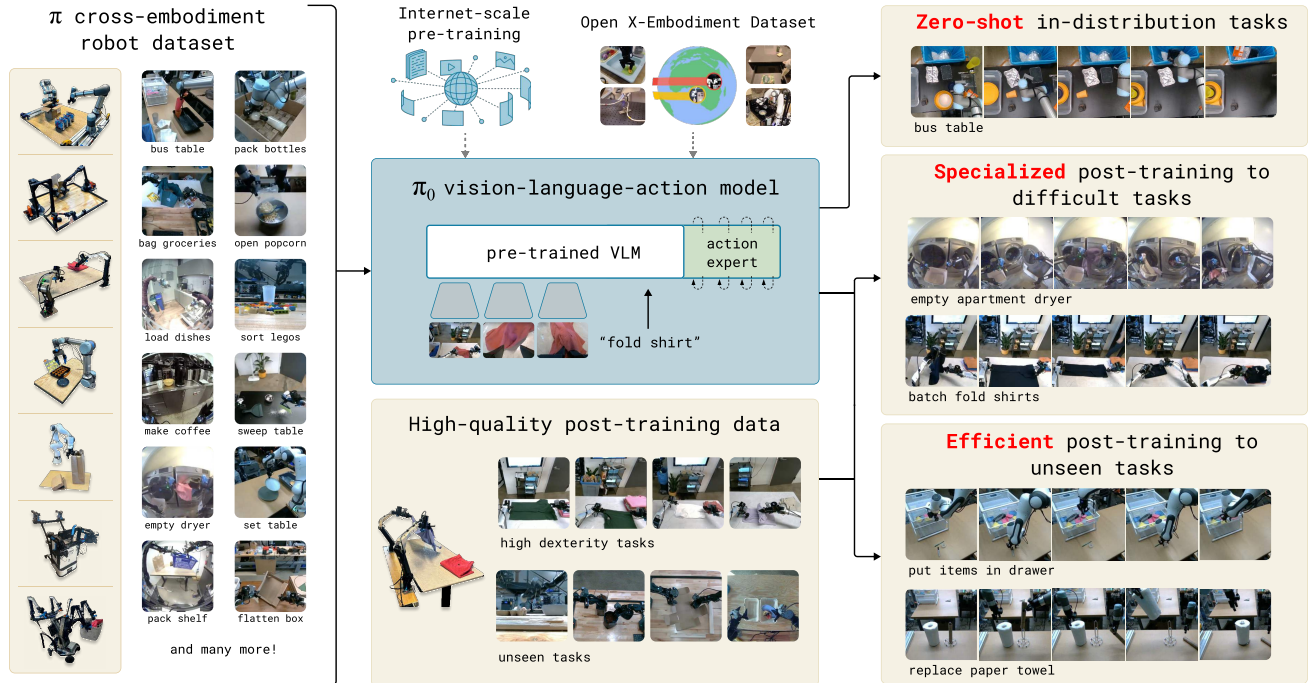https://physicalintelligence.company/blog/pi0

Fig. 1: Our generalist robot policy uses a pre-trained vision-language model (VLM) backbone, as well as a diverse cross-embodiment dataset with a variety of dexterous manipulation tasks. The model is adapted to robot control by adding a separate *action expert* that produces continuous actions via flow matching, enabling precise and fluent manipulation skills. The model can then be used directly to perform tasks based on a prompt, or fine-tuned on high-quality data to enable complex multi-stage tasks, such as folding multiple articles of laundry or assembling a box.

*Abstract*—Robot learning holds tremendous promise to unlock the full potential of flexible, general, and dexterous robot systems, as well as to address some of the deepest questions in artificial intelligence. However, bringing robot learning to the level of generality required for effective real-world systems faces major obstacles in terms of data, generalization, and robustness. In this paper, we discuss how generalist robot policies (i.e., robot foundation models) can address these challenges, and how we can design effective generalist robot policies for complex and highly dexterous tasks. We propose a novel flow matching architecture built on top of a pre-trained vision-language model (VLM) to inherit Internet-scale semantic knowledge. We then discuss how this model can be trained on a large and diverse dataset from multiple dexterous robot platforms, including single-arm robots, dual-arm robots, and mobile manipulators. We evaluate our model in terms of its ability to perform tasks via direct prompting, follow language instructions from people and from a high-level VLM policy, and its ability to acquire new skills via fine-tuning. Our results cover a wide variety of tasks, such as laundry folding, table cleaning, and assembling boxes.

Physical Intelligence, San Francisco, California, USA. Correspondance to:
research@physicalintelligence.company

Fig. 2: $\pi_0$ **controls a mobile manipulator to fold laundry.** Our model is pre-trained on diverse data from 7 distinct robot configurations and 68 tasks, and can then either be prompted directly or fine-tuned to complex downstream tasks, as in the case of this laundry folding policy, which fetches laundry from a dryer, packs it into a hamper, brings the hamper to a folding table, and then folds each article of clothing.

## I. INTRODUCTION

*A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects.*

Robert A. Heinlein, *Time Enough for Love*

Artificial intelligence systems come in all shapes and sizes, from highly specialized systems that solve complex problems inaccessible to the human mind, such as predicting the conformation of a protein [21], to systems that can produce lifelike high-resolution images or videos based on textual prompts [40]. However, the axis along which human intelligence most outpaces machine intelligence is *versatility*: the ability to solve diverse tasks situated in varied physical environments, while responding intelligently to environmental constraints, language commands, and unexpected perturbations. Perhaps the most tangible progress toward this kind of versatility in AI can be seen in large language- and vision-language models [1, 48]: systems that are pre-trained on large and very diverse corpora of images and text from the web, and then fine-tuned ("aligned") using more carefully curated datasets meant to induce the desired pattern of behavior and responsiveness. While such models have been shown to exhibit broad instruction-following and problem-solving abilities [53, 27], they are not truly *situated* in a physical world the way that people are, and their understanding of physical interaction is based entirely on abstract descriptions. If such methods are to make tangible progress toward AI systems that exhibit the kind of physically situated versatility that people possess, we will need to train them on physically situated data — that is, data from embodied robot agents.

Flexible and general-purpose models that can be tasked to perform a variety of robot behaviors have tremendous practical ramifications, but they may also offer solutions to some of the toughest challenges facing robot learning today, such as availability of data, generalization, and robustness. In natural language [1] and computer vision [39], general-purpose foundation models that are pre-trained on diverse multi-task data tend to outperform narrowly tailored and specialized solutions. For example, if the goal is to recognize birds in photographs, it is likely more expedient to pre-train on many different image-language associations and then fine-tune or prompt for the bird recognition task, than it is to train on only bird recognition data. Similarly, we may find that for effective specialized robot systems, it is more effective to first pre-train on highly diverse robot data, and then fine-tune or prompt for the desired task. This can resolve the data scarcity challenge, because many more sources of data are available to a generalist model — including data from other tasks, other robots, or even non-robot sources — and it may resolve robustness and generalization challenges, because the diverse data exhibits a greater coverage of observations and actions, providing a variety of scenes, corrections, and recovery behaviors that might not be present in more narrow specialized data. Thus, adopting a large-scale pre-training approach to robot learning has the potential to address many of the field's challenges and make practical learning-enabled robots a reality, while at the same time furthering our understanding of the deepest problems in artificial intelligence.

However, developing such generalist robot policies — i.e., robot foundation models — involves a number of major challenges. First, any such research must be done at a very large scale, because the full benefits of large-scale pre-training are often not present at smaller scales [54]. Second, it requires developing the right model architectures that can effectively make use of diverse data sources, while at the same time being able to represent the intricate and subtle behaviors necessary to interact with complex physical scenes. Third, it requires the right training *recipe*. This is perhaps the most important ingredient, as much of the recent progress with large models in NLP and computer vision has relied heavily on delicate strategies for curating pre-training and post-training data [35].

In this paper, we present a prototype model and learning framework, which we call $\pi_0$, that illustrates how each of these three bottlenecks could be tackled. We illustrate our model and system in Figure 1. To incorporate diverse data sources, we begin by utilizing a pre-trained vision-language model (VLM) to import Internet-scale experience. By basing our model on a VLM, we inherit the general knowledge, semantic reasoning, and problem-solving abilities of language- and vision-language models. We then further train our model to incorporate robot actions, turning it into a vision-language-

action (VLA) model [7]. In order to make it feasible to utilize a variety of diverse robot data sources, we employ *cross-embodiment training* [10], where data from many robot types is combined into the same model. These different robot types have different configuration spaces and action representations, including single and dual-arm systems, as well as mobile manipulators. Additionally, in order to make it possible to perform highly dexterous and intricate physical tasks, we use an action chunking architecture [57] with flow matching (a variant of diffusion) to represent complex continuous action distributions [28, 32]. This enables our model to control robots at frequencies of up to 50 Hz for dexterous tasks such as laundry folding (see Figure 1). To combine flow matching with VLMs, we use a novel *action expert* that augments the standard VLM with flow-based outputs.

As with language models, the architecture of our model is only part of our method. In order to flexibly and robustly perform complex tasks, we need the right training recipe. Our recipe mirrors the pre-training/post-training separation commonly seen in exascale language- and image-language models [1, 48], where the model is first pre-trained on a very large and diverse corpus, and then fine-tuned on more narrow and more carefully curated data to induce the desired pattern of behavior — in our case, dexterity, efficiency, and robustness. Intuitively, training only on high-quality data does not teach the model how to recover from mistakes, since mistakes are rarely seen in such data. Training on only lower-quality pre-training data does not teach the model to act efficiently and robustly. Combining both provides the desired behavior: the model attempts insofar as possible to act in a manner similar to the high-quality data, but still has a repertoire of recoveries and corrections that it can deploy in the case of a mistake.

