Scaling Proprioceptive-Visual Learning with Heterogeneous Pre-trained Transformers

Lirui Wang[1] **Xinlei Chen**[2] **Jialiang Zhao**[1] **Kaiming He**[1]
[1]MIT CSAIL [2]Meta, FAIR
https://liruiw.github.io/hpt

Abstract

One of the roadblocks for training generalist robotic models today is heterogeneity. Previous robot learning methods often collect data to train with one specific embodiment for one task, which is expensive and prone to overfitting. This work studies the problem of learning policy representations through *heterogeneous pre-training* on robot data across different embodiments and tasks at scale. We propose Heterogeneous Pre-trained Transformers (HPT), which pre-train a large, shareable trunk of a policy neural network to learn a task and embodiment agnostic shared representation. This general architecture aligns the specific proprioception and vision inputs from distinct embodiments to a short sequence of tokens and then processes such tokens to map to control robots for different tasks. Leveraging the recent large-scale multi-embodiment real-world robotic datasets as well as simulation, deployed robots, and human video datasets, we investigate pre-training policies across heterogeneity. We conduct experiments to investigate the scaling behaviors of training objectives, to the extent of 52 datasets. HPTs outperform several baselines and enhance the fine-tuned policy performance by over 20% on unseen tasks in multiple simulator benchmarks and real-world settings.

1 Introduction

Building robotic policies today is hard: it often requires collecting *specific* data for each robot, task, and environment, and the learned policies do not generalize beyond these specific settings. A historical lesson that has revolutionized machine learning is that pre-training [34, 27, 29] on large-scale, high-quality, and diverse data can bring *general* models that usually outperform specific models. Recent progress in open-source large-scale data collection [14, 76] has made this path possible, but the heterogeneity (such as varying robot hardware and different environments) present in large-scale robotic data has posed a significant challenge. A central question for the field now is how to leverage the heterogeneous robot data to pre-train robotic foundation models [3].

Foundation models from natural language processing [60, 58] and computer vision [37] have shown a paradigm to achieve general-purpose task-agnostic models through pre-training on massive amounts and diversity of data. In addition to the benefits from more data, training with diverse tasks also enforces the representation to be more generalized. These foundation models can achieve high task success rates for various tasks, are more robust to outliers, and are flexible for adapting to new tasks. These approaches map input signals from distinct domains and tasks into a high-dimensional representation space, and exhibit consistent scaling behaviors [34, 29]. After that, minimal fine-tuning is required to transfer the representation for downstream tasks to achieve good performance.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

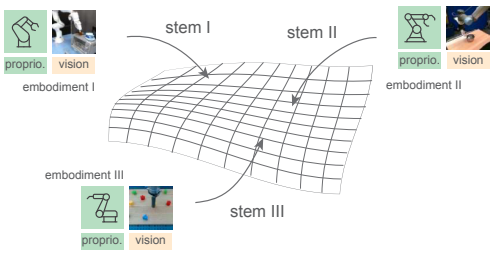

Figure 1: The **Heterogeneous Pre-training** concept. It maps different embodiments, each with its own *proprioception* and *vision* sensors, onto a *shared* latent space by embodiment-specific tokenizers ("stems"). This *aligns* the heterogeneous data from different embodiments into a joint representation space. This allows us to train a shared Transformer trunk on the union of all heterogeneous datasets. The pre-trained Transformer can be transferred to a new embodiment, with a small, new tokenizer learned at transferring time.

The heterogeneity in robotics presents a distinct challenge: different robots are physically different embodiments[1] of hardware acting in different environments. Each embodiment can have a distinct *proprioception*, including different degrees of freedom, end-effectors, motion controllers, and workspace configurations built for a specific application. Another common heterogeneity in robotics is *vision* heterogeneity. Robots are often equipped with different camera sensors mounted at different places (e.g. wrist and/or third-person) and the visual appearance of each robot varies dramatically due to environments and tasks. Both proprioception and vision information are crucial for complex, contact-rich, long-horizon behaviors in robotics. Poor learning of such information can lead to overfitting behaviors such as repeating motions for a particular scene and task or even trajectory.

In this work, we propose to address this issue by aligning the proprioception and vision information from different embodiments to a shared "language" of policies through *heterogenous pre-training* (Figure 1). With such a shared representation, a new embodiment only requires minimal data and training to "translate" its specific setup to the shared "languages". In other words, we want to pre-train task-agnostic and embodiment-agnostic foundational models that can map raw sensor signals from individual embodiments into a shared latent space. Previous works have made significant progress in pre-training only the vision part of the policy on human videos [49, 54, 35, 62] and pre-training the full policy [6, 14, 56] with a unified model and dataset format (e.g. using languages [5]). Additionally, they assume no proprioception in pre-training and add it post hoc in transfer learning.

We introduce Heterogeneous Pre-trained Transformers (HPT), a family of architecture designed to scalably learn from data across heterogeneous embodiments. HPT modularizes a general policy network architecture (Figure 2) and pre-trains the *policy representation* of a latent transformer with supervised learning. Inspired by learning from multimodal data [1, 74, 20, 31], we use *embodiment-specific* tokenizers, dubbed "stem", to align various sensor inputs such as camera views and proprioception inputs. The "trunk" is *shared* and pre-trained across datasets and is transferred when adapting to new embodiments and tasks that are unknown during the pre-training times. Moreover, we use task-specific action decoders, dubbed "head", to produce the action outputs. Crucially, after "tokenizing each embodiment", HPT operates on a shared space of a short sequence of *latent tokens*. This hierarchy is motivated by how humans handle feedback loops between specific motor responses and perceived stimuli at the level of the spinal cord's neural circuitry [69].

We extensively investigated the scaling behaviors and various designs of policy pre-training to the extent of more than 50 individual data sources (2 times more than [56]) and model size of over 1 billion parameters. Analogous to the scaling laws [27, 29], we found that to some extent, HPT scales with the dataset quantity and diversity as well as the model and training compute.

In addition, heterogeneity can occur in different embodiment domains, such as real robot hardware, simulation domains, and human videos. We incorporate many available embodied datasets in different embodiments such as real robots [14, 76, 39], simulation [82, 90, 50, 21, 86, 81] and internet human videos [15] in the pre-training process and demonstrate the generality of our framework including embodiments beyond expensive real-world on-robot teleoperations.

Through transfer learning experiments across multiple simulation benchmarks [90, 50, 82] and real-world dexterous tasks, we compare with several baselines and the from-scratch counterparts. Overall, based on the pre-training objectives, HPT can scale with the model, data, compute, and the heterogeneity of the robotic datasets across real robots, simulations, and human videos. These pre-

[1]Embodiment can be defined differently according to the context of robotics and AI. In this work, we consider robots equipped with a distinct set of sensors and actuators with the associated observation and action space to be a unique embodiment.

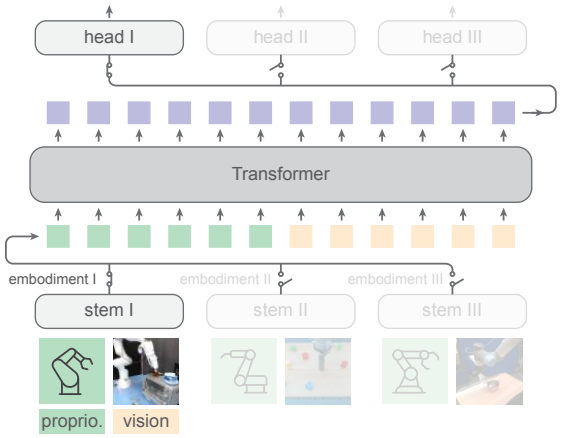

Figure 2: **HPT architecture.** HPT is modularized into stems, trunk, and heads. The stem, consisting of a proprioception tokenizer and a vision tokenizer, maps the vision and proprioception observations of different embodiments to a fixed number (e.g. 16) of tokens. The *shared* trunk, which is a Transformer, maps the concatenated tokens into shared representations. The head then maps the processed tokens to actions in different downstream tasks. For a specific embodiment, one stem/head pair is activated (denoted by the switch). The trunk is shared and pre-trained on action-labeled data with supervised learning and then transferred to new embodiments. This procedure scales up to 52 datasets and 1B parameters.

training procedures and models can simplify building reliable robotic policies for new embodiments and new tasks in terms of data requirements and generalized performance. As an attempt to scale heterogeneous pre-training, our code and weights are open-sourced, and we hope that HPT can shed some light on learning robot representations from heterogeneous embodiments and tasks.

2 Related Works

Pre-training and Transfer Learning. Pre-training [2], through direct supervision [38] and/or self-supervision [57, 25, 12, 22, 10], has been shown to learn representation useful for unseen downstream tasks in computer vision [7, 37] and natural language [58], and their intersections [60]. The representation learned from ImagetNet [16] or web-scale data [38, 60, 18] shows robustness to distribution shifts and can be transferred to new tasks.

The recent surge of foundation models [3] scales these representation learning methods [27, 29] by applying task-agnostic objectives to multitask data. Moreover, recent works [46, 42, 45] show that small projection layers can be used to align the pre-trained feature spaces of the foundation models. Different from other fields, robotics has less data quantity and diversity but much more heterogeneity.

Alignment. Recent works such as Flamingo [1], Perceiver [31], and ImageBind [20] proposed ways to combine representations from *multimodal data* such as image, language, and audio by aligning these different modalities to the same latent space in the pursuit of representation learning. Our architecture design is also motivated by methods such as LLaVA[45] in the multimodal learning community. Very recently, GPT-4o [58], Gemini [77], MM1 [52], X-VILA [89], and Chameleon [75] demonstrated the capabilities of heterogeneous pre-training a universal transformer from and for multiple modalities. The idea of alignment, across modalities and/or embodiments, is important as we scale to use heterogeneous embodiments and reuse data from distinct embodiments.

