Assistive Tele-op: Leveraging Transformers to Collect Robotic Task Demonstrations

Henry M. Clever[1,2], **Ankur Handa**[*1], **Hammad Mazhar**[1], **Kevin Parker**[1], **Omer Shapira**[1], **Qian Wan**[1], **Yashraj Narang**[1], **Iretiayo Akinola**[1], **Maya Cakmak**[1], **Dieter Fox**[1]
[1]NVIDIA, USA. [2]Georgia Institute of Technology, Atlanta, GA, USA.

Abstract

Sharing autonomy between robots and human operators could facilitate data collection of robotic task demonstrations to continuously improve learned models. Yet, the means to communicate intent and reason about the future are disparate between humans and robots. We present Assistive Tele-op, a virtual reality (VR) system for collecting robot task demonstrations that displays an autonomous trajectory forecast to communicate the robot's intent. As the robot moves, the user can switch between autonomous and manual control when desired. This allows users to collect task demonstrations with both a high success rate and with greater ease than manual teleoperation systems. Our system is powered by transformers, which can provide a window of potential states and actions far into the future – with almost no added computation time. A key insight is that human intent can be injected at any location within the transformer sequence if the user decides that the model-predicted actions are inappropriate. At every time step, the user can (1) do nothing and allow autonomous operation to continue while observing the robot's future plan sequence, or (2) take over and momentarily prescribe a different set of actions to nudge the model back on track. We host the videos and other supplementary material at `https://sites.google.com/view/assistive-teleop`.

1 Introduction

Manually teleoperating robots to collect task demonstrations at scale is laborious and challenging. We present a shared-autonomy-based method using neural networks to forecast robot trajectories that substantially reduces manual teleoperation time while maintaining a high success rate. Specifically, we adapt a learned model to do trajectory auto-complete, *i.e.,* given an initial sequence of states and actions, the network learns to complete the rest of the trajectory. The user can either accept the model's suggestions or provide manual corrections while observing their effect on the model forecast. We leverage transformers [29] for modeling the states and actions through time, which are well-suited to modeling long sequences of information with complex dependencies (see Fig. 1-*left*). Their self-attention mechanism can holistically understand a robot trajectory, rather than emphasizing adjacent connections between states. When taking as input a sequence of past actions, the transformer can look far into the future and predict future actions. By integrating this into a robot manipulation environment with VR, a user can decide if executing the future actions is appropriate, or otherwise take momentary control of the system to provide a better demonstration.

This work explores transformer model prediction for a set of 7 manipulation tasks involving pick-and-place across industrial, household, and caregiving robot settings (Fig. 1-*right*). We train on a large number of demonstrations (> 500) from the open-source RoboTurk [18] dataset and perform few-shot learning by fine-tuning on a small number of expert demonstrations ($<= 60$) and show that it is able to succeed autonomously for 67.1% of task scenarios during test time. When the model predicts a wrong sequence of actions, the user takes control and gives the robot a nudge to get it back on track. This significantly improves performance, and resulting in a 96.1% success rate. Importantly, our Assistive Tele-op system reduces manual control time to collect demonstrations by a factor of 5. It is worth stressing that while the use of interventions is similar to DAgger [27], we aid the user by

[*]corresponding author: ahanda@nvidia.com

NeurIPS 2021 Workshop on Robot Learning: Self-Supervised and Lifelong Learning, Virtual, Virtual

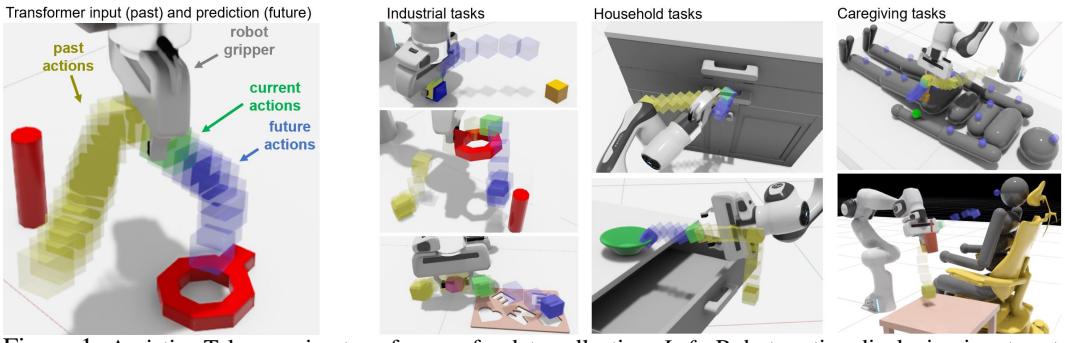

Figure 1: Assistive Tele-op using transformers for data collection. *Left:* Robot motion displaying input past actions (yellow) and output predicted actions (blue) to the user. *Right:* We test our method on 7 task scenarios across industrial (block stacking, nut assembly, kit assembly), household (cabinet drawer and bowl manipulation), and caregiving (itch scratching and drinking) tasks.