The contributions of our work consist of a novel generalist robot policy architecture based on VLM pre-training and flow matching, and an empirical investigation of pre-training/post-training recipes for such robot foundation models. We evaluate our model out of the box with language commands, with fine-tuning to downstream tasks, and in combination with a high-level semantic policy that outputs intermediate language commands to perform complex and temporally extended tasks. While our model and system make use of a variety of ideas presented in recent work, the combination of ingredients is novel, and the empirical evaluation demonstrates a level of dexterity and generality that goes significantly beyond previously demonstrated robot foundation models. We evaluate our approach by pre-training on over 10,000 hours of robot data, and fine-tuning to a variety of dexterous tasks, including laundry folding (see Figure 2), clearing a table, putting dishes in a microwave, stacking eggs into a carton, assembling a box, and bagging groceries.

## II. RELATED WORK

Our work builds on recently proposed methods in large-scale robot learning, as well as multimodal language models. Our work is most closely related to recently proposed vision-language action (VLA) models, which use pre-trained VLMs that are fine-tuned for robot control [7, 24, 55]. Such models employ autoregressive discretization to represent actions in a manner analogous to text tokens. In contrast, our model employs a novel design that fine-tunes a VLM to produce actions via flow matching [32, 28], a variant of diffusion [20, 46]. This allows us to handle high-frequency action chunks [57] (up to 50 Hz) and highly dexterous tasks, which we show pose a major challenge for prior autoregressive VLAs [7]. This resembles a number of recent works on diffusion models for action generation [9, 60]. In contrast to these works, our model uses a pre-trained VLM backbone [5]. Our contribution is also fundamentally integrative, focusing on a *framework* for robot foundation models, including not only the model architecture itself but also a pre-training recipe, pre-training and post-training phases, and a range of real-world experiments.

Outside of robot control, many models have been proposed that combine pre-trained language models with diffusion [40, 41, 14], including models that specifically hybridize diffusion and autoregressive large language models [19, 29, 59]. Such models are typically concerned with image generation, but our action generation model builds on a number of previously proposed concepts. Like Zhou et al. [59], we train our model via a diffusion-style (flow matching) loss applied on individual sequence elements, in lieu of the standard cross-entropy loss for decoder-only transformers. Like Liu et al. [29], we use a separate set of weights for the tokens corresponding to diffusion. Incorporating these concepts into a VLA model, we introduce what to our knowledge is the first flow matching VLA that produces high-frequency action chunks for dexterous control.

Our work also builds on a rich history of prior works on large-scale robot learning. Early work in this area often utilized self-supervised or autonomous data collection [26, 22, 8], providing a tractable data source for simple tasks such as grasping [18, 37] or pushing [56], but without the complexity of more dexterous behaviors. More recently, a number of high-quality datasets have been collected for robot control that enable broad generalization [23, 10, 52, 33, 34, 43, 13, 6], but typically for simpler tasks that consist of object relocation and rudimentary furniture manipulation (e.g., drawer opening) [31, 15]. More dexterous tasks have been studied at a smaller scale, typically with 10s or 100s of training trajectories [57], equivalent to 10 or less hours. Since one of our aims is to study complex and dexterous behaviors, we utilize a much larger dataset, with about 10,000 hours of demonstrations, complemented by the open-source OXE dataset [10]. To our knowledge, this represents by far the largest robot learning experiment in terms of the amount of robot data. At this scale, we show that a more sophisticated pre-training/post-training recipe is highly effective — analogously to the recipes used for large language models, a pre-training phase endows our model with a broad base of knowledge, which is then refined in a post-training phase with higher-quality curated data to achieve the desired behavior.

The complexity of the tasks we illustrate goes significantly beyond prior work. While recent work has illustrated a number
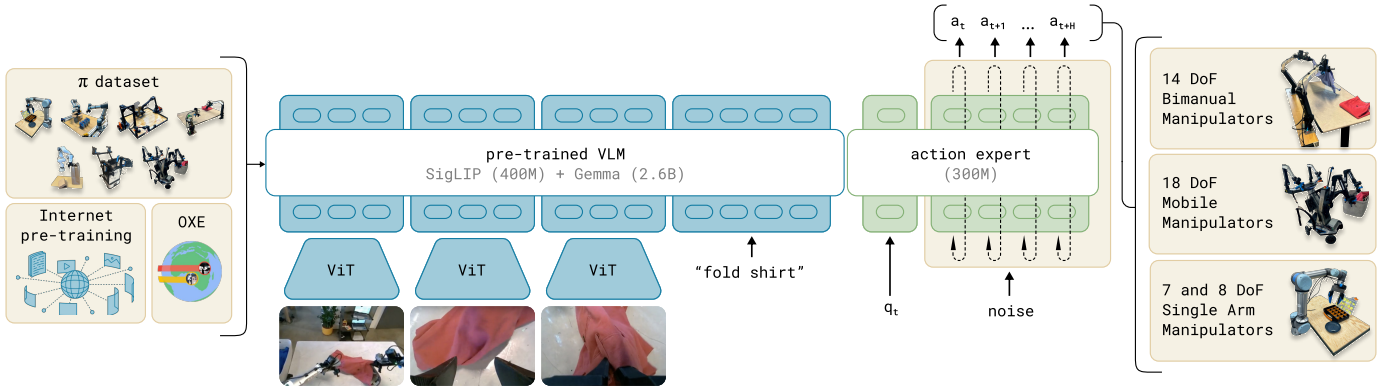
Fig. 3: **Overview of our framework.** We start with a pre-training mixture, which consists of both our own dexterous manipulation datasets and open-source data. We use this mixture to train our flow matching VLA model, which consists of a larger VLM backbone and a smaller *action expert* for processing robot states and actions. The VLM backbone weights are initialized from PaliGemma [5], providing representations learned from large-scale Internet pre-training. The resulting $\pi_0$ model can be used to control multiple robot embodiments with differing action spaces to accomplish a wide variety of tasks.

of more complex and dexterous behaviors, such as tying shoelaces [58] or cooking shrimp [17], we show that our framework can learn very long tasks, sometimes tens of minutes in length, for behaviors that combine both physical dexterity and combinatorial complexity. For example, our laundry folding task requires the robot to manipulate a variety of clothing items that can start in any configuration, and fold multiple items in sequence. Our table bussing task requires discerning the class of novel objects (trash or dishes). We show that a single cross-embodiment model can be used as the base model for these tasks. To our knowledge, our work demonstrates the longest dexterous tasks in the end-to-end robot learning literature.

## III. OVERVIEW

We provide an outline of our model and training procedure in Figure 3. In our training framework, we first assemble a pre-training mixture consisting of a weighted combination of our own dexterous manipulation datasets (Section V-C), collected on 7 different robot configurations for 68 different tasks, and the entire OXE dataset [10], which contains data from 22 robots. The pre-training phase (Section V-A) also uses diverse language labels, combining *task names* and *segment annotations* (fine-grained labels for sub-trajectories, typically about 2 seconds in length). The purpose of the pre-training phase is to train a *base model* that exhibits broad capabilities and generalization, but is not necessarily specialized for high performance on any one task. This base model can follow language commands and perform a variety of tasks at rudimentary proficiency. For complex and dexterous tasks, we then employ a post-training procedure (Section V-A), which uses high-quality curated data to adapt the model to specific downstream tasks. We study both efficient post-training with small to moderate amounts of data, and high-quality post-training with larger datasets for complex tasks such as laundry folding and mobile manipulation.

Our model, which we describe in Section IV, is based on the PaliGemma vision-language model [5], which we then further train with our data mixture. To turn the base PaliGemma VLM into $\pi_0$, we add action outputs that use flow matching [32, 28] to generate continuous action distributions. We describe this design in detail in the following section. Note that we use PaliGemma for convenience and because of its comparatively small size (which is useful for real-time control), but our framework is compatible with any base pre-trained VLM.

## IV. THE $\pi_0$ MODEL

The $\pi_0$ model, illustrated in Figure 3, consists primarily of a language model transformer backbone. Following the standard late fusion VLM recipe [3, 11, 30], image encoders embed the robot's image observations into the same embedding space as language tokens. We further augment this backbone with robotics-specific inputs and outputs — namely, proprioceptive state and robot actions. $\pi_0$ uses conditional flow matching [28, 32] to model the continuous distribution of actions. Flow matching provides our model with high precision and multimodal modeling capability, making it especially well suited to high-frequency dexterous tasks. Our architecture is inspired by Transfusion [59], which trains a single transformer using multiple objectives, with tokens[1] corresponding to continuous outputs supervised via a flow matching loss and tokens corresponding to discrete outputs supervised via a cross-entropy loss. Building on Transfusion, we additionally found that using a separate set of weights for the robotics-specific (action and state) tokens led to an improvement in performance. This design is analogous to a mixture of experts [45, 25, 12, 16] with two mixture elements, where the first element is used for image and text inputs, and

---

[1]In this paper, we use the word "token" to refer to an input/output slot along the sequence dimension, whether the slot corresponds to a discrete variable (e.g., a language token) or a continuous variable (e.g., an image patch or a robot action).

the second is used for robotics-specific inputs and outputs. We refer to the second set of weights as the *action expert*.