Representation Learning in Robotics. Representation learning has been explored in the robotic community. Previous works such as R3M [54], VC-1[49], Voltron[35], and SpatialVLM [11] investigate visual representations by training the policy with human videos and robotic data [70]. Recent works [61, 17, 4, 88, 71, 40] also align representations from multiple modalities and data distributions for robotic tasks. After pre-training, transfer learning with the frozen representation and/or finetuning is conducted in the target domains.

Generalist Policies. Large-scale policy learning in robotics has leveraged diverse data from real robots [6, 73], human videos [54, 49], and simulation domain [33, 63, 83, 80] separately. There are also works in multi-task learning [65, 66, 85, 23], meta-learning [79, 55, 19], few-shot learning [84], and fleet learning [82]. Recently, RT-X, Octo, OpenVLA [6, 14, 56, 36] train generalist vision-language-action robotic policies on datasets from diverse robotic embodiments.

Compared with these works, HPT handles broader heterogeneity including proprioception and vision, explores scaling behaviors on more heterogeneous domains including real robots, human videos, and simulation data, and is evaluated at a larger scale in simulation benchmarks.

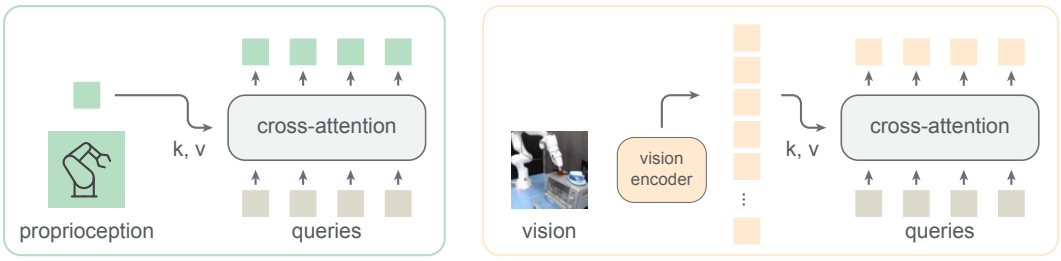

Figure 3: **Stem Architecture in HPT.** In the HPT stem, the proprioceptive tokenizer uses an MLP to map proprioceptive information to a feature which is then attended by 16 learnable tokens. The vision tokenizer uses pre-trained encoders and similarly uses an attention mechanism to map vision features into 16 fixed tokens. The architecture flexibly handles sequences of inputs without increasing the size of tokens.

Mixture of Experts. Our architecture design is related to works in conditional computation and MoE [51, 44, 72], where we create one expert for each embodiment, and the router (for the whole network) is determined by the embodiment. This technique has been used to scale language models to a substantial size [32].

3 Heterogenoues Pre-trained Transformers (HPT)

In *heterogeneous robot learning* with cross embodiments, the data are generated from different domains such as simulation and real robots, across sensory modalities such as RGB images, language instructions, depth maps, 3D point clouds, and tactile images. Each robot is a unique hardware embodiment with varying degrees of freedom, end-effectors, sensor configurations, controller and action spaces, and application-specific physical setups.

In the following sections, we discuss the HPT network architecture and the training procedure to address the heterogeneity above. We modularize the network architecture (Figure 2) into the embodiment-specific stem, the shared trunk, and the task-specific heads. Intuitively, the stems, shown in Figure 3, are earlier layers of the neural network that align sensory inputs from heterogeneous embodiment and modalities into the shared representation space. The *shared* middle part of the network is called the trunk, which processes the sensory representation into a latent representation that can be used for multiple tasks. Finally, the last part of the network is the head, which maps that latent representation to the action space in individual tasks of interest. The training procedure, dubbed *heterogeneous pre-training*, assigns and aligns specific stem/head pairs based on the sampled embodiment and task data, and still enjoys the benefits of joint training in the shared trunk. This can be thought of as tokenizing each embodiment using neural networks and alleviating the need to unify embodiments into a homogeneous data form in standard training procedures.

3.1 Network Architecture

Stem. The stem θ_{stem} (Figure 3) in HPT is composed of a proprioceptive tokenizer and a vision tokenizer. These tokenizers map heterogeneous inputs from different embodiments to a *fixed number* of tokens with *fixed dimensions*, which enables the trunk to treat them in the same manner despite large heterogeneity, as well as enjoy the scaling and inference benefits on fixed context length. The key idea is to leverage attention [78, 31, 9] to attend a fixed number of learnable tokens to features of the observations. Although we mainly focus on proprioception and vision, handling other kinds of sensor heterogeneity in tactile, 3D, and action inputs can be flexibly extended in stems.

- **Proprioception Tokenizers.** In Figure 3 (left), for embodiment k, the proprioceptive tokenizer maps any sequence of robot proprioceptive information with dimension d_p^k (e.g. 7 for end-effector pose) into N_p (e.g. $N_p = 16$) tokens with dimension d with values ranging from 128 to 1024. To achieve this, we first use an MLP to map the proprioceptive input into a feature space with dimension d. We then apply sinusoidal position encoding and use attention across the state feature and the learnable tokens, to map into 16 tokens with dimension d. Proprioceptive information is critical in robot policy learning, but its usage is often as simple as feature concatenation with a vision encoder [41].

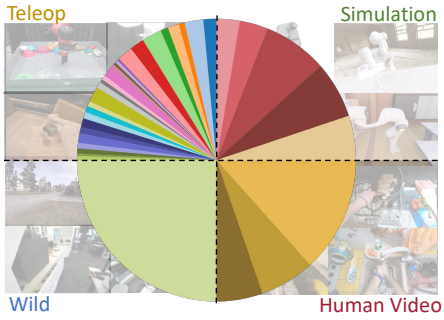

Figure 4: **Dataset Heterogeneity in Robotics.** We show illustrations of dataset mixtures (each color is a distinct embodiment) from different domains including real robot teleop [14], deployed robots [39], simulations, and human videos [15]. See Appendix Section A for dataset mixture details.

	# Depth	# Width	# Attn. Heads	# Param.
HPT-Small	16	128	8	3.1M
HPT-Base	16	256	8	12.6M
HPT-Large	16	512	8	50.5M
HPT-XLarge	32	768	16	226.8M
HPT-Huge	80	1024	16	1.1B

Table 1: **Network Details of HPT.** The width denotes the latent dimension size of the trunk transformer and the depth denotes the number of blocks. The *default* setup is the HPT-Small model.

	# Dataset	# Traj.	# Samples	# Batch Size
Default	27	16k	5M	256
Scaled	52	270k	155M	2048

Table 2: **Dataset Details of Pre-train Settings.** The *default* setup is trained with 27 datasets from RT-X with 16k trajectories (maximum 1000 trajectories each) and *scaled* setup involves more data and compute.

- **Vision Tokenizers.** In Figure 3 (right), the vision tokenizer can map any sequence of camera images (videos of multiple views) with dimension $H \times W \times 3$ into N_v (we use $N_v = 16$ by default) tokens with dimension d. To achieve this, we first use *pre-trained frozen feature networks* (e.g. 7 by 7 features from ResNet) and then flatten the features. After that, we again use attention across these features and learnable tokens, to map the vision input into 16 tokens with dimension d.

After processing each modality individually in the time sequence order, we concatenate all modality tokens and add additional modality embeddings and sinusoidal positional embeddings. This is used as the input sequence to the trunk that we introduce below. To avoid overfitting, the stem only has a small number of parameters (one MLP and one attention layer).

Related works such as Octo [56] and others [54, 49, 6] mostly focus on pre-training the vision backbone of the policy through masking or self-supervision. They often stack sequences of single-view images along channels [6] for a particular robot or use a large number of tokens (256 in [56]). In contrast, HPT uses stems with pre-trained vision encoders to map arbitrary image sequences to a short sequence of tokens (16). Moreover, rather than *add in* proprioception during transfer in related works, HPT *jointly pre-trains* the vision and proprioception parts, from heterogeneous datasets.

Trunk. As the central component for pre-training, the trunk architecture follows a transformer, parametrized by θ^{trunk} in the latent space with dimension d. The output token sequence length L is the same as the input token sequence length. The output token sequence is simply pooled as the final combined feature for the observation. The trunk is shared across different embodiments and tasks to capture the complex input-output relationships (i.e. the number of trunk parameters is fixed independent of the number of embodiments and tasks).

Head. The policy head θ_{head} takes the output of the trunk transformer and maps it to the action space $\mathcal{A}$ in each dataset. For each embodiment and task, the policy head can be an arbitrary architecture (e.g. MLP) that takes as input the pooled feature of the trunk and outputs a normalized action trajectory. The policy head is reinitialized for transferring to a new embodiment.

3.2 Training Objective

Given a total of K datasets with heterogeneous embodiments sampled from different distributions $\mathcal{D}_1, ..., \mathcal{D}_k, ..., \mathcal{D}_K$, we let $\mathcal{D}_k = \{\tau^{(i)}\}_{1 \le i \le M_k}$ denote a set of M_k trajectories in dataset $\mathcal{D}_k$, with $\tau^{(i)} = \{o_t^{(i)}, a_t^{(i)}\}_{1 \le t \le T}$ denoting the i-th trajectory of maximum length T of observation and action tuples. The objective is to minimize the following loss across datasets

$$\min_{\theta} \sum_{k=1}^{K} \mathcal{L}(\theta_k^{\text{stem}}, \theta^{\text{trunk}}, \theta_k^{\text{head}}; \mathcal{D}_k). \tag{1}$$

$\mathcal{L}$ is behavior cloning loss computed as the Huber loss between the normalized action labels based on dataset statistics and the network's action predictions. $\theta = \bigcup_{k=1}^{K} \{\theta_k^{\text{stem}}, \theta_k^{\text{head}}\} \cup \theta^{\text{trunk}}$ denotes the network parameters comprised of embodiment-specific stem and head $\theta_k^{\text{stem}}, \theta_k^{\text{head}}$ for dataset k, and a single set of shared trunk parameters θ_{trunk} across all embodiments. This training procedure has two axes of data scaling: the quantity M_k for one dataset D_k and the total number of datasets K. In the pre-training stage, only the trunk parameters are updated at every iteration, and the stems and heads for each heterogeneous embodiment and task are updated based on the training batch sampling. See implementation details in Appendix Section A.3.