displaying a live forecasting model of the robot trajectory so corrections can be made well before the robot makes a mistake.

Our transformer embedding has a single state space of fixed length that we can map a variety of tasks into. Further, by defining the robot state and actions entirely in end-effector space like [19] and learning a world model [11] in this space, the transformer can be interchanged between robots and simulation environments. Our transformer is pre-trained on demonstrations collected in a different physics simulator and robot than what we use in this work. While only the predicted robot actions are necessary for control, we use an additional loss on object pose states in the scene, which boosts performance. Transformers are better able to parse sequence information using a positional embedding vector, which we compute using the cumulative distanced traversed by the end effector in Euclidean space (similar in spirit to [26]). Finally, by training with a BERT-style zero padding [6] on the input to the transformer corresponding to future states, we can make an arbitrary number of model predictions far into the future – which allows the user to understand what the robot is about to do.

In summary, the work makes the following contributions: (1) Evidence that pre-trained transformers can be fine-tuned for few-shot generalization to new robot manipulation tasks, and (2) Assistive Tele-op, a VR system with live model forecasting, to assist users with collecting robotic task demonstrations at a high success rate and with substantially reduced manual control time. As the user collects more demonstrations, they can be fed back to the model for continual learning.

2 Related Work

Task demonstrations for robot manipulation can be collected using automatic methods such as trajectory optimization [3, 15] and reinforcement learning [28, 16, 27], as well as manual methods of kinesthetic teaching and teleoperation [2]. The former require carefully tuned reward functions, while the latter can be laborious to collect. Virtual reality [7, 31] can help, but the human effort remains considerable for complex tasks, and when many demonstrations are required. Shared autonomy [12, 9, 25, 14] offers a better solution to collecting large-scale data. These works blend robot and user intent using optimization [9], reinforcement learning [25], and learned coarse-to-fine user precision [14], while ours lets the user look far into the future to understand the autonomous prediction. A similar forecasting method was proposed by Liu *et al.* [17], but it is used in a behavior cloning loss function, rather than for communicating intent to the user. We take some insight from Pérez-D'Arpino and Shah [22], who overlay a series of robot configuration renderings through time to show planned motions. Later, they used this feature for human robot teaming to allow an operator to either accept a suggested motion plan or momentarily intervene [21].

Transformers have only recently gained traction in robotics. Janner *et al.* [13] reframed RL as sequence modeling, and used transformers to control humanoid walking. Chen *et al.* [5] concurrently explored this in the context of a game environment. Common transformer implementations have used sinusoidal positional embeddings to better model the order of words [29]. However, Chen *et al.* [5] used an episodic timestep positional embedding and Press *et al.* [23] added a linear bias to each attention score. We take inspiration from these and use a cumulative distance embedding to provide information for how far the end effector has traversed.

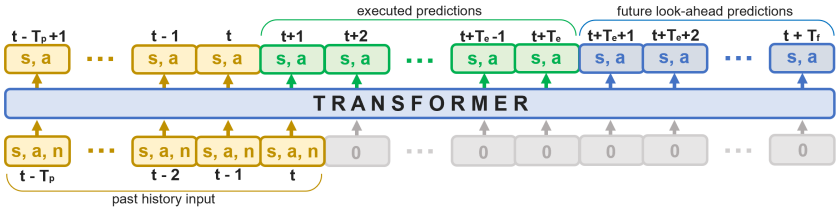

Figure 2: The transformer takes as input a past history of T_p states, actions, and positional embeddings, and outputs predicted states and actions. The user can observe predictions far into the future (e.g. $T_f = 300$ timesteps) and may choose to execute $T_e <= T_f$ of those predictions. The input for future states is padded with zero, forcing the model to learn future predictions only from past inputs.