Formally, we want to model the data distribution $p(\mathbf{A}_t|\mathbf{o}_t)$, where $\mathbf{A}_t = [\mathbf{a}_t, \mathbf{a}_{t+1}, ..., \mathbf{a}_{t+H-1}]$ corresponds to an *action chunk* of future actions (we use $H = 50$ for our tasks), and $\mathbf{o}_t$ is an observation. The observation consists of multiple RGB images, a language command, and the robot's proprioceptive state, such that $\mathbf{o}_t = [\mathbf{I}_t^1, ..., \mathbf{I}_t^n, \ell_t, \mathbf{q}_t]$, where $\mathbf{I}_t^i$ is $i^{\text{th}}$ image (with 2 or 3 images per robot), $\ell_t$ is a sequence of language tokens, and $\mathbf{q}_t$ is a vector of joint angles. The images $\mathbf{I}_t^i$ and state $\mathbf{q}_t$ are encoded via corresponding encoders and then projected via a linear projection layer into the same embedding space as the language tokens.

For each action $\mathbf{a}_{t'}$ in the action chunk $\mathbf{A}_t$, we have a corresponding *action token* that we feed through the action expert. During training, we supervise these action tokens using a conditional flow matching loss [28, 32],

$$L^\tau(\theta) = \mathbb{E}_{p(\mathbf{A}_t|\mathbf{o}_t), q(\mathbf{A}_t^\tau|\mathbf{A}_t)} ||\mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t) - \mathbf{u}(\mathbf{A}_t^\tau|\mathbf{A}_t)||^2,$$

where subscripts denote robot timesteps and superscripts denote flow matching timesteps, with $\tau \in [0, 1]$. Recent work in high-resolution image [14] and video [38] synthesis has shown that flow matching can achieve strong empirical performance when combined with a simple linear-Gaussian (or optimal transport) probability path [28], given by $q(\mathbf{A}_t^\tau|\mathbf{A}_t) = \mathcal{N}(\tau\mathbf{A}_t, (1-\tau)\mathbf{I})$. In practice, the network is trained by sampling random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, computing the "noisy actions" $\mathbf{A}_t^\tau = \tau\mathbf{A}_t + (1-\tau)\epsilon$, and then training the network outputs $\mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t)$ to match the denoising vector field $\mathbf{u}(\mathbf{A}_t^\tau|\mathbf{A}_t) = \mathbf{A}_t - \epsilon$. The action expert uses a full bidirectional attention mask, so that all action tokens attend to each other. During training, we sample the flow matching timestep $\tau$ from a beta distribution that emphasizes lower (noisier) timesteps. See Appendix B for more details.

At inference time, we generate actions by integrating the learned vector field from $\tau = 0$ to $\tau = 1$, starting with random noise $\mathbf{A}_t^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We use the forward Euler integration rule:

$$\mathbf{A}_t^{\tau+\delta} = \mathbf{A}_t^\tau + \delta\mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t),$$

where $\delta$ is the integration step size. We use 10 integration steps (corresponding to $\delta = 0.1$) in our experiments. Note that inference can be implemented efficiently by caching the attention keys and values for the prefix $\mathbf{o}_t$ and only recomputing the suffix corresponding to the action tokens for each integration step. We provide more details regarding the inference procedure, including the inference time for each part of the model, in Appendix D.

While in principle our model can be initialized from scratch or fine-tuned from any VLM backbone, in practice we use PaliGemma [5] as our base model. PaliGemma is an open-source 3 billion parameter VLM that offers a convenient trade-off between size and performance. We add 300M parameters for the action expert (which is initialized from scratch) for a total of 3.3 billion parameters. We provide a full description of the model architecture in Appendix B.

**Non-VLM baseline model.** In addition to our main VLA model, we also trained a similar baseline model that did not use a VLM initialization for ablation experiments. This model, which we refer to as $\pi_0$-small, has 470M parameters, does not use VLM initialization, and has a number of small differences that we found to be helpful for training on our data without VLM initialization, which are summarized in Appendix C. This model is used in our comparisons to evaluate the benefits of incorporating VLM pertaining.

## V. DATA COLLECTION AND TRAINING RECIPE

Broadly capable robot foundation models require not only an expressive and powerful architecture, but also the right dataset and, more importantly, the right training *recipe*. In the same way that LLM training is typically divided into pre-training and post-training phases, we employ a multi-stage training procedure for our model. The goal of the pre-training phase is to expose the model to a diverse range of tasks so that it can acquire broadly applicable and general physical capabilities, while the goal of the post-training phase is to provide the model with the ability to skillfully and fluently execute the desired downstream task. Because of this, the requirements for the pre-training and post-training datasets are distinct: the pre-training dataset should cover as many tasks as possible, and within each of those tasks should cover a diversity of behaviors. The post-training dataset should instead cover behaviors that are conducive to effective task execution, which should exhibit a consistent and fluent strategy. Intuitively, the diverse (but lower quality) pre-training data allows the model to recover from mistakes and handle highly varied situations, which might not otherwise occur in the high-quality post-training data, while the post-training data teaches the model to perform the task well.
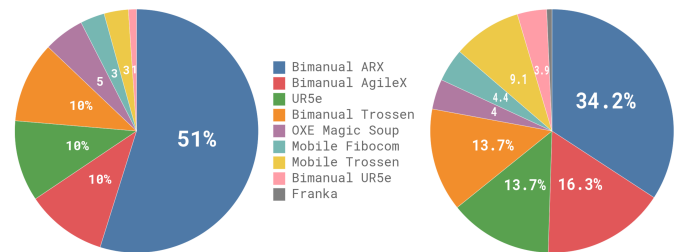
### A. Pre-training and post-training



Fig. 4: **Overview of our dataset**: The pre-training mixture consists of a subset of OXE [10] and the $\pi$ dataset. We use a subset of OXE, which we refer to as OXE Magic Soup [24]. The right figure illustrates the weight of the different datasets in the pre-training mixture. The left figure illustrates their relative sizes as measured by the number of steps.

We provide an overview of our pre-training mixture in Figure 4. Since each training example corresponds to a timestep — i.e., a tuple $(\mathbf{o}_t, \mathbf{A}_t)$, — we will quantify data in terms of timesteps in this discussion. 9.1% of the training mixture consists of open-source datasets, including OXE [10], Bridge

v2 [52], and DROID [23]. The robots and tasks in these datasets typically have one or two cameras and use low-frequency control, between 2 and 10 Hz. However, these datasets cover a wide range of objects and environments. To learn dexterous and more complex tasks, we also use 903M timesteps of data from our own datasets, where 106M steps are from single-arm robots and 797M are from dual-arm robots. This data has 68 tasks, where each task is composed of complex behaviors — e.g., the "bussing" task involves putting a wide range of different dishes, cups, and utensils into a bussing bin, and a wide array of trash items into the garbage. Note that this definition of task is significantly different from prior work, which typically uses any combination of noun and verb (e.g., "pick up the cup" vs. "pick up the plate") to constitute a distinct task. Therefore, the actual range of behaviors in our dataset is significantly broader than this number of "tasks" would imply. We discuss the specific robots and tasks in our dataset in more detail in Section V-C.

Since the datasets are somewhat imbalanced in size (e.g., the more difficult laundry folding tasks are overrepresented), we weight each task-robot combination by $n^{0.43}$, where $n$ is the number of samples for that combination, such that over-represented combinations are down-weighted. The configuration vector $\mathbf{q}_t$ and action vectors $\mathbf{a}_t$ always have the dimensionality of the largest robot in the dataset (18 in our case, to accommodate two 6-DoF arms, 2 grippers, a mobile base, and a vertically actuated torso). For robots with lower-dimensional configuration and action spaces, we zero-pad the configuration and action vectors. For robots with fewer than three images, we also mask out the missing image slots.

In the post-training phase, we fine-tune our model with a smaller task-specific dataset to specialize it to particular downstream applications. As mentioned previously, our definition of "task" is fairly broad — e.g., the "bussing" task requires manipulating a wide range of different objects. Different tasks require very different datasets, with the simplest of the tasks necessitating only 5 hours and the most complex tasks using 100 or more hours of data.

### B. Language and high-level policies

More complex tasks that require semantic reasoning and high-level strategy, such as table bussing, can also benefit from a high-level policy that decomposes high-level tasks (such as "bus the table") into more immediate subtasks (such as "pick up the napkin" or "throw the napkin into the trash"). Since our model is trained to process language inputs, we can use a high-level VLM to make these semantic inferences, a method that is analogous to LLM/VLM planning methods such as SayCan [2]. We use such a high-level policy to assist our model with high-level strategy for several of our experimental tasks, as we will discuss in Section VI.

### C. Robot system details

Our dexterous manipulation datasets include 7 different robot configurations and 68 tasks. We summarize these platforms in Figure 5, and discuss them below:



Fig. 5: **The robots used in our experiments.** These include single and dual-arm manipulators with 6-DoF and 7-DoF arms, as well as holonomic and nonholonomic mobile manipulators. $\pi_0$ is trained jointly on all of these platforms.

**UR5e.** An arm with a parallel jaw gripper, with a wrist-mounted and over-the-shoulder camera, for a total of two camera images and a 7-dimensional configuration and action space.

**Bimanual UR5e.** Two UR5e setups, for a total of three camera images and a 14-dimensional configuration and action space.

**Franka.** The Franka setup has two cameras and an 8-dimensional configuration and action space.

**Bimanual Trossen.** This setup has two 6-DoF Trossen ViperX arms in a configuration based on the ALOHA setup [4, 57], with two wrist cameras and a base camera, and a 14-dimensional configuration and action space.