3.3 Transfer Learning

The policy transfer process is similar to aligning the features of the new domain (through pre-trained stem encoders) to the pre-trained embedding space of the trunk [42, 45]. Given a new dataset $\mathcal{D}_{K+1}$ from a new embodiment, the objective can be the same as pre-training or alternatives [13]. We reinitialize the head and stem parameters with embodiment-specific input and output dimensions (such as different proprioception and action dimensions), and freeze the weights of the trunk.

4 Experiments on Pre-training

In this section, we aim to answer the following question: Does HPT pre-training have a *scaling behavior* under heterogeneous data across domains?

Default Setting. We use 27 robot teleoperation datasets, including a subset of the recently public Open-X Embodiment dataset [14] as the training corpus. By default, we use one camera view of the scene with the pre-trained frozen ResNet18 image encode to compute the vision features. We use proprioception information, such as end-effector poses and joint positions, whenever they are available and provided. We use a maximum of 1000 trajectories from each dataset and a total number of 16k trajectories, and a held-out validation dataset with a maximum 200 trajectories per data source. Furthermore, we use a model with a trunk size of 3.17 million parameters, which is denoted as HPT-Small (Table 1). The training uses a batch size of 256 for 80k iterations, which is around 0.65B tokens in the latent space that feeds into HPTs and around 5B tokens in the vision and proprioception token spaces (horizon-dependent). While we do not align or preprocess action space or observation space [56, 87] other than normalization, data cleanup and filtering would be very helpful.

Scaled Setting. We use 200k trajectories with 52 datasets, including simulation (e.g. [50]), deployed robots (e.g. [39]), human videos (e.g. [15]), from distinct embodiments in the training process. This includes many public and accessible robotic datasets. In addition to different tasks in different institutes, these heterogeneous mixtures of datasets (Fig. 4 and Fig. 13) come with multiple views, language inputs, and different observation inputs in different environments.

4.1 Protocol

We evaluate the HPT pre-training performance with the *averaged validation loss* (prediction errors on unseen trajectories) at the last iteration of pre-training. These validation datasets are fixed independent of the trajectory counts and models during training. Unless particularly noted, the validation datasets come from the same 27 datasets in the *Default Setting*. Note that it is unrealistic to evaluate the pre-trained models on many real-world robotic environments at scale and there are very few evaluation alternatives to measure large-scale pre-training if we ignore this objective. In fields such as NLP[30, 34], training loss objective (e.g. perplexity) is often used to measure the progress of pre-training. Admittedly, there are several caveats to this metric including the closed-loop performance gap and the task success rate gap. We will address these issues in Section 5 on HPT transfer learning. See Appendix Section A and Section D for more details and discussions.

4.2 Scaling Behaviors

Data Scaling. In Figure 5 (a), we observe stable and scaling validation losses even on increasingly heterogeneous embodiments. Moreover, we found the compute (e.g. samples seen per training run) and the data amounts needed to scale in tandem [34] to get closer to convergence in the training process. In the red line in Figure 5 (a), we observe better validation losses as we scale up the total number of trajectories, by using a larger model and doubling the batch size every order of magnitude increase in trajectory counts. Strictly increasing data while keeping others bottlenecked (HPT-S

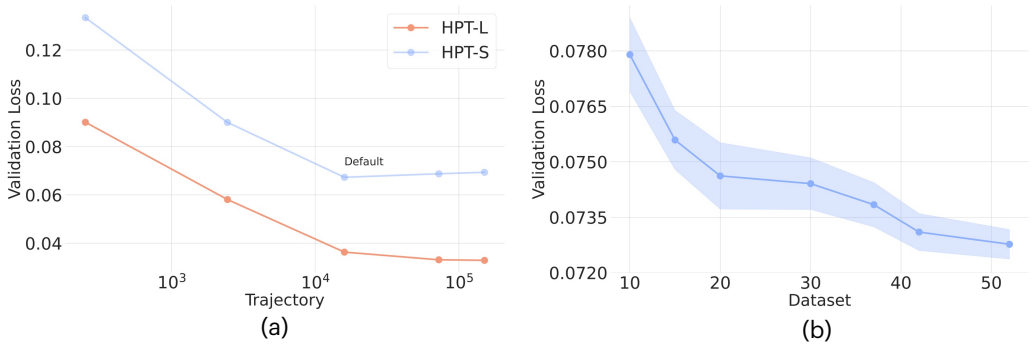

Figure 5: **Data Scaling**. We run scaling HPT experiments along dataset sizes and the number of datasets. Each point is the validation loss of a full training run. (a) We evaluate the losses on 27 datasets with the number of total trajectories ranging from a maximum of 10 trajectories per dataset (270 in total) to a maximum of 100000 trajectories per dataset (170k in total). We compare two model sizes, HPT-S/L, where HPT-L is a bigger model trained with 4 times more tokens than HPT-S. (b) We compute the validation losses for a fixed subset of 10 datasets with a fixed number of epochs (2). We compute mean and standard deviations for 4 runs across model sizes from HPT-S to HPT-XL and across dataset counts from 10 to 52.

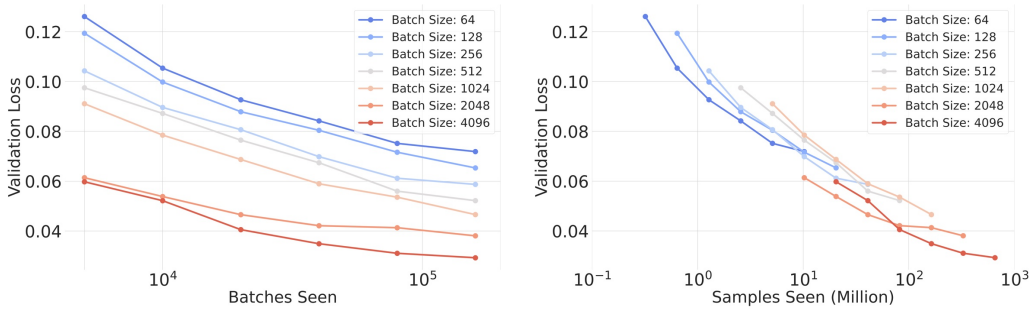

Figure 6: **Epoch Scaling**. We run scaling HPT experiments along the number of total samples. Each point is the validation loss of a full pre-training run. Setting: HPT-S, 27 datasets with a maximum of 1000 trajectories for each dataset. Left) We scale up the number of batch sizes and measure the changes in validation losses. Right) Derived from the left figure, we multiply the batches seen by the number of samples in each batch.

and fixed iterations) might cause an early plateau performance at around 1000 trajectories max per dataset, as shown in the blue line in Figure 5. In Figure 5 (b), we also pre-train on an increasing number of datasets with a fixed number of epochs and evaluate on the fixed subset (first 10 datasets). We hypothesize that training with more embodiments contributes to the generalization of the trunk. These experiments can scale to the extent of 200k trajectories and 52 datasets.

Model Scaling. In Figure 7, we fix the number of datasets (27) in RT-X and use a maximum of 1000 trajectories for each dataset. We scale along model size (from 1M to 1B) and gradually increase the batch sizes from 256 to 2048 (doubles every order of model size increase) and use the larger dataset with 170k trajectories. We observe that when we scale to bigger models with larger amounts of compute (red line), the pre-training can achieve low validation losses until it is plateaued. We do not find a significant difference between scaling depth or scaling width.

Epoch Scaling. In this experiment, we fix the number of datasets (27) and use a maximum of 1000 trajectories for each dataset. In Figure 6, we observe that increasing batch sizes (Left), which effectively scales training tokens (Right), can generally improve the model performance until convergence. Another observation we have is to use distributed workers to load from as many datasets as possible to aggregate each batch. We hypothesize that the large variance of training on heterogeneous datasets can be reduced by using a large batch size. See Appendix B for more experiment details.

4.3 Pre-training on Synthetic Data and Internet Human Videos

We experiment beyond real-world robot teleop data, which is expensive to collect and scale. For the additional datasets, we consider 7 simulation datasets across many popular simulators Drake [82], Mujoco [90, 50], Isaac Sim [21], and PyBullet [86, 81], as well as Sapien [53] and Flex [67], with

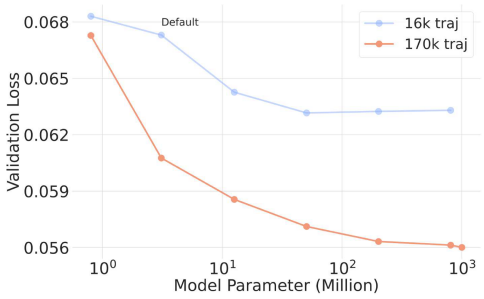

Figure 7: **Model Scaling**. We run scaling HPT experiments along model sizes. Each point is a full training run. Setting: 27 datasets with a maximum of 1000 trajectories for each dataset. We scale along model size (from 1M to 1B) for both the blue and red lines. The red line is trained with increasing data and epochs to reach convergence. Specifically, we gradually increase the batch sizes from 256 to 2048 (doubles every order of model size increase) and use 170k trajectories.

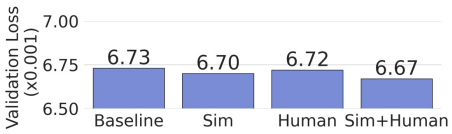

Figure 8: **Joint Pre-training with Simulation and Human Videos.** The baseline denotes the default setting without simulation and human datasets. Setting: We run the experiments with a training corpus of datasets with 1000 trajectories maximum.

image inputs and expert demonstrations. For the human datasets that lack proprioception and action information, we use poses and 2D positions as surrogates for the supervised policy learning objectives. We use in total 300 trajectories from EPIC kitchen [15] and PoCo [83] with a maximum trajectory length 1000. See Appendix Figure 13 and Table 4 for more details on the dataset compositions.

