PLEX: Making the Most of the Available Data for Robotic Manipulation Pretraining

Anonymous Author(s)
Affiliation
Address
email

Abstract: A rich representation is key to general robotic manipulation, but existing approaches to learn one require a lot of multimodal demonstrations. In this work we propose PLEX, a transformer-based architecture that learns from a small amount of *task-agnostic visuomotor* trajectories accompanied by a much larger amount of *task-conditioned* object manipulation *videos* —- a type of data that is available in quantity. The key insight behind PLEX is that visuomotor trajectories help induce a latent feature space and train a robot to execute task-agnostic manipulation routines, while diverse video-only demonstrations can efficiently teach the robot how to plan in this feature space for a wide variety of tasks. In contrast to most works on robotic manipulation pretraining, PLEX learns a generalizable sensorimotor multi-task policy, not just an observational representation. We also show that using relative positional encoding in PLEX's transformers greatly helps PLEX in low-data regimes when learning from human-collected demonstrations. Experiments showcase PLEX's generalization on Meta-World benchmark and SOTA performance in challenging Robosuite environments.

Keywords: Robot learning, Robotic manipulation, Visuomotor representations

1 Introduction

The use of transformers [1] has lead to breakthroughs in training large-scale general-purpose representations for computer vision (CV) and natural language processing (NLP) [2], enabling zero-shot adaptation and fast finetuning [3]. At the same time, despite impressive progress, transformer-based representations haven't shown the same versatility for robotic manipulation. Some attribute this gap to the lack of suitable training data for robotics [3]. We argue instead that data relevant to training robotic manipulation models is copious but has important structure that most existing training methods ignore and fail to leverage. These insights lead us to propose a novel transformer-based model, called *PLEX*, that is capable of zero-shot adaptation and effective finetuning thanks to being tailored to the realities of robotic manipulation data.

We observe that robotics-relevant data falls into three major categories. **(1)** The most plentiful category is comprised of "in-the-wild" video datasets [4–6]. Some of them, e.g., Epic Kitchens [7, 8], focus on object manipulation. In aggregation, these datasets cover an immense variety of tasks and are typically annotated with activity descriptions but contain no explicit action information for a robotic to mimic. **(2)** The second category consists of matching sequences of percepts and actions. In some datasets of this type, these sequences don't correspond to meaningful task and are generated by a scripted exploration policy [9]. In others, they come from well-defined tasks, but even in the largest such datasets, e.g., [10], the task coverage is modest compared to video-only data such as [6]. Nonetheless, these sensorimotor sequences capture valuable correlations between a robot's actions and changes in the environment. **(3)** The third data category, the scarcest one, consists of high-quality demonstrations for a target task in a target environment. Thus, a scalable model architecture

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

for robotic manipulation must be able to learn primarily from videos, while being extra data-efficient on sensorimotor training sequences and the available demonstrations from the target environment.

PLEX, the ***PL***anning-***EX***ecution architecture we propose, achieves exactly that. A PLEX model has two major transformer-based components: **(I)** a task-conditioned observational *planner* that, given a task specification and an estimate of the current world state, determines the next state to which the robot should attempt to transition, and **(II)** an *executor* that, having received the desired next state from the planner, produces an action that should lead there from the current state. The executor is trained by optimizing an inverse dynamics loss over exploratory sensorimotor data of the aforementioned category **(2)**, while the planner is trained by minimizing a loss of its autoregressive predictions computed with respect to video-only trajectories of category **(1)**. The target-task data of category **(3)** can be optionally used to efficiently finetune the planner, the executor, or both.

We make three design choices that greatly help the data efficiency of PLEX's training:

- *Learning to plan in the observation embedding space.* Rather than generating videos of proposed task execution using, e.g., stable diffusion as in Du et al. [11], PLEX learns to plan and execute in the low-dimensional space of observation embeddings.
- *Asymmetric learning of the embedding space.* The space in which the executor and the planner operate is induced either by a pretrained frozen feature-rich observation encoder, such as R3M [12], or by the executor's training loss *only*. Keeping the planner's gradients from entering observation encoders makes training significantly cheaper and removes the risk of latent space collapse.
- *Relative positional encodings.* We adopt the relative positional encodings [13] in PLEX. We empirically show that in robotic manipulation the relative positional encodings significantly improve training efficiency from human-collected data compared with the *absolute* positional encodings [1] commonly used in the literature on transformers.

Most approaches that use video-only demonstrations for pretraining in robotic manipulation produce purely visual representations (see, e.g., [12, 14–16]). The majority of algorithms that produce sensorimotor models need most or all of the video demonstrations to be accompanied by action sequences that generated the videos, a requirement that holds only for a small fraction available manipulation data [10, 17–22]. Few approaches have a dedicated trainable planning component; e.g. [21, 23–26] plan in a skill space, which PLEX can be modified to do as well. Conceptually, PLEX falls under the paradigm of learning from observations (LfO), but existing LfO approaches don't have multitask zero-shot planning capability [27–30] or demostrate it only in low-dimensional environments across similar tasks [31]. Of the works that have used transformers for robotic manipulation [20, 22, 26, 32, 33], only Brohan et al. [22] have analyzed their data efficiency, and none have looked at positional embeddings as a way to improve it. Overall, the closest approach to PLEX is the concurrently proposed UniPi [11]. It also has counterparts of PLEX's planner and executor, but its planner operates using diffusion *in the image space* [5], which is expensive both datawise and computationally, and may fail to model manipulation-relevant 3D object structure consistently [5]. A more extensive discussion of prior work is provided in Appendix A.