3 Methods

The transformer takes as input a trajectory of states and actions $\tau_x = \{\mathsf{s_x}, \mathsf{a_x}\}$ and positional encoding vector $\boldsymbol{n}$, and outputs a predicted trajectory $\widehat{\tau}_y = \{\widehat{\mathsf{s}}_\mathsf{y}, \widehat{\mathsf{a}}_\mathsf{y}\}$. As shown in Fig. 2, the input consists of T_p past timesteps of information, while the output contains additional information on future predictions up to timestep T_f. The state at each timestep $\mathsf{s_t}$ is a vector consisting of a global robot end effector pose $\mathsf{s_{r,t}} \in \mathbb{R}^7$, continuous gripper state $\mathsf{s_{g,t}} \in \mathbb{R}^1$, and the local pose of J objects in the environment $\{\mathsf{s_{o1,t}} \ldots \mathsf{s_{oJ,t}}\} \in \mathbb{R}^{7\times J}$ relative to the robot end effector. Each 7D pose contains a position and quaternion. At the input, this is fed into a state embedding function $\mathcal{F}_\mathsf{s}$ which we represent with a single fully connected network layer: $\mathsf{s_{emb,t}} = \mathcal{F}_\mathsf{s}(\mathsf{s_t})$, where $\mathsf{s_{emb,t}} \in \mathbb{R}^{128}$. The action at each timestep $\mathsf{a_t}$ is a vector consisting of the local target pose of the robot end effector $\mathsf{a_{r,t}} \in \mathbb{R}^7$ and the binary gripper command $\mathsf{a_{g,t}} \in \mathbb{R}^1$. At the input, this is fed into an action embedding function $\mathcal{F}_\mathsf{a}$, which we represent with a single fully connected network layer: $\mathsf{a_{emb,t}} = \mathcal{F}_\mathsf{a}(\mathsf{a_t})$, where $\mathsf{a_{emb,t}} \in \mathbb{R}^{128}$. Additionally, the network contains a positional embedding to measure the distance and rotation the end effector has traversed at each timestep along the trajectory from timestep $1 \ldots t$. The positional embedding $\mathsf{n_t} \in \mathbb{R}^1$ is an integer token computed at each timestep as: $\mathsf{n_t} = \sum_{t'=2}^{t} \sum_{j=1}^{8} ||\mathsf{c_{r,j,t'}} - \mathsf{c_{r,j,t'-1}}||_2$ where each $\mathsf{c_r}$ is coordinate of a corner of a 3D bounding box around the end effector at a given timestep, and is a function of end effector position $\mathsf{p_r}$ and quaternion $\mathsf{q_r}$ in the global frame [1]. We chose this pose representation because it casts rotation in position space, which mitigates the problem of combining heterogenous terms in the same function. The token $\mathsf{n_t}$ is fed into a learned positional embedding layer, $\mathcal{F}_\mathsf{n}$: $\mathsf{n_{emb,t}} = \mathcal{F}_\mathsf{n}(\mathsf{n_t})$, where $\mathsf{n_{emb,t}} \in \mathbb{R}^{128}$. This is added to each state and action embedding. At each timestep the transformer receives the input vector computed as $\mathsf{x_t} = \mathsf{LN}\big((\mathsf{s_{emb,t}} + \mathsf{n_{emb,t}}) \oplus (\mathsf{a_{emb,t}} + \mathsf{n_{emb,t}})\big)$ where $\mathsf{x_t} \in \mathbb{R}^{256}$ and LN represents layer normalization. The transformer outputs predicted vector $\widehat{\mathsf{y}}_\mathsf{t} \in \mathbb{R}^{256}$, which is then decoded with linear layers on the output mirroring those on the input, represented by $\mathsf{g_s}$ and $\mathsf{g_a}$ for the states and actions. The output of these decoding layers contain predicted states and actions, the sequence of which forms trajectory $\widehat{\tau}_y$. The robot is controlled with the predicted end-effector pose actions by using Riemannian motion policies [24]. See Appendix A for training details.

4 Evaluation

We evaluated the transformer and Assistive Tele-op system across tasks representing industrial [18, 30], household [4, 8], and caregiving [7, 10] task scenarios. For this, we used both existing data from RoboTurk [18] and new data that we collected with VR. See Appendix B for details.