In Figure 8, we use a maximum of 1000 trajectories for each dataset and compare against the baseline of 27 datasets with evaluation on all the pre-trained datasets. We show that pre-training on additional embodiment datasets such as simulation and human video datasets can be possible, despite the large embodiment gaps with real robots. These datasets provide complimentary embodiment data to pure teleop data, and they illustrate how much heterogeneity can be handled in the HPT framework.

5 Experiments on Transfer Learning

In the previous section, we evaluate pre-training using the validation losses. In this section, we answer the following question with task success rates in transfer learning: Can the pre-trained HPT model be transferred to new embodiments, tasks, and environments in simulation and the real world?

5.1 Transfer to Embodiments in Simulations

Protocol. We evaluate the pre-trained representations on robot manipulation simulation benchmarks Meta-world [90], RoboMimic [50], and Fleet-Tools [82]. Each training dataset uses from 20-100 trajectories per task and each testing covers 50 episodes with different initial conditions. The policies use HPT-Small as the pre-trained trunk and reinitialize the stem and head for transferring.

During the evaluation phase, we compare the following models: `No Trunk` uses only the stem and head without the trunk in the middle and trains from scratch as common practice [41]. `From Scratch` trains the entire policy from scratch with the trunk, `Pretrained Frozen` uses and freezes the pre-trained trunk during transfer learning and `Pretrained Finetuned` loads the pre-trained HPT-Base trunk and finetunes the whole network end-to-end, and `Pretrained Finetuned (HPT-XL)` uses the same fine-tuning procedure with a pre-trained HPT-XL trunk with a lower pre-training validation loss. To reduce the variance, we conduct independent training runs and evaluations 5 times and average for each model. The inference time during transfer on an RTX 3070 GPU is 47Hz for HPT-base and 19Hz for HPT-XL, while a more recent GPU like A100 can be 3-4 times faster.

Experiment. In Figure 10 (a), we test the model on the downstream tasks in closed-loop simulation and observe improved task success rate using the pre-trained models ranging from HPT-B to HPT-XL, although pre-training for the simulation experiments only happens in the real-world embodiments.

In Figure 10 (b), we run HPT on the recently released Simpler [43] Benchmark, which allows for comparing with Octo [56], RT1-X, and RT2-X [14] on a high-fidelity simulation. We focus on three different tasks `Close Drawer`, `Move Near`, and `Pick Coke Can` in the Google EDR embodiment. For each task, we test several different initializations with a total of over 300 episodes for all tasks.

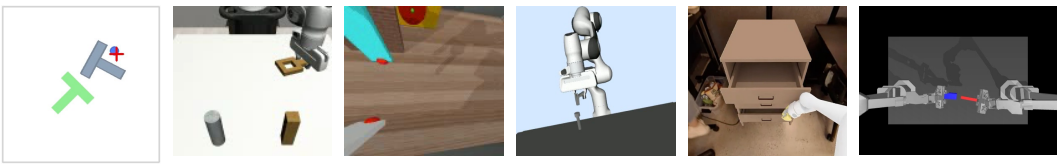

Figure 9: **Simulation Evaluation Tasks.** We evaluate HPT across several simulation benchmarks and show policy rollout visualizations of the experiments. Experiment details can be found in Section 5.1 and A.4.

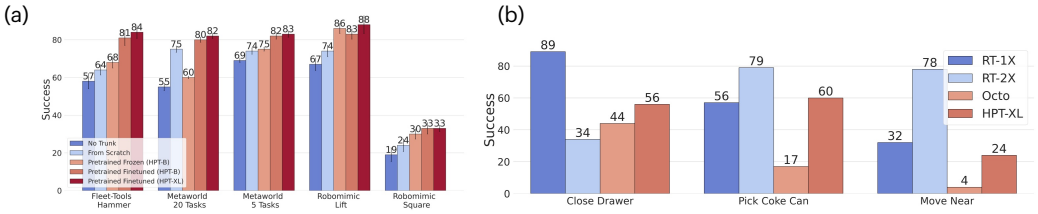

Figure 10: **Success Rates in Simulation Experiments.** (a) We evaluate transfer learning performance of models from HPT-B to HPT-XL on tasks across 4 different simulator benchmarks. (b) We compare with several generalist models in the recent Simpler [43] benchmark with Google GDR embodiment. The pre-trained trunks are trained from the Scaled Settings. The success rates are computed over 150 rollouts per approach.

Note that the pre-training corpus of HPT-S does not include [6], and simulation tasks have a focus on language conditioning and do not expose proprioception inputs, which is not suitable for HPT. To address these issues, we finetune HPT on the supervised datasets with around 50 trajectories under the simulation protocol. We use HPT-base as the backbone for this experiment. We use the baseline results from [43]. See Section A.4 for more implementation and experiment details.

5.2 Transfer to Embodiments in the Real World

Protocol. For the real-world experiments, we evaluate the HPTs on two different embodiments for tasks in pet care and assembly, which are not covered in the pre-training datasets [14]. In particular, for these two robots, we experiment with different observation spaces 1 camera v.s. 2 cameras as well as different action spaces relative pose v.s. absolute pose. For data collection, we experiment with both an Oculus Quest to collect relative pose control as action labels as well as kinesthetic teaching. The episode lengths of real-world teleoperation vary from 50 steps to 150 steps with 10 Hz control frequencies. We experiment with the tasks `Sweep Leftover`, `Fill Water`, `Scoop Food` and `Switch Insertion`, which require 5-20 seconds of interactions with granular or small objects with fine contacts, shown in Figure 11. We collect around 100 demos for each task and evaluate them for 15 trials to measure the average success rate.

Experiment. We adopt a similar transfer learning method in the previous section and evaluate the pre-trained HPT representations under real-world evaluation protocols. We train the policy with 20000 iterations with a batch size of 256 and a learning rate of $5e^{-6}$. We defer implementation details to Appendix Section A.5. Quantitatively in Figure 12, we observe pre-trained policies attain a better success rate over the `No-Trunk` and the `From-Scratch` baselines. In particular, the `From-Scratch` baselines in `Fill-Water` use the state-of-the-art diffusion policy architecture to illustrate the flexibility of the pre-trained representations. In Figure 11, qualitatively, we observe better generalization and robustness to varying poses and numbers of granular objects, and varying camera configurations and lighting conditions with pre-trained HPT.

On Table 3, we perform an ablation study for the `Sweep Leftover` task. We also compare with R3M [54], Voltron [35], and VC-1 [49]. We use a finetuned model with the released backbone and weights. We note that these previous works focus on only pre-training the vision encoders of the policies with human videos. Finally, we compared with policies that train from scratch (`From Scratch`) and policies that do not use proprioception during pre-training (`No Prop. Finetuned`) and add in proprioception afterwards. All of our experiments use pre-trained encoders and the trainable parameters (stem and head) can be as few as 2% of the parameters.

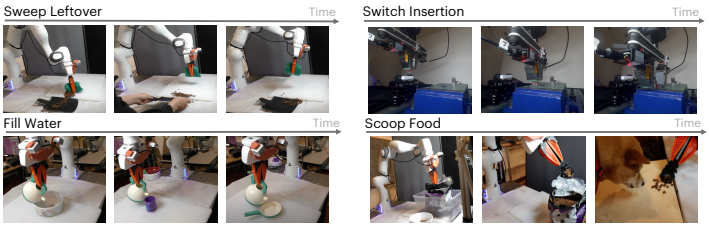

Figure 11: **Real World Qualitative Results**. Pre-trained HPT policies can perform dynamic and long-horizon contact-rich precision tasks in pet care and assembly. The policies show robust and generalized behaviors under scene changes and disturbances.

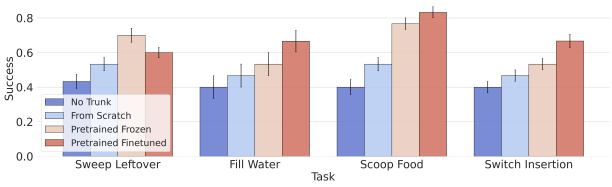

Figure 12: **Transfer Learning in the Real World.** We evaluate the pre-trained HPTs on four tasks / two embodiments. The average success rate with standard deviations is computed for 45 trials per approach. We use the default pre-training setup with HPT-Base for this experiment. See Section 5.2 for detailed descriptions.

Method	**Success (%)**
From Scratch No Prop.	26.7±3.3
From Scratch	43.3±3.8
R3M [54]	50.0±3.0
Voltron [35]	46.7±3.8
VC-1 [49]	53.3±2.6
No Prop. Finetuned	63.3±2.6
HPT-B Finetuned	70.0±3.0
HPT-XL Finetuned	**76.7**±3.3

Table 3: **Comparison on the `Sweep Leftover`.** We compare the fine-tuned HPT models with several baselines including vision-only pre-trained models.

6 Conclusion

There is room for improvement for many aspects including the dataset curation and pre-training objectives. Specifically, the embodiment splits in our balanced dataset mixture are rather simple. Moreover, careful data filtering to ensure the data quality is under-explored in this work. Also, this work has focused on supervised learning as the pre-training objective and the data size in tokens and training compute sizes in FLOPs only reach a moderate scale of LLM training to ensure full convergence. Although the model architecture and training procedure are modular and independent of embodiment setups, heterogeneous pre-training can converge slowly. For evaluation, both the simulation and real-world evaluation tasks are restricted to short-horizon manipulation tasks with a fixed embodiment, which might limit the benefits of using a higher-capacity model. Furthermore, the learned policies still do not offer very high reliability on the tested tasks (typically below 90%). See Appendix §C for some failure modes.

Given the recent surge of scaled data, robot learning is still limited by its generality because of the heterogeneity, including different embodiments, tasks, and environments where the robots are operated. To handle the heterogeneity common in robotics, we propose HPT, a modular architecture and framework to embrace this heterogeneity through pre-training. We explore and scale HPT with heterogeneous datasets to over 50 available datasets. The learned representation can be transferred and improve performance in both simulation and the real world, and it shows correlations with pre-training performance. The code[2] is open-source for future research. We hope this perspective will inspire future work in handling the *heterogeneous nature* of robotic data for robotic foundation models.