We experimentally show that PLEX's planner-executor design can effectively exploit the structure of realistically available robotic manipulation data to achieve efficient learning. On the multi-task Meta-World [34] benchmark, despite pretraining mostly on video data, PLEX exhibits strong zero-shot performance on unseen tasks and can be further improved by finetuning on a small amount of video-only demonstrations. We empirically show on the challenging Robosuite/Robomimic [35, 36] benchmark that, contrary to conclusions from NLP [13], the use of relative positional encodings significantly improves the data efficiency of PLEX learning from human-collected demonstrations.

2 Problem statement and relevant concepts

2.1 Problem statement

We consider the problem of learning a generalist task-conditioned policy for goal-directed object manipulation. Namely, we seek a policy that can control a robotic manipulator to successfully

accomplish tasks that the robot may not have encountered during the policy training process; such a policy formally can be viewed a solution to a task-conditioned partially observable Markov decision process (POMDP) described in Appendix B. In practice, learning a generalist policy that performs well on a broad distribution of tasks zero-shot is very challenging, as the coverage and amount of publicly available training data are limited. Therefore, in this work we consider a two-phased learning process: (1) pretraining, during which a generalist policy is trained, and (2) finetuning, during which this policy is adapted to a target task.

2.2 Data for training robotic manipulation models

We consider three broad groups of datasets relevant to training robotic manipulation systems:[1]

Multi-task video demonstrations ($\mathcal{D}_{\mathbf{mtvd}}$). Being the most abundant category, it comprises data collections ranging from general YouTube videos to curated benchmarks such as Ego4D [6], Epic Kitchens [7, 8], and YouTube-8M [4] showing *an* agent – either a robot or a person – performing a meaningful object manipulation task with an end-effector. This data contains demonstration-quality sequences of video observations and descriptions of tasks they accomplish, but not the action sequences whose execution generated these videos.

Visuomotor trajectories ($\mathcal{D}_{\mathbf{vmt}}$). These trajectories consist of paired sequences of observations and robots' actions. Although some of them may be high-quality demonstrations of specific tasks, e.g., as in the Bridge Dataset [10], many of these trajectories are generated by activities that most people will not find meaningful, e.g., grabbing random objects in a tray, as in the RoboNet [9]. Since no strong quality, quantity, or task association requirements are imposed on $\mathcal{D}_{\text{vmt}}$ data, it is relatively easy to collect for any target embodiment and environment.

Target-task demonstrations ($\mathcal{D}_{\mathbf{ttd}}$). This is the most scarce but also most desirable data category, since it encompasses high-quality trajectories for a specific task in question, ideally collected on the target embodiment (robot). Note, however that we don't require that these demonstrations be visuomotor. In fact, our experiments show that PLEX needs only video demonstrations for finetuning to learn a high-quality policy for a target task.

A key data assumption we make in this work is that $|\mathcal{D}_{\text{ttd}}| \ll |\mathcal{D}_{\text{vmt}}| \ll |\mathcal{D}_{\text{mtvd}}|$.

2.3 Transformers and positional encodings

A transformer-based architecture consists of several specially structured *self-attention layers* and, in general, maps an input *set* (often called a *context*) of K elements (called *tokens*) to an output of the same size K [1]. In most applications, such as language translation, transformers need to map *ordered* sets (i.e. sequences) to other ordered sets, and therefore add special vectors called *positional encodings* to each input element to identify its position in a sequence. These encodings can be learned as part of transformer's training or be hand-crafted.

The most common scheme is the *absolute positional encoding*, where each position in the transformer's K-sized context gets a positional vector [1]. Some transformers, e.g., Chen et al. [37], use what we call a *global positional encoding*. It is similar to the absolute one, but assigns a separate vector to each position *in the entire input sequence* rather than just the K-sized context, up to some maximum length $T \gg K$. Finally, models based on Transformer-XL [13, 20, 22], instead condition the attention computation on the *relative* positions between different pairs of input tokens within a context. In this work, we argue that on robotic manipulation finetuning datasets that consist of small numbers of human-gathered demonstrations, relative positional encoding is significantly more data-efficient than absolute or global one.

3 PLEX architecture and training

3.1 Intuition

[1] Static image datasets, e.g., ImageNet, aren't treated by PLEX in a special way and we don't discuss it here, but can be used to pretrain PLEX's image encoders.

PLEX (shown in Figure 1) separates the model into two transformer-based submodules: *1)* a *planner* that plans in the observation embedding space based on a task specification, and *2)* an *executor* that takes the embeddings of the historical and the planned future observations and outputs an action to control the robot. This design is motivated by structures of $\mathcal{D}_{\text{mtvd}}$, $\mathcal{D}_{\text{vmt}}$, and $\mathcal{D}_{\text{ttd}}$ dataset categories, which as we explain below make them suitable for three complementary learning objectives.