**Bimanual ARX & bimanual AgileX.** This setup uses two 6-DoF arms, and supports either ARX or AgileX arms, with three cameras (two wrist and one base) and a 14-dimensional configuration and action space. This class encompasses two distinct platforms, but we categorize them together because of their similar kinematic properties.

**Mobile Trossen & mobile ARX.** This setup is based on the Mobile ALOHA [57] platform, with two 6-DoF arms on a mobile base, which are either ARX arms or Trossen ViperX arms. The nonholonomic base adds two action dimensions, for a 14-dimensional configuration and 16-dimensional action space. There are two wrist cameras and a base camera. This class encompasses two distinct platforms, but we categorize them together because of their similar kinematic properties.

**Mobile Fibocom.** Two 6-DoF ARX arms on a holonomic base. The base adds three action dimensions (two for translation and one for orientation), for a 14-dimensional configuration and 17-dimensional action space.

We summarize the proportion of our dataset from each robot in Figure 4.

## VI. EXPERIMENTAL EVALUATION

Our experimental evaluation consists of out-of-box evaluation experiments that compare our base (pre-trained) model
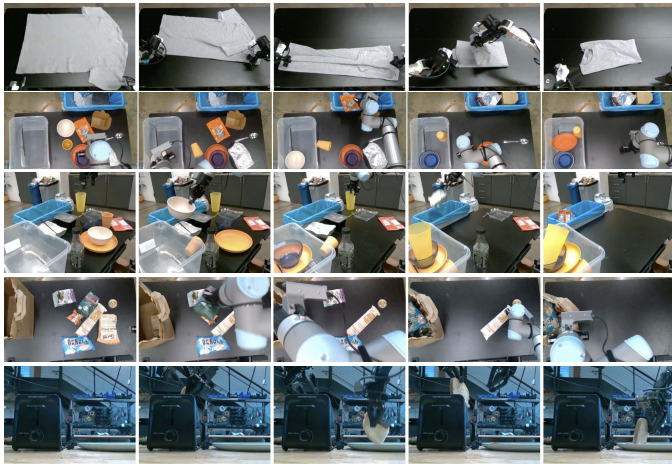
Fig. 6: **Out-of-box evaluation tasks**: To evaluate our base model, we run it after pre-training on five tasks: **shirt folding**, **bussing easy**, **bussing hard**, **grocery bagging**, and **toast out of toaster**. The tasks require a combination of dexterous manipulation, multi-stage behaviors, and semantic recognition.

to alternative model designs with direct prompting, as well as detailed fine-tuning experiments that evaluate our model on challenging downstream tasks, comparing it to other methods that have been proposed for dexterous manipulation. We study the following research questions:

**How well does $\pi_0$ perform after *pre-training* on a variety of tasks that are present in the pre-training data?** We study this question by directly evaluating $\pi_0$, with comparisons to other robot foundation models.

**How well does $\pi_0$ follow language commands?** These experiments compare $\pi_0$ to $\pi_0$-small, a smaller version of our model without VLM initialization, to evaluate its performance on following language commands. We evaluate with both human-provided commands and commands specified by a high-level VLM policy, as discussed in Section V-B.

**How does $\pi_0$ compare to methods that have been proposed specifically for addressing dexterous manipulation tasks?** These experiments study downstream tasks for which we can either fine-tune our model from the pre-trained initialization, or train it from scratch on task-specific data, comparing to prior methods that were proposed for dexterous manipulation. We aim to evaluate both the benefits of our architecture and our pre-training procedure.

**Can $\pi_0$ be adapted to complex, multi-stage tasks?** In our final set of experiments, we fine-tune $\pi_0$ to a set of particularly complex tasks, including folding laundry and bussing a table. These tasks take between 5 and 20 minutes to complete. Some require guidance from a high-level policy.

### A. Evaluating the base model

In our first set of experiments, we evaluate the model after pre-training on our full mixture, without any post-training, to evaluate how well our base model can perform a variety of tasks. We compare to other robot foundation models in

the literature: both VLAs and smaller models that are trained from scratch on the same pre-training mixture. We evaluate on the following tasks, visualized in Figure 6, with each task commanded to the same base model via a language command.

**Shirt folding:** the robot must fold a t-shirt, which starts flattened.

**Bussing easy:** the robot must clean a table, putting trash in the trash bin and dishes into the dish bin. The score indicates the number of objects that were placed in the correct receptacle.

**Bussing hard:** a harder version of the bussing task, with more objects and more challenging configurations, such as utensils intentionally placed on top of trash objects, objects obstructing each other, and some objects that are not in the pre-training dataset.

**Grocery bagging:** the robot must bag all grocery items, such as potato chips, marshmallows, and cat food.

**Toast out of toaster:** the robot removes toast from a toaster.

Providing comparisons for these experiments is challenging because very few prior models can operate at this scale. We compare to OpenVLA [24], a 7B parameter VLA model that was originally trained on the OXE dataset [10]. We train OpenVLA on our full mixture. This is a very difficult mixture for OpenVLA, which does not support action chunking or high-frequency control. We also compare to Octo [50], a smaller 93M parameter model. While Octo is not a VLA, it does use a diffusion process to generate actions, providing a valuable point of comparison for our flow matching VLA. We also train Octo on the same mixture as our model. Due to time constraints, we were unable to train OpenVLA and Octo for the same number of epochs as our full model. We therefore also compare to a "compute parity" version of our model, which is trained for only 160k steps (as opposed to 700k steps for our main model), which is equal to or lower than the number of steps provided to the baselines (160k for OpenVLA, 320k for Octo). We also include a version of the OpenVLA model that we fine-tuned only on the UR5e data, without cross-embodiment training, in the hopes of providing an even stronger baseline on the UR5e tasks. Finally, we include a comparison to the $\pi_0$-small model described in Section IV, which can be viewed as a scaled-down version of our model without VLM pre-training.

The evaluation metric uses a normalized score averaged over 10 episodes per task and method, where an episode receives a score of 1.0 for a full success, and a fractional score for partial success. For example, the score for bussing is the fraction of objects that are correctly placed in the proper receptacle. We describe the scoring rubrics in Appendix E. The results, shown in Figure 7, show that $\pi_0$ attains by far the best results across the board on all the out-of-box tasks, with near perfect success rates on shirt folding and the easier bussing tasks, and large improvements over all baselines. The "parity" version of $\pi_0$, which is trained for only 160k steps, still outperforms all the baselines, and even $\pi_0$-small outperforms OpenVLA and Octo. OpenVLA struggles on these tasks because its autoregressive discretization architecture does not support action chunks. The UR5e-only OpenVLA model performs better, but is still far
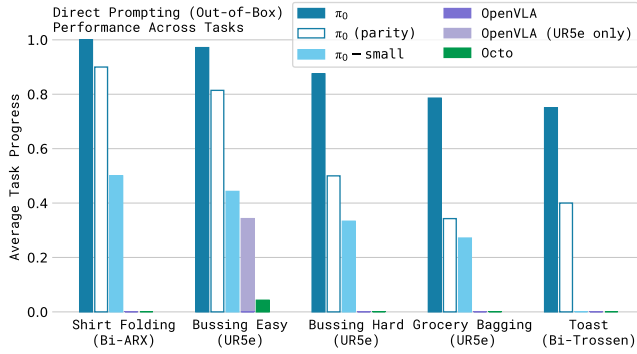
Fig. 7: **Out-of-box evaluation results**: We evaluate $\pi_0$ trained for the full 700k steps, a version trained for 160k steps that matches the number of updates for baseline models, $\pi_0$-small, and three baselines: OpenVLA and Octo trained on all of our data, and OpenVLA trained only on the UR5e tasks (which we found to work better on UR5e tasks). Across all tasks and all comparisons, even the "parity" version of our model outperforms all baselines, and the full version of our model achieves the best results by a large margin.

below the performance of $\pi_0$. Octo does support action chunks, but has a comparatively limited representational capacity. This comparison illustrates the importance of combining large, expressive architectures with the ability to model complex distributions via flow matching or diffusion. Additionally, the comparison to $\pi_0$-small illustrates the importance of incorporating VLM pre-training. Unfortunately, it is hard to make this last comparison fair: $\pi_0$-small uses fewer parameters, but larger models are difficult to use without pre-training. Overall, these experiments show that $\pi_0$ provides a powerful pre-trained model with the ability to effectively perform a variety of tasks with a variety of robots, with much better performance than prior models.

### B. Following language commands

In the next set of experiments, we fine-tune the base $\pi_0$ model to follow language commands in a set of evaluation domains. We compare this fine-tuned $\pi_0$ model with the $\pi_0$-small model described in Section IV, which we found to be the strongest baseline in the previous section. Recall that $\pi_0$-small does *not* use a VLM initialization. This experiment therefore aims to measure how much VLM pre-training boosts our model's ability to follow language instructions. Note that $\pi_0$-small is also a significantly smaller model — unfortunately, it is difficult to remove this confounder, because VLM initialization serves both to make it practical to train a much larger model without overfitting, and to improve language instruction following. We nonetheless hope that this experiment sheds light on the language capabilities of $\pi_0$. The language instructions for each task consist of objects to pick up and locations to place those objects, with language-labeled segments that are about 2 seconds in length. Each full



Fig. 8: **The tasks in our language evaluation.** We evaluate our model on 3 different language-conditioned tasks, each of which requires following a sequence of intermediate language commands. The tasks involve bussing a table (top) to put dishes in a bin and garbage in a trash bin, setting a table (middle) by taking items out of a bin, and packing a shopping bag (bottom).

task consists of numerous such segments. The tasks in this evaluation consist of:

**Bussing:** the robot must clean a table, placing dishes and cutlery in a bin, and trash into a trash bin.

**Table setting:** the robot must take out items from a bin to set a table, including a place mat, dishes, silverware, napkin, and cups, and adjust them according to language instructions.