Automatic model prediction. First, we evaluated the pretrained model. We trained it for $>$ 2 days on raw RoboTurk data with the Baxter robot in MuJoCo, and evaluated it in a reconstructed environment with the Franka robot in NVIDIA Omniverse. We evaluated the pretrained model on both 50 scenes from the training data and on 50 test scenes. The training data evaluation provides a measure of the sim2sim transfer between simulators. The test data evaluation shows generalization to previously unseen initial item locations. We conducted more tests in tasks A-G in Omniverse (see Table 1). For each, we fine-tuned a transformer from the pretrained model and trained a transformer from scratch (no pretraining). We used a fixed-time budget for this comparison. Each model is tested on 50 new object configurations, except itch scratching, which is tested on 57.

Assistive Tele-op: We evaluated the human-in-the-loop Assistive Tele-op system using both task success rate and manual demonstration time elapsed. A researcher used the HTC Vive VR system to communicate with the transformer prediction when controlling the robot, as shown in Fig. 3. For each task, the model began in automatic mode, and the user clicked a button on the interface to stage an intervention when the robot moved in an inappropriate direction (e.g. away from the bowl rather

Table 1: Success rate and demonstration time for models trained from scratch and pretrained models.

Task	No. training demos	Success rate				Manual demonstration time (s)		
		Manual tele-op	Auto no pretr.	Auto w/pretr.	Assistive Tele-op	Manual tele-op	Auto	Assistive Tele-op
RoboTurk pick/place [18]	533	-	0.84 / 0.66†	-	-	-	-	-
A. Block stacking	35	1.00	0.60	0.74	1.00	14.5	0.0	3.5
B. Round nut assembly [18]	50	1.00	0.38	0.70	0.94	72.3*	0.0	7.9
C. Assembly kit - hexagon [30]	35	1.00	0.00	0.10	0.92	38.5	0.0	5.6
D. Cabinet drawer opening	35	1.00	0.98	1.00	1.00	14.0	0.0	N/A
E. Put bowl in drawer	50	1.00	0.52	0.64	1.00	28.3	0.0	4.5
F. Humanoid itch scratching	57	1.00	0.46	0.70	0.93	23.0	0.0	4.2
G. Humanoid drinking	35	1.00	0.68	0.82	0.94	18.8	0.0	7.7
Overall (A-G average)	-	**1.000**	**0.517**	**0.671**	**0.961**	**29.9**	**0.0**	**5.5**

**Based on approximate RoboTurk data frequency of 15 Hz.* †*Results on training data / results on test data. All results collected in NVIDIA Omniverse with Franka.*

than toward it when picking it up). We score Assistive Tele-op success as the ability to complete a demonstration on a new task scenario – either with fully automatic prediction, or with intervention assistance. Assistive Tele-op can only improve the demonstration success rate. For all Assistive Tele-op scenarios, we used the transformer with pretraining. Time taken for manual demonstration is compared among manual, automatic, and assistive modes. In manual mode, this is the average time to collect each full demonstration. In automatic, it is 0, because no human effort is required. In Assistive Tele-op mode, it is the average intervention time for all demonstrations per task.

5 Results and Discussion

Pre-trained transformers can be used for few shot generalization to new tasks. For each task, we compared models trained from scratch to those fine-tuned starting with a pre-trained model. The pre-trained and fine-tuned models performed better across all tasks (Table 1). Testing scenarios were successful in most cases, except Task C. See Appendix C for details.

Models trained with our method can transfer between different simulators and robots. The RoboTurk dataset was collected in MuJoCo using a Baxter robot. However, we tested the transformer model using a Franka robot in the Omniverse [20] simulator. These environments have a different robot configuration, control method, data collection rate, and simulation method, but the model performs well, showing good sim2sim transfer. Task B, also from RoboTurk, provides further evidence for this. Formulating the model in end effector space is key to this transfer, by obviating the configuration space representation that is different between Baxter and Franka.

Human interventions can get the model back on track. When a human intervenes in the event of failure to nudge the robot back on track, success increases from 67.1% to 96.1%. See Fig. 3.

Collecting Assistive Tele-op demonstrations with model prediction is easier. For purely manual teleoperation, the average demonstration time is 29.9 seconds. For Assistive Tele-op, the average human demonstration time to get the robot back on track is 5.5 seconds across task scenarios that otherwise cannot be completed with fully automatic model prediction.