1. **Learning to execute state transitions.** The visuomotor trajectories from $\mathcal{D}_{\text{vmt}}$, collected on the target robotic manipulator or a similar one, show the robot how to execute a wide variety of state transitions. By sampling an observation-action tuple $\langle o_{t-H}, \ldots, o_t, a_t, o_{t+L} \rangle$, the agent can learn to infer a_t from $o_{t-H}, \ldots, o_t$, and o_{t+L} using *inverse dynamics*, where t is the current time step, H is an observation history length, and L is a lookahead parameter.

2. **Learning to plan for tasks.** In order to recommend a meaningful action at each step, inverse dynamics inference needs the (embedding of) the desired future observation. Determining the desired future observation *given a task description* is something that can be learned from multi-task video-only data $\mathcal{D}_{\text{mtvd}}$, since this data shows what progress towards a successful completion of a specified task should look like.

3. **Improving target-task performance.** While learning to plan and execute on diverse $\mathcal{D}_{\text{mtvd}}$ and $\mathcal{D}_{\text{vmt}}$ data can result in a robotic manipulation foundation model [3] with strong zero-shot performance, on many tasks it may be far from perfect. Small datasets $\mathcal{D}_{\text{ttd}}$ of high-quality target-task demonstrations (e.g., through teleoperation) can provide additional grounding to the target domain to further improve a pretrained model.

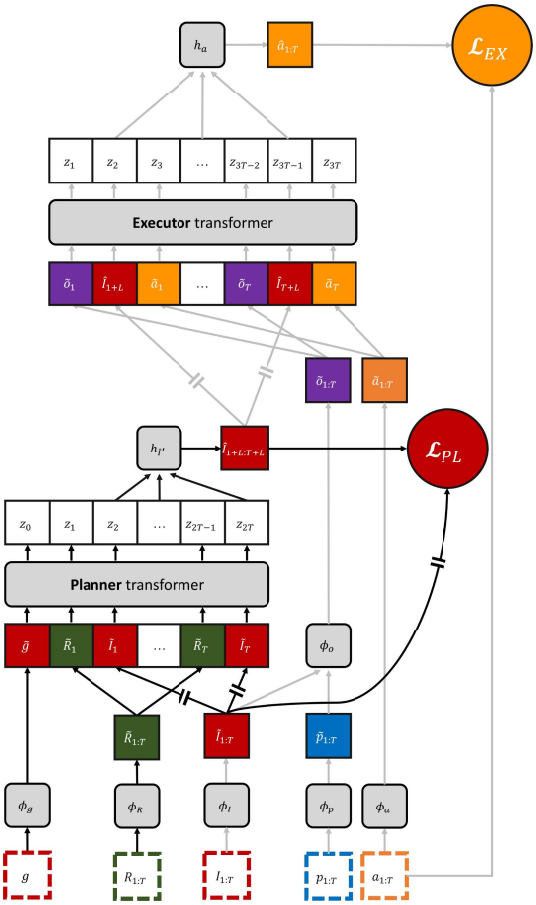

Figure 1: **PLEX architecture.** PLEX is optimized using the planner's loss $\mathcal{L}_{PL}$ (computation shown with black arrows ↑), and the executor's loss $\mathcal{L}_{EX}$ (computation shown with gray arrows ↑). The symbols '=' and '=' denote stopgrads, where backpropagation is halted. Each input modality m is embedded using a modality-specific encoder ϕ_m. Video demonstration embeddings $\tilde{g}, \tilde{I}_{1:T}$, and (optionally) $\tilde{R}_{1:T}$ are used to train the planner with the prediction loss $\mathcal{L}_{PL}$ in the embedding space. Visuomotor trajectory embeddings $\tilde{I}_{1:T}, \tilde{p}_{1:T}, \tilde{a}_{1:T}$ are passed to the executor to compute the inverse dynamics loss $\mathcal{L}_{EX}$. Note that if the image encoder ϕ_I isn't frozen, $\mathcal{L}_{EX}$'s gradients will update ϕ_I. In contrast, the planner's own loss $\mathcal{L}_{PL}$ never affects ϕ_I (see stopgrad symbol =).

3.2 Architecture

Following the above intuitions, we train PLEX's executor using data $\mathcal{D}_{\text{vmt}}$ and PLEX's planner using data $\mathcal{D}_{\text{mtvd}}$, in addition to a small dataset $\mathcal{D}_{\text{ttd}}$ of target-task trajectories (which, if available, can be used to train both the planner and executor). Specifically, let $\tau = g, R_1, I_1, p_1, a_1 \ldots, R_T, I_T, p_T, a_T = g, R_{1:T}, I_{1:T}, p_{1:T}, a_{1:T}$ denote a trajectory, where g is a task specification, I_t is a tuple of camera image observations, p_t is a proprioceptive state, a_t is an action, and R_t is a return-to-go at time t; the length T can vary across trajectories. As Figure 1 shows, PLEX processes these input modalities using corresponding encoders ϕ_g, ϕ_I, ϕ_p, ϕ_a, and ϕ_R to obtain an embedded sequence $\tilde{g}, \tilde{R}_{1:T}, \tilde{I}_{1:T}, \tilde{p}_{1:T}, \tilde{a}_{1:T}$. When a modality is missing, it is replaced by trainable placeholder vectors during embedding. Missing modalities are common in robotic manipulation datasets; e.g., few datasets have rewards. Since PLEX's executor and planner