**Grocery bagging:** the robot must pack grocery items, such as bags of coffee beans, barley, marshmallow, seaweed, almonds, spaghetti, and cans into a bag.

In Figure 8, we show the language-conditioned tasks in our evaluation and present the evaluation results. We evaluate five different conditions. $\pi_0$-flat (and $\pi_0$-small-flat) corresponds to directly command the model with the task description (e.g., "bag the groceries"), without intermediate language commands. $\pi_0$-human (and $\pi_0$-small-human) provides intermediate step commands (e.g., which object to pick and where to place it) from an expert human user. These conditions evaluate each model's ability to follow more detailed language commands: while these intermediate commands provide considerable information for how to perform the task, the model must be able to understand and follow those commands to benefit from them. Finally, $\pi_0$-HL evaluates $\pi_0$ with high-level commands provided by a high-level VLM, as discussed in Section V-B. This condition is also autonomous, without any human expert.

The results in Figure 9, averaging over 10 trials per task, show that the language following accuracy of $\pi_0$ is significantly better than that of $\pi_0$-small. This suggests a significant improvement from the larger pre-trained VLM initialization. This capability translates to an improvement in performance with expert human guidance ($\pi_0$-human) and with high-level model guidance ($\pi_0$-HL). The results indicate that $\pi_0$'s language following ability directly translates into better autonomous performance on complex tasks with high-level guidance.

### C. Learning new dexterous tasks

In the next set of experiments, we evaluate our model on new tasks that differ significantly from the pre-training data,
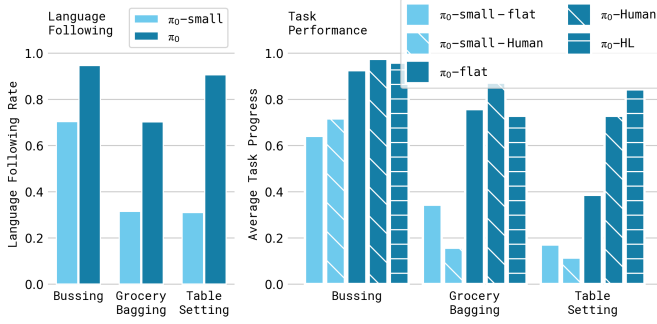
Fig. 9: **Language evaluation.** We compare "flat" versions of our policies, −flat, which receive only the overall task command (e.g., "bag the groceries") with a method that receives intermediate commands from a human expert, −human, or a high-level VLM policy, −HL. We also compare our model to a small non-VLM variant under the "expert" condition, $\pi_0$ and $\pi_0$-small, in terms of language following accuracy. The results show a significant improvement with $\pi_0$ from intermediate language commands provided by a human expert and to a lesser degree by an autonomous high-level policy. Notably, due to $\pi_0$-small's limited language following ability, overall it does not gain with the addition of a high-level expert.
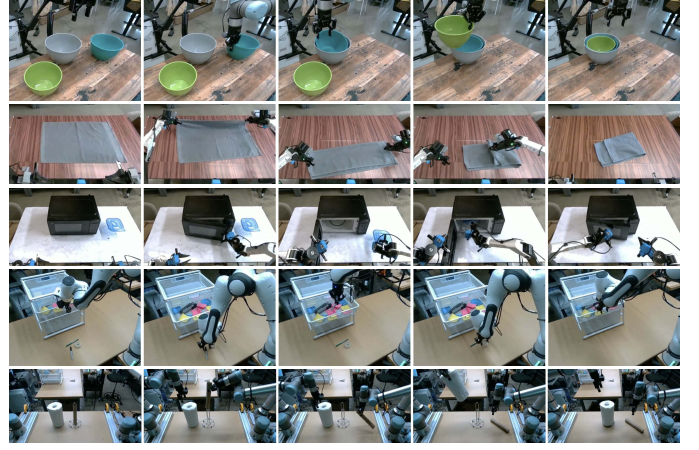


Fig. 10: **Fine-tuning evaluation tasks**: We fine-tune our model to a variety of downstream tasks that are distinct from the tasks seen in pre-training. Our tasks represent a range of similarity from the pre-training tasks, with tasks that are most similar to pre-training (**stack bowls** and **towel folding**), a task that introduces an unseen new element (a microwave), and tasks that require new motions and new object types (**Franka items in drawer** and **paper towel replacement**).

requiring entirely new behaviors. For these evaluations, we fine-tune the model using various amounts of data for each new task. While each task is new, we partition the tasks into "tiers" depending on how much they differ from tasks in the pre-training data. The tasks, shown in Figure 10, are:

**UR5e stack bowls.** This task requires stacking bowls, with four bowls of different sizes. Since this task requires grasping and moving dishes like the bussing task in the pre-training data, we place it in the "easy" tier. The training data contains a variety of bowls, and the evaluations use a mix of seen and unseen bowls.

**Towel folding.** This task requires folding a towel. Since this is similar to shirt folding, which is present in pre-training, we place it in the "easy" tier.

**Tupperware in microwave.** This task requires opening a microwave, putting a plastic container inside it, and closing it. The containers come in different shapes and colors, and the evaluations use a mix of seen and unseen containers. The container manipulation resembles pre-training data, but the microwave is not found in pre-training.

**Paper towel replacement.** This task requires removing an old cardboard paper towel tube from a holder and replacing it with a fresh paper towel roll. Because no such items are found in pre-training, we consider this "hard."

**Franka items in drawer.** This task requires opening a drawer, packing items into a drawer, and closing it. Because there is no similar task with the Franka robot in pre-training, we consider this "hard."

We compare our model after fine-tuning both to Open-VLA [24] and Octo [50], which also employ a pre-training and fine-tuning recipe. Since our aim is to evaluate the specific models (rather than the architectures), we use the publicly available pre-trained checkpoints for these models, which are trained on OXE [10], and then fine-tune them to each task. We also compare to ACT [57] and Diffusion Policy [9], which are designed specifically for learning dexterous tasks from smaller datasets. ACT and Diffusion Policy are trained *only* on the fine-tuning datasets, which are of similar size to the individual datasets used in the ACT and Diffusion Policy experiments [9, 57]. We evaluate $\pi_0$ by fine-tuning from our pre-trained base model, as well as by training from scratch. This comparison is meant to evaluate the individual benefits of the $\pi_0$ architecture and our pre-training procedure. We hypothesize that the $\pi_0$ architecture with VLM initialization should already provide a stronger starting point for the individual tasks, while the pre-training procedure should further improve its performance, especially with smaller fine-tuning datasets.

Figure 11 shows the performance across all of the tasks for a variety of methods, averaging over 10 trials per task, with different amounts of fine-tuning data on each task. We include all of the baselines on the **stack bowls** and **Tupperware in microwave** tasks. Since OpenVLA and Octo attain significantly worse performance, we only run these for one of the dataset sizes, due to the time cost of evaluating so many models in the real world. The results show that $\pi_0$ generally outperforms other methods. Interestingly, the strongest prior models are the ones that are trained entirely from scratch on the target tasks, suggesting that leveraging pre-training in these domains presents a major challenge for prior approaches. While the 5-hour policy for $\pi_0$ on the Tupperware task performs similarly to the baselines, the 1-hour version is significantly better. As expected, pre-training leads to larger improvement for tasks
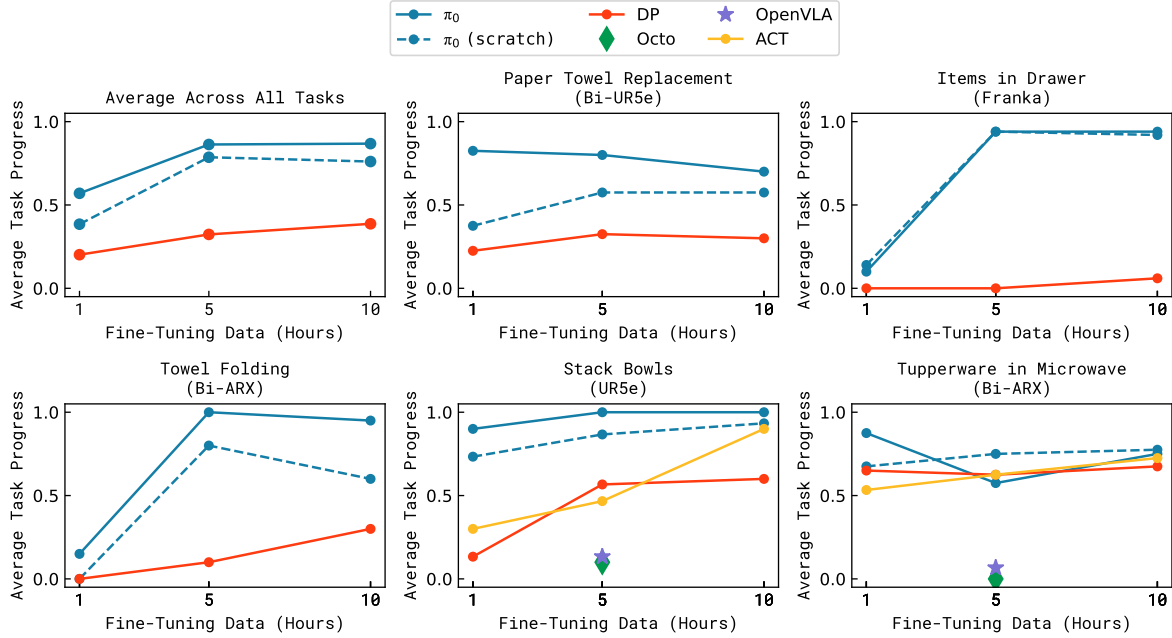
Fig. 11: **Fine-tuning with varying amounts of data.** $\pi_0$ can learn some easier tasks even with smaller amounts of data, and the pre-trained model often attains a larger improvement over the model trained from scratch.

that are more similar to the pre-training data, though the pre-trained model is frequently better than the non-pre-trained model, sometimes by as much as 2x.