Auxiliary object pose loss boosts performance. We ablated the auxiliary loss on the objects in the scene, and found that for the round nut assembly (Task B) success for the pretrained model, performance drops from 70% to 58%.

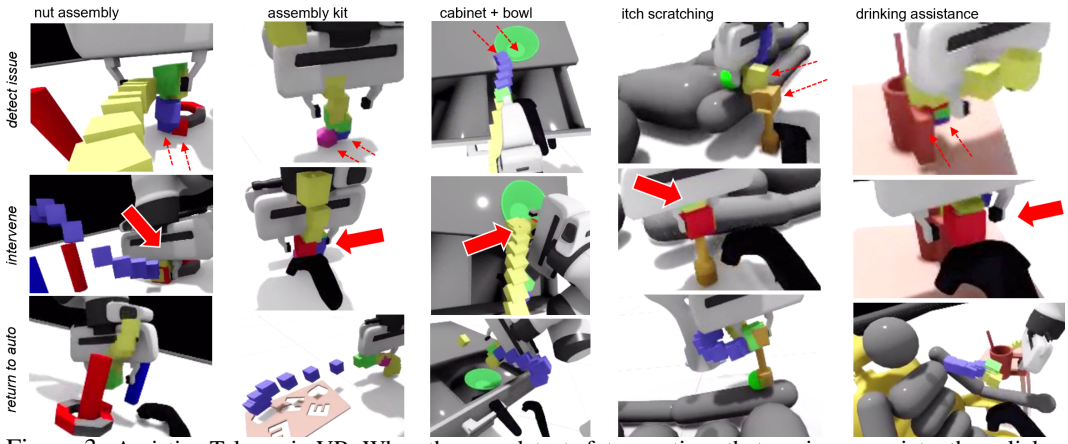

Figure 3: Assistive Tele-op in VR. When the user detects future actions that are inappropriate, they click a button to take over control. After a momentary nudge, they return control to the transformer.

References

[1] Arthur Allshire, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger. *arXiv preprint arXiv:2108.09779*, 2021.

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] Arthur E Bryson and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. Routledge, 2018.

[4] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[5] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Zackory Erickson, Yijun Gu, and Charles C Kemp. Assistive vr gym: Interactions with real people to improve virtual assistive robots. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 299–306. IEEE, 2020.

[8] Caelan Reed Garrett, Chris Paxton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. Online replanning in belief space for partially observable task and motion problems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5678–5684. IEEE, 2020.

[9] Deepak Gopinath, Siddarth Jain, and Brenna D Argall. Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE Robotics and Automation Letters*, 2(1):247–254, 2016.

[10] Phillip M Grice and Charles C Kemp. In-home and remote use of robotic body surrogates by people with profound motor deficits. *Plos one*, 14(3):e0212904, 2019.

[11] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *arXiv preprint arXiv:1809.01999*, 2018.

[12] Ioannis Havoutis and Sylvain Calinon. Learning from demonstration for semi-autonomous teleoperation. *Autonomous Robots*, 43(3):713–726, 2019.

[13] Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*, 2021.

[14] Hong Jun Jeon, Dylan P Losey, and Dorsa Sadigh. Shared autonomy with learned latent actions. *arXiv preprint arXiv:2005.03210*, 2020.

[15] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, volume 27, pages 1071–1079. Citeseer, 2014.

[16] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[17] Yifang Liu, Diego Romeres, Devesh K Jha, and Daniel Nikovski. Understanding multi-modal perception using behavioral cloning for peg-in-a-hole insertion tasks. *arXiv preprint arXiv:2007.11646*, 2020.

[18] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[19] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1010–1017. IEEE, 2019.

[20] NVIDIA. Omniverse multi-GPU real-time simulation and collaboration platform. Available at `https://developer.nvidia.com/nvidia-omniverse-platform` (2021/09/17), 2021.

[21] Claudia Pérez-D'Arpino, Rebecca P Khurshid, and Julie A Shah. Experimental assessment of human-robot teaming for multi-step remote manipulation with expert operators. *arXiv preprint arXiv:2011.10898*, 2020.

[22] Claudia Pérez-D'Arpino and Julie A Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4058–4065. IEEE, 2017.

[23] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation.

[24] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

[25] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Shared autonomy via deep reinforcement learning. *arXiv preprint arXiv:1802.01744*, 2018.

[26] Jürgen Schmidhuber Róbert Csordás, Kazuki Irie. The devil is in the detail: Simple tricks improve systematic generalization of transformers. *arXiv preprint arXiv:2108.12284*, 2021.