are designed to be trainable on task-agnostic visuomotor $\mathcal{D}_{\text{vmt}}$ data and task-conditioned video-only demonstrations $\mathcal{D}_{\text{mtvd}}$, respectively, each of these components is specialized to operate only on the (embeddings of) modalities available in their prevalent training data. In particular, per Figure 1, task description and return embeddings $\tilde{g}$ and $\tilde{R}_{1:T}$ don't get routed to the executor, since they are missing from $\mathcal{D}_{\text{vmt}}$ data. Similarly, the planner only receives $\tilde{g}$, $\tilde{I}_{1:T}$ and, optionally, $\tilde{R}_{1:T}$ embeddings, since they are present in $\mathcal{D}_{\text{mtvd}}$ data. This separation holds even at the deployment time, when the trajectory generated by the robot has all modalities.

Planner The planner's sole purpose is to determine *where* the agent should go in the observation embedding space. As shown in Figure 1, given embeddings $\tilde{g}, \tilde{I}_{1:T}$ of a task-conditioned video-only training demonstration, the planner outputs a sequence $\hat{I}_{L:T+L}$ of embeddings corresponding to the observations the agent should ideally see L steps in the future from its current time step; L is a hyperparameter. The planner's training minimizes the prediction loss

$$\mathcal{L}_{PL}(\tilde{g}, \tilde{R}_{1:T}, \tilde{I}_{1:T}) = \sum_{t=1+L}^{T+L} \|\tilde{I}_t - \hat{I}_t\|_2^2. \tag{1}$$

where we set $\tilde{I}_t = \tilde{I}_T$ for $t = T+1, ..., T+L$.

Crucially, $\mathcal{L}_{PL}$'s gradients *don't backpropagate* into the encoders ϕ_g and ϕ_I, to prevent representation collapse of the image embedding space (denoted as $\mathscr{E}_o$); note the stopgrad symbols on $\mathcal{L}_{PL}$'s computation paths in Figure 1. The embedding space $\mathscr{E}_o$ either comes from pretrained and frozen encoders or is learned along with inverse dynamics during executor training.

Executor Like the planner, the executor has a specific role at the deployment time. Given the observation-action sequence $o_{1:t}, a_{1:t}$ so far, as well as the target observation embedding $\hat{I}_{t+L}$ produced by the planner, the executor infers an action $\hat{a}_t$ for the current step. The executor should do this in a task-agnostic way, as the knowledge of the task is already incorporated by the planner into the $\hat{I}_{t+L}$ prediction. For an input trajectory from $\mathcal{D}_{\text{vmt}}$, we optimize the executor via the inverse dynamics loss

$$\mathcal{L}_{EX}(I_{1:T}, p_{1:T}, \hat{I}_{1+L:T+L}, a_{1:T},) = \sum_{t=1}^{T-1} \|a_t - \hat{a}_t\|_2^2 \tag{2}$$

A major difference between $\mathcal{L}_{EX}$ and $\mathcal{L}_{PL}$ optimization is that the former's gradients can backpropagate into the encoders ϕ_I, ϕ_o, ϕ_p, and ϕ_a: the computation path for $\mathcal{L}_{EX}$ through these encoders in Figure 1 doesn't have a stopgrad. This allows executor training to shape the embedding space $\mathscr{E}_o$.

Relative positional encoding Like the Decision Transformer (DT) [37], PLEX's planner and executor transformers are derived from GPT-2. However, DT's use of global positional encoding implicitly assumes that all training trajectories have the same length T. PLEX, in contrast, uses relative encoding from Dai et al. [13] as the default. As we show empirically, in robotic manipulation settings where tasks are usually goal-oriented and training demonstrations vary a lot in length, global positional embedding performs poorly and even the fixed absolute positional encoding common in NLP [1] performs much better. Especially, for human-collected demonstrations where variability is significant, our experimental results show that relative encoding [13] perform significantly better.

3.3 Training PLEX

Training PLEX generally involves both pretraining and finetuning, though the experiments in Section 4.2 show that pretraining alone already gives PLEX solid zero-shot performance.

Pretraining PLEX consists of two sub-stages:

1. Pretraining the executor by optimizing the $\mathcal{L}_{EX}$ loss (Equation (2)) over a $\mathcal{D}_{\text{vmt}}$ dataset.

2. Pretraining the planner by optimizing the $\mathcal{L}_{PL}$ loss (Equation (1)) over a $\mathcal{D}_{\text{mtvd}}$ dataset.

If the observation encoders are expected to be trained or finetuned by the inverse dynamics loss $\mathcal{L}_{EX}$, rather than pretrained and frozen beforehand, it is critical for executor pretraining to be done before training the planner. Indeed, the planner is expected to make predictions in the observation encoders' embedding space, which will change if the inverse dynamics loss affects the encoders. If the encoders are frozen from the start, however, the pretraining stages can proceed asynchronously.