Acknowledgement. We would like to thank Russ Tedrake for discussions and suggestions, Liane Xu for helping with real-world experiments, Tianhong Li for helping with cluster experiments, and Remi Cadene for helping with the LeRobot implementation. We thank MIT Supercloud for providing computing resources to process training data. This work is supported in part by the Amazon Greater Boston Tech Initiative and Amazon PO No. 2D-06310236. Toyota Research Institute provided funds to partially support this work.

[2] https://github.com/liruiw/HPT and https://github.com/liruiw/lerobot

References

[1] 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.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[4] Rogerio Bonatti, Sai Vemprala, Shuang Ma, Felipe Frujeri, Shuhang Chen, and Ashish Kapoor. Pact: Perception-action causal transformer for autoregressive robotics pre-training. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3621–3627. IEEE, 2023.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[7] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

[8] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[10] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[11] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. *arXiv preprint arXiv:2401.12168*, 2024.

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[13] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[14] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. https://robotics-transformer-x.github.io, 2023.

[15] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[18] Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Alexander Toshev, Vaishaal Shankar, Joshua M Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. *arXiv preprint arXiv:2401.08541*, 2024.

[19] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[20] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190, 2023.

[21] Ran Gong, Jiangyong Huang, Yizhou Zhao, Haoran Geng, Xiaofeng Gao, Qingyang Wu, Wensi Ai, Ziheng Zhou, Demetri Terzopoulos, Song-Chun Zhu, et al. Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[22] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[23] Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

[24] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[25] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

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

[29] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[30] Chip Huyen. Understanding evaluation metrics for language models. *The Gradient*, 2023.

[31] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[32] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[33] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: Robot manipulation with multimodal prompts. 2023.

[34] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[35] Siddharth Karamcheti, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023.

[36] 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.

[37] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[39] FrodoBots Lab. Frodobots-2k dataset. https://huggingface.co/datasets/frodobots/FrodoBots-2K, 2024. Accessed: 2024-05-27.

[40] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[41] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[42] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[43] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[44] Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Gosh, Luke Zettlemoyer, and Armen Aghajanyan. Moma: Efficient early-fusion pre-training with mixture of modality-aware experts. *arXiv preprint arXiv:2407.21770*, 2024.

[45] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

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

[47] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[48] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

[49] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? 2023.

[50] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[51] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293, 2014.

[52] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*, 2024.

[53] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

[54] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[55] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[56] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. https://octo-models.github.io, 2023.

[57] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[58] OpenAI. Gpt-4 technical report, 2023.

[59] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[60] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[61] Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot learning with sensorimotor pre-training. In *Conference on Robot Learning*, pages 683–693. PMLR, 2023.

[62] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

[63] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid locomotion as next token prediction. *arXiv preprint arXiv:2402.19469*, 2024.

[64] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[65] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[66] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[67] Gautam Salhotra, I-Chun Arthur Liu, Marcus Dominguez-Kuhne, and Gaurav S Sukhatme. Learning deformable object manipulation from expert demonstrations. *IEEE Robotics and Automation Letters*, 7(4):8775–8782, 2022.

[68] Saumya Saxena, Mohit Sharma, and Oliver Kroemer. Multi-resolution sensing for real-time control with vision-language models. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[69] Lucia Seminara, Strahinja Dosen, Fulvio Mastrogiovanni, Matteo Bianchi, Simon Watt, Philipp Beckerle, Thrishantha Nanayakkara, Knut Drewing, Alessandro Moscatelli, Roberta L Klatzky, et al. A hierarchical sensorimotor control framework for human-in-the-loop robotic hands. *Science Robotics*, 8(78):eadd5434, 2023.

[70] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[71] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. Mutex: Learning unified policies from multimodal task specifications. *arXiv preprint arXiv:2309.14320*, 2023.

[72] 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.

[73] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[74] Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Gupta, Philip M Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. Omninet: Omnidirectional representations from transformers. In *International Conference on Machine Learning*, pages 10193–10202. PMLR, 2021.

[75] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models, 2024.

[76] DROID Team. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

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

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[79] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

[80] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. *arXiv preprint arXiv:2310.01361*, 2023.

[81] Lirui Wang, Yu Xiang, Wei Yang, Arsalan Mousavian, and Dieter Fox. Goal-auxiliary actor-critic for 6d robotic grasping with point clouds. In *Conference on Robot Learning*, pages 70–80. PMLR, 2022.

[82] Lirui Wang, Kaiqing Zhang, Allan Zhou, Max Simchowitz, and Russ Tedrake. Fleet policy learning via weight merging and an application to robotic tool-use, 2024.

[83] Lirui Wang, Jialiang Zhao, Yilun Du, Edward Adelson, and Russ Tedrake. Poco: Policy composition from and for heterogeneous robot learning, 2024.

[84] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[85] Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. *arXiv preprint arXiv:2305.02968*, 2023.

[86] Manuel Wüthrich, Felix Widmaier, Felix Grimminger, Joel Akpo, Shruti Joshi, Vaibhav Agrawal, Bilal Hammoud, Majid Khadiv, Miroslav Bogdanovic, Vincent Berenz, et al. Trifinger: An open-source robot for learning dexterity. *arXiv preprint arXiv:2008.03596*, 2020.

[87] Jonathan Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *arXiv preprint arXiv:2402.19432*, 2024.

[88] Jonathan Yang, Dorsa Sadigh, and Chelsea Finn. Polybot: Training one policy across robots while embracing variability. *arXiv preprint arXiv:2307.03719*, 2023.

[89] Hanrong Ye, De-An Huang, Yao Lu, Zhiding Yu, Wei Ping, Andrew Tao, Jan Kautz, Song Han, Dan Xu, Pavlo Molchanov, et al. X-vila: Cross-modality alignment for large language model. *arXiv preprint arXiv:2405.19335*, 2024.

[90] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[91] Jialiang Zhao, Yuxiang Ma, Lirui Wang, and Edward H Adelson. Transferable tactile transformers for representation learning across diverse sensors and tasks. *arXiv preprint arXiv:2406.13640*, 2024.

[92] 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.

A Implementation Details

Experiment Details. We conduct pre-training experiments across several orders of magnitudes in computes and data. The number of trajectories and transitions in the dataset is limited by the maximum number of trajectories in each of the constituent datasets. We use maximum episode counts per dataset ranging from 10 trajectories to 100000 trajectories, and the total trajectories range from around 300 trajectories and 6000 transitions to around 300k trajectories and 5 million data points. When training with 80k iterations, the approximate training epochs with fixed batch size 512 range from 200 epochs to 2 epochs. In terms of tokens, our experiment model ranges from 0.5 million to 1 billion parameters, the dataset tokens from all modalities range from approximately 32 million tokens to 5 billion tokens, and the tokens in a batch range from 0.03 million tokens to 2 million tokens (including sequence length). The compute FLOPs range from 0.03GFlops to 31GFlops. See Table 5 for more details of the scale and see Figure 13 for example lists of dataset mixtures. To facilitate future research, we will open-source the data processing scripts.

Different from previous work [56, 87], we use minimal amounts of processing and cleaning of the observation and actions in the raw trajectories. Specifically, the default training setup is to train 80000 iterations with a batch size 256, which is around 0.65B tokens in the latent space that feeds into HPTs and around 5B tokens in the perception token spaces of the raw perception inputs (such as image patches). Due to resource limits, for some bigger datasets such as Droid [76], we did not process the full size.

A.1 Dataset Details

Real Robot Teleoperation Dataset. In total, we use a subset of 42 datasets in the Open-X Embodiment dataset [14], including the recent Droid [76] dataset. These public datasets have high heterogeneity including distinct embodiment, environments to operate on, tasks to solve, etc.

Simulation Dataset. For the additional 7 simulation dataset, we use the simulator benchmarks across all popular simulators Drake [82], Mujoco [90, 50], Isaac Sim [21], and PyBullet [81], as well as Sapien [53] and Flex [67], with image inputs and expert demonstrations. These are used as additional training data from the simulation domains.

Human Video Dataset. Since the human datasets do not contain proprioception and action information, we use hand poses and 2D positions in the image space as surrogates for the supervised learning objectives. In the PoCo [83] dataset, we use 3D positions of the hand and use 6-Dof poses extracted by ICP as actions, and for EPIC-Kitchen [15], we use normalized 2D centers of the detection box as proprioceptions and the difference to the next frame as actions. We use in total of 2000 trajectories of video clips from EPIC kitchen with a maximum trajectory length of 500.

Deployed Robot Dataset. To further increase the heterogeneity in the pre-training dataset, we consider FrodoBot-2k [39] dataset that involves plays of driving robots in the wild for gaming. This dataset is composed of deployed mobile robots in the wild. We use the front camera for this dataset. The action space of this robot is parametrized by linear and angular velocity actions and the proprioception space includes measurements from the IMU. We use a total of 150 trajectories and each trajectory contains more than 500 steps.

See Figure 13 for visualization of some examples of heterogeneous dataset compositions. In practice when training on the mixture of these datasets, users can define customized sampling weights or apply stratified sampling methods. For generality, in this work, we use a balanced weight sampling method, commonly used in multitask learning. For these datasets, we process the visual features separately and save them on the disks.