[27] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[30] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020.

[31] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018.

Assistive Tele-op: Leveraging Transformers to Collect Robotic Task Demonstrations **APPENDIX**

Henry M. Clever[1,2], **Ankur Handa**[*1], **Hammad Mazhar**[1], **Kevin Parker**[1], **Omer Shapira**[1], **Qian Wan**[1], **Yashraj Narang**[1], **Iretiayo Akinola**[1], **Maya Cakmak**[1], **Dieter Fox**[1]
[1]NVIDIA, USA. [2]Georgia Institute of Technology, Atlanta, GA, USA.

Appendix A: Network Training

During training, a sub-sequence of length $T_s = 400$ from is sampled from a task demonstration of trajectory length T_d, where $T_d > T_s$ (recall Fig. 2). The network takes input of sequence length T_s of state and action pairs, τ_x, associated with each time-step. Inspired by BERT [2] masking, we only keep the first T_p time-steps and mask the remaining $T_s - T_p$ time-steps with zeros. The network is trained to predict the corresponding state and action pairs, τ_y, at masked out time-steps. The input length T_p is chosen at random during training to force the transformer to make future predictions of an arbitrary horizon length. It is sampled from a uniform distribution $T_p \sim \mathcal{U}(1, 350)$, such that the future prediction length is at least 50 timesteps.

We denote $\mathsf{s}_{\mathsf{r},\mathsf{t}} = [\mathsf{p}_{\mathsf{r},\mathsf{t}}, \mathsf{q}_{\mathsf{r},\mathsf{t}}]$ to be the state of the robot end-effector composed of position and orientation and time t and $\mathsf{a}^{\mathsf{r}}_{\mathsf{t}} = [\mathsf{p}_{\mathsf{r}_{\mathsf{target}},\mathsf{t}}, \mathsf{q}_{\mathsf{r}_{\mathsf{target}},\mathsf{t}}]$ as the action composed of target end-effector position and orientation. Similarly, we define current gripper state $\mathsf{s}_{\mathsf{g},\mathsf{t}} \in [0, 1]$ and target gripper state $\mathsf{a}_{\mathsf{g},\mathsf{t}} \in [0, 1]$. Object i in the scene is also represented by its state vector $\mathsf{s}_{\mathsf{oi},\mathsf{t}} = [\mathsf{p}_{\mathsf{oi},\mathsf{t}}, \mathsf{q}_{\mathsf{oi},\mathsf{t}}]$ composed of its position and orientation and there is no action associated with it. The network takes an input sequence of states $\mathsf{s}_\mathsf{x}{=}\{\mathsf{s}_{\mathsf{r},\mathsf{t}}, \mathsf{s}_{\mathsf{o}1,\mathsf{t}} \cdots \mathsf{s}_{\mathsf{o}\mathsf{J},\mathsf{t}}, \mathsf{s}_{\mathsf{g},\mathsf{t}}\}^{\mathsf{k}}_{\mathsf{t}=0}$ composed of robot end-effector state, gripper state and states of J objects in the scene and actions $\mathsf{a}_\mathsf{x}{=}\{\mathsf{a}_{\mathsf{r},\mathsf{t}}, \mathsf{a}_{\mathsf{g},\mathsf{t}}\}^{\mathsf{k}}_{\mathsf{t}=0}$ associated with the robot end-effector and predicts the future sequence of states $\mathsf{s}_\mathsf{y}{=}\{\mathsf{s}_{\mathsf{r},\mathsf{t}}, \mathsf{s}_{\mathsf{o}1,\mathsf{t}} \cdots \mathsf{s}_{\mathsf{o}\mathsf{J},\mathsf{t}}, \mathsf{s}_{\mathsf{g},\mathsf{t}}\}^{\mathsf{T}}_{\mathsf{t}=\mathsf{k}+1}$ and corresponding actions $\mathsf{a}_\mathsf{y}{=}\{\mathsf{a}_{\mathsf{r},\mathsf{t}}, \mathsf{a}_{\mathsf{g},\mathsf{t}}\}^{\mathsf{T}}_{\mathsf{t}=\mathsf{k}+1}$. We predict the future states and actions given the current states and actions using a GPT-style transformer: $\widehat{\mathsf{s}}_\mathsf{y}, \widehat{\mathsf{a}}_\mathsf{y} = \mathsf{GPT}(\mathsf{s}_\mathsf{x}, \mathsf{a}_\mathsf{x})$.