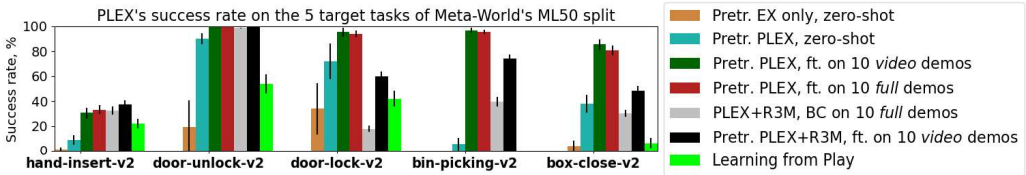

Figure 2: PLEX's generalization experiments. The confidence intervals are computed with 10 seeds.

Finetuning involves adapting PLEX using a target-task demonstration dataset $\mathcal{D}_{\text{ttd}}$. As with any finetuning, this involves deciding which part of PLEX to adapt.

Since $\mathcal{D}_{\text{ttd}}$ can be viewed both as a small $\mathcal{D}_{\text{mtvd}}$ and a small $\mathcal{D}_{\text{vmt}}$ dataset, it can be used to train any component of PLEX—executor, planner, and observation encoders. As with pretraining, if $\mathcal{D}_{\text{ttd}}$ is used for finetuning the encoders, it is critical to complete their finetuning before finetuning the planner. In Section 4.2, we show that finetuning just the last layer of the planner's transformer, which constitutes 5% of the parameters of the PLEX instance in the experiment, is sufficient for significantly boosting a pretrained PLEX's performance.

$\mathcal{D}_{\text{ttd}}$ can also be employed for optimizing a behavior cloning loss $\mathcal{L}_{BC}$, which is equivalent to a stopgrad-free version of $\mathcal{L}_{EX}$, whose gradients are allowed to backpropagate through the entire PLEX model, including the planner and/or the encoders. The experiments in Section 4.3 demonstrate the efficiency of BC-based finetuning thanks to the use of a relative position encoding.

4 Experiments

We conduct two sets of experiments to answer the following questions: ***(i)*** *Does PLEX pretrained on task-agnostic sensorimotor data and task-annotated video data generalize well to downstream tasks?* ***(ii)*** *How does the use of relative positional encodings affect PLEX's policy quality?* Appendix C provides the details about our PLEX implementation.[2]

4.1 Benchmarks and training data

Meta-World: Meta-World [34] is a collection of 50 tasks featuring a Sawyer arm. We use Meta-World-v2 with image observations (see details in Appendix D.1). We consider the ML50 split consisting of 45 training and 5 target tasks (***door-lock***, ***door-unlock***, ***hand-insert***, ***bin-picking***, and ***box-close***). We use these 5 target tasks for evaluation. Meta-World comes with high-quality scripted policies for all tasks. To get ***video demonstration data*** ($\mathcal{D}_{\text{mtvd}}$), we use these scripted policies to generate 100 successful video-only demonstrations for each of the 45 training tasks, i.e., $|\mathcal{D}_{\text{mtvd}}| = 4500$. To generate ***visuomotor trajectories*** ($\mathcal{D}_{\text{vmt}}$), for the 5 target tasks' environments, we add zero-mean Gaussian noise with standard deviation 0.5 to the actions of the scripted policies and record the altered actions. We collect 50 trajectories per task, i.e., $|\mathcal{D}_{\text{vmt}}| = 250$. Finally, for ***target-task demonstrations*** ($\mathcal{D}_{\text{ttd}}$), we employ the original scripted policies to produce 75 demonstrations per target task and sample 10 of them in a finetuning experiment run, i.e., $|\mathcal{D}_{\text{ttd}}| = 10$.

Robosuite: Robosuite benchmark [35], compared Meta-World, has robotic manipulation tasks with a significantly more complicated dynamics and action space. We use 9 of its tasks involving a single robot arm (Panda) (***Lift***, ***Stack***, ***Door***, ***NutAssemblyRound***, ***NutAssemblySquare***, ***PickPlace-Bread PickPlaceCan***, ***PickPlaceMilk***, and ***PickPlaceCereal***). Robosuite's details are provided in Appendix D.1. Importantly, the training data for Robosuite was collected from human demonstrations, *not* generated by scripted policies as in Meta-World. See Appendix D.4 for details.

4.2 Generalization experiments

In these experiments, we focus on pretraining PLEX with multi-task Meta-World data. The results are shown in Figure 2. We train a 16,639,149-parameter PLEX instance (including the ResNet-18-based image encoder) from scratch with random initialization. We use the success rate on the 5 target

[2]We implement PLEX using the GPT-2 of the DT codebase [37] but without return conditioning.

tasks as the performance metric. For baselines, we experiment with PLEX with a frozen ResNet-50-based R3M [12], an observational representation pretrained on the large Ego4D dataset [6]. We denote it as *PLEX +R3M*. In addition, we use an adapted *Learning from Play* (LfP) approach [17]. The hyperparameters and details can found Appendices C and D. In summary, the experimental results show that PLEX can perform well without seeing a single sensorimotor expert demonstration during pretraining and finetuning.