### D. Mastering complex multi-stage tasks

In our final set of experiments, we tackle a range of challenging multi-stage tasks via a combination of fine-tuning and language. For some of these tasks, data is present in pre-training, but fine-tuning is required to attain mastery. For some, no data is present in pre-training. The tasks in this evaluation, shown in Figure 12, are:

**Laundry folding:** This task requires a static (non-mobile) bimanual system to fold articles of clothing. The clothing items start in a randomized crumpled state in a bin, and the goal is to take out the item, fold it, and place it on top of a stack of previously folded items. The randomized initial configuration of the crumpled laundry presents a major challenge, since the policy needs to generalize to any configuration. This task is present in pre-training.

**Mobile laundry:** Here, the Fibocom mobile robot in Figure 5 has to fold laundry, facing many of the same challenges while controlling orientation and translation. This task is present in pre-training.

**Dryer unloading:** Here, the Fibocom mobile robot has to take laundry out of a dryer and place it into a hamper. This task is present in pre-training.

**Table bussing:** This task requires bussing a table with a diverse array of novel objects in a clutter scene, presenting a much greater challenge than the benchmark in our out-of-box evaluation: the policy must generalize to unseen objects

of varying shapes and sizes, and perform complex dexterous motions, such as twisting the gripper to pick up large plates and carefully grasping thin, delicate items such as glasses. The robot must handle dense clutter and intelligently sequence various behaviors — for example, to clean off a plate with trash, it must first pick up the plate, then shake its contents into the garbage, and then place the plate in the bin. This task is not present in pre-training.

**Box building:** The robot has to assemble a cardboard box that starts in a flattened state. This task presents a number of major challenges: the box needs to bent in the right way, and the robot needs to hold down parts of the box while folding others, utilizing both arms and even the surface of the table to brace during folding motions. The robot might need to retry some folds, requiring a reactive and intelligent strategy. This task is not present in pre-training.

**To-go box:** This task requires moving several food items from a plate into a to-go box, requiring packing the items into the box so that they do not stick out, and then closing the box with both arms. This task is not present in pre-training.

**Packing eggs:** The robot needs to take six eggs out of a bowl and pack them into an egg carton, and then close the carton. The eggs need to be grasped in a manner appropriate to their pose inside the bowl, and then placed into open slots in the carton. This presents challenges due to the egg shape, slipperiness, and the need for careful placement. Closing the box requires the use of both arms. This task is not present in pre-training.

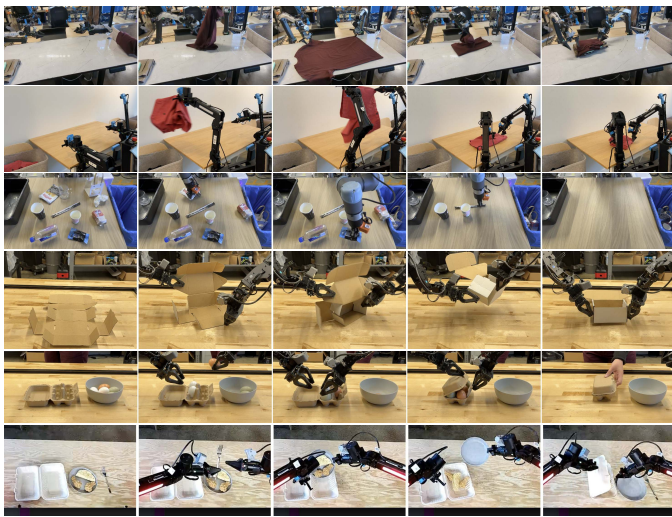The results, showing average scores per task over 10 trials,

Fig. 12: **We evaluate a range of complex and temporally extended tasks.** This includes: folding laundry from a bin with a stationary (a) or mobile (b) robot, bussing a real lunch table (c), assembling a box (d), packing eggs into a carton (e), and packing food into a to-go box (f). These tasks require combining dozens of individual behaviors, such as grasping, stacking, folding, and flattening, generalization to a huge variety of object configurations, and complex physical properties, such as deformable objects or flexible cardboard.



Fig. 13: **Post-training results on complex tasks** in terms of average scores over 10 trials. The full pre-trained $\pi_0$ model attains more than 50% of the maximum score across all of the tasks, and typically outperforms the ablations, with especially significant improvements on the hardest tasks.

are presented in Figure 13. The scoring rubrics are in Appendix E. A score of 1.0 represents a perfect execution, while partial scores correspond to partially completed tasks (e.g., 0.5 indicates that half the objects were bussed correctly). These tasks are very difficult, and we were not able to solve them with other methods. We therefore use these tasks to compare to ablations of our approach, evaluating $\pi_0$ after pre-training and fine-tuning, out of the box after pre-training only ("out-of-box"), and training on the fine-tuning data without any pre-training ("scratch"). The results show that $\pi_0$ can solve many of these tasks, with our full pre-training and fine-tuning recipe performing best across the board. Note that many of these more difficult tasks show a very large improvement from using the pre-trained model, indicating that pre-training is especially useful with harder tasks. The absolute performance of $\pi_0$ varies across the tasks, likely due to differences in task difficulty and the degree to which the tasks are represented in pre-training. We recommend that readers watch the task videos on the accompanying website for a more complete impression of these tasks and their complexity. We believe that this level of autonomous performance on such challenging tasks represents a new state of the art in dexterous robot manipulation with learned policies.

## VII. DISCUSSION, LIMITATIONS, AND FUTURE WORK

We presented a framework for training a robot foundation model, which we refer to as $\pi_0$, that consists of pre-training on highly diverse data, followed by either out-of-box evaluation or fine-tuning to complex downstream tasks.
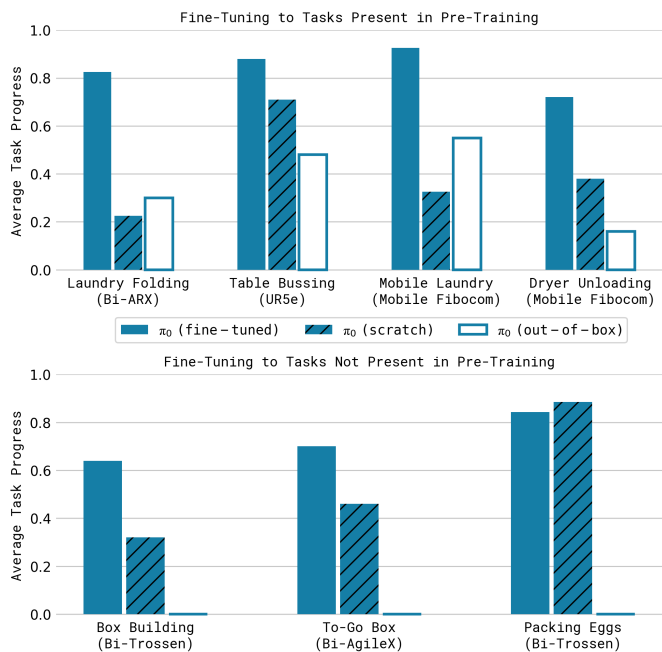
Our empirical evaluation studies tasks that combine dexterity, generalization, and temporally extended multi-stage behaviors. Our model incorporates Internet-scale vision-language model (VLM) pre-training with flow matching for representing complex high-frequency action chunks. Our pre-training mixture consists of 10,000 hours of dexterous manipulation data from 7 different robot configurations and 68 tasks, in addition to large amounts of previously collected robot manipulation data from OXE [10], DROID [23], and Bridge [52]. To our knowledge, this represents the largest pre-training mixture ever used for a robot manipulation model. Our fine-tuning experiments include over 20 tasks, where we show that our model outperforms a variety of baselines, including prior VLA models [24] and models designed specifically for dexterous manipulation [57, 9]. We also examine how our post-training recipe can enable highly complex tasks, such as folding multiple articles of clothing from arbitrary initial configurations or assembling boxes.

Our framework broadly resembles the training procedures employed for large language models, which typically consist of pre-training a base model on very large datasets scraped from the web, followed by a post-training procedure that aims to "align" the model to enable it to follow instructions and perform user commands. It is generally recognized that most of the "knowledge" in such models is acquired in the pre-training phase, while the post-training phase serves to tell the model how it should leverage that knowledge to fulfill user commands. Our experiments imply that an analogous

phenomenon might take place with robot foundation models, where pre-trained models have some zero-shot capabilities, but complex tasks like laundry following require fine-tuning with high-quality data. Training on only this high-quality data results in a brittle model that does not reliably recover from mistakes, while running the pre-trained model in zero shot does not always exhibit the fluent strategies demonstrated in the post-training data.