A.2 Network Details

Stem. For vision inputs, we resize each image to the standard square size (224x224) before feeding into a ResNet18 [26] into a 7x7 vision modality token. These tokens are specifically the features before the final global pooling. If multiple views are available, we create individual projector MLP for each image view and then concatenate the vision tokens. For the vision encoders. In Figure 18, We have experimented with multiple vision encoders such as MAE ViT base [24], Dino V2[59], and CLIP ViT Base [60]. We choose ResNet and the default image size for their simplicity and common

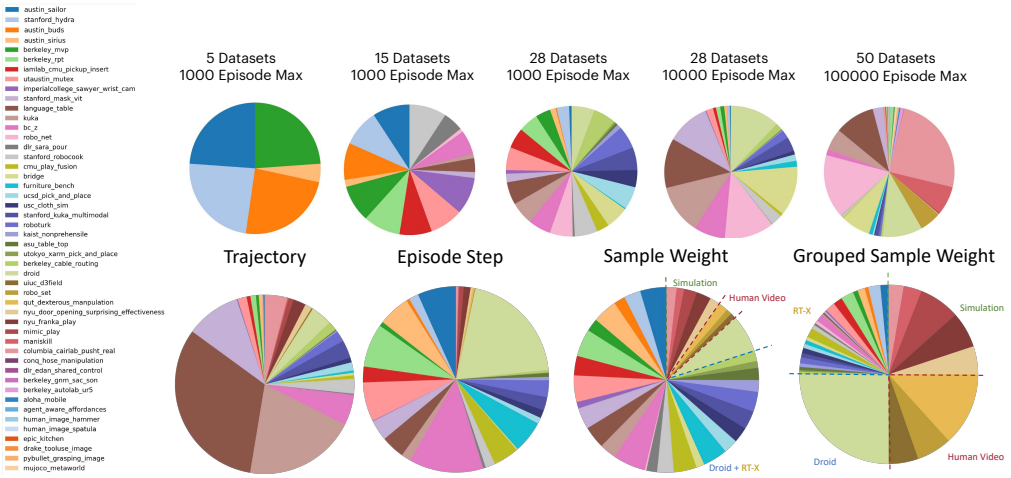

Figure 13: **Large-scale Dataset Heterogeneity in Robotics.** We show different dataset mixtures at increasing scales (top row) across trajectory counts, dataset sample counts, and sampling weights (bottom row). We also show illustrations of the different embodiments including real robots, simulations, and human videos. By default, during training, we use a uniform distribution to sample from each of the embodiment datasets.

usage in policy learning. The investigation of more complex fusion and processing for vision features is left to future works.

When language is used, we use T5 [64] to encode trajectory-level language modality tokens. Rather than using raw data and computing the tokens each time, these tokens are processed offline and saved as datasets before training.

For low-dimensional inputs such as proprioceptions and actions, we first flatten these vectors and then apply normalization to each dimension, as a single token. We have also experimented with increasing the dimensions of these modalities by adding sinusoidal position embeddings. For multiple steps in the observation horizon, we concatenate the sequence for each modality separately. At inference time, these tokens are forwarded once and cached to avoid multiple calculations.

We apply cross attention (with sinusoidal position encodings) and a shallow MLP projector [42, 46] within the stem to map different various sequences of tokens into a fixed number of tokens for that modality. The cross-attention layer has 8 heads and 64 hidden dimensions per head. We map the modality tokens into 16 tokens, 16 tokens, and 8 tokens respectively for image, proprioception, and language.

Trunk. The trunk is parametrized by a decoder-only transformer architecture with embedding dimension h that we ablate from 64 to 2048, and with block numbers ranging from 16 to 80. Note that the number of parameters for the trunk scales quadratically with the dimension size and linearly with the number of layers. The trunk also supports loading from existing large language models. Refer to Table. 1 for more details on model sizes.

The code is modularized so that the trunk training and transfer are independent of the encoder architecture or pre-trained weights of the stem, which can be ImageNet pre-trained ResNet [26], R3M [54], Voltron [35], Dino v2 [59], etc, and can be fintuned during transfer learning. It is also independent of the head, which can be diffusion policies, MLP, or transformer.

Head. For the head architecture, we normalize the action spaces of each embodiment dataset to be between -1 and 1 element-wise, based on the dataset statistics, such that the output scale of the heads and the loss and gradients remain at similar scales. Since action trajectories and observation history can often help with the robotic problem, we pick a fixed action horizon (length 8) and observation horizon (length 4) for each embodiment. We then apply random masking along the time dimensions for each batch during training to be suitable for downstream tasks with different horizons.

Dataset	Trajectory	Trajectory %	Sample	Sample %
Austin Sailor Dataset	205	0.09%	290167	1.85%
Stanford Hydra Dataset	487	0.22%	300217	1.92%
Austin Buds Dataset	42	0.02%	27894	0.18%
Austin Sirius Dataset	478	0.21%	235175	1.50%
Berkeley MVP	410	0.18%	34039	0.22%
Berkeley RPT	776	0.35%	326504	2.08%
IAMLAb CMU Pickup Insert	539	0.24%	118055	0.75%
UT Austin Mutex	1283	0.57%	295042	1.88%
Imperial College Sawyer Wrist Cam	145	0.06%	4519	0.03%
Stanford Mask VIT	7788	3.49%	155760	0.99%
Language Table	29554	13.24%	175855	1.12%
Kuka	15903	7.13%	73158	0.47%
BC-Z	4365	1.96%	559009	3.57%
Robo Net	47127	21.12%	895413	5.72%
DLR Sara Pour	171	0.08%	19462	0.12%
Stanford Robocook	2103	0.94%	73493	0.47%
CMU Play Fusion	492	0.22%	192988	1.23%
Bridge	21768	9.75%	500331	3.19%
Furniture Bench Dataset	2763	1.24%	2424703	15.48%
UCSD Pick And Place Dataset	1158	0.52%	45162	0.29%
USC Cloth Sim	684	0.31%	60876	0.39%
Stanford Kuka Multimodal Dataset	2565	1.15%	100021	0.64%
Roboturk	1535	0.69%	127523	0.81%
KAIST Nonprehensile	171	0.08%	25478	0.16%
ASU Table Top	94	0.04%	22029	0.14%
UTokyo Xarm Pick And Place	78	0.03%	4860	0.03%
Berkeley Cable Routing	1266	0.57%	19328	0.12%
Droid	29437	13.19%	2800000	17.88%
UIUC D3Field	164	0.07%	9803	0.06%
Robo Set	15603	6.99%	1042887	6.66%
QUT Dexterous Manipulation	171	0.08%	150698	0.96%
NYU Door Opening Surprising Effectiveness	372	0.17%	11418	0.07%
NYU Franka Play Dataset	311	0.14%	25536	0.16%
Mimic Play	323	0.14%	303738	1.94%
ManiSkill Dataset	21346	9.57%	2969893	18.96%
Columbia CairLab Pusht Real	103	0.05%	20375	0.13%
Conq Hose Manipulation	96	0.04%	4078	0.03%
DLR EDAN Shared Control	88	0.04%	6698	0.04%
Berkeley GNM SAC Son	2526	1.13%	183078	1.17%
Berkeley Autolab UR5	766	0.34%	66425	0.42%
Aloha Mobile	236	0.11%	401754	2.57%
Agent Aware Affordances	101	0.05%	127403	0.81%
Epic Kitchen	58	0.03%	173012	1.10%
PoCo Hammer	220	0.10%	12517	0.08%
PoCo Spatula	142	0.06%	7517	0.05%
Drake Tooluse	925	0.41%	16650	0.11%
PyBullet Grasping Image	1788	0.80%	51852	0.33%
MuJoCo MetaWorld	741	0.33%	34725	0.22%
MuJoCo RoboMimic	180	0.08%	6735	0.04%
Isaac Arnold Image	3214	1.44%	16070	0.10%
PyBullet TriFinger	147	0.07%	89383	0.57%
MuJoCo Adroit	90	0.04%	8010	0.05%
FrodoBot	60	0.03%	12891	0.08%

Table 4: **Detailed Dataset Mixture.** We include the detailed number of trajectories and the number of dataset samples in the training mixture. These include 41 dataset from Open-X [14], 7 datasets from simulation, 3 datasets from human video, and 1 from in-the-wild deployed dataset.

We support various types of policy heads in the network architectures such as standard MLP, transformer decoder, and diffusion policy. For the MLP head, we pooled the trunk feature (e.g. averaging) and then applied a 3-layer MLP. For the transformer decoder, we concatenate learnable tokens to the tokens before feeding into the trunk and use 1D convolution layer on these output tokens to regress actions. For diffusion policy in real world experiments, we use a diffusion head and train with DDPM [28]. Finally, the actions are unnormalized based on dataset statistics and Huber losses are applied for regression. The reason behind Huber loss is to balance between the "difficult frame" in robot trajectories and the easy but lengthy part of the trajectories.

As discussed, HPT is a meta-level architecture where the stem, trunk, and head code are modular and each supports various architectural changes with pre-trained weights. The inference time, which includes all the pre-processing and encoder time, on the local computer with an old NVIDIA RTX 3070 GPU is 39Hz for HPT-base and 33Hz for HPT-xlarge. A modern GPU such as A100 will likely improve the speed by 3-4 times.

A.3 Pre-training Experiment Details.

We train HPT with AdamW [47] optimizer with a weight decay ratio 0.05, and a base learning rate of 0.0002 with a cosine learning rate schedule with warmups and dropouts. We apply proportional scaling of the base learning rate depending on the batch size. To support various horizons during

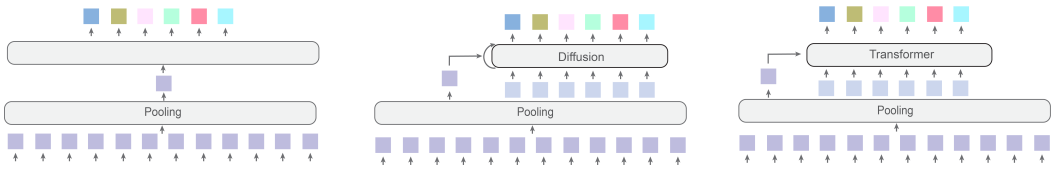

Figure 14: **Flexible Head Architectures in HPT.** We highlight that our HPT architecture is a meta-level architecture for policy learning, and it can work with various head architectures. The important takeaway is the scalable transformer architecture in the middle of the policy to absorb the diverse data and provide tokens for these policy heads to regress on the action outputs.