PLEX demonstrates strong zero-shot performance We show that PLEX pretrained on as few as 4500 video demonstrations ($\mathcal{D}_{\text{mtvd}}$) from the training environments and 250 dynamics trajectories ($\mathcal{D}_{\text{vmt}}$) from the target environments (denoted as *Pretr. PLEX, zero-shot* in Figure 2) exhibits good downstream performance *zero-shot*. To show this performance is really due to planning learned from video-only data as opposed to the executor inadvertently exploiting biases in the data, we consider a PLEX variation (denoted as *Pretr. EX only, zero-shot*), where we only pretrain the *executor* (on $\mathcal{D}_{\text{vmt}}$) not the planner.[3] The results of *Pretr. EX only, zero-shot* shows the baseline performance one can get with knowledge in the dynamics data $\mathcal{D}_{\text{vmt}}$ alone. We see *Pretr. EX only, zero-shot* underperforms *Pretr. PLEX, zero-shot*, which shows the importance of learning from $\mathcal{D}_{\text{mtvd}}$ via the PLEX's planner. As a baseline, we also report *Learning from Play* performance, which is in the middle of the two PLEX models above. This is expected: LfP has planning capability and therefore outmatches *Pretr. EX only, zero-shot*; however, LfP doesn't use either the video-only data $\mathcal{D}_{\text{mtvd}}$ or the target-task demonstrations $\mathcal{D}_{\text{ttd}}$, the ability to utilize which gives PLEX a large advantage.

PLEX can fast adapt to samll video-only demonstrations We further show that finetuning only 5% of PLEX's parameters (the last transformer layer of the planner) on just 10 *video-only* demonstrations for a given task significantly boosts PLEX's success rate there. For all 5 downstream tasks, this policy outperforms *Pretr. EX only, zero-shot* by $\geq 2\times$. The improvement is drastic especially in the case of *hand-insert-v2*, *bin-picking-v2*, and *box-close-v2*.

Video-only demonstrations is all PLEX needs during finetuning Interestingly, we find that full demonstrations (with both video and action sequences) don't increase PLEX's performance beyond video-only ones. This can seen from the experimental results of *Pretr. PLEX, ft. on 10 full demos*, where we finetune PLEX (the action head and last transformer layer of PLEX's planner, executor; $\approx 11\%$ of PLEX) on 10 *full* (sensorimotor) demonstrations for each task. We think this is due to PLEX's image encoder being pretrained only on observations from $\mathcal{D}_{\text{vmt}}$ and frozen during finetuning. Because of this, finetuning couldn't help the encoder learn any extra features for modeling inverse dynamics *over the observation space region covered by* $\mathcal{D}_{ttd}$, even if such features would improve PLEX's performance.

The issue of impoverished observation coverage in $\mathcal{D}_{\text{vmt}}$ dataset can be addressed by using a frozen encoder pretrained on an independent large dataset, as the results of single-task *PLEX +R3M, BC* and of *pretrained PLEX +R3M* in Figure 2 suggest. There, PLEX's R3M encoder was never trained on *any* Meta-World observations but enables PLEX to perform reasonably well. This experiment also shows two other aspects of using observation-only representations like R3M: (1) The sensorimotor representation that PLEX learns *on top of* R3M clearly helps generalization — *pretrained PLEX +R3M* performs much better than the single-task one, despite seeing just video-only demonstrations at finetuning. (2) Fully frozen somewhat R3M limits PLEX's performance – PLEX that pretrained its own encoder outperforms PLEX +R3M on 3 of 5 tasks.

4.3 Positional encoding experiments

In the Meta-World experiments, all training data was generated by scripted policies. In real settings, most such data is generated by people teleoperating robots or performing various tasks themselves. A hallmark of human-generated datasets compared to script-generated ones is the demonstration variability in the former: even trajectories for the same task originating in the same state tend to be different. In this section, we show that in low-data regimes typical of finetuning on human-generated

[3] At run time we feed the embedding of the task's *goal* image as the predictions that the *executor* conditions on (since no planner is trained).

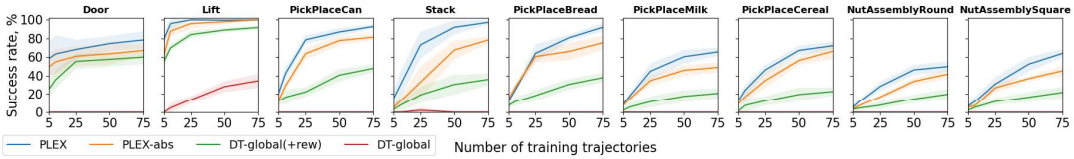

Figure 3: Data efficiency of PLEX's relative positional encoding in single-task mode on Robosuite's single-arm tasks with $|\mathcal{D}_{\text{ttd}}|$ varying from 5 to 75. **PLEX** (with relative encodings) in most cases significantly outperforms and at worst matches the performance of its version **PLEX-abs** with absolute positional encodings. Both versions significantly outperform DT.

demonstrations, PLEX with relative positional encoding yields superior policies for a given amount of training data than using absolute encoding. The results are in Figure 3.