We hope that our results will serve as a stepping stone toward general and broadly applicable robot foundation models. Our experiments suggest that such models may soon be a reality, but there are a number of limitations and ample room for future work. First, our experiments do not yet provide a comprehensive understanding of how the pre-training datasets should be composed: we combined all data available to us, but understanding what type of data is more helpful to add and how it should be weighted remains an open problem. Not all tasks in our evaluation work reliably, and it remains unclear how to predict how much and what kind of data is needed to attain near-perfect performance. Finally, it remains to be seen how much positive transfer there is in combining highly diverse data, particularly from different tasks and different robots: although our results suggest that universal pre-trained robot foundation models might become a reality, it is left for future work to understand whether this universality extends to much more distinct domains, such as autonomous driving, navigation, and legged locomotion.

## REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning.

*Advances in neural information processing systems*, 35: 23716–23736, 2022.

[4] Jorge Aldaco, Travis Armstrong, Robert Baruch, Jeff Bingham, Sanky Chan, Kenneth Draper, Debidatta Dwibedi, Chelsea Finn, Pete Florence, Spencer Goodrich, et al. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation. *arXiv preprint arXiv:2405.02292*, 2024.

[5] 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.

[6] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. RoboAgent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4788–4795. IEEE, 2024.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[8] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[10] OX-Embodiment Collaboration, A Padalkar, A Pooley, A Jain, A Bewley, A Herzog, A Irpan, A Khazatsky, A Rai, A Singh, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 1(2), 2023.

[11] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-

e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[12] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

[13] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[14] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

[15] Haritheja Etukuru, Norihito Naka, Zijin Hu, Seung-jae Lee, Julian Mehu, Aaron Edsinger, Chris Paxton, Soumith Chintala, Lerrel Pinto, and Nur Muhammad Mahi Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. *arXiv preprint arXiv:2409.05865*, 2024.

[16] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[17] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[18] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018.

[19] Wanggui He, Siming Fu, Mushui Liu, Xierui Wang, Wenyi Xiao, Fangxun Shu, Yi Wang, Lei Zhang, Zhelun Yu, Haoyuan Li, et al. Mars: Mixture of auto-regressive models for fine-grained text-to-image synthesis. *arXiv preprint arXiv:2407.07614*, 2024.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[22] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[23] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[24] 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.

[25] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[26] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[27] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[28] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[29] Bingchen Liu, Ehsan Akhgari, Alexander Visheratin, Aleks Kamko, Linmiao Xu, Shivam Shrirao, Joao Souza, Suhail Doshi, and Daiqing Li. Playground v3: Improving text-to-image alignment with deep-fusion large language models. *arXiv preprint arXiv:2409.10695*, 2024.

[30] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[31] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[32] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

[33] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. RoboTurk: A crowdsourcing platform for robotic skill learning through

imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[34] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[35] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[36] 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.

[37] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[38] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

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

[40] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[41] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[42] V Sanh. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[43] Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

[44] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[45] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[46] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[47] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

[48] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[49] 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.

[50] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[52] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. BridgeData v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[53] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[54] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[55] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

[56] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto

Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 30–37. IEEE, 2016.

[57] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[58] Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

[59] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024.

[60] Minjie Zhu, Yichen Zhu, Jinming Li, Junjie Wen, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, et al. Scaling diffusion policy in transformer to 1 billion parameters for robotic manipulation. *arXiv preprint arXiv:2409.14411*, 2024.

# APPENDIX

## A. Contributions

The authors contributed to the following areas (listed alphabetically):

**Data and operations:** Noah Brown, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Liyiming Ke, Suraj Nair, Lucy Shi, and Anna Walling.

**Evaluation experiments:** Kevin Black, Michael Equi, Chelsea Finn, Brian Ichter, Liyiming Ke, Adrian Li-Bell, Suraj Nair, Karl Pertsch, and Lucy Shi.

**Model design:** Kevin Black, Brian Ichter, Sergey Levine, Karl Pertsch, Lucy Shi, and Quan Vuong.

**Post-training:** Michael Equi, Chelsea Finn, Liyiming Ke, Adrian Li-Bell, Suraj Nair, and Lucy Shi.

**Pre-training:** Kevin Black, Danny Driess, Brian Ichter, Sergey Levine, Karl Pertsch, Lucy Shi, and Quan Vuong.

**Robot hardware:** Noah Brown, Adnan Esmail, Chelsea Finn, Tim Jones, and Mohith Mothukuri.

**Robot software:** Karol Hausman, Szymon Jakubczak, Sergey Levine, James Tanner, and Haohuan Wang.

**Training infrastructure:** Kevin Black, Michael Equi, Sergey Levine, Adrian Li-Bell, Suraj Nair, Quan Vuong, Haohuan Wang, and Ury Zhilinsky.

**Writing and illustration:** Kevin Black, Chelsea Finn, Lachy Groom, Karol Hausman, Brian Ichter, Sergey Levine, and Quan Vuong.

## B. Model Architecture Details

In this section, we provide a full description of the model architecture. We follow the PaliGemma VLM [5] design, with the following differences: (1) additional input and output projections for the robotics-specific tokens, including the state vector $\mathbf{q}_t$ and action vectors $\mathbf{A}_t = [\mathbf{a}_t, ..., \mathbf{a}_{t+H-1}]$, (2) an additional MLP for incorporating the flow matching timestep information $\tau$, and (3) a second, smaller set of weights for the action expert.

**Additional inputs and outputs.** The standard PaliGemma architecture takes in a sequence of images $[\mathbf{I}_t^1, ..., \mathbf{I}_t^n]$ followed by a language prompt $\ell_t$. We add an input $\mathbf{q}_t$ for the robot's proprioceptive state, which is mapped to the transformer embedding dimension using a linear projection. The final set of input tokens correspond to the noisy action chunk $\mathbf{A}_t^\tau = [\mathbf{a}_t^\tau, ..., \mathbf{a}_{t+H-1}^\tau]$, with the number of tokens equal to the action horizon ($H = 50$ for our tasks). We only use the transformer outputs corresponding to the $H$ noisy actions, which are decoded into $\mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t)$ using a linear projection.

**Incorporating the flow matching timestep.** The noisy action chunk $\mathbf{A}_t^\tau$ is mapped to the transformer's embedding dimension using an MLP that also incorporates the flow matching timestep $\tau$. For each noisy action $\mathbf{a}_{t'}^\tau$, the expression for the corresponding embedding that is fed into the transformer is $W_3 \cdot \text{swish}(W_2 \cdot \text{concat}(W_1 \cdot \mathbf{a}_{t'}^\tau, \phi(\tau)))$, where $\phi : \mathbb{R} \to \mathbb{R}^w$ is a sinusoidal positional encoding function [51], $W_1 \in \mathbb{R}^{w \times d}$, $W_2 \in \mathbb{R}^{w \times 2w}$, $W_3 \in \mathbb{R}^{w \times w}$, $d$ is the action dimension, and $w$ is the embedding dimension (or *width*) of the action expert.

**Attention mask.** $\pi_0$ uses a blockwise causal attention mask with 3 blocks: $[\mathbf{I}_t^1, ..., \mathbf{I}_t^n, \ell_t]$, $[\mathbf{q}_t]$, and $[\mathbf{a}_t^\tau, ..., \mathbf{a}_{t+H-1}^\tau]$. Within each block, there is full bidirectional attention, whereas the tokens in each block cannot attend to the tokens in future blocks. The first block includes the input modalities from PaliGemma's VLM pre-training, which are prevented from attending to future blocks (which include new inputs) to minimize distribution shift from said pre-training. The robot state $\mathbf{q}_t$ is its own block because it does not change with each flow matching integration step; preventing it from attending to the final block allows its corresponding keys and values to be cached during sampling. The final block corresponds to the noisy actions $\mathbf{A}_t^\tau$, which can attend to the full input sequence.

**Action expert.** $\pi_0$ is implemented as a single transformer with two sets of weights (also known as experts [45]), where each token is routed to one of the experts; the weights interact only through the transformer's self-attention layers. The images and language prompt, $[\mathbf{I}_t^1, ..., \mathbf{I}_t^n, \ell_t]$, are routed to the larger VLM backbone, which we initialize from PaliGemma. The inputs not seen during VLM pre-training, $[\mathbf{q}_t, \mathbf{A}_t^\tau]$, are routed to the action expert. PaliGemma is based on the Gemma 2B [49] language model, which uses multi-query attention [44] and a configuration of {*width*=2048, *depth*=18, *mlp_dim*=16,384, *num_heads*=18, *num_kv_heads*=1, *head_dim*=256}. Since the experts interact only in the self-attention layers, *width* and *mlp_dim* do not necessarily need to match between experts. To speed up inference (which requires multiple forward passes of the action expert), we downsize the action expert to {*width*=1024, *mlp_dim*=4096}, resulting in a parameter count of ∼300M.