Method	Dataset	Trajectories	Model Size	Heterogeneous Proprioception
RT-1 [6]	1	0.1M	16M	×
RT-2X [14]	12	-	55B	×
Octo [56]	25	0.8M	93M	×
OpenVLA [36]	25	1M	7B	×
HPT	52	0.2M	1.1B	✓

Table 5: **Experiment Statistics.** By leveraging heterogeneous datasets, the embodiment diversity in data and training scales reaches across several orders. Note that the training flops and the number of tokens are approximated from a single iteration and the model size only counts the trunk parameters (stem and head only have a small active parameter count. HPT is also provided with multiple open-source implementations and extensive simulation evaluation tasks across 6 different benchmarks.

transfer learning, we apply random masking along the time dimensions for each batch during training. Since action trajectories can have imbalanced losses along prediction horizons, we use Huber loss with $\delta = 0.1$ (empirically found). We found the pre-training stage to be stable across various hyperparameters in learning rate schedules and optimizers, and the choice of the validation dataset. The code is open-sourced and the pre-released model can be downloaded easily from Huggingface.

In practice, since training losses can vary across different datasets and our goal is to perform well on all embodiments and tasks, we apply a weighted sampling procedure for data loading. For every training iteration, we sample a dataset with inverse probabilities based on an exponential of its dataset size as a temperature. Specifically, we compute the squared root of each dataset size and sum these sizes to compute a normalization constant. For each batch item, we then sample from these dataset with the corresponding probability. This prevents large datasets from dominating a full training epoch, which is a common practice in multitask learning.

Note that the stem and head for each embodiment are updated in different frequencies than the trunk, similar to a mixture-of-expert [72] training procedure. Especially under distributed training settings, each stem and head is trained with data from a particular embodiment and tasks, and the trunk will accumulate gradients from all batches from training workers. The compute resources for these pre-training experiments range from 8 V-100s to 128 V-100s and the training time spans from half a day to 1 month. The total dataset disk size is around 10Tb and the RAM memory requirement is below 50Gb. See Table 5 for summary details of the experiment.

A.4 Simulation Experiment Details

For simulation benchmarks, we use the released datasets as the expert demonstrations [90, 82, 13]. In summary, Metaworld [90] uses wrist camera view, and Robomimic [50] as well as Simpler [43] uses third-person view, with their own proprioception definitions by the dataset. Fleet-Tools [82] uses both views as inputs and uses the end effector poses as the proprioception inputs. We encode the image using pre-trained frozen ResNet features and normalize the proprioception inputs before passing them into the stem. We train single-task policies for all these simulation benchmarks except for Metaworld.

For the Simpler [43] benchmark, we focus on the `Close-Drawer`, `Move Near`, and `Pick Coke Can` task and Google EDR embodiment with a visual matching setting. We test 9 different initializations with a total of 218 episodes. Note that the simulation tasks have a focus on language conditioning and do not expose proprioception inputs, which is not the most suitable testbed for HPT. To address

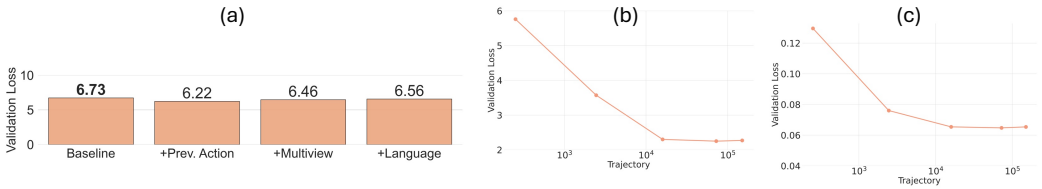

Figure 15: **Additional Architectural Ablation.** (a) We found that architecture changes on HPT-Base such as adding previous actions as inputs, multiview as inputs, and language input can help with HPT pre-training performance. (b,c) We ablate other policy head architectures such as discrete classification heads as well as action token heads by scaling along the number of trajectories. The experiment is conducted under the default setting with HPT-Base, fixed 27 datasets with 1000 max trajectories in each dataset.

these issues, we finetune HPT on the RTX supervised datasets with 79 trajectories as other simulation benchmarks. We use HPT-base as the backbone for this experiment.

By default, we train with 20000 iterations with batch size 512 and small learning rate $1e^{-5}$. The image and state stem are one-layer MLP with hidden dimension 128 and the head is two-layer MLP. We only use an observation window of length 1 and MLP as the policy head. Each training dataset uses from 10-100 trajectories per task and each test covers 50 episodes with different initial conditions. Each trajectory in the simulation has a slight difference in scene initialization. To reduce the variance, we conduct independent training runs 5 times, and the average for each baseline. In Figure 9, we show illustrations of some simulation tasks we evaluated.

A.5 Real-World Experiment Details

Task Definition. We experiment with robotic tool-use tasks `Sweep Leftover`, `Fill Water`, `Scoop Food` and `Switch Insertion` across two different robot setups. While for both setups, we use Franka Panda as the robot, we note that the sensor locations as well as the action spaces are drastically different. We collect approximately 100 demos for each task and evaluate each task for 15 trials to measure the average success rate.

During evaluation, a human supervises the robot at all times. An evaluation episode can be terminated due to safety concerns, robot faults, timeout, etc. An episode is considered successful if it accomplishes the task. In the `Fill Water` task, the success score of 1 means some water is poured into the bowl. In the `Sweep Leftover` tasks, a success score of 1 means all the piles are pushed into the plate, and a success score of 0.5 means some piles are pushed into the plate. In the `Scoop Food` task, a success score of 1 means some dog food is scooped and all is poured into the bowl and a score of 0.5 means some food is scooped. In the `Switch Insertion` task [91], a success score of 1 means the switch is precisely inserted into the three pins on the PCB board. The robot moves to a pre-defined pose before it attempts the insertion. We pick these challenging tasks as they require contact-rich interactions with tools and granular objects, and they require high-precision as well as dense contacts. We make sure the initial conditions of the robots are the same. Due to the complexity of these tasks and human errors, the initial conditions of the object setups are not exactly the same.

Transfer Learning. For the policy head in the real-world experiments, we have experimented with both MLPs and diffusion policies [13]. Fine-tuning only has active parameters of no more than 3Mb, compared to much bigger models (e.g. 100M) often used for a single task in other works. We use an observation history window of 2 with a small learning rate $2e^{-5}$. We train with batch size 256 on a single NVIDIA RTX 2080Ti GPU for 20000 iterations (around 4 hours of wall time).

B Additional Experiments

In this section, we present some additional experiments and ablation studies.

B.1 Additional Simulation Experiments

Based on the training curves of the four baselines in Figure 16 (a), we observe that leveraging a pre-trained HPT representation can often achieve a lower validation loss curve (lower) during fine-tuning.

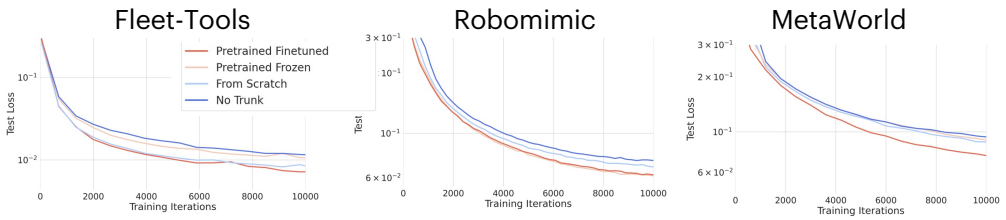

Figure 16: **Transfer Learning Objective.** We run transfer learning across several simulator benchmarks [82, 50, 90]. We compare the validation loss curves of several baselines with and without pre-trained HPT trunks. The pre-trained trunks are trained from the Default Settings.

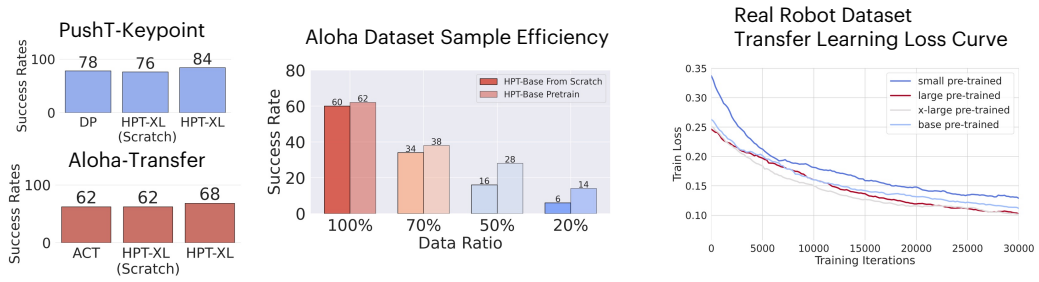

Figure 17: **Simulation Task Performance compared with Single-Task Policy in LeRobot Implementation.** We do evaluation in a different implementation in unseen simulation benchmarks. Left) we show that an improvement in performance can be achieved with pre-trained HPT trunks and outperforms single-task state-of-the-art architectures. We note that HPT trunks have not been pre-trained with diffusion heads and transformer decoder heads. Middle) we show the sample efficiency ablation study for HPT-Base. Right) We show model size ablation study in loss curves.

In Figure 10 (a), we run HPT on the Simpler [43] Benchmark, which allows for comparing with Octo [56], RT1-X, and RT2-X [14] on a high-fidelity simulation. We focus on three different tasks `Close Drawer`, `Move Near`, and `Pick Coke Can` in the Google EDR embodiment. For each task, we test several different initializations with a total of over 300 episodes for all tasks. Note that the pre-training corpus of HPT-S does not include [6], and simulation tasks have a focus on language conditioning and do not expose proprioception inputs, which is not suitable for HPT. To address these issues, we generate training data using the RT-1[6] as the expert, and finetune HPT on the RT-1X supervised datasets with around 50 trajectories and the simulation protocol. We use HPT-base with language tokenizers as the backbone for this experiment. We use the results from the paper [43].