Baselines, training and evaluation protocol. To analyze data efficiency and compare to prior results on Robosuite, we focus on an extreme variant of finetuning – training from scratch. For each of the 9 Robosuite tasks and each of the evaluated encodings, we train a separate 36,242,769-parameter PLEX instance using only that task's $\mathcal{D}_{\text{ttd}}$ dataset of full sensorimotor human-generated demonstrations. We compare PLEX with relative positional encoding to PLEX with absolute one and to two flavors of the Decision Transformer (DT) [37], which use global positional embedding. Appendix D.5 and Figure 3 provide more details about model training dataset collection, and the baselines. For each task/dataset size/approach, we train on 10 seeds.

Results. As Figure 3 shows, PLEX learns strong policies using at most 75 demonstrations, despite having to train a 36M-parameter model including randomly initialized vision models for tasks, most of which have complex dynamics and broad initial state distributions. Moreover, PLEX with relative positional encoding (denoted simply as *PLEX* in the legend) outperforms the alternatives by as much as 20 percentage points (pp) on Robosuite's human-generated demonstration data while never losing to them. In particular, *DT-global(+rew)* and, especially, *DT-global* perform far worse of both *PLEX* and *PLEX-abs*. Since all models share most of the implementation and are trained similarly when *PLEX* and *PLEX-abs* run in BC mode, we attribute PLEX's advantage only to the combined effect of using human-generated training data and positional encodings. We have also trained PLEX and PLEX-abs for Meta-World's 5 target tasks from the previous experiment for various amounts of the available – scripted – demonstrations for these tasks and noticed no significant performance difference between PLEX and PLEX-abs on any task. This provides additional evidence that the utility of relative positional enconding manifests itself specifically on human-generated demonstration data.

In fact, relying on relative positional encoding allows PLEX to achieve *state-of-the art* performance on all Robosuite tasks in this experiment, as we show and analyze empirically in Appendix D.4.

5 Conclusion and limitations

We have introduced PLEX, a transformer-based sensorimotor model architecture that can be pretrained on robotic manipulation-relevant data realistically available in quantity. Our experimental results show that PLEX demonstrate strong zero-shot performance and can be effectively finetuned with demonstrations to further boost its performance. In particular, PLEX shows superior performance on human-collected demonstrations because of its usage of relative positional encoding.

Limitations We believe that PLEX has great potential as a model architecture for general robotic manipulation, but in most of our experiments so far, the training data came from the same robot on which the trained model was ultimately deployed. In reality, most available multi-task video demonstration data $\mathcal{D}_{\text{mtvd}}$ is generated by other robots or even people. This can cause a mismatch between the demonstrations and the target robot's capabilities and setups. Planning hierarchically first in the skill space as, e.g., in Lynch et al. [38], and then in the observation embedding space may address this issue. In addition, so far we have trained PLEX on simulated data. Pretraining it on a large or even internet-scale data corpus will be a more conclusive test of PLEX's capabilities. Also, with the rise of powerful LLMs such as Ouyang et al. [39], switching PLEX to language for task specification can help it generalize much better across tasks.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[3] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang. On the opportunities and risks of foundation models. *arXiv*, 2021.

[4] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark, 2016.

[5] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, and T. Salimans. Imagen Video: High definition video generation with diffusion models. *arXiv*, 2022.

[6] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, M. Martin, T. Nagarajan, I. Radosavovic, S. K. Ramakrishnan, F. Ryan, J. Sharma, M. Wray, M. Xu, E. Z. Xu, C. Zhao, S. Bansal, D. Batra, V. Cartillier, S. Crane, T. Do, M. Doulaty, A. Erapalli, C. Feichtenhofer, A. Fragomeni, Q. Fu, A. Gebreselasie, C. Gonzalez, J. Hillis, X. Huang, Y. Huang, W. Jia, W. Khoo, J. Kolar, S. Kottur, A. Kumar, F. Landini, C. Li, Y. Li, Z. Li, K. Mangalam, R. Modhugu, J. Munro, T. Murrell, T. Nishiyasu, W. Price, P. R. Puentes, M. Ramazanova, L. Sari, K. Somasundaram, A. Southerland, Y. Sugano, R. Tao, M. Vo, Y. Wang, X. Wu, T. Yagi, Z. Zhao, Y. Zhu, P. Arbelaez, D. Crandall, D. Damen, G. M. Farinella, C. Fuegen, B. Ghanem, V. K. Ithapu, C. V. Jawahar, H. Joo, K. Kitani, H. Li, R. Newcombe, A. Oliva, H. S. Park, J. M. Rehg, Y. Sato, J. Shi, M. Z. Shou, A. Torralba, L. Torresani, M. Yan, and J. Malik. Ego4d: Around the world in 3,000 hours of egocentric video, 2022.

[7] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[8] D. Damen, H. Doughty, G. M. Farinella, , A. Furnari, J. Ma, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130: 33–55, 2022.

[9] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. In *CoRL*, 2019.