A.1 Loss Function

The loss function used to train the transformer model has many different components that induce different properties on the learned model. We compute two loss functions related to end-effector state and action, two for gripper state and action prediction, and one loss function on the state for each object in the scene.

It is worth noting that since the state and actions are composed of positions and orientations, computing the loss function by weighting rotation and translation with a hyper-parameter is not ideal. Instead we compute the loss directly in euclidean space by representing 3D locations of the 8 corners of a cube with corresponding position and orientation and computing the loss function as euclidean distance between predicted and ground truth 3D positions [1].

We compute the end-effector state loss by mapping its position and orientation to 8 corners of the rigid cube both for the predictions and ground truth:

*corresponding author: ahanda@nvidia.com

$$\widehat{\mathsf{c}}_{\mathsf{s}_\mathsf{r}} = \mathsf{CORNERS}(\widehat{\mathsf{p}}_{\mathsf{s}_\mathsf{r}}, \widehat{\mathsf{q}}_{\mathsf{s}_\mathsf{r}}) \tag{1}$$

$$\mathsf{c}_{\mathsf{s}_\mathsf{r}} = \mathsf{CORNERS}(\mathsf{p}_{\mathsf{s}_\mathsf{r}}, \mathsf{q}_{\mathsf{s}_\mathsf{r}}) \tag{2}$$

where $\mathsf{c}_{\mathsf{s}_\mathsf{r}} \in \mathbb{R}^{8\times 3}$ are the 3D positions of the 8 corners of the bounding box extents of the end effector. The loss function is the L_2 distance of the corresponding 8 corners of ground truth and predictions:

$$\mathcal{L}_{\mathsf{s}_\mathsf{r}} = \sum_{t=T_p}^{T_s} ||\widehat{\mathsf{c}}_{\mathsf{s}_\mathsf{r},\mathsf{t}} - \mathsf{c}_{\mathsf{s}_\mathsf{r},\mathsf{t}}||_2 \tag{3}$$

Similarly, we can define the loss function on the action space where the predictions are end-effector target positions and orientations as well as loss function object state predictions which also involve the position and orientations.

To compute loss on the gripper state we use binary cross entropy

$$\mathcal{L}_{\mathsf{s}_\mathsf{g}} = \sum_{t=T_p}^{T_s} \mathsf{BCE}(\widehat{\mathsf{s}}_{\mathsf{g},\mathsf{t}}, \mathsf{s}_{\mathsf{g},\mathsf{t}}) \tag{4}$$

$$\mathcal{L}_{\mathsf{a}_\mathsf{g}} = \sum_{t=T_p}^{T_s} \mathsf{BCE}(\widehat{\mathsf{a}}_{\mathsf{g},\mathsf{t}}, \mathsf{a}_{\mathsf{g},\mathsf{t}}) \tag{5}$$

The total loss is sum of the individual losses:

$$\mathcal{L}_{\mathsf{TOTAL}} = \mathcal{L}_{\mathsf{s}_\mathsf{r}} + \mathcal{L}_{\mathsf{a}_\mathsf{r}} + \sum_{i=1}^{J} \mathcal{L}_{\mathsf{s}_{\mathsf{oi}}} + \lambda(\mathcal{L}_{\mathsf{s}_\mathsf{g}} + \mathcal{L}_{\mathsf{a}_\mathsf{g}}) \tag{6}$$

Appendix B: Evaluation Details

B.1 Data collection

Existing data - Roboturk. The Roboturk simulation dataset was collected in the Mujoco [3] simulator with a Baxter robot, but we found that many demonstrations played back successfully in the NVIDIA Omniverse simulator with a Franka robot when controlling the robot in end effector space. This dataset consists of over 6000 crowd-sourced human demonstrations for pick and place tasks with 4 objects (cereal box, milk jug, bread, and coke can) and nut assembly tasks. Of these, we selected 533 pick and place demonstrations for pretraining the transformer. We chose the first 533 demonstrations that had an overall time of less than 900 timesteps, because we observed that shorter demonstrations had better quality. We hand-selected demonstrations for the round nut assembly (Task B) by choosing the first 50 that played back smoothly in our recreation of the scene in Omniverse with the Franka robot.