**Sampling the flow matching timestep.** The original flow matching papers [28, 32] sample the flow matching timestep
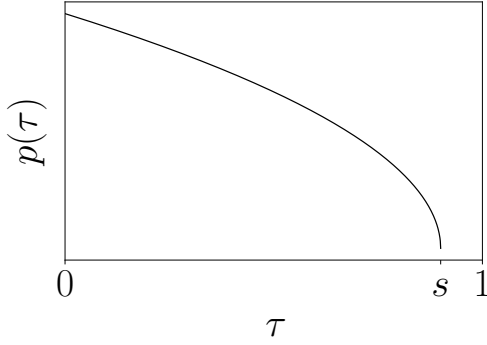
Fig. 14: **Flow matching timestep sampling distribution**. We sample $\tau$ from a shifted beta distribution that emphasizes lower timesteps (corresponding to noisier actions), and does not sample timesteps at all above a cutoff value $s$. We use $s = 0.999$ in our experiments.

from a uniform distribution: $\tau \sim \mathcal{U}(0, 1)$. Esser et al. [14] instead propose sampling from a logit-normal distribution that emphasizes the middle timesteps; the authors posit that at high timesteps (low noise levels), the model needs only to learn the identity function, and at low timesteps (high noise levels), the model needs only to learn the mean of the data distribution. However, we hypothesize that the task of action prediction is subtly different from high-resolution image synthesis — while it may be relatively easy to predict the mean image conditioned on a text label, predicting the mean action conditioned on a robot observation (i.e., learning $\mathbb{E}[\mathbf{A}_t | \mathbf{o}_t]$) is a much harder problem; this is because the observation $\mathbf{o}_t$ is very *informative* in that it should constrain the distribution of possible actions much more than a text label constrains the distribution of possible images. As a result, we designed a timestep sampling distribution that emphasizes low timesteps (high noise levels); additionally, timesteps above a given threshold $s$ are not sampled at all, since they are not needed so long as the integration step $\delta$ is greater than $1-s$. The distribution is given by $p(\tau) = \text{Beta}(\frac{s-\tau}{s}; 1.5, 1)$ and is visualized in Figure 14. We use $s = 0.999$ in our experiments, which allows for $\delta > \frac{1}{1000}$, or up to 1,000 integration steps.

### C. Non-VLM Baseline Architecture

Our baseline architecture $\pi_0$-small is *not* based on a VLM backbone. Hence, we use it to evaluate the benefits of VLM-pre-training. We design it to be sufficiently expressive to fit our large dataset while still providing good performance when trained from scratch. This model has about 470M parameters, and differs from our main model in the following ways: (1) We use DistilBERT [42] to encode the language tokens of the language command $\ell_t$, since this model does not use a language model backbone; (2) The action expert cross-attends to the outputs of the observation encoder, akin to a traditional encoder-decoder transformer [51], rather than our main model which is more like a decoder-only mixture of experts [45]; (3) The images are encoded with a smaller pre-trained ViT

encoder (specifically, the R26-S-32 ResNet-ViT hybrid from Steiner et al. [47]); (4) The ViT image encoders do not share weights; (5) The transformer backbone that encodes the observations (which comes after the ViT image encoders) is not pre-trained on Internet data; (6) The action expert uses the DiT architecture [36] rather than the Gemma architecture, and hence incorporates the flow-matching timestep $\tau$ using AdaLN-Zero layers. Besides this, the models are broadly similar: both use pre-trained ViT image encoders, both use separate weights for the observation encoder and the action expert, both take in the same observation format, and both perform 10 steps of flow matching to predict the action chunk.

### D. Inference

Recall that our model takes an observation $\mathbf{o}_t = [\mathbf{I}_t^1, ..., \mathbf{I}_t^n, \ell_t, \mathbf{q}_t]$ and the noisy actions $\mathbf{A}_t^\tau$ and outputs the vector field that needs to be integrated to obtain the next flow matching step, $\mathbf{v}_t^\tau$. Each time we predict a new action chunk $\mathbf{A}_t$, we must encode each of the images $\mathbf{I}_t^1, ..., \mathbf{I}_t^n$, run a forward pass on the tokens corresponding to $\mathbf{o}_t$, and then run 10 steps of flow matching, where each step requires running a forward pass on the tokens corresponding to $\mathbf{A}_t^\tau$ (the keys and values corresponding to $\mathbf{o}_t$ are cached). Table I summarizes the computation time for this operation with 3 camera images. The operations were timed on an NVIDIA GeForce RTX 4090 consumer-grade GPU. For the mobile robot, inference was done off-board over a Wi-Fi connection, adding a small amount of network latency. Further optimizations, quantization, and other improvements might further reduce inference times.

Since the model generates an entire $H$-step action chunk at once, we can execute up to $H$ actions before we need to run inference again. However, we may run inference more often than that, as well as combine actions from different inference calls using various aggregation strategies. We tried temporal ensembling [57] early on and found that it hurt policy performance, so we opted not to aggregate actions and instead execute action chunks open-loop. For the 20Hz UR5e and Franka robots, we run inference every 0.8 seconds (after executing 16 actions), and for all other robots, which run at 50Hz, we run inference every 0.5 seconds (after executing 25 actions).

| model part | inference time |
|---|---|
| image encoders | 14 ms |
| observation forward pass | 32 ms |
| x10 action forward pass (flow) | 27 ms |
| network latency (if off-board) | 13 ms |
| total on-board inference | 73 ms |
| total off-board inference | 86 ms |

TABLE I: Inference time of our model on an NVIDIA GeForce RTX 4090 GPU.

### E. Evaluation Details

For each task, we design a score rubric that measures progress on the task, and use this for our quantitative results.

We describe this rubric for each task below:

**A. Evaluating the base model**

**Shirt folding:** Shirt folding is recorded as either success or failure. We begin each shirt folding eval by laying the shirt flat on the table. Success is defined as having folded in the sleeves and performed one half-fold along the length of the shirt. Our eval includes 4 small t-shirts and 1 medium t-shirt. We run 2 evals for each item for a maximum of 15000 steps or approximately 5 minutes each.

**Bussing easy:** This task is scored out of 7, where there are 7 different objects on the table, and 1 point is given for each correctly sorted object.

**Bussing hard:** This task is scored out of 12, where there are 12 different objects on the table, and 1 point is given for each correctly sorted object. This version of the task includes particularly challenging settings, like a chopstick on top of a piece of trash.

**Grocery bagging:** This task is scored out of 7. For each 7 grocery items, a point is given for putting it in the bag.

**Toast out of toaster:** This task is scored out of 4. For each piece of toast, 1 point is given for picking it from the toaster and another for putting it on the plate.

**B. Language instruction following.** The policy is scored on successfully repositioning each object and whether it follows instructions.

**Bussing:** The robot has to follow the command to pick up the correct object and place each of them into the correct receptacle. The robot receives 12 objects in total and around 30 instructions in one episode.

**Table setting:** The robot arranges all dishes, utensils, and napkins and makes adjustments according to language specification. The robot receives 7 objects in total and around 20 instructions in one episode.

**Grocery bagging:** The robot picks up the correct item (among bag of coffee beans, bag of barley, bag of marshmallow, cat food, spaghetti, bag of seaweed, bag of almonds), and bags them into a paper bag. The robot receives 7 objects in total and around 14 instructions in one episode.

**C. Learning new dexterous tasks**

**Stack bowls:** This task is scored out of 3. One point for each of two bowls stacked in larger bowls, and one for the neatness of the final product.

**Towel folding:** This task is scored out of 3. One point for the first half-fold of the towel, one point for the second half-fold of the towel, and one point for neatness of the final product.

**Tupperware in microwave:** This task is scored out of 4. One point for opening the microwave, one point for picking up the Tupperware, one point for putting the Tupperware in the microwave, and one point for closing the microwave.

**Paper towel replacement:** This task is scored out of 4. One point is given for grasping the old roll, and another point is given for removing it. Then, one point is given for grasping the new paper towel roll, and the final point is given for placing it on the dispenser.

**Items in drawer:** This task is scored out of 5. One point for opening the drawer, one point for each of 3 items picked and placed into the drawer, and one point for closing the drawer.

**D. Mastering complex multi-stage tasks**

**Laundry folding:** This task is scored out of 4. Our evaluation includes five items, three shirts of size M, L, and XL and two shorts of size 28 and 36. We perform two trials for each item, and the items left to be evaluated start randomly crumpled in a laundry bin (while previously evaluated items start in a folded stack). One point is given for picking an item out of the bin and putting it on the table. Another point is given for flattening the shirt or shorts. A third point is granted for folding the shirt or shorts. A final point is given for either placing the item in the corner of the table (if it is the first item evaluated), or stacking it onto an existing stack of folded clothes. We run each eval for a maximum of 15000 steps or approximately 5 minutes.

**Mobile laundry:** This evaluation follows the same protocol as laundry folding. The three shirts are sized M, M, and XL, and the shorts are sized 32 and 31 W.

**Table bussing:** This task is scored out of 12, where there are 12 different objects on the table, and 1 point is given for each correctly sorted object. This version of the task includes particularly challenging settings, like a chopstick on top of a piece of trash.

**Box building:** This task is scored out of 5. One point is given for successfully picking up the box to begin the task. One point is given for folding the box in half, so the flaps can be closed. One point is given for closing the right flap. One point is given for closing the left flap. The final point is given for neatly centering the final product.

**Packing eggs:** This task is scored out of 7. One point for each egg placed in the correct slot in the carton, and one point for closing the lid.

**Packing food:** This task is scored out of 5. One point for picking up the plate of food, one point for each of 3 food items placed in the to-go box, and one point for closing the to-go box.

**Dryer unloading:** This task involves having the robot approach a dyer with a laundry basket and unload the clothes into the basket. We score this eval out of five, where one point is given for properly approaching the dryer. Another for placing the laundry basket on the stool. A third for opening the dryer. A fourth for putting all the clothes in the basket and a fifth point for closing the dryer. We eval with 3 shirts and 2 shorts that start in a random configuration inside the dryer.