[10] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *RSS*, 2022.

[11] Y. Du, M. Yang, B. Dai, H. Dai, O. Nachum, J. B. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *arXiv*, 2023.

[12] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3M: A universal visual representation for robot manipulation, 2022.

[13] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In A. Korhonen, D. R. Traum, and L. Màrquez, editors, *ACL*, 2019.

[14] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin. Learning to see before learning to act: Visual pre-training for manipulation. In *ICRA*, 2020.

[15] A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. In *RSS*, 2021.

[16] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell. Real world robot learning with masked visual pre-training. In *CoRL*, 2022.

[17] C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data. In *RSS*, 2021.

[18] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002, 2021.

[19] Z. Mandi, F. Liu, K. Lee, and P. Abbeel. Towards more generalizable one-shot visual imitation learning. In *ICRA*, 2022.

[20] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent, 2022.

[21] S. Nasiriany, T. Gao, A. Mandlekar, and Y. Zhu. Learning and retrieval from prior data for skill-based imitation learning. *arXiv preprint arXiv:2210.11435*, 2022.

[22] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. RT-1: Robotics transformer for real-world control at scale. *arXiv*, 2022.

[23] K. Hakhamaneshi, R. Zhao, A. Zhan, P. Abbeel, and M. Laskin. Hierarchical few-shot imitation with skill transition models. *arXiv preprint arXiv:2107.08981*, 2021.

[24] A. Ren, S. Veer, and A. Majumdar. Generalization guarantees for imitation learning. In *Conference on Robot Learning*, pages 1426–1442. PMLR, 2021.

[25] B. Xihan, O. Mendez, and S. Hadfield. Skill-il: Disentangling skill and knowledge in multitask imitation learning. *arXiv preprint arXiv:2205.03130*, 2022.

[26] O. Mees, L. Hermann, and W. Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

[27] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *ICRA*, 2017.

[28] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. In *IROS*, 2021.

[29] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[30] B. Baker, I. Akkaya, P. Zhokhov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In *NeurIPS*, 2022.

[31] H. Xu, L. Jiang, J. Li, and X. Zhan. A policy-guided imitation approach for offline reinforcement learning. In *arXiv*, 2022.

[32] S. Dasari and A. Gupta. Transformers for one-shot visual imitation. In *CoRL*, 2020.

[33] H. Kim, Y. Ohmura, and Y. Kuniyoshi. Transformer-based deep imitation learning for dual-arm robot manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8965–8972. IEEE, 2021.

[34] T. Yu, D. Quillen, Z. He, R. Julian, A. Narayan, H. Shively, A. Bellathur, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

[35] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín. Robosuite: A modular simulation framework and benchmark for robot learning, 2020.

[36] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *CoRL*, 2021.

[37] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021.

[38] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *coRL*, 2019.

[39] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[40] E. Chane-Sane, C. Schmid, and I. Laptev. Learning video-conditioned policies for unseen manipulation tasks. *arXiv preprint arXiv:2305.06289*, 2023.

[41] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023.

[42] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *RSS*, 2018.

[43] A. Zhou, E. Jang, D. Kappler, A. Herzog, M. Khansari, P. Wohlhart, Y. Bai, M. Kalakrishnan, S. Levine, and C. Finn. Watch, try, learn: Meta-learning from demonstrations and reward. In *ICLR*, 2020.

[44] J. Li, T. Lu, X. Cao, Y. Cai, and S. Wang. Meta-imitation learning by watching video demonstrations. In *International Conference on Learning Representations*, 2022.

[45] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017.

[46] Z. Mandi, P. Abbeel, and S. James. On the effectiveness of fine-tuning versus meta-reinforcement learning. *CoRL 2022 Workshop on Pre-training Robot Learning*, 2022.

[47] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. In *CoRL*, 2020.

[48] K. Schmeckpeper, O. Rybkin, K. Daniilidis, S. Levine, and C. Finn. Reinforcement learning with videos: Combining offline observations with interaction. In *CoRL*, 2020.

[49] D. Venuto, S. Yang, P. Abbeel, D. Precup, I. Mordatch, and O. Nachum. Multi-environment pretraining enables transfer to action limited datasets. *arXiv preprint arXiv:2211.13337*, 2022.

[50] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[51] M. Janner, Q. Li, and S. Levine. Reinforcement learning as one big sequence modeling problem, 2021.

[52] J. Shang, K. Kahatapitiya, X. Li, and M. S. Ryoo. Starformer: Transformer with state-action-reward representations for visual reinforcement learning. *arXiv preprint arXiv:2110.06206*, 2021.

[53] R. Loynd, R. Fernandez, A. Celikyilmaz, A. Swaminathan, and M. Hausknecht. Working memory graphs. In *ICML*, 2020.

[54] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, M. Botvinick, N. Heess, and R. Hadsell. Stabilizing transformers for reinforcement learning. In *ICML*, 2020.

[55] C. R. Dance, J. Perez, and T. Cachet. Conditioned reinforcement learning for few-shot imitation. In *International Conference on Machine Learning*, pages 2376–2387. PMLR, 2021.

[56] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *CoRL*, 2022.

[57] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[58] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.