New data. An HTC Vive virtual reality headset was used by a researcher to collect data in Omniverse with Franka. To create variation in each scene, we sampled initial scene object poses from the following uniform noise distributions for both training and testing scenes: (A) block stacking - blue picked block from $\pm$5cm planar translation and $\pm 45^\circ$ rotation, orange stacked block from $\pm$5cm planar translation. (C) assembly kit - pink hexagon from $\pm$7.5cm planar translation and $\pm 180^\circ$ rotation. (D) cabinet, $\pm 45^\circ$ rotation. (E) put bowl in cabinet, cabinet from $\pm 45^\circ$ rotation, green bowl from $\pm$10cm uni-directional translation relative to the cabinet. (F) itch scratching, scratch tool from ± 5 cm unidirectional translation, humanoid root pose from ± 5 cm planar translation, and 19 unique itch scratching locations on the humanoid. 3 manual demonstrations are collected for each location (57 total). (G) humanoid drinking, mug w/straw from $\pm$5cm unidirectional translation and $\pm 30^\circ$ rotation over the table, and humanoid/wheelchair root from $\pm$7.5cm planar translation.

B.2 Transformer hyperparameters

The input sequence length was set to $T_p = 250$ and the future prediction length to $T_f = 150$. Each time a forward pass runs on the transformer, the simulator executes $T_e = 10$ actions, which are

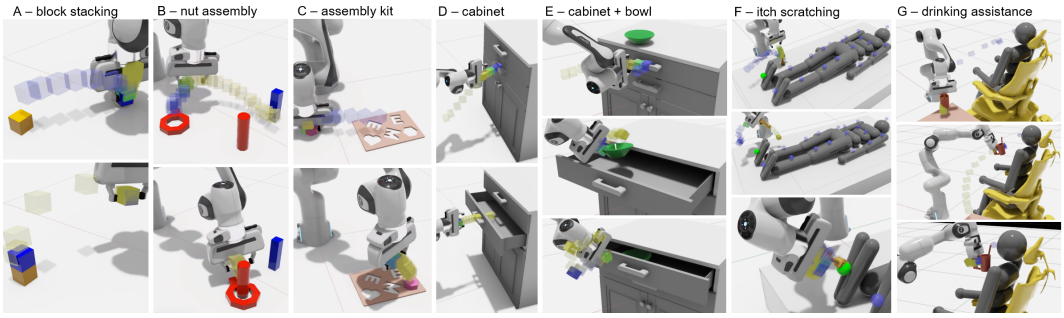

Figure 4: Task examples that the transformer completes successfully. (A) the blue block is stacked upon the orange block (B) the round nut is placed on the round peg. (C) the pink hexagon is fit into the assembly kit board. (D) the bottom cabinet drawer is opened. (E) the green bowl is put in the top cabinet drawer. (F) an itch scratching tool is picked up and used to scratch an itch on the bottom of the left foot. (G) a mug with a straw is picked and brought to a person in a wheelchair.

fed back into the transformer. Predicted actions are fed directly back to the transformer, while real simulator states resulting from the actions are input to the model. The transformer contains 6 layers, 8 heads, a hidden layer size of 256, and it is trained with a batch size of 128. During pre-training, we used a learning rate of 1e-4 that linearly decreased to 5e-5, and a learning rate of 5e-5 for training on other task data.

Appendix C: Results

We present additional qualitative results in Fig. 4 showing success during automatic model prediction on new task scenarios. This provides further evidence for the success of few-shot learning with transformers. All examples in the figure are for the pretrained model that was fine-tuned on individual tasks.

C.1 Limitations

The transformer model has poor generalization performance for precision tasks with few (<= 50) examples. The Assembly Kit from TransporterNets consists of five precisely fitting shapes into a board. The performance is low for a single precisely fitting shape (the Hexagon, at 10%), as shown in Table 1 of the main paper. However, if the goal criteria is set more loosely (i.e. the shape is next to the goal but not quite in the slot), then performance is 46%. It also has some difficulty picking the hexagon, because the hexagon is almost as wide as the open gripper max width.

Most failures happen largely due to imprecise grasping and the robot unable to recover from these failures. In some cases, the robot grasps the object but stops midway and never reaches the goal or stays frozen after grasping. This may be to due to out of distribution errors due to limited demonstrations.

References

[1] Arthur Allshire, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger. *arXiv preprint arXiv:2108.09779*, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.