Additionally, under the LeRobot [8] implementation, we compare HPT with state-of-the-art single-task policy architecture such as Diffusion Policy [13] and ACT [92]. In particular, we inherit a simple version of these complex policies as the head architectures (Figure 14). We run full training runs with 50 evaluation trials every 20000 training steps, and use the maximum task success rates during the 100000 training steps for comparisons. In Figure 17, we compare HPT with DP on the PushT task with keypoint representations using the diffusion head and achieve similar success rates at 78%. We also compare with ACT on the Aloha Transfer Box task with the image representations using the transformer decoder head and achieve similar success rates at 60%. This showcases the flexibility of HPT architecture to work with state-of-the-art policy architectures for high-frequency control in fine-grained tasks. Moreover, in the experiment with 50 episodes in total and HPT-base, we observe an improvement in sample efficiency with pre-trained models. We also see improvement in transfer learning loss with pre-trained models at increasing scales.

In Figure 10 (b), we ablate on the number of visual and proprioceptive tokens in the simulation transfer learning experiments. We observe that missing either information hurts the performance of the downstream policies. See Section A.4 for more implementation and experiment details.

B.2 Ablation Study on the Stem

In this experiment, we fix the number of datasets (27) in Open-X datasets and use a maximum of 1000 trajectories for each dataset. We consider several ablations on the stem part of the HPTs.

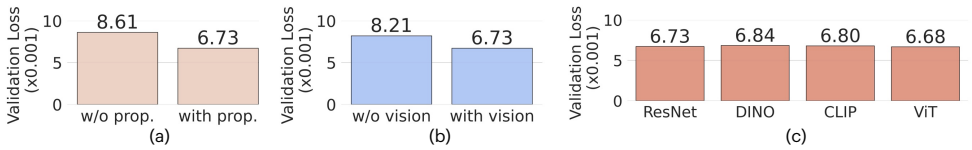

Figure 18: **Ablation Study on HPT Stem.** We ablate the pre-training performance for (a) proprioception, (b) vision stems, and (c) vision encoders. Setting: HPT-S, batch 256, iterations 80000, 27 datasets with a maximum of 1000 trajectories for each dataset.

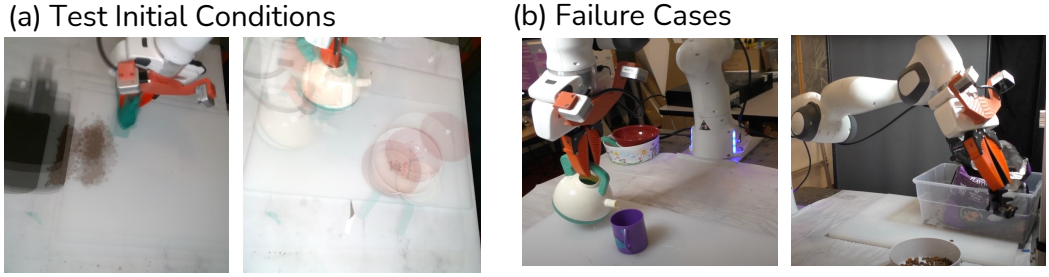

Figure 19: **(a) Initial Condition Overlay.** We visualize different rollout initial conditions during test times. **(b) Failure Cases of the Learned Policy in the Real World.** The robot sometimes has issues executing very precise manipulation.

Specifically, in Figure 18 (a, b), we observe an increase in validation losses when not using either the proprioception information or the vision. Intuitively, not using such proprioception or vision information would make learning action predictions from heterogeneous datasets very challenging. This also implies that both information are critical for policy pre-training at scale.

We also conduct an ablation experiment over vision backbones on a smaller subset of the training datasets among several popular vision encoders such as ViT-base [24] and DiNO [59] (Figure 18). Further ablating on input image resolution or joint finetuning of the vision backbone on the downstream task success rates will be interesting future work. Although the default implementation focuses on single-view visual information, the stem can naturally extend to multiple views and other modalities such as languages and action history.

B.3 Pre-training Ablation Study

In Figure 15, we conduct several ablations to pre-train the HPTs, such as using multiple views to provide 3D information implicitly, using languages for task guidance, and using previous action trajectories as additional ablations. We found these ablations to improve over the default HPT setting with a single view and vision and proprioceptive inputs. These ablations are more pronounced in certain datasets such as multiple views for insertion [68], and language modality for Language Table [48]. Using previous action trajectories is helpful in providing additional context and embodiment information as well. We believe that integrating multiple modalities and investigating how to handle missing modalities is an exciting future direction. We leave the exploration of these ablations to future work.

From the architecture perspective, we also ablate over the token sizes and observation history and do not find a big effect on the averaged validation loss. We hypothesize there is a trade-off between the information given to the policy and the desired generalization. In Figure 15(b,c), we have also experimented with a discretized version [6] of the policy head and architecture that uses additionally learned position encodings as action tokens for transformer, with conv1D head for action regression. We opt for regression on continuous values for its generality. An initial investigation of the attention map of the transformer blocks shows that there is a dense attention weight attributed to the proprioception and vision tokens.

C Failure Cases

In Figure 19, we show some failure cases of the learned HPT policies in the real world. One of the common issues is overshooting or undershooting. For example, the policies tend to pour the water before it reaches the mug or pour the dog food a bit in front of the bowl. These issues could be due to the spatial precision of the policies and due to the data qualities failing to uncover the causal relationships. More targeted data recollection and better finetuning of the vision encoders might help address these issues.

D Discussion and Future Directions

We first discuss the metric used for measuring pre-training performance, or intrinsic evaluation. Validation (or training) loss is a common metric to evaluate the progress of large-scale pre-training [34]. It is based on the goal of training a generalist model, where even fitting all the data can be a challenge [58], as opposed to training a specialist model where overfitting is often appreciated. This is also considered a step towards more scientifically studying pre-training and scaling behavior in robotics [30]. Previously, people often rely on a single binary metric for evaluating task success rates in the real world, with only some amount of test trials.

Admittedly, there are several caveats to this metric. From the evaluation perspective, the averaged validation loss depends on the overall multitask losses of all datasets. For example, increasing the number of trajectories in one dataset might lower validation loss on the associated dataset, but may not lead to an overall loss decrease. In practice, the exact subset for evaluation and the number of training steps in each dataset also play a role in the averaged validation loss metric. For example, when evaluating whether additional datasets to the default setting contribute to the representation learning, selecting the default datasets allows us to compare on the same metric. But these datasets are inevitably trained less, if we fix the number of total training iterations. Moreover, the validation loss during pre-training and the downstream policy learning performance have an evaluation gap due to task differences, and downstream policy learning and task execution have another evaluation gap due to closed-loop execution.

Given the recent surge of scaled data, robot learning is still limited by its generality because of the heterogeneity, including different embodiments, tasks, and environments where the robots are operated. We propose HPT, a novel architecture and framework to embrace this heterogeneity through pre-training. We align proprioception and vision information of different embodiments through a modular stem, a shared scalable transformer trunk, and task-specific heads to actions. We explore and scale HPT with heterogeneous datasets to over 50 available datasets. The learned representation can be transferred and improved performance in both simulation and the real world. We hope this perspective will inspire future work in handling the *heterogeneous nature* of robotic data.

Since fine-tuning is still required for robotics generalist models, future research directions include exploring different algorithms including network architectures that incorporate embodiment-specific structures, such as URDF, into network architecture as well as tokenization. One can also explore training objectives beyond supervised learning. As we move towards scaling robot pre-training, more high-quality diverse datasets with clear annotations would be crucial, including teleoperation data, simulation data, human videos, and deployed robot data. Understanding what kinds of mixture will contribute to better representations is interesting future work.

Due to the complexity of real-world evaluation, large-scale unified simulation benchmarks with varying dexterity and generalization challenges would be very crucial to consistently compare among different models. For real-world policy performance, we believe that extending to longer-horizon fine manipulations with a bimanual or mobile setup would be interesting future works.

Future works can also study and extend scaling laws in policy learning, and explore representations from heterogeneous data in other domains beyond robotic policy learning. Finally, leveraging more modalities and domains in robotics such as 3D point clouds, tactile data, simulation domains, and human data, etc, is worth investigating.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction, we make the relevant claims on heterogeneous pre-training, scaling behaviors, and transfer learning experiments as backed by our experiment sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: In Section 6, we discuss the limitations and potential future work of this project.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: There is no theoretical results associated with this work.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The implementation details are disclosed to full extent in the appendix. The code are also open-source for checking experiments relevant to the main results.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will open-source the code with detailed instructions to reproduce the main experimental results. The data can be generated on the fly as they come from online public datasets in Figure 13.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Section §4 and Section §A, we discuss all the relevant training and test detials, including dataset splits, training hyperparameters, type of optimizers. We will also open-source our code to follow open research.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in the appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Most training curve experiments (measured with validation losses) in Figure 8, 6, 5,7 are relatively stable with statistical significance. For simulation experiments in Figure 16 with task successes as the metric, we run experiments and average across 5 trials, which results in stable and small variances. For real world task performance experiments in Figure 11 and 12 and 3, we report the standard errors across different trials for each model for each task.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Section A.3, we discuss the compute resources, including workers, memory, time of execution for experiments in this work.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The code of ethics is strongly followed in this work.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The project focuses on pre-training from heterogeneous robotic data. There is no direct societal impacts associated with it.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for the responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We released a series of robotic models that are trained on action data and output action data. The data all have public licenses and do no contain any sensitive or risky information.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes] The released code, data, and models are under MIT license.

    Justification: The code and models are free to use by most audience for the purpose of advancing the field.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes] The new released models and codebases are well documented for the released version.

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA] .

    Justification: This project does not include crowdsourcing experiments with human subjects. The involved human video datasets are directly used from the dataset. The robot teleoperation is conducted by the authors.

    Guidelines: [NA]

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

    Justification: There is no human subjects with crowsourcing experiments in this work